THE STATE COMMITTEE FOR COMMUNICATION, INFORMATIZATION AND TELECOMMUNICATION TECHNOLOGIES OF THE REPUBLIC OF UZBEKISTAN

TASHKENT UNIVERSITY OF INFORMATION TECNOLOGIES

As the manuscript

UDK 004.056

Islomov Shahboz Zokir ugli

Using of image recognition methods in the system of information protection

5A330302 – Information security

Dissertation
is written for taking the degree
master academic

Scientific advisor:
d.t.s., prof. Karimov M.M

CONTENT

INTRODUCTION

For solving the problems on the computerization of society, the development of information technology and security June 27, 2013y DP-1989 Decree of the President of the Republic of Uzbekistan " On measures for further development of the National Information and Communication System of the Republic of Uzbekistan" The measures defined by the Decree , the establishment of national systems provide information, conditions for mass adoption in the economy and the life of every member of society, computer and information technologies, increase the competitiveness of the domestic economy in the world market [1].

Uzbek face recognition experts have contracted with department Freelancer.in which is the platform for clients to meet evaluate their professional competencies and undertake the hiring process after being satisfied with their abilities to provide quality work.

To ensure the development, installation and maintenance of such a highly effective system, the creativity and expertise of Uzbek face recognition professionals are necessary. This is where Freelancer.in will help us meet, interact, evaluate and hire the most qualified software developers to handle your project. As you will realize, the face recognition system uses technology with a man-to-machine interface that guarantee no contact at all. This biometric system replaces the otherwise time consuming, tradition ID card or password-based technology.

Rationale of research and its relevance. Face recognition is one of the most relevant applications: pattern recognition, neural networks, computer graphics, image processing and psychology of image analysis. Face recognition was included as an unavoidable preprocessing step for face recognition, and as an issue by itself, because it presents its own difficulties and challenges, sometimes quite different from face recognition. We have soon recognized that the amount of published information is unmanageable for a short term effort, such as required of a errors and problems in this area, so in agreement with my supervisor we have stopped at a reasonable time, having reviewed most conventional face detection and face recognition approaches, leaving advanced issues, such as video face

recognition or expression invariance, for the future work in the framework of a doctoral research. I have tried to gather much of the mathematical foundations of the approaches reviewed aiming for a self contained work, which is, of course, rather difficult to produce.

Object and inventory of research: master's dissertation object is the software of face recognition. Inventory is the system of protecting information, video observing and access control, processes face recognition.

The goal of the research. Analyses of face recognition methods and development program module on the base finally algorithm.

The tasks:

- Studying face recognition and face detection modules and their localization;
- Analyses of face recognition methods;
- Experimenting face recognition algorithms by adding of modification methods;
- Creating the algorithm of program for face recognition;
- Developing software on the base finally algorithm

The hypothesis and main problems of the research. The need for a robust, accurate, and easily trainable face recognition system becomes more pressing as real world applications such as biometrics, law enforcement, and surveillance continue to develop. However, extrinsic imaging parameters such as pose, illumination and facial expression still cause much difficulty in accurate recognition. Recently, component-based approaches have shown promising results in various object detection and recognition tasks such as face detection, person detection, and face recognition.

The hypothesis is theory of numbers, mathematical analysis, module arithmetic, combination operator and comparison methods which used to develop of face recognition software.

Short analyses of references by the research. A literature study to find general information on face recognition and issues regarding circumvention of face

recognition products and the human perception of faces. 1332 articles are published in only one year – 2012. First researcher was Woodrow W. Bledsoe in this area. He was started Panoramic Research in Palo Alto, California in1960. The South Korea are researching more with face recognition and detection. Ph.D. Candidate in Dept. of EE in KAIST Dae Hoe Kim is researching in theme "Development of Mammography Computer Aided Detection (CAD) - Image enhancement" [3]. His recent studies have shown that computer-aided detection (CAD) systems could play an important role in increasing the breast cancer detection rate in face technology. M.S. and Ph.D. degrees Yong Ju Jung, Ph.D Candidate in Dept. of EE and Ph.D Candidate in Dept. of EE from Korea Advanced Institute of Science and Technology (KAIST) are experimenting in project "Attention model-based visual comfort assessment for stereoscopic 3D videos" [4]. Their research interests include 3D image/video processing, 3D visual comfort assessment, 2D/3D video quality, human 3D perception. Article in theme "Model Estimation and Selection towards Unconstrained Real-Time Tracking and Mapping" [11] is published by Department of Computer Science , University of California, Santa Barbara, Sweeney, C., Ventura, J., Turk, M.. They presented an approach and prototype implementation to initialization-free real-time tracking and mapping that supports any type of camera motion in 3D environments, that is, parallax-inducing as well as rotation-only motions.

Theoretical and practical significance of the results research's. Theoretical significance of dissertation work is estimating and analysis of default hardware-software algorithms and practical significance is making new face recognition software and using inuniversities of Uzbekistan.

Scientific novelty is creating software for face detection and recognition using by based on finally algorithm.

The composition of research. This dissertation work consists of an introduction, three chapters, conclusion, literatures and appendix. Also, there are *9* tables, *72* figures and *2* block schema. Total dissertation pages is *79*.

4 thesis and 2 articles were published on this dissertation theme.

Chapter I. Problems of using face recognition and detection technologies in information and communication systems

1. Classification applications of imaging recognition

Face recognition is one of the most relevant applications of image analysis. It's a true challenge to build an automated system which equals human ability to recognize faces. Although humans are quite good identifying known faces, we are not very skilled when we must deal with a large amount of unknown faces. The computers, with an almost limitless memory and computational speed, should overcome humans limitations.

Face recognition remains as an unsolved problem and a demanded technology - see table.1. A simple search with the phrase "face recognition" in the IEEE Digital Library throws 9422 results. 1332 articles in only one year - 2012. There are many different industry areas interested in what it could offer. Some examples include video surveillance, human-machine interaction, photo cameras, virtual reality or law enforcement. This multidisciplinary interest pushes the research and attracts interest from diverse disciplines. Therefore, it's not a problem restricted to computer vision research. Face recognition is a relevant subject in pattern recognition, neural networks, computer graphics, image processing and psychology. In fact, the earliest works on this subject were made in the 1950's in psychology. They came attached to other issues like face expression, interpretation of emotion or perception of gestures [25].

Engineering started to show interest in face recognition in the 1960's. One of the first researches on this subject was Woodrow W. Bledsoe. In 1960, Bledsoe, along other researches, started Panoramic Research, Inc., in Palo Alto, California. The majority of the work done by this company involved AI-related contracts from the U.S. Department of Defense and various intelligence agencies. During 1964 and 1965, Bledsoe, along with Helen Chan and Charles Bisson, worked on using computers to recognize human faces. Because the funding of these researches was provided by an unnamed intelligence agency, little of the work was published. He

continued later his researches at Stanford Research Institute. Bledsoe designed and implemented a semi-automatic system. Some face coordinates were selected by a human operator, and then computers used this information for recognition. He described most of the problems that even 50 years later Face Recognition still suffers - variations in illumination, head rotation, facial expression and aging. Researches on this matter still continue, trying to measure subjective face features as ear size or between-eye distance. For instance, this approach was used in Bell Laboratories by A. Jay Goldstein, Leon D. Harmon and Ann B. Lesk. They described a vector, containing 21 subjective features like ear protrusion, eyebrow weight or nose length, as the basis to recognize faces using pattern classification techniques. In 1973, Fischler and Elschanger tried to measure similar features automatically. Their algorithm used local template matching and a global measure of fit to find and measure facial features [3].

There were other approaches back on the 1970's. Some tried to define a face as a set of geometric parameters and then perform some pattern recognition based on those parameters. But the first one that developed a fully automated face recognition system was Kenade in 1973. He designed and implemented a face recognition program. It ran in a computer system designed for this purpose. The algorithm extracted sixteen facial parameters automatically. In he's work, Kenade compares this automated extraction to a human or manual extraction, showing only a small difference. He got a correct identification rate of 45-75%. He demonstrated that better results were obtained when irrelevant features were not used.

In the 1980's there were a diversity of approaches actively followed, most of them continuing with previous tendencies. Some works tried to improve the methods used measuring subjective features. For instance, Mark Nixon presented a geometric measurement for eye spacing. The template matching approach was improved with strategies such as "deformable templates". This decade also brought new approaches. Some researchers build face recognition algorithms using artificial neural networks.

The first mention to eigenfaces in image processing, a technique that would become the dominant approach in following years, was made by L. Sirovich and M. Kirby in 1986. Their methods were based on the Principal Component Analysis. Their goal was to represent an image in a lower dimension without losing much information, and then reconstructing it. Their work would be later the foundation of the proposal of many new face recognition algorithms.

The 1990's saw the broad recognition of the mentioned eigenface approach as the basis for the state of the art and the first industrial applications. In 1992 Mathew Turk and Alex Pentland of the MIT presented a work which used eigenfaces for recognition. Their algorithm was able to locate, track and classify a subject's head. Since the 1990's, face recognition area has received a lot of attention, with a noticeable increase in the number of publications. Many approaches have been taken which has lead to different algorithms. Some of the most relevant are PCA, ICA, LDA and their derivatives. Different approaches and algorithms will be discussed later in this work.

The technologies using face recognition techniques have also evolved through the years. The first companies to invest in such researches where law enforcement agencies - e.g. the Woodrow W. Bledsoe case. Nowadays diverse enterprises are using face recognition in their products. One good example could be entertainment business. Products like Microsoft's Project Natal or Sony's PlayStation Eye will use face recognition. It will allow a new way to interact with the machine. The idea of detecting people and analyzing their gesture is also being used in automotive industry. Companies such as Toyota are developing sleep detectors to increase safety. These and other applications are raising the interest on face recognition. It's narrow initial application area is being widened (table.1).

To resist intruders, you must be able to find them. Axon’s Intellect integrated Facial Recognition module notifies operators when it spots a human face within the video frame. It automatically detects and captures the image of that person and compares it to known photos of persons of interest [4].

Table.1.

Applications of face recognition

Areas	Applications
Information security	Access security (OS, data bases), data privacy (e.g. medical records) and user authentication (trading, on line banking)
Access management	Secure access authentication (restricted facilities), permission based systems, access log and audit trails
Biometrics	Person identification (national IDs, passports, voter registrations, driver licenses) and automated identity verification (border controls)
Law enforcement	Video surveillance, suspect identification, suspect tracking (investigation), simulated aging and forensic reconstruction of faces from remains
Personal security	Home video surveillance systems and expression interpretation (driver monitoring system)
Entertainment leisure	Home video game systems and photo camera applications

2. Define of face recognition and detection modules

Face Recognition module. You can use this detection tool to create a database of employees or other individuals (fig.1): a camera situated in an entrance can record all who pass through the turnstile and save images of their faces to a database [5]:

- Identifies people regardless of facial hair, hairstyle, eyewear, aging or angle, and in a variety of background conditions;
- A non-intrusive, contact-free process, unlike other biometrics;
- Compatible with legacy databases;
- Real-time notification about recognized persons;

- Easy integration with existing systems;
- Real-time notification of identity matches and alerts;
- Automatic synchronization with users database;
- Integration with access control systems;
- Recognition algorithms powered by Cognitec SDK v 5.0;

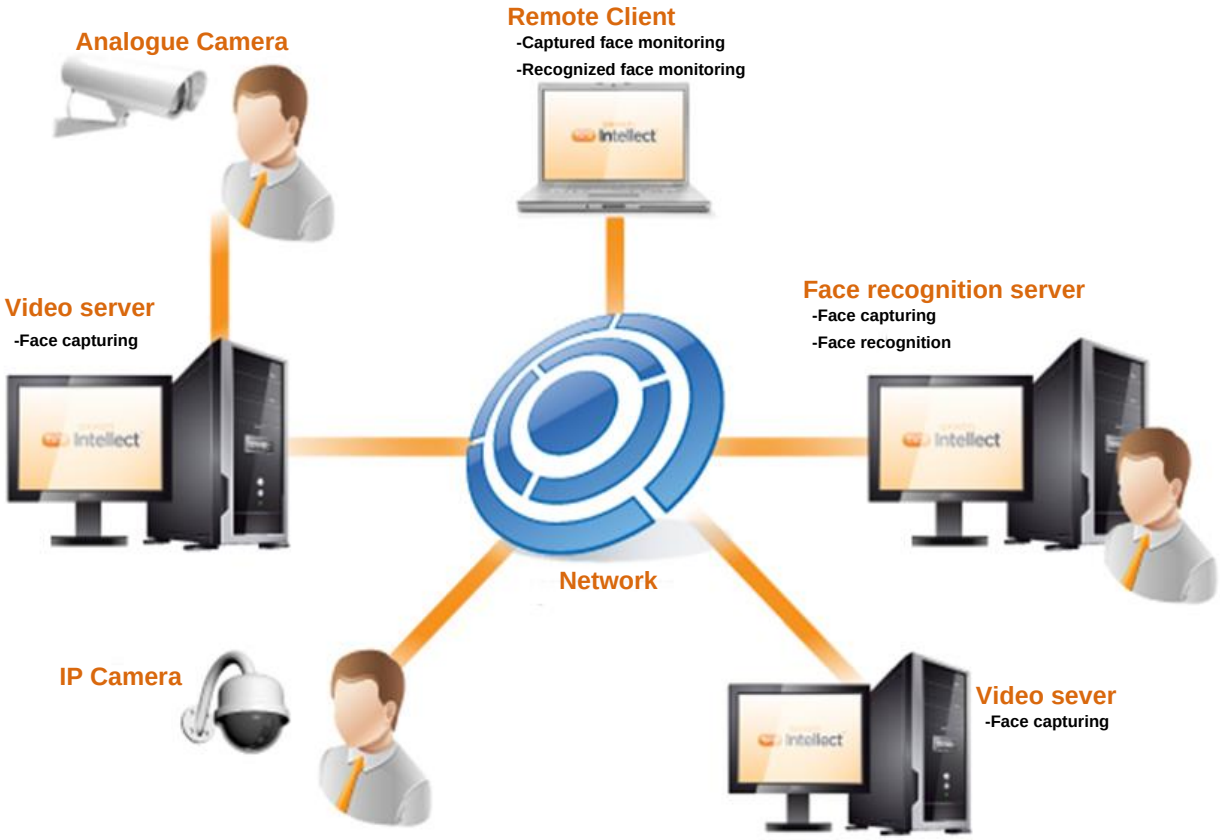


Fig.1.Face recognition module

The face recognition module automatically compares an image chosen by the face capture module with images stored in a database. Identification algorithms, powered by the Cognitec engine, guarantee high probability of correct recognition and quick search of databases containing hundreds or thousands of images. The Facial Recognition module integrates with various biometric systems for identifying human faces, from checkpoints to criminal databases.

Face search module. The face search module, powered by the Cognitec or VeriLook engine, creates a database of all faces captured by video cameras and lets you search the database for similar faces. To search, indicate a frame in the video

archive containing a face, indicate a link (URL) to an image, or upload an image of a face to the system. The results are displayed as a list of photographs sorted by similarity. This module makes your search in video footage for persons of interest dramatically faster, as well as collecting statistics on capture by various cameras.

Diverse Applications. Intellect's Facial Recognition and Face Search modules are designed for use at public places, airports, stadiums, border control zones, prisons, critical infrastructure and military sites.

Investigation and search activities. Upon recognition of a person recorded into an investigational database by the module, the operator receives all available information and immediately notifies law enforcement authorities. The Face Search module is a great time-saver for investigation and search activities based on video footage.

Restricted access objects requiring the highest level of security. Traditional access control systems cannot prevent an unauthorized person from using an access card. The Facial Recognition module authenticates a card holder automatically, by comparing the face in the video frame with the image in the database

Face identification at border crossings (Intellect Enterprise can connect to external databases with images of terrorists and wanted criminals), with simultaneous cross-check of facial image to passport/ID.

While traditional face recognition systems have to date been primarily designed to operate under controlled environments based on previously acquired photographs and videos, it is now becoming increasingly important to identify humans in real-time under unconstrained environments. Designing such a system is challenging because the acquisition conditions are subject to variable pose of the subject, low illumination and possible occlusions.

Our goal is to utilize the robustness and parallel computational abilities of a distributed camera network to address these challenges and achieve accurate, real-time human identification.

Our main areas of research are:

- Design of middleware services that utilize the geometry of the network to collaborate at run-time and enable the rapid extraction of multi-view face data and soft-biometric data (such as gait and human anthropometric features);

- Camera configuration techniques that enhance the probability of acquiring suitable face images for recognition while minimizing the overall cost of deployment;

- Efficient fusion algorithms and multi-view face recognition techniques that are optimized for operation in a dynamic, continually streaming mode (fig.2).

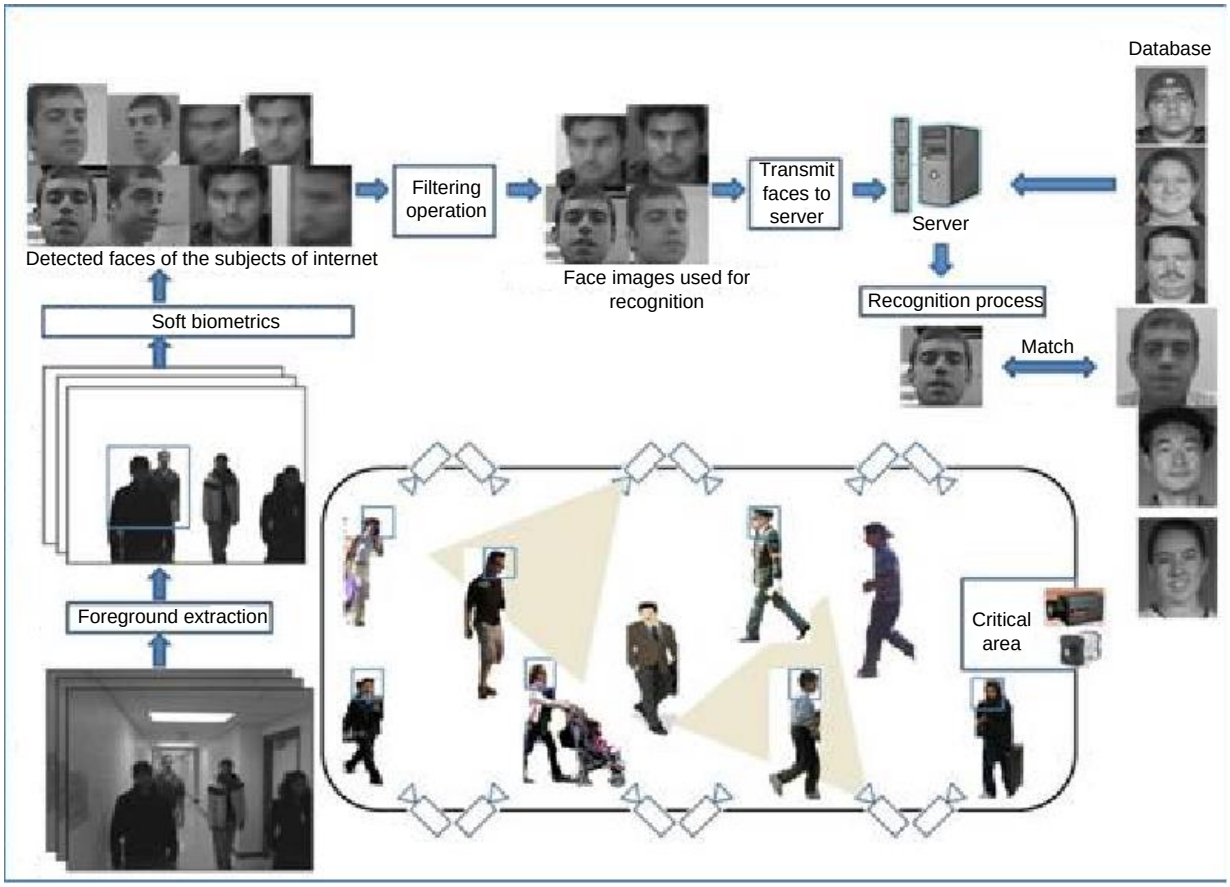


Fig.2. Types of research feces

Face detection module. Face Detection is an innovative technology which can be integrated into any webpage as a Silver light plug in. It takes snapshots using a web camera, searches for one or multiple faces in an image using a sophisticated two-dimensional DSP algorithm.

Our brand new intellectual algorithm goes beyond the limits of all existing technologies as it obtains the unique functional features that enable the application to [6]:

- work stably with any skin color;
- accept low-quality images (e.g. pixilated, blurred, noised, darkened, etc.);
- detect faces with glasses;
- determine the position of the particular facial features (e.g. eye pupils, nose, lips, etc.);
- detect several faces on one image.

Face Detection technology may be used as a core component (basis) for a great number of applications with a wide sphere of usage.

1. *Smart capture.* It is a variety of an already existing capture which uses graphic images and sounds. However, using a face and/or motion detection technology you are not going to bother users with recognizing bizarre letters and unclear sounds any more. All the user needs is to show his face in motion so that the website owner is sure that he is a human being, not a machine. No more losing partners and visitors, no worry of being spammed by robots.

2. *Webcam based energy/power saver*. This is an absolutely innovative idea which may be helpful to everyone who uses a PC, TV or some other types of household appliances. The main feature here is to switch on a sleep mode or reduce the brightness of a screen whenever a user's face is away from his device for more than 5-10 minutes and to set it back to work automatically when a user's face is detected. There is no need to say that nowadays the problem of power saving is one of the most urgent. Using face detection module you will be closer to its solving than ever.

3. *Time tracking service*. If you are interested in tracking the billed hours of your employees this is exactly what you need. No way to cheat you, no more money spent in vain. You will only pay for hours spent by your employee on his working place. In case a web camera no longer detects a face and/or motion in

front of it the time tracking is stopped. In the moment a person comes back it starts again. As easy as ABC.

4. *Outdoor surveillance camera service*. Face and motion detection technology may help making a good service to keep you secure. Firstly, it will give you a signal of somebody's coming even before he rings the doorbell. Secondly, with its help you will be able not to watch the whole video recording when it is necessary, but to choose the periods when motion or face was detected, so that it saves your time considerably.

5. *Video chat service*. If you are an owner of a video chat our application should meet some of your needs. All of your customers obviously have a webcam. Unfortunately, there are the people who tend to spoil your reputation making the rest of the users see indecent pictures of a sexual nature. Using the face detection technology you can be sure that all of the visitors of your website behave properly and feel comfortable even if their children use this service. Moreover, it could be a sound idea to use the face detection module to define the status of the visitor. If he is not present in front of the screen he obtains an "Away" status, when he comes back and his face is detected, "Online" status appears and the rest of the users are able to see him again.

To design an Automated Face Recognition system, one needs to address several related problems:

- detection of an image pattern as a subject and then as a face against either uniform or complex background;
- detection of facial landmarks for normalizing the face images to account for geometrical and illumination changes;
- identification and verification of face images using appropriate classification algorithms and possibly post processing the results using model-based schemes and logistic feedback.

Concerning with the face detection task, we develop the algorithm that decides first if a face is present, and if the face is present it crops ('box') the face. The approach used for face detection involves three main stages, those of location,

cropping, and post processing. The first stage finds a rough approximation for the possible location of the face box, the second stage will refine it, and the last stage will possibly decide whether face is present in the image. The first stage locates **BOX1,** possibly surrounding the face, using simple but fast algorithms in order to focus processing for the cropping stage. The location stage consists of three steps: (i) histogram equalization, (ii) edge detection, and (iii) analysis of projection profiles. The cropping stage labels each 8x8 window from **BOX1** as face or non-face using decision trees (DT). The DT are induced ('learned') from both positive ('face') and negative ('non- face') examples expressed in terms of features such as entropy, mean, and standard deviation (SD). The labeled output of the cropping stage, is post processed to (i) decide if a face is present, and if it is present to (ii) derive its **BOX2** in terms of straight boundaries, and to (iii) normalize the box to yield uniform sized boxes BOX3 for all face images. The architecture for implementing the above procedure for detecting the face box is shown below in Fig.3.

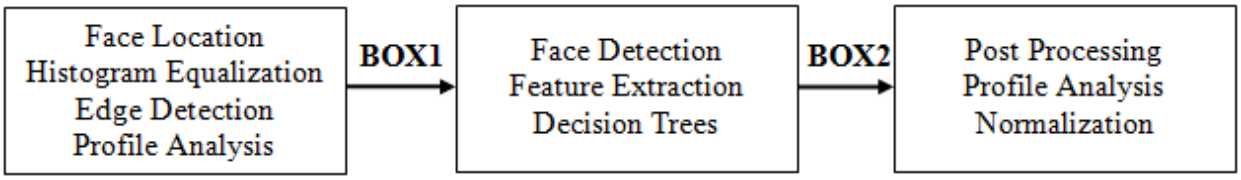


Fig.3. Face detection module

3. Detect the localization of persons in facial technologies

Face detection is a computer technology that determines the locations and sizes of human faces in digital images. It detects face and ignores anything else, such as buildings, trees and bodies. Face detection can be regarded as a more general case of face localization. In face localization, the task is to find the locations and sizes of a known number of faces (usually one). In face detection, face is processed and matched bitwise with the underlying face image in the database. Any slight change in facial expression, e.g. smile, lip movement, will not match the face.

Face localization is a fundamental step in the process of face recognition. Its aim is to decide whether there is a face in a given image and, in the positive case, to determine the coordinates of the face. The accuracy of the detected face coordinates has a heavy influence on the recognition performance.

Face detection can be regarded as a specific case of object-class detection. In object-class detection, the task is to find the locations and sizes of all objects in an image that belong to a given class. Examples include upper torsos, pedestrians, and cars.

Face-detection algorithms focus on the detection of frontal human faces. It is analogous to image detection in which the image of a person is matched bit by bit. Image matches with the image stores in database. Any facial feature changes in the database will invalidate the matching process.

A facial recognition system is a computer application for automatically identifying or verifying a person from a digital image or a video frame from a video source. One of the ways to do this is by comparing selected facial features from the image and a facial database.

It is typically used in security systems and can be compared to other biometrics such as fingerprint or eye iris recognition systems.

In I presented a method for robust frontal face detection based on the Hausdorff distance. This algorithm uses a predefined edge model of the human face to find face candidates in the image. While it is possible to use a simple ellipse as a model, the detection performance can be improved by using a more detailed model. However, it must still represent a wide variety of faces.

This section gives a brief overview of the underlying face detection algorithm. A more detailed description can be found in article *Audio- and Video-Based Person Authentication* [7].

The face detection problem can be stated as follows: given an input image, decide whether there is a face in the image or not. If a face was found, return the face coordinates inside the image. In our case these are the coordinates of the left

and right eye centers, which is sufficient if the problem is restricted to frontal view faces in images of constant aspect ratio.

For simplicity, we concentrate on the task of finding a single face in an image. The extension to finding multiple faces is straightforward.

Some facial recognition algorithms identify facial features by extracting landmarks, or features, from an image of the subject's face. For example, an algorithm may analyze the relative position, size, and/or shape of the eyes, nose, cheekbones, and jaw. These features are then used to search for other images with matching features. Other algorithms normalize a gallery of face images and then compress the face data, only saving the data in the image that is useful for face recognition. A probe image is then compared with the face data. One of the earliest successful systems is based on template matching techniques applied to a set of salient facial features, providing a sort of compressed face representation.

Recognition algorithms can be divided into two main approaches, geometric, which looks at distinguishing features, or photometric, which is a statistical approach that distills an image into values and compares the values with templates to eliminate variances.

Popular recognition algorithms include Principal Component Analysis using eigenfaces, Linear Discriminate Analysis, Elastic Bunch Graph Matching using the Fisherface algorithm, the Hidden Markov model, the Multilinear Subspace Learning using tensor representation, and the neuronal motivated dynamic link matching.

A newly emerging trend, claimed to achieve improved accuracies, is three-dimensional face recognition. This technique uses 3D sensors to capture information about the shape of a face. This information is then used to identify distinctive features on the surface of a face, such as the contour of the eye sockets, nose, and chin [8].

One advantage of 3D facial recognition is that it is not affected by changes in lighting like other techniques. It can also identify a face from a range of viewing angles, including a profile view. Three-dimensional data points from a face vastly

improve the precision of facial recognition. 3D research is enhanced by the development of sophisticated sensors that do a better job of capturing 3D face imagery. The sensors work by projecting structured light onto the face. Up to a dozen or more of these image sensors can be placed on the same CMOS chip—each sensor captures a different part of the spectrum.

Even a perfect 3D matching technique could be sensitive to expressions. For that goal a group at the Technion applied tools from metric geometry to treat expressions as isometries A company called Vision Access created a firm solution for 3D facial recognition. The company was later acquired by the biometric access company Bioscrypt Inc. which developed a version known as 3D FastPass.

The Media Analysis Management Interface (MAMI) enables the understanding of the real world at a low cost by using analysis engines such as video image processing engines, sensor data analysis engines, and so on. It also enables various services to be easily provided, such as physical security, environmental load reduction, and intelligent accessibility services. The MAMI Incubator Group (MAMI-XG) discussed the requirements and determined the feasibility of the MAMI, which consists of the data models and exchange protocols for the analysis data of various media. In this document, the findings of the group are summarized(fig.4).

We use an edge-based method for finding faces. Therefore we first calculate an edge magnitude image with the Sobel operator. The relevant edge feature points are extracted by a locally adaptive threshold filter to compensate variable illumination. We assume that this procedure will produce a characteristic arrangement of segmentation points in the facial area.

Based on the typical layout that strongly depends on the segmentation steps, we use a face model which itself consists of a set of feature points and can be represented as a binary image.

The feature points of the face model are chosen in a way that the pattern stored in the model is somehow similar to the typically observed patterns in the images' face area.

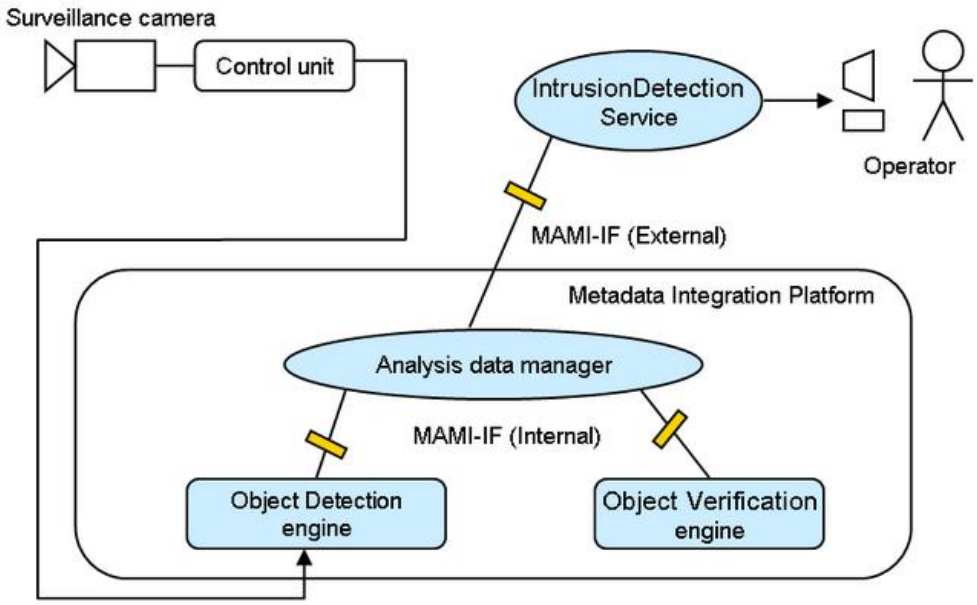


Fig.4. The Media Analysis Management Interface

To detect a face, the model is superimposed over the image at several discrete positions. At each position the similarity between the translated model and the covered part of the image is calculated. A face is considered to be located at the position yielding the highest similarity between model and covered image part. The procedure is illustrated in fig.5. Note that the model can be scaled to allow detecting faces of different sizes.

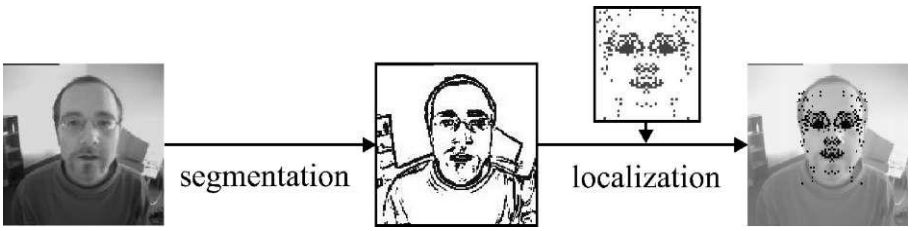


Fig.5. Face finding procedure

An efficient yet powerful method to calculate the similarity of two binary images is the Hausdorff distance, a metric between two point sets. We use a slightly adapted measure, called the (directed) *modified Hausdorff distance* (MHD) to calculate the similarity between the image and the model. Given the two point sets A and B and some underlying norm on the points, the With the two-dimensional point set A representing the image and $T_p(B)$ representing the translated and scaled model with transformation parameters p, the formula

$$dp = \min \text{hmod}(A, Tp(B))$$

attention calculates the distance value of the best matching position and scale. The parameters of the corresponding transformation are represented by the parameter set p.

Genetic Algorithms are a powerful tool that can help in finding an appropriate model for face localization. The presented framework led to a model that performed considerably better than a simple hand-drawn model.

Face localization can be improved by a multi-step detection approach that uses more than one model in different grades of detail. Each of these models can then be optimized separately. This does not only speed up the localization procedure but also produces more exact face coordinates [20].

Face detection tasks are becoming required more frequently in the modern world. It's caused by the development of security systems as an answer to acts of terrorism. In addition, these algorithms are widely used in interactive user interfaces, in advertisement industry, entertainment services, video coding, etc. However, many researchers mostly paid their attention to Face Recognition algorithms considering Face Detection tasks (necessary first step for all face recognition systems) to be almost solved.

In this work we represent faces by coordinates of the centers of the eyes (i.e. centers of the pupils), because first, this representation looks to be more opportune in terms of the results comparison; second, usually face recognition algorithms require the centers of eyes matching for learning samples; third, experts mark eyes faster, easier and more precisely than they mark faces by rectangles. Thus, to unify the resulting comparison method we suggest eyes reconstruction model, which receives a face location in rectangle representation and returns estimated coordinates of the centers of eyes.

To make an efficient implementation possible, we use a discrete grid for the model point positions. A model can then be represented by a binary image where white pixels represent the model points. The resolution of this image has to be high enough to be able to represent enough detail but has to be as low as possible to

minimize computation time for both the localization procedure and the model optimization process. We used a 45 x 47 model grid which has turned out to be a good trade-off.

One of the major problems of the Hausdorff distance method is the actual creation of the face model. While a simple "hand-drawn" model will be sufficient for the detection of simple objects, a general face model must cover the broad variety of different faces [26].

The task of finding a well-suited model for Hausdorff distance based face localization can be formulated as a discrete global optimization problem. An exhaustive search would produce the optimal result (with respect to a given sample set), but due to the exponential complexity it is not computationally feasible. In the broad area of global optimization methods, Genetic Algorithms (GA) form a widely accepted trade-off between global and local search strategy. They were chosen here for they are well-investigated and have proven their applicability in many fields.

One of the major problems in model-based face detection is the creation of a proper face model. We have presented a genetic algorithm approach for obtaining a binary edge model that allows localization of a wide variety of faces with the Hausdorff search method.

The experiments showed that the GA performs better when starting from scratch than from a hand-drawn model. With this method, the localization performance could be improved to more than 90% compared to roughly 60% for the hand-drawn model.

If detected faces are represented by the centers of the eyes (fig.6.), let's consider them to be correctly detected, if and only if detected eyes belong the area around the true eyes location with the diameter D_{Eyes}. Which depends on the distance between eyes' centers and a, has been taken equal to 0.25 (This criterion was originally used by Jesorsky et al.), and calculates as $D_{Eyes} = 2a \times l_{Eyes}$.

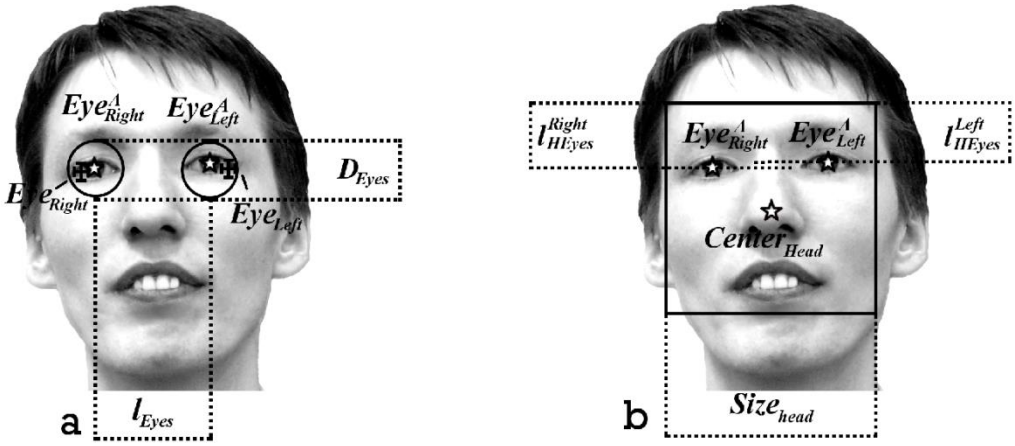


Fig.6. Schematic face representation

Assume there is a full face portrait image with no incline (fig.6.b), and the algorithm has found its center and size - *(Center$_{Head}$* and *Size$_{Head}$* respectively). Obviously, the eyes on this image are located symmetrically about the vertical axis (i.e., at the half the distance between them: *l$_{Eyes}$/2)* and at the same distance *(l$_{HEyes}$)* from the top border of the face's rectangle. Thus the absolute coordinates of eyes can be estimated as:

$$^{Eye}\text{Right} = {}^{Eye}\text{left} = {}^{Center}\text{Head} + l_{HEyes} - {}^{Size}\text{Head}$$

$$^{Eye}\text{Right} = {}^{Center}\text{Head} - 2\ {}^{l}\text{Eyes}$$

$$^{Eye}\text{left} = {}^{Center}\text{Head} + 2\ {}^{l}\text{Eyes}$$

Let's try to estimate the parameters of the algorithm, namely *l$_{Eyes}$* and *l$_{HEye}$s*, as an average of the huge amount of images with experts' labeled eyes. Based of such analysis, the following coefficients have been founded: *A* - average proportion of distance between top border of the face and center of the eyes *(l$_{HEyes}$)* to the size of the face rectangle; and *B* - average proportion of the distance between eyes *(l$_{Eyes}$)* to the size of the face rectangle *(SizeHead).*

It should be noted that such "conversion" of face representation could deteriorate the localization accuracy for algorithms describing faces by rectangles. However, all benefits from the "conversion" (discussed in the beginning of this section) are much bigger than risks mentioned above [21].

4. Problems and errors in face recognition systems

Implementing a system which enables or restricts access to the VISL lab, using automatic computer based recognition. Performance requirements of the face detection stage were a high detection rate and a low false alarm rate, while maintaining high scanning rate during face detection. In addition a low error rate was required during the face recognition stage.

The system has to deal with varying face size and location in the frame, changing facial expressions, and a non constant lighting. The system was built on a regular PC, using a simple video camera.

The difficulty in the face detection process stems mainly from a large number of potential face candidates (about $2*10^5$) per frame. The classification of each candidate is not negligible, and in order to reach desired detection and false alarm rates, an even more complex preprocess is needed. Even after sampling the picture and using motion detection, we are left with a large number of about 5000 candidates.

Previous systems used classifiers of constant input size, in order to scan the frame using all locations. In order to find faces at different distances from the camera, smaller versions of the picture were created. These versions were also scanned by the classifier. This process requires a lot of time and resources. In fact, it is the bottleneck of the scanning process, preventing it from working in real time. We call his problem “The Pyramid Problem”.

In order to overcome these difficulties, we will be using a novel approach developed by Viola and Jones, and augmented by Linheart et. al. For the remaining segments, an SVM implementation shall be given.

When the authorities decide which face recognition product to use, two important criteria are the False Acceptance Rate (FAR) and False Rejection Rate (FRR) of the products. Traditional estimation of FAR of face recognition products is usually based on zero effort impostors'. In a real border control environment traditional estimation of FAR with *zero effort* impostors are not necessarily representative for the real amount of false acceptances. Potential attackers with or

without plenty of resources could use several technological and physical techniques to circumvent the system. This could involve physiological alteration of their appearances using masks, facial make-up, different facial hair or plastic surgery, or technological techniques to alter information about an applicant for a visa. Also, identical twins is traditionally problematic when using face recognition, although a supplier of 3D face recognition claims to have countered this problem. To obtain a more realistic evaluation of FAR it is therefore important to examine possible attacks, their influence on the FAR, and resources needed to perform them. Such research will enable the authority and other users of face recognition products to perform more enlighten evaluation of face recognition products, and make them aware of the problems so that they can execute necessary countermeasures.

There exists little or no publicly available data on face recognition products response to attackers that perform an effort other than simply supplying their own biometric data hoping that they will circumvent the system. How can those employing such systems know which system to use when the evaluation is based surely on zero effort circumvention? This could very well result in the choice of the lesser product. Also, those who employ such system should be aware of the different approaches that exist for circumvention, so that they can make measures to thwart this. This thesis will provide an overview of these attacks and how they are done.

Institutions that are employing face recognition products will undoubtedly benefit from a survey that has demonstrated the effect of non-zero effort impostors, since this would make them more aware of the potential differences between traditional estimation of FAR, and when it is based on non-zero effort attacks. Hopefully this will make for a demand for more realistic evaluations of FAR, more in accordance with the environments in which it will be employed. Users may then avoid potentially costly pitfalls. In Norway such stakeholders could be UDI, which are heavily involved in the introduction of face recognition in the new NORVIS

(the Norwegian version of VIS) system, and other institutions that decide to use face recognition products [10].

The intention of the thesis is to see if the face recognition products available today are adequate in a setting such as the new Visa Information System. To evaluate this, the authority performing the evaluation should have information of the potential threats that could arise. This thesis will provide such information by giving an overview of some of the threats that exists towards face recognition software, and an evaluation of the probability for such threats occurring in a border control environment, in a setting like the new Visa Information System.

The new visa system involves biometrics such as fingerprint and face recognition. This thesis will examine the different methods for circumventing face recognition products, involving the resources and skills needed and the potential cost. The focus of this thesis will be on the use of face recognition in a border control environment with non-zero effort attackers and the effect these will have on face recognition products reliability and performance. In order to find out how impostors will affect the face recognition systems, the following issues will be examined.

When deciding which research methods to use, we used J.W. Creswell's book *Research design* as a basis. In this thesis we look at different methods for circumvention of face recognition products that affect the false acceptance rate. The primary method used for analyzing this problem is literature survey. To establish the impact non-zero effort attacks have on the FAR of a face recognition product in a border control environment, we have used a mixed methods approach. The intention was to contact individuals in the face recognition community, to see if they had literature or knowledge of literature about circumvention of face recognition systems that could be used in the thesis work. This however did not result in any usable material on circumvention, but we did receive information on face recognition in general and more specific information about the Visa Information System. A thorough examination of available literature through the

Internet and different libraries provided the material necessary to conduct my thesis.

A literature is studied to find general information on face recognition and issues regarding circumvention of face recognition products and the human perception of faces. The literature study was used to gain an increased knowledge within these areas, and to obtain ideas on a useful experiment. To be able to use a literature study, there has to be relevant literature available, and we should have access to the necessary databases. Access to the Gjovik University College library and the databases available through this library, IEEE and Citeseer, combined with web searches provided most of the necessary literature to perform the study. Contacts within the face recognition community and the VIS provided additional literature that were necessary to perform this study.

Email correspondence with contacts within the face recognition community and the government to obtain knowledge beyond what is possible from searching the web and using the library. To be able to correspond with such contacts, information about such contacts should be available. This was achieved through a former employer and the teaching supervisor we where able to come in contact with such contacts. We did receive vital information about face recognition and the Visa Information System using this method.

An experiment on human comparison of face image-pairs with and without hair, to evaluate the effect hair has on human ability to recognize faces. The image-pairs used in the experiment had already been accepted as the same individuals of a computer- based face recognition system. The success of this experiment depended on enough people being willing to participate in the experiment. By using this method we were able to measure the effect hair has on human ability to recognize faces. This way we were provided an indication of how easy or difficult it is to circumvent both a computer-based face recognition system and a human supervisor.

False acceptance and false rejection. There are two types of error a biometric system camera make: False rejection which is when a legitimate user is

rejected, and false acceptance which is when an illegitimate user is accepted as someone else. The probability that a genuine person is rejected is called false rejection rate (FRR), while false acceptance rate (FAR) is the probability that an impostor is accepted as a legitimate person [11].

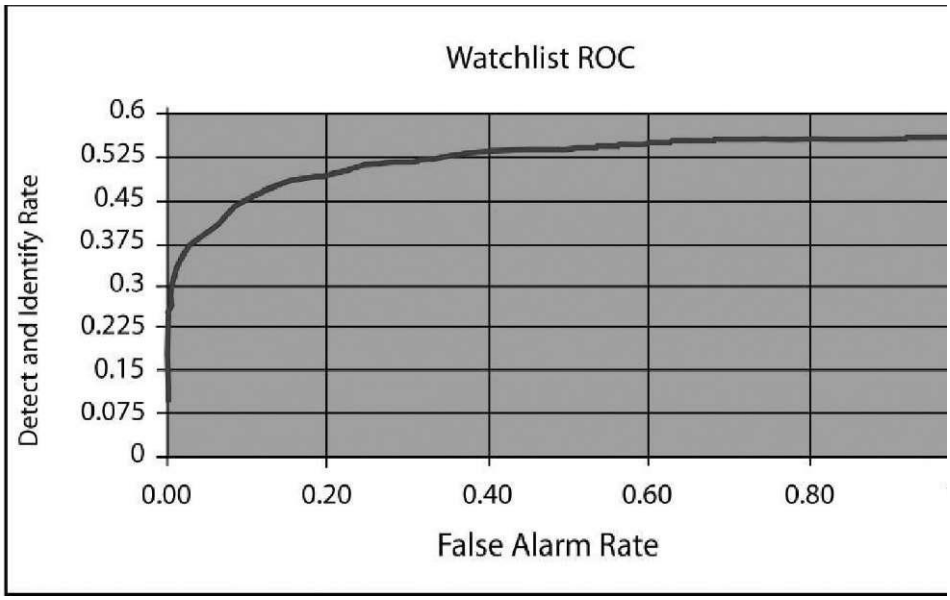


Fig.7. Watchlist receiver operating characteristic

The figure is taken from *Biometrics, Duane M. Blackburn*. The detect and identify rates and their associated false alarm rate is plotted into the diagram. A WROC helps to better see the give-and-take relationship between false alarm rate and the correct detect and identify rate.

Minimizing in error rate for face recognition. Automated face recognition is a relatively new concept. Developed in the 1960s, the first semi-automated system for face recognition required the administrator to locate features (such as eyes, ears, nose, and mouth) on the photographs before it calculated distances and ratios to a common reference point, which were then compared to reference data. In the 1970s, Goldstein, Harmon, and Lesk used 21 specific subjective markers such as hair color and lip thickness to automate the recognition. The problem with both of these early solutions was that the measurements and locations were manually computed. In 1988, Kirby and Sirovich applied principle component analysis, a standard linear algebra technique, to the face recognition problem. This was considered somewhat of a milestone as it showed that less than one hundred

values were required to accurately code a suitably aligned and normalized face image. In 1991, Turk and Pentland discovered that while using the eigenfaces techniques, the residual error could be used to detect faces in images; a discovery that enabled reliable real-time automated face recognition systems. Although the approach was somewhat constrained by the environmental factors, the nonetheless created significant interest in furthering development of automated face recognition technologies. The technology first captured the public's attention from the media reaction to a trial implementation at the January 2001 Super Bowl, which captured surveillance images and compared them to a database of digital mug shots. This demonstration initiated much-needed analysis on how to use the technology to support national needs while being considerate of the public's social and privacy concerns. Today, face recognition technology is being used to combat passport fraud, support law enforcement, identify missing children, and minimize benefit/identity fraud. As one of the most successful applications of image analysis and understanding, face recognition has recently gained significant attention. Over the last ten years or so, it has become a popular area of research in computer vision and one of the most successful applications of image analysis and understanding.

A facial recognition system identifies a person by comparing a newly captured image to a database of stored images. When the system is linked to a video surveillance system, an algorithm program is used to search for faces. Facial scan technology works well with commercial off-the-shelf (COTS) PC video capture cameras, and generally requires 320 x 240 resolution and at least three to five frames per second. Higher frame per second rates directly translate into higher performance. Facial recognition systems use programs that take facial images and measure the unique characteristics (geometry) of the face and create a template file. Using these templates, the software then compares the image with a stored image and produces a score that measures how similar the images are to each other. Typical sources of images in facial recognition include video camera signals and pre-existing photos, such as those in driver's li-cense databases. Once the

images are extracted, a template is created. The template is a compressed amalgamation of images and is usually less than 1/100 the size of the original. The leading approaches in face recognition today are Local Feature Analysis (LFA), Laplacian-Faces and Eigen-Face.

Facial biometrics gets complex when lighting and angles change. Hence, face recognition performance suffers at longer distances. For example, conventional 3D face recognition cannot be used at distances beyond 2 meters without help of specialized surveillance cameras. People do not stop for cameras nor do they often look directly into the camera. Non-cooperative behavior creates angle and lighting variability that decreases accuracy. Two-dimension cameras have accuracy issues when a subject is in motion and as light varies. 3D biometrics can resolve this through multiple solutions, including infrared. However, 3D cameras are rather expensive, and suffer from various problems typical for immature technologies.

As a result, most of currently available face recognition systems are subject to a relatively high levels of False Accept Rate and False Reject Rate which either compromise the reliability of the systems, or makes their use uncomfortable due to high level of False Rejection. Note that most systems can be calibrated such that there is a trade off between two error rates. Therefore, improving one of these features of the algorithm is expected to improve the overall usefulness of the system.

In this paper we propose a novel approach to reducing the False Acceptance error rate, and therefore significantly improving the usability of such systems. This is done using a collaborative algorithm, aiming for reducing the effect of malicious attacks on the system. The proposed method relies on the fact that although at certain times, certain cameras may mistakenly grant access to an unauthorized person, collaboratively processing these events may reduce the number of such cases dramatically. In order to do so, we use a collaborative algorithm for monitoring malicious applications in mobile phones, first proposed in. Originally, this algorithm was designed to analytically guarantee that a dynamic mobile

network would become immune to a stream of attacks by malicious applications. Adapted to the domain of collaborative face recognition, we show that this algorithm can also provide the ability to reduce the combined error rates of a system of face recognition cameras for admission control.

This paper is a position paper, presenting the problem and a proposed solution. Extensive simulation experiments will be carried out in the future, and reported in a full version of this paper. The rest of the paper is organized as follows: Section 2 presents the collaborative face recognition problem, while Section 3 presents our proposed solution. Theorem 1 shows an upper bound on the convergence time as well as the overall network overhead of the algorithm. Theorem 2 presents a lower bound over the improvement factor that is guaranteed by using the proposed algorithm, while Corollary 1 shows that this factor is monotonically increasing with the size of the network, n. Section 4 concludes the paper and discusses future work.

Given a face recognition system, comprising cameras, each having a False Accept Rate of A and False Reject Rate of we are interested in implementing a coordination layer, on top of the existing system, that would guarantee a decreased error rates. We would also like this layer to have as small overhead as possible.

We assume that each camera can respond with 3 possible results, each time it is asked to grant access to a person: Accept, Reject, Unsure. When responding with Unsure, the system is assumed to treat this result as if it was Accept. Nonetheless, although for the person seeking access there are only two possible results (namely, Accept, Reject), the system can store the entries for which it had responded with an unsure result.

Let denote a malicious person, disguised as an authorized person a, and occasionally recognized by the system as Unsure, with probability AM. We would like to minimize this probability. Notice that as a actively tries to deceive the system, we can assume that its acceptance probability is greater than the average False Acceptance rate, namely: $\lambda_m > \lambda_a$.

We assume that O(ln n) reports that the same person was given access based on an Unsure result is enough in order to safely classify this person as a malicious attacker.

Let us assume that attacker a encounters the system on average once every days. We assume that the face recognition system is distributed (due to security considerations) but that units can pass messages on a random network overlay

In order to collaboratively monitor attack attempts on the system, we shall utilize the TPP collaborative monitoring algorithm, first presented in. This algorithm was first developed in order to implement a collaborative monitoring scheme for mobile devices against malicious applications. The algorithm assumes that an agent is activated in any mobile device that chose to participate in this service. This agent periodically monitors one or more mobile applications, derives conclusions concerning their maliciousness, and reports its conclusions to a small number of other mobile devices. Each mobile de-vice that receives a message (conclusion) propagates it to one additional mobile device. Each message has a predefined TTL. Upon receiving enough alerting messages (namely, more than a decision threshold p), a unit can classify an application as malicious. In this work we shall use the TPP algorithm for collaboratively monitor attacks against a distributed face recognition system. Every time a person seeking access would be recognized with an Unsure result, the system would still grant this person access, but would react as though this was an application being identified as malicious. Namely, the unit would generate alerts messages and propagate them throughout the network. When a receiving unit had received more than p messages, it would stop granting access to this person, when it is recognized as Unsure.

Every D days, some of the n units of the system is approached by a and in probability AM is accepted, with an Unsure result. In this case, the unit with which a is currently engaged would generate an alert message, ac-cording to the TPP algorithm. This message would be propagated throughout the network of face recognition units, and would guarantee that after several encounters of some of the

system's units with a, the vast majority of the units would be aware that the acceptance of a is frequently done on the basis of its identification as Unsure.

In order to verify that the system meets the requirements of the TPP algorithm, we would select its parameters as follows. The TPP algorithm guarantees that a large enough portion of the network is vaccinated against a given malicious threat. This portion equals $(1 - P_{max}) \bullet n$, where p_{max} denotes the penetration threshold of the system. Note that when the system is initialized person a have admission access in probability of XM. Note also that relying on the properties of the TPP algorithm, once the algorithm converges the new acceptance probability of a equals:

$$P_{max} * \lambda_m \lambda`_a$$

In this work we have presented the algorithm for collaborative face recognition, aimed for reducing False Accept and False Reject Error rates of any system of face recognition cameras. The algorithm relies on the TPP collaborative application monitoring algorithm that was presented in. Using the analytic properties of the TPP algorithm, we have shown that the error rates of any system comprising a multitude of face recognition cameras can be improved.

The improvement factor relies on the ratio of λ_m and λ_a, namely — what is the benefit of actually trying to deceive the system, compared to its "usual" False Acceptance Rate, and is given in Theorem expected due to the fact that the system is collaboratively using the collective experience of its units, the improvement factor grows monotonically with the size of the network.

The idea of collaboratively performing a face recognition mission was proposed in the past in the context of annotating photos for web and social networks applications.

Based on the typical layout that strongly depends on the segmentation steps, we use a face model which itself consists of a set of feature points and can be represented as a binary image.

Summary of chapter I

We can conclude that face recognition is most interesting because the subject is not necessarily aware that his identity is being verified, this is very useful for surveillance applications. Circumvention is an important factor to consider when choosing face recognition for authentication purposes, furthermore permanence and uniqueness of the face might remain a limitting factor of this biometric solution.

In 2012, the performance of the latest face recognition algorithms was evaluated in the Face Recognition Grand Challenge. High-resolution face images, 3D face scans, and iris images were used in the tests. The results indicated that the new algorithms are 1000 times more accurate than the face recognition algorithms of 2012 and 100 times more accurate than those of 2005. Some of the softwares were able to outperform human participants in recognizing faces and could uniquely identify identical twins.

So, in this chapter is destroyed problems of using face recognition and detection technologies in information and communication systems, classification applications of imaging recognition, defining of face recognition and detection modules, detecting the localization of persons in facial technologies, errors in face recognition systems.

Chapter II. Analysis of face recognition algorithms and methods for solving problems

1. Face recognition algorithms base on point and distance between key-points

Facial detection is a particular case of an object detection. It uses basic detection algorithm which can be used to detect objects of any type (eyes, mouth corners, other facial feature points, hands, cars, etc.) Firstly, the algorithm normalizes snapshot's brightness and contrast. Secondly, it moves a 'window' through a snapshot by gradually changing its left, top, width and height, which is called window-sliding. Every window is analyzed by an object classifier that decides whether a particular region of the snapshot is a face or not. As a result, we are having a list of regions where face (or another object) is detected. Regions are grouped in order to remove duplicated detections of the same object.

Four baseline face recognition algorithms have been developed. They are [27]:

- A standard PCA or Eigen-faces algorithm;
- A combination PCA and LDA algorithm based upon the University of Maryland algorithm in the FERET tests;
- A Bayesian Intrapersonal/Extra-person Image Difference Classifier based upon the MIT algorithm in the FERET tests;
- An Elastic Bunch Graph Matching Algorithm that uses localized landmark features represented by Gabor jets. This algorithm is based upon the USC algorithm in the FERET tests.

To review how these algorithms perform today on the four FERET probe sets used in much of the original FERET evaluation see our summary pages. To understand how these pages are generated, review our system User's Guide and download your own copy of the CSU Face Identification Evaluation System.

The object classifier is defined as a set of Haar features with weights and thresholds. A 2-rectangle Haar feature can be defined as a difference of the sum of pixels of areas inside a rectangle, which can be located at any position and scale

within the original image. The classifier also uses 3-rectangle features and 4-rectangle features. The values indicate certain characteristics of a particular area of the image. Each feature type can indicate the existence (or absence) of certain characteristics in the image, such as edges or changes in texture [18].

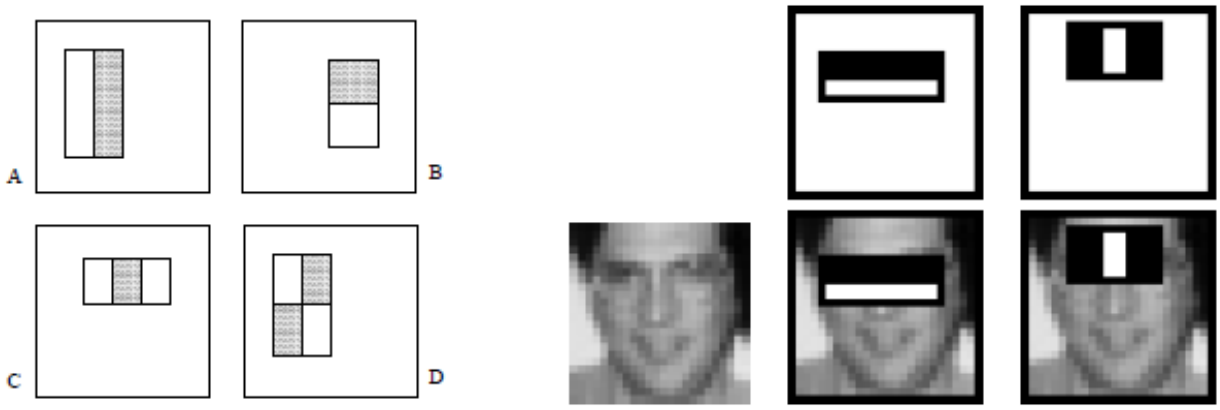


Fig.8. Haar features

For example, a 2-rectangle feature can indicate where the border lies between a dark region and a light region.

The haar like feature with polarity (+1, -1) and threshold defines a weak classifier. Feature value is multiplied by polarity and compared with the threshold. As a result, we have a boolean decision of classification. The decision of weak classifier has low accuracy (normally, the correct classification rate is 50%-80%). In order to improve accuracy, multiple statistically independent weak classifiers are grouped together. Weighted sum of weak decisions is used to make a final decision of a strong classifier. The features, weights and thresholds are obtained using Adaboost training algorithm offline and stored in XML file. Typical face classifier contains 500-1000 weak classifiers. In order to increase speed of object detection, weak classifiers are grouped in stages, a stage usually contains 4-100 weak classifiers.

Automatic Facial Land-marking using Active Shape Models (ASMs). This is my primary research area. I am looking into ways of improving the fitting accuracy of ASMs to enable the automatic annotation of unseen faces. ASMs are deformable templates that can be trained to automatically position a pre-defined set of landmarks along the contours of an object of interest. They are most commonly

used for facial landmarking and if initialized well, using a suitable face detector, produce fairly accurate results. Currently my implementation works fairly well with frontal faces and can automatically annotate them with a dense set of 79 landmarks. I am working towards extending it to handle pose and expression variation. There are several potential applications of this tool including face recognition (accurate registration can be made possible), expression analysis, pose estimation using regression models, generation of 3D facial models etc.

Fig.9. Landmarks of Active Shape Models

This work was abled to apply ASMs to the task of tracking the previously mentioned 79 facial landmarks across the frames of different video sequences in which the subject showed rapid in-plane rotation and out of plane pose variation. This task was accomplished by Kalman filtering the landmark coordinates. The predictive mechanism of the Kalman filter ensured accurate initialization of the ASM on the next frame (without the need for face detection) while its corrective

mechanism, treated the landmark locations produced by the ASM as noisy observations that were refined to produce more accurate results. We benchmarked our approach against naive methods that 1) did not harness any temporal information and treated each frame independently 2) initialized the ASM on frame n+1 using the fitting results on frame n but without Kalman filtering and found that our approach produced far lower fitting error. The applications of this work include tracking and facial recognition in surveillance footage. Our system is not currently capable of operating in real-time but can be used for post processing [9].

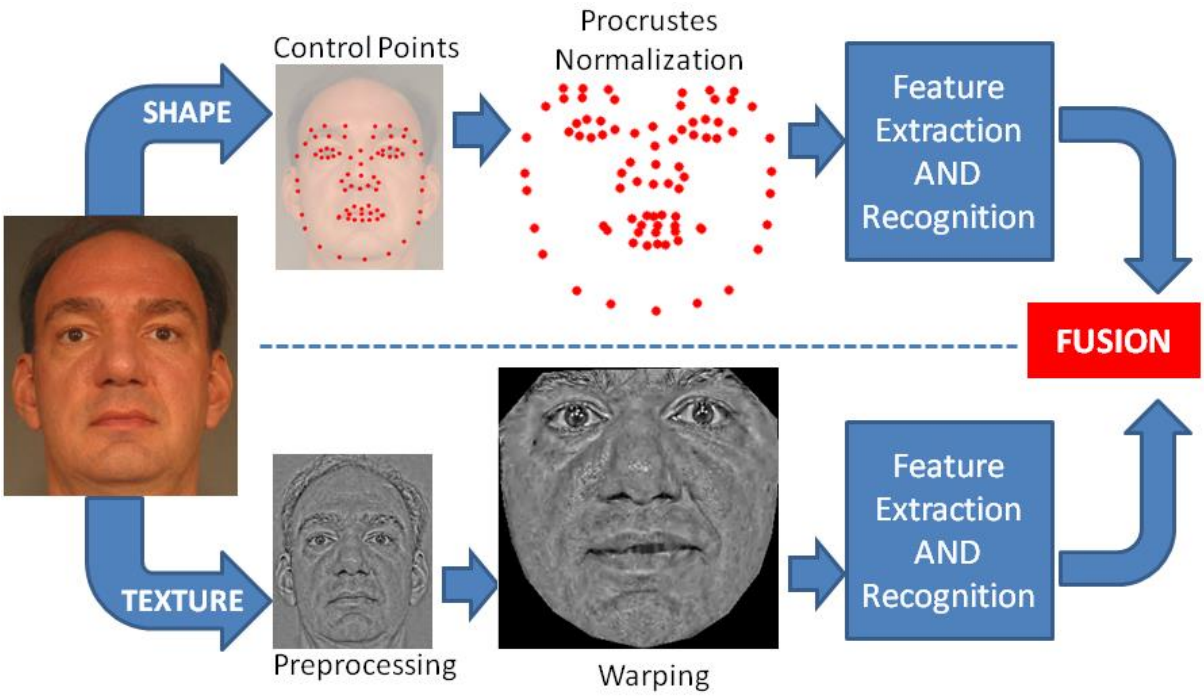


Fig.10. Using shape and texture information for improving facial Recognition

Conventional face recognition algorithms simply crop out the facial region, extract features from the crop and match based on these features. We aimed at a different approach to harness more information from images and hence boost facial recognition rates. Using our dense land-marking scheme, we performed experiments in which the shape information contained purely in the locations of these landmarks was harnessed while shape-free texture information was harnessed separately by warping all faces to a common mean face. Purely shape based features and purely texture based features were then used for matching and the results were subsequently fused. The results we obtained demonstrated that 1)

purely shape based recognition is fairly discriminative 2) significant performance improvement can be gained by better registration of texture using shape information and 3) fusing shape and texture based methods consistently boosts the performance of subspace-based approaches such as Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA) etc. To the best of our knowledge, this is the first large scale study of the roles of shape and texture in facial recognition as we carried out our tests on the vast and challenging NIST Facial Recognition Grand Challenge (FGRC) version 2.0 database.

Kernel Discriminant Transformation (KDT) for Linear Subspaces. It presents an automatic face recognition system based on a novel kernel discriminant transformation (KDT) algorithm. Compared to existing face recognition algorithms regarding a single test image, the proposed system utilizes facial image sets consisting of arbitrary head poses, facial expressions and lighting conditions. Since the distribution of the image sets are nonlinear and highly-overlapped, we generate a corresponding kernel subspace for each image set in a high-dimensional feature space using kernel principal component analysis (KPCA). An optimal KDT matrix is then formulated to maximize the similarity between the kernel subspaces of the same subject and simultaneously minimize that between the kernel subspaces of different subjects. While the KDT matrix cannot be computed explicitly, we present an iterative kernel discriminant transformation algorithm that alternatively solves the matrix in an implicit way. For matching each pair of the kernel subspaces generated from the image sets, we address another perspective of similarity measure, namely canonical difference, to simplify the formulation. The recognition result is obtained by computing the canonical difference between each pair of the test and the training kernel subspaces. The proposed face recognition system is demonstrated to outperform existing still image based as well as image-set-based recognition systems.

The second row displays corresponding manifolds constructed from the first row, showing that the facial images are highly non-linearly distributed in 3D Eigen-faces [19].

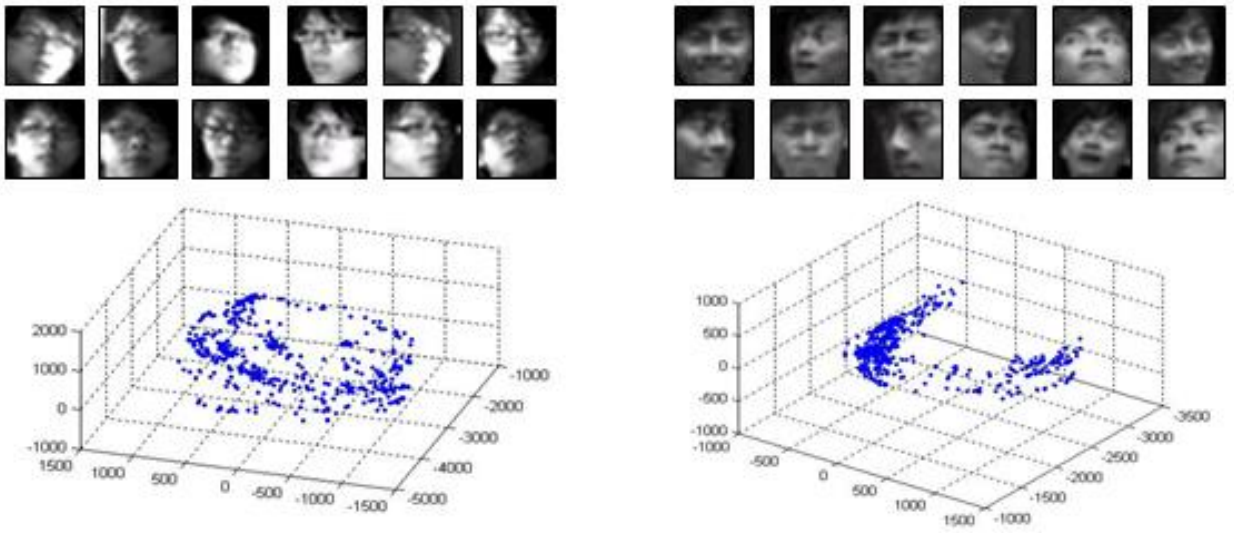


Fig.11. Examples of image sets containing unconstrained head motions with different lighting conditions (left) and facial expressions (right).

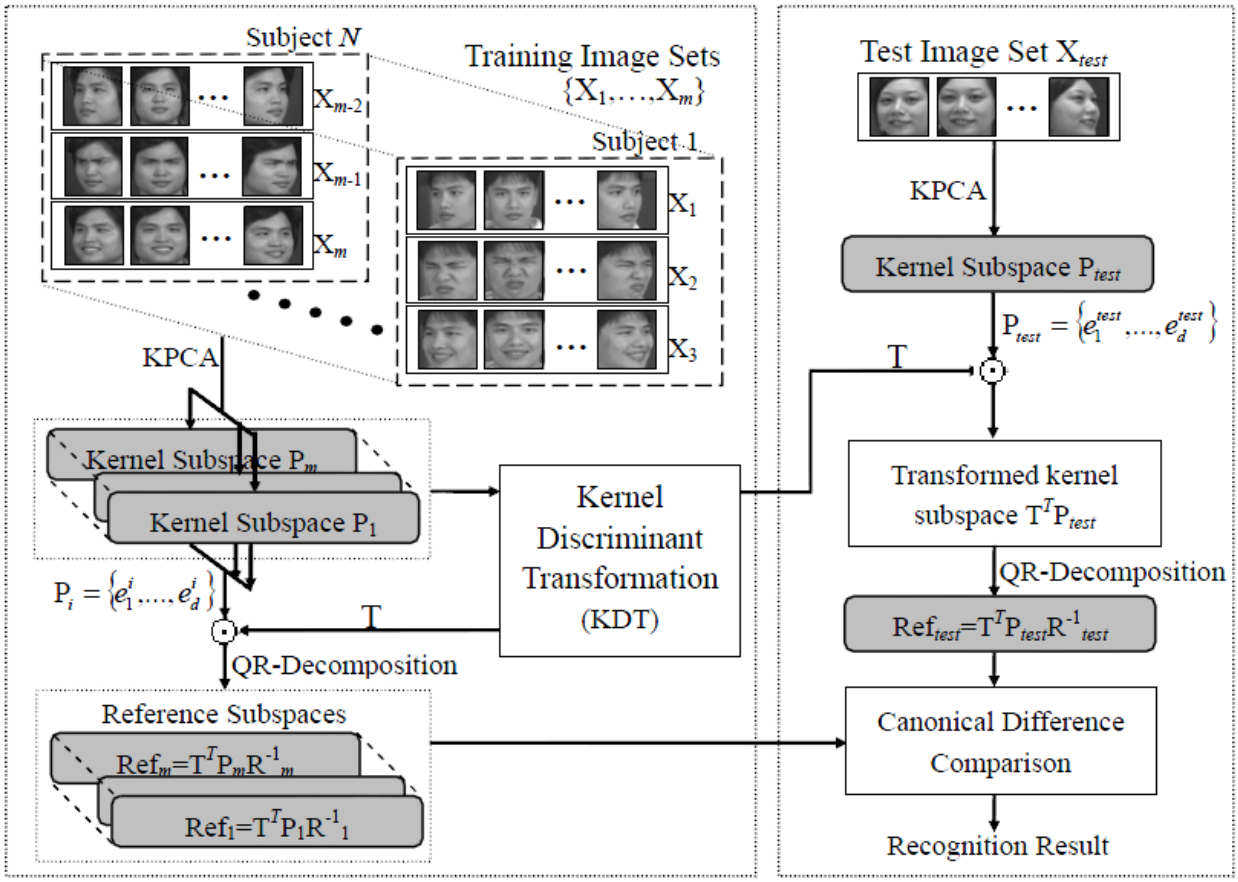


Figure 2.5. Flowchart of proposed face recognition system (left: training process, right: test process)

An optimal transformation matrix T is trained and given to the test process to maximize the canonical correlation of intra-subspaces while minimizing that of inter-subspaces.

Component-Based Constraint Mutual Subspace Method. The objective of this method is to recognize faces with temporal image sequences captured by arbitrary CCD camera. The difficulties of face recognition are primarily the variations of pose, illumination, expression and occlusion. We propose a component-based face recognition system based on canonical correlations between subspaces constructed from facial image sets. For identifying the facical images, the system is capable of automatically collecting patches of five facial components through cascaded pre-processing steps. Then, Constrained Mutual Subspace Method (CMSM) is performed on subspaces constructed from each facial component. Finally, the similarity of two sequences is calculated as the summation of weighted similarity of facial components between different subjects [22].

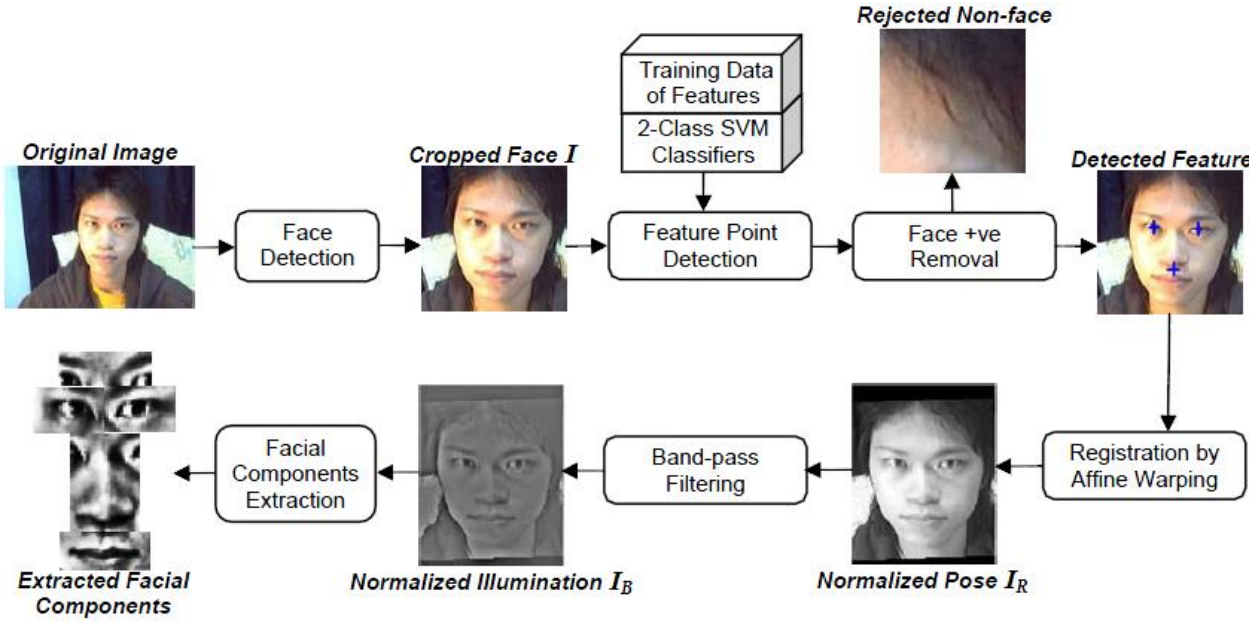


Fig.12. A schematic representation of the proposed facial component extraction process

Different color represents for different subject. It is obvious that the distribution of component subspace after projection is much easier to be classified.

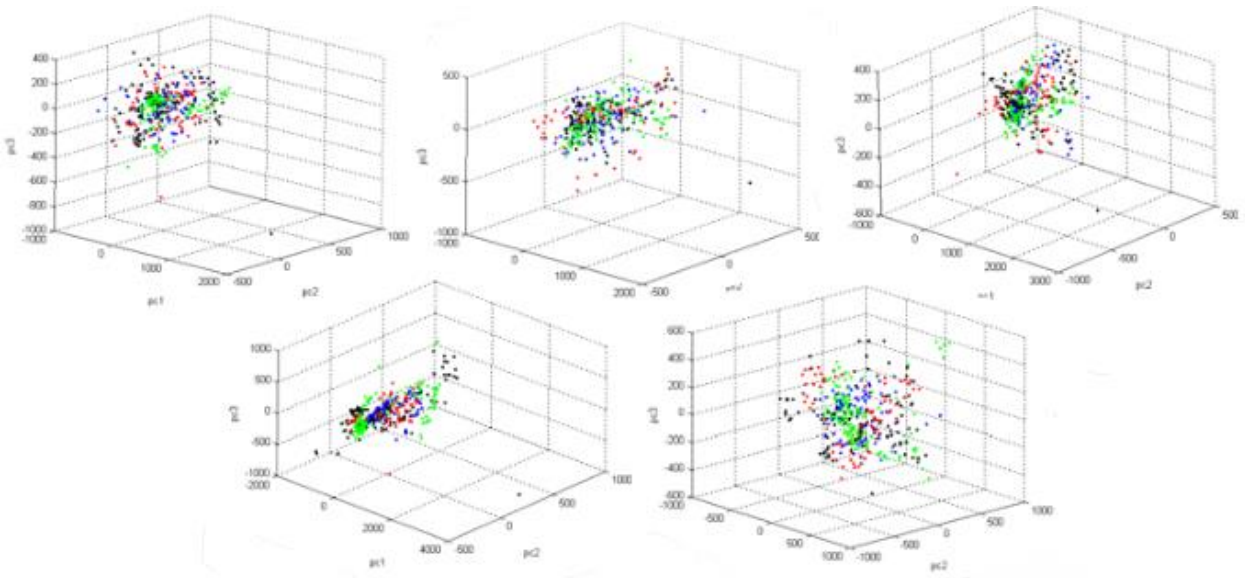


Fig.13. Top/bottom rows display the five subspaces of each facial component subspace before/after the projection on corresponding constraint subspaces

Cam Shift Algorithm (Haar Face Detection Algorithm). Eyes are the most important features of the human face. So effective usage of eye movements as a communication technique in user-to-computer interfaces can find place in various application areas.

Eye tracking and the information provided by the eye features have the potential to become an interesting way of communicating with a computer in a human-computer interaction (HCI) system. So with this motivation, designing a real-time eye feature tracking software is the aim of this project.

The purpose of the project is to implement a real-time eye-feature tracker with the following capabilities:

- Real Time face tracking with scale and rotation invariance;
- Tracking the eye areas individually;
- Tracking eye features;
- Eye gaze direction finding;
- Remote controlling using eye movements.

Adaptive Mean Shift algorithm is used for tracking human faces and is based on robust non-parametric technique for climbing density gradients to find the mode (peak) of probability distributions called the mean shift algorithm. As faces are tracked in video sequences, mean shift algorithm is modified to deal with

the problem of dynamically changing color probability distributions. The block diagram of the algorithm is given below [10]:

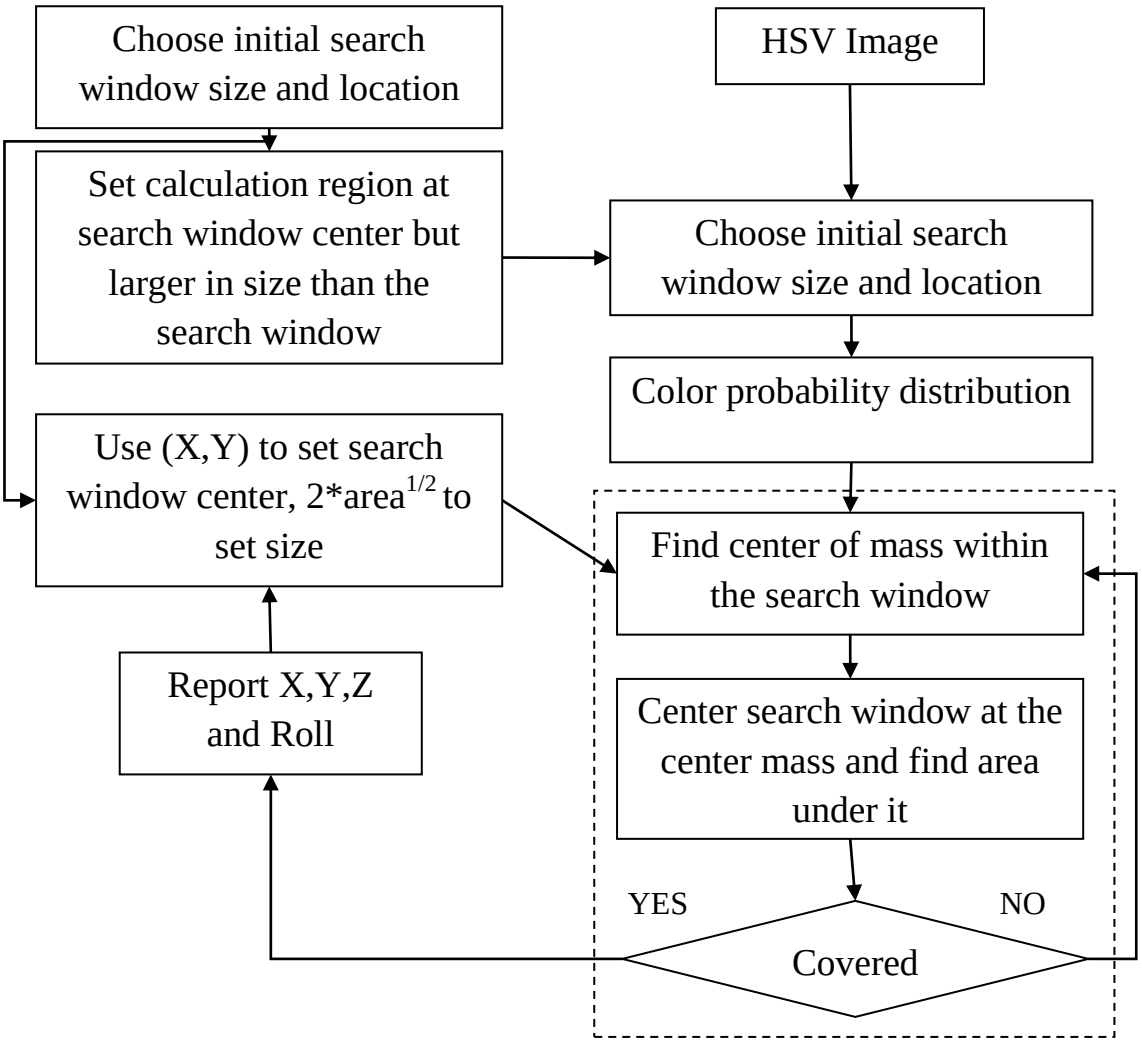


Fig.14. Haar-Face Detection Method

The second face detection algorithm is based on a classifier working with Haar-Like features (namely a cascade of boosted classifiers working with Haar-like features). First of all it is trained with a few hundreds of sample views of a face. After a classifier is trained, it can be applied to a region of interest in an input image. The classifier outputs a "1" if the region is likely to show face and "0" otherwise. To search for the object in the whole image, one can move the search window across the image and check every location using the classifier. The classifier is designed so that it can be easily "resized" in order to be able to find the objects of interest at different sizes, which is more efficient than resizing the image itself.

Template-Matching is a well-known method for object detection. In our template matching method, a standard eye pattern is created manually and given an input image, the correlation values with the standard patterns are computed for the eyes. The existence of an eye is determined based on the correlation values. This approach has the advantage of being simple to implement. However, it may sometimes be inadequate for eye detection since it cannot effectively deal with variation in scale, pose and shape.

Adaptive Eigen Eye Method. Adaptive Eigen Eye Method is based on the well-known method Eigen Faces. However as the method is used for eye detection we named it as "Eigen Eye Method". The main idea is to decompose eye images into a small set of characteristics feature images called Eigen eyes, which may be thought of as the principal components of the original images. These Eigen eyes function as the orthogonal basis vectors of a subspace called eye space. However we know that the Eigen face method is not scale invariant. To provide the scale invariance we can resize the eye-database once with the information gathered by the face detection algorithm (Eye Width / Face Width ? 0.35), we can provide scale-invariant detection using only one database. A simple rectangular Haar-like feature can be defined as the difference of the sum of pixels of areas inside the rectangle, which can be at any position and scale within the original image. This modified feature set is called 2-rectangle feature. Viola and Jones also defined 3-rectangle features and 4-rectangle features. The values indicate certain characteristics of a particular area of the image [11].

We are currently working on a more sophisticated geometric transformation to cover more facial expressions. Although the results in section show only a small improvement (with the exception of MLP where the improvement was significant), we suppose there is great potential of using samples with synthetic expression. The triangular model of face enables to extend the generation algorithm by other possibilities like generation of samples with different poses and illumination conditions.

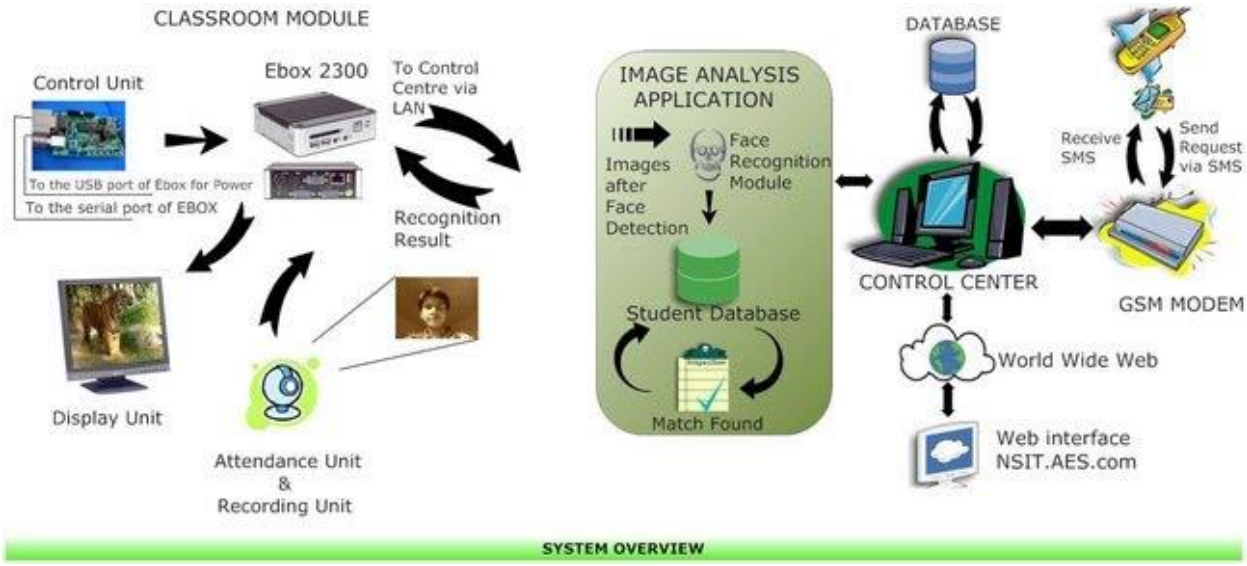


Fig.15. Facial Detection method through Haar Detection Method

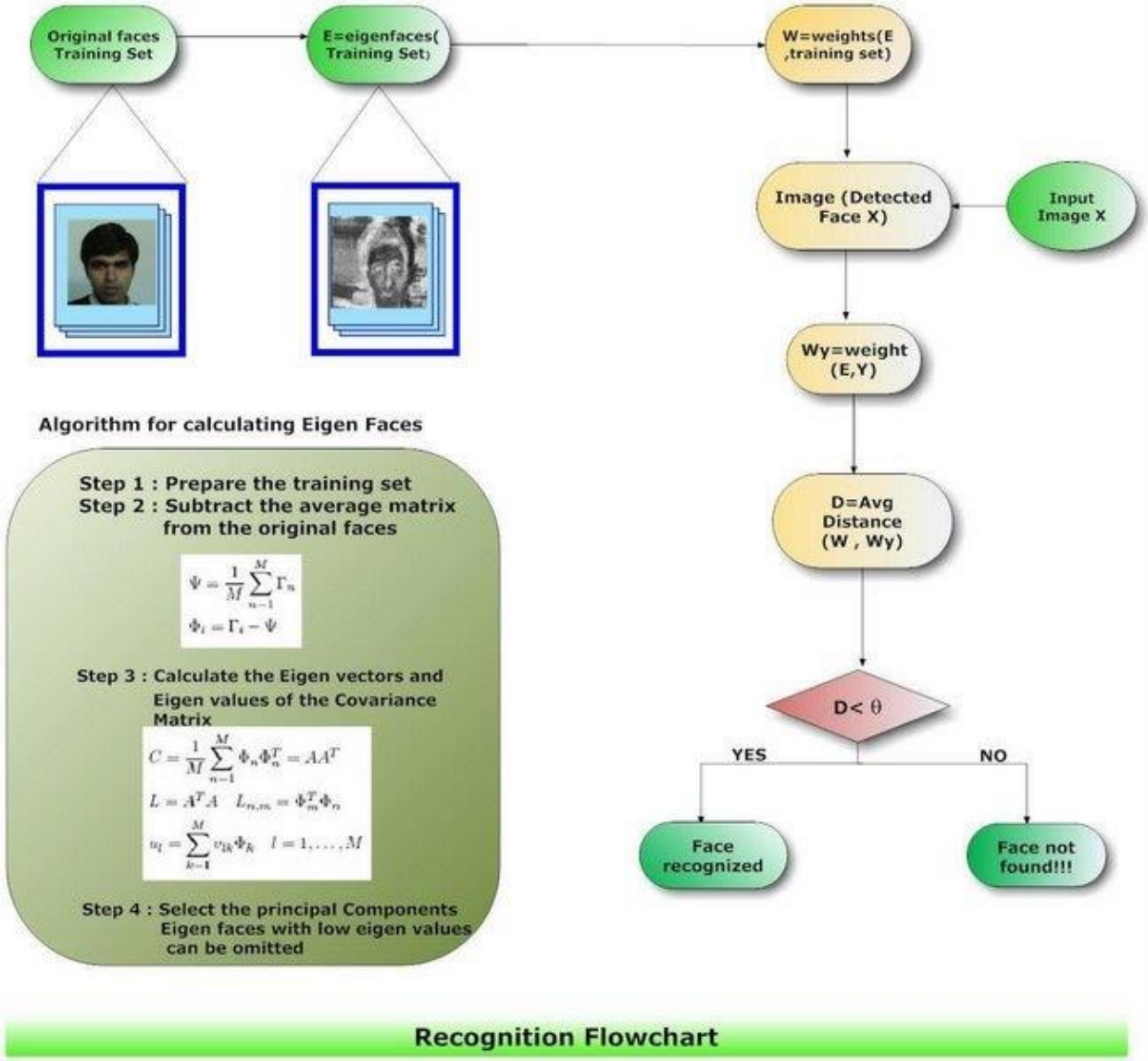


Fig.16. Search algorithm

2. Compare analyses of face recognition algorithms

We illustrate the influence of number of training samples per subject to recognition efficiency for selected methods. We use PCA (principal component analysis), MLP (multilayer perceptron), RBF (radial basis function) network, kernel methods and LBP (local binary patterns). We compare results using single and multiple training samples per person for images taken from FERET database. For our experiments, we selected large image set from FERET database.

Highlighting other relevant important facts related to single sample recognition.

We analyze some relevant facts that can influence further development in this area. We also outline possible directions for further research.

Generally, we can divide the face recognition methods into three groups: holistic methods, local methods and hybrid methods.

Holistic methods like PCA (Eigen faces), LDA (fisher faces) or SVM need principally more image samples per person in the training phase. To solve the one sample problem there are basically two ways how to deal with it [12]:

- To extend the classical methods to be trained from single sample more efficiently. 2D-PCA, (PC)2A, E(PC)2A, SPCA, APCA, FLDA, Gabor+PCA+WSCM.
- To enlarge the training set by new representations or generating new views.

Local methods can be divided into 2 groups:

Local feature based, which mostly work with some type of graph spread over the face regions with corners in important face features – face recognition is formulated as a problem of graph matching. These methods deal with the one sample problem better than the typical holistic methods. EBGM (Elastic Bunch Graph Matching) or DCP (directional corner points) are examples of this type of methods.

Local appearance-based methods extract information from defined local regions. The features are extracted by known methods for texture classification

(Gabor wavelets, LBP, etc.) and the feature space is reduced by known methods like PCA or LDA.

We use various methods in order to deeply explore the behavior of face recognition methods for single sample problem and to compare the methods using multiple face samples - both real-world samples and virtually generated samples. Used methods are briefly introduced in this subchapter.

Principal component analysis(PCA). One of the most successful techniques used in face recognition is principal component analysis (PCA). The method based on PCA is named Eigen face and was pioneered by Turk and Pentland. In this method, each input image must be transformed into one dimensional image vector and set of these vectors forms input matrix. So the main idea behind PCA is that each *n*-dimensional face image can be represented as a linearly weighted sum of a set of orthonormal basis vectors.

This standard statistical method can be used for feature extraction. Principal component analysis reduces the dimension of input data by a linear projection that maximizes the scatter of all projected sample.

Two-dimensional PCA(2D PCA). PCA is well-known feature extraction method mostly used as a baseline method for comparison purpose. Several extensions of PCA have been proposed. A major problem of using PCA lies in computation of covariance matrix what is computationally expensive. This computation can be significantly reduced by computing PCA features for columns (or rows) without previous matrix-to-vector conversion. This approach is also called two dimensional PCA. Main idea behind 2D PCA is the projection of image columns (rows) onto covariance matrix computed as the average of covariance matrices of each column for all training images. Let A be an m by n image matrix and average image A defined as $A=1/M\sum kAk$, where M is number of all k training images. Then covariance matrix can be calculated by [13],

$$G=1M\sum k=1M\sum i=1m(A\{i\}k-A-(i))T(Ak(i)A-(i))$$

Equation reveals that the image covariance matrix can be obtained from the outer product of column (row) vectors of images, assuming the training images have zero mean.

For that reason, we claim that original 2D PCA works in the column direction of images. Result of feature extraction is then a matrix instead of a vector. Feature matrix has the same number of columns (rows) as width (height) of face image.

The extraction of image features is computationally more efficient using 2D PCA than PCA since the size of the image covariance matrix is quite small compared to the size of a covariance matrix in PCA (by using Turk & Pentlands optimization it depends on number of training images). 2D PCA is not only more efficient than PCA but it is possible to reach even higher recognition accuracy.

Kernel PCA(KPCA). PCA is a linear algorithm that is not able to work with nonlinear data. Kernel PCA is a method computing a nonlinear form of PCA. Instead of directly doing nonlinear PCA, it implicitly computes linear PCA in high-dimensional feature space that is in non-linear relation to input space.

Support vector machine (SVM). Support vector machines (SVM) are based on the concept of decision planes that define optimal boundaries. Its fundamental idea is very simple: the boundary is located to achieve the largest possible distance for the vectors of different sets. Example of this is shown in the fig.17 . This figure illustrates linearly separable problem. In the case of linearly no separable problem, kernel methods are used. The concept of kernel method is a transformation of the vector space into a higher dimensional space.

Optimal boundary of support vector machine. The kernel function is defined as follows:

$$K(x, x`) = \Phi(x)T\Phi(x`)$$

Kernel function is equivalent to the distance between x and x` measured in the higher dimensional space transformed by a nonlinear mapping Φ [14].

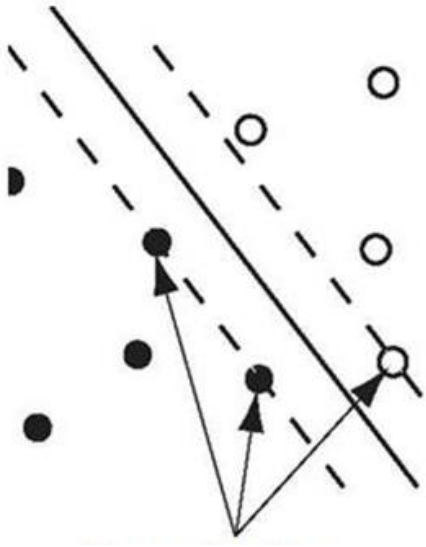


Fig.17. Support vectors

Multilayer perceptron (MLP). Multilayer perceptron (MLP) is a layered feed forward network consisting of input, hidden and output layers.

Multilayer perceptron operates with functional and error signals. The functional signal propagates forward starting at the network input and ending at the network output as an output signal. The error signal originates at output neurons during the learning and propagates backward. MLP is trained by back propagation algorithm. MLP represents nested sigmoidal scheme, its form for single output neuron is,

$$F(x,w)=\rho\sum jw_jk\rho(...\rho(\sum iwl_ix_i)...)$$

where$\rho(\cdot)$ is a sigmoidal activation function, w_oj is the synaptic weight from neuron j in the last hidden layer to the single output neuron o, and so on for the other synaptic weights, x_i is the i-th element of the input vector x. The weight vector w denotes the entire set of synaptic weights ordered by layer, then neurons in a layer, and then number in a neuron.

Radial basis function network(RBF). Radial basis function network (RBF) is a feed forward network consisting of input, one hidden and output layer. Input layer distributes input vectors into the network, hidden layer represents RBFs h_i. Linear output neurons compute linear combinations of their inputs. RBF network topology is shown in fig.18.

RBF network is trained in three steps:

1. Determination of centers of the hidden neurons;

2. Computation of additional parameters of RBFs;

3. Computation of output layer weights.

RBF network from Fig.2.12 can be described as follows:

$$f(x)=w_o+\sum(i=1)mw_ih_i(x)$$

where x is the input of RB activation function h_i and w_i are weights. Output of network is a linear combination of RBFs [15].

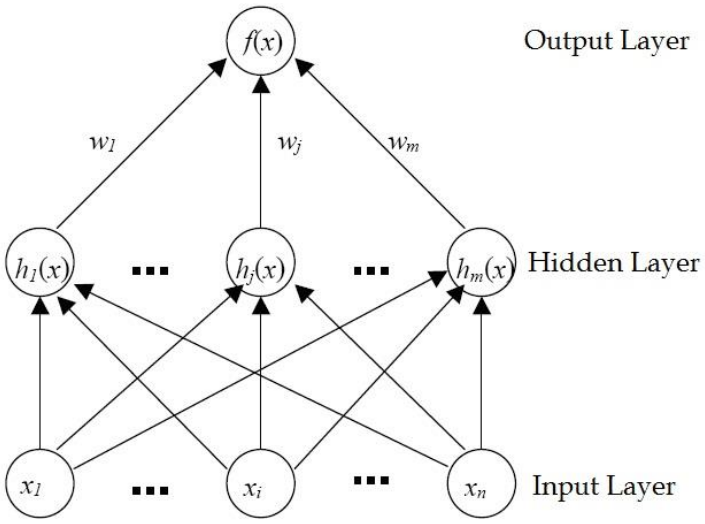


Fig.18. RBF network topology

Local binary patterns (LBP). LBP was first described in. It is a computationally efficient descriptor to capture the micro-structural properties and was proposed for texture classification. Later the LBP operator has been extended to use circle neighborhoods of different sizes - the pixel values are bilinearly interpolated (fig.19).

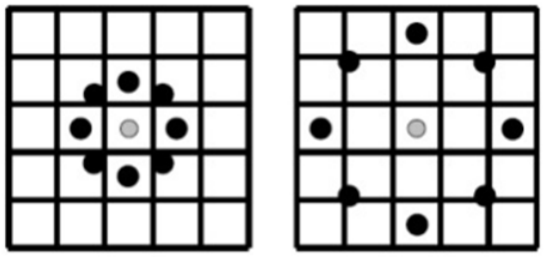


Fig.19. LBP pixel values

Another extension uses just uniform patterns. A local binary pattern is called uniform if it contains at most two bitwise transitions from 0 to 1 or vice versa when the binary string is considered circular. For example, 00000000, 00011110 and 10000011 are uniform patterns. Such patterns represent important features on the image like corners or edges. Uniform patterns account for most of the pattern in images.

A system using LBP for face recognition is proposed in. Image is divided into non-overlapping regions. In each region a histogram of uniform LBP patterns is computed, the histograms are concatenated into one histogram (see fig.20 for illustration), which represents features extracted from the image in 3 levels (pixel, region and whole image).

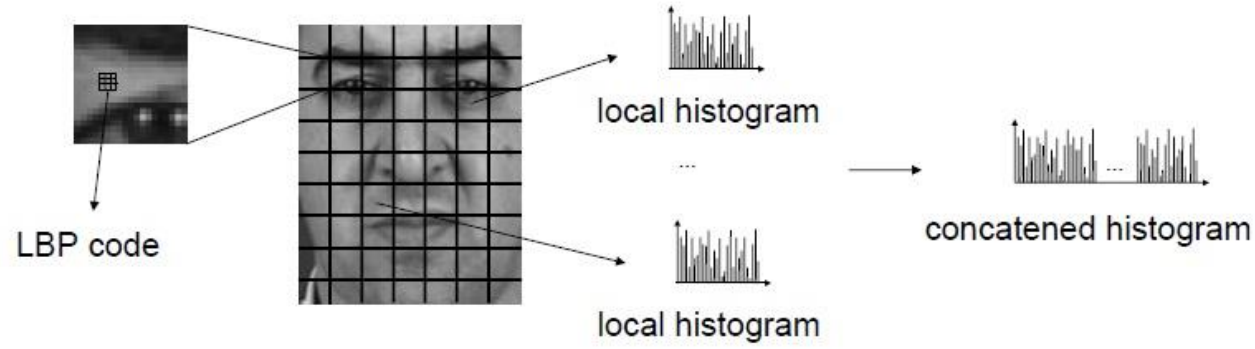


Fig.20. Histograms of LBP

Description of face using concatenated LBP histogram.

The $\chi 2$ metric is used as the distance metric for comparing the histograms:

$$x_2(S,M)=\sum(S(i)-M(i))^2 S(i)+M(i)$$

where S and M are the histograms to be compared and i is the i-th bin of histogram.

Simulation results for 1 - 4 original training images per subject. In our experiments with original training images we compared the efficiency of several algorithms in scenario with 1 (single sample problem), 2, 3 and 4 images/subject. We carefully selected algorithms generally considered to play the major role in

today face recognition research. Also standard PCA was included for comparison purposes. Face recognition methods [16].

Results for different training sets (dependence of face recognition accuracy in % with regard to number of samples per subject in the training set)

In each test with different number of images in training set we made 4 runs with different selection of the images into the training set: original one with choosing the first images alphabetically by name and 3 additional training/testing collections with randomly shuffled images. The final test results are the average from these 4 values[17].

Table.2.

Face recognition methods and their results in real time

	1_train	2_train	3_train	4_train
PCA	72,26	82,43	86,6	89.43
2D-PCA	73,41	81,9	86,36	89,02
KPCA	73,97	82,28	87,83	91,62
PCA+SVM	79.97	91.52	95.76	97,18
SVM	66.32	68.76	91.73	95.05
MLP	61.69	72.86	83.40	85.86
RBF	66.41	85.26	93.16	96.79
LBP-5x5	83.02	89.37	92.06	94.21
LBP 7x7	85.29	91.47	94.45	95.99
LBP 7x7w	**85.81**	**92.91**	**95.05**	**96.59**

Herein we consider different situation: only one original sample is available and we try to enhance recognition accuracy by generating new samples to the training sets in artificial manner. Thus, we try to enlarge the training sets by generating new (virtual, artificial) samples. We propose to generate new samples by modifying single available original image in different ways – this is why we will use the term *image modification* (or *modified image*).

In our tests we use different modifications of available single per person images: adding noise, applying wavelet transform and performing geometric transformation.

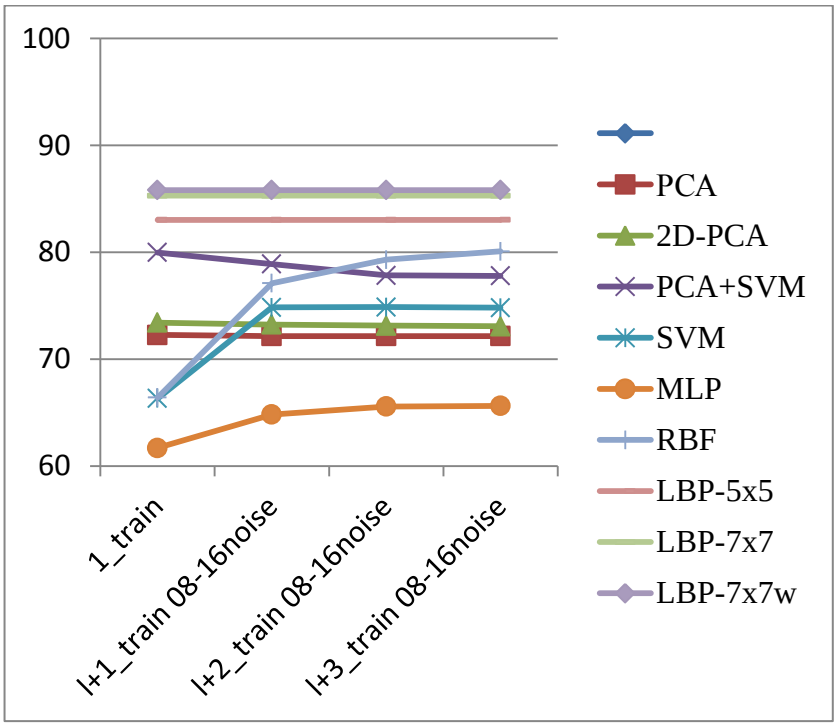


Fig.21. Face recognition methods and their results

Our results are summarized in table.2 and in fig.21 All figures and tables in this chapter contain values whose meaning is recognition accuracy in % achieved on test sets. The notation *n_train* means *n images (samples) per subject in training sets*.

Graphic comparison of the results for different training sets (dependence of face recognition accuracy in % with regard to number of samples per subject in the training set)

Neural networks and SVM. For single sample per person training sets, methods based on neural networks (RBF network and MLP) and also SVM achieved less favorable results (below 70%). The extension of the training sets by second sample per person slightly increased face recognition test results for MLP and SVM methods. For RBF network, the second sample improved the result to the value above 85%. Impact of adding third sample per person into training sets caused a significant improvement of test results (for RBF and SVM above 90%

accuracy was achieved). Adding more than four samples per person into training sets has only a minimal effect on increasing the face recognition results and has a negative impact on the computational and time complexity. The larger training sets the better recognition results were achieved.

PCA-based methods. PCA with Euclidean distance metric as a reference method shows that more images per subject in training set lead to more accurate recognition results, improving from 68% with 1 img./subj. to 89% with 4 img./subj. Although there was reported that 2D PCA can reach higher accuracy in term of precision, PCA slightly overcome 2D PCA in our experiments. However, 2D PCA still has big advantage in comparison to PCA which lies in faster training time due to using smaller covariance matrix. As it is shown in, 2D PCA is equal to block-based PCA and it means that it uses only several parts of covariance matrix used in PCA. In other words we lose information from rest of covariance matrix that can lead to worse recognition rates. KPCA achieved slightly better results compared to 2D PCA (KPCA is included for comparison purposes here and it will not be used further within this chapter).

PCA+SVM. PCA+SVM method is a two-stage setup including both feature extraction and classification. Features are first efficiently extracted by PCA with optimal truncating the vectors from the transform matrix. The parameters for the selection of the transformation vectors are based on our previous research. The classification stage is performed by SVM. SVM model is created with the best parameters found using cross-validation on the training set. PCA+SVM has very good recognition rate even with 1 img./subj. and with 3 and 4 img./subj. it outperforms all other methods in our tests reaching 97% recognition rate with 4 img./subj.

LBP. In our experiments, we used local binary patterns method for face recognition in 3 different modifications. The image is divided into 5x5 or 7x7 blocks from which the concatenated histogram is computed. The “LBP 7x7w” modification adds also weighting of the histogram with different weights according to corresponding image regions.Results for all LBP methods are the best in our

tests and were outperformed only slightly with PCA+SVM method with 3 and 4 img./subj. The main characteristic of LBP is that the recognition results are very good even for 1 img./subj.. From the graph in table.3 we see that the recognition rates for the three LBP methods go parallel with each other. The LBP is starting with 83% reaching 94% accuracy with 4 img./subj. LBP 7x7 is approximately 1.5% better than the 5x5modification and the LBP 7x7w more than 2% better reaching almost 97% accuracy with 4 img./subj.

Table.3.

Compare methods of LBP in real time

Method	l_train
LBP-5x5	83,02
LBP 7x7	85,29
LBP 7x7w	85,81

Method	l_train
LBP 7x7 130xl50pix	84,82
LBP 7x7w 130xl50pix	86,66
LBP 10x10 130xl50pix	87,48
LBP l0xl0w 130xl50pix	88,34

Within this chapter, we work with images of size 65x75 pixels after preprocessing. In table.3, results for image size 130x150pix (FERET default standard) are shown for illustration. Generally, larger size of images can yield slightly better recognition rates.

3. Face recognition algorithms by adding of modification methods

In this section, we explore face detection and recognition algorithm by adding of modification methods. Experiments include normalization of a face database to increase recognition performance through various pre-processing methods. We propose a novel algorithm for automatic face detection and post-processing method to further improve the face recognition performance for smartphone environment and PC.

Generally, face recognition system requires face image data that used for learning and features are normalized and registered for face recognition process. These face image data are collected for some time intervals in various conditions.

We have collected 3 set of face database for this experiment called "Sejong face database". A set of images of each person are used for learning and other sets are used for face recognition system. A session contains three images for each person (fig.22).

Fig.22. Sejong Face Image Database (SJDB)

The images of each person are collected one or two images a day in the same physical setup such as pose and lightning called 'session". The session of N different people are collected without distinction of sex. In this way, we collected 100 different sessions about 70 male and 30 female. Each session contains three frontal images which were collected with similar illumination condition (170-220 lux).

For any face recognition system noise-data is a primary factor of degradation. Therefore, the process for remove noise is essential to improve recognition accuracy. This process is called pre-processing and it begins with geometric transformation (rotation, translation, and rescaling). The correction of the geometric transformation uses the intermediate point of the eyes and mouth. We remove the unnecessary data of the outer face region using an elliptical mask. The pre-processed image is properly modified with several feature point (eyes, nose, mouth) to have equal dimension regardless of personal face size.

Table.4.

Pre-Process Time For One Face Data (Not Integerized)

Image size	1200 by 1600	120 by 160	30 by 40
Processing time (ms) on desktop computer	1639.75	449.02	414.24
Processing time (ms) on smartphone	4475.24	1064.88	849.58

The face images for this experiment has horizontal versus vertical ratio of 3:4. As a result, face images are modified to 1200 by 1600, 120 by 160, and 30 by 40 size.

Fig.23. Preprocessing Criteria, Processing and Modified Census Transform Results

The pre-processing time is shown in Table 1 which was performed based on Intel Core™2 i5 computer with 2.6GHz and 4GByte configuration and Samsung Galaxy SII smartphone environment. In this paper the pre-processing was performed with fully automatic setup with modified census transform (MCT) and Adaboost based algorithm. The image with size of 120 by 160 does the best performance reference of Table.4. In this experiment, we have applied series of algorithms including histogram equalization and filtering. As for the image, noise data was removed and the feature data is emphasized. In addition, intensity information of the image was redistributed through histogram equalization. The structure of our system is shown in fig.23.

We have used principal component analysis (PCA) and linear discriminant analysis (LDA) for feature extraction. As for the feature data, the change of recognition rate is presented with different cutoff ratio (percentage of feature vectors used) as shown in Table 2. We set the cutoff ratio at 80% and designed our face recognition system which we considered as most appropriate performance. We have measured the distance between feature vectors based on L2 norm. The choice of the automatic face detector was based on Adaboost algorithm which was trained using FERET, XM2VTS, CMU PIE and Sejong face database. The size of learning data is using face (22 by 22), eyes (10 by 6) and mouth (20 by 9) pixels. The 3,513 positive data and 10,777 negative pieces of image data are collected and

used. As a result, we succeeded in the face detection from near frontal face images ranging from 0-15 degrees.

Table.5.

A Face Recognition Result by the Change of Image Size and Cutoff Ratio in Smartphone Environment

Image size / Cutoff	1200 by 1600	120 by 160	30 by 40
100 %	96%	98%	97%
80%	97%	98%	96%
70%	94%	95 %	91 %

For learning the face detection data using Adaboost algorithm, negative (wrong) data is equally important as positive (correct) data. After appropriate training, the face detector can locate feature points (the center of each eyes and the center of the mouth) from a various rotated angle and front face image. After pre-processing of face image, face recognition takes about 7 fps (frame per second) which is reasonable for real-time processing. The total processing time requires 1.1 fps (frame per second) including the face detection. Face detection and face recognition process show rate of 87% detection and 95% recognition percentage each.

We have implemented four major face recognition functions to performed general smartphone environment. Biometric function is composed of face detection, face recognition and face evaluation module designed based on Google Android platform. Generally, face recognition algorithms are expressed by normalized numerical value. Such real number based algorithms are used for graphical algorithms such as image processing, 3D rendering, etc. There exists noticeable performance drops between floating and integer number based calculation. Moreover, this difference appears greatly on smartphone environment has weak floating point unit (FPU) compared to desktop computer.

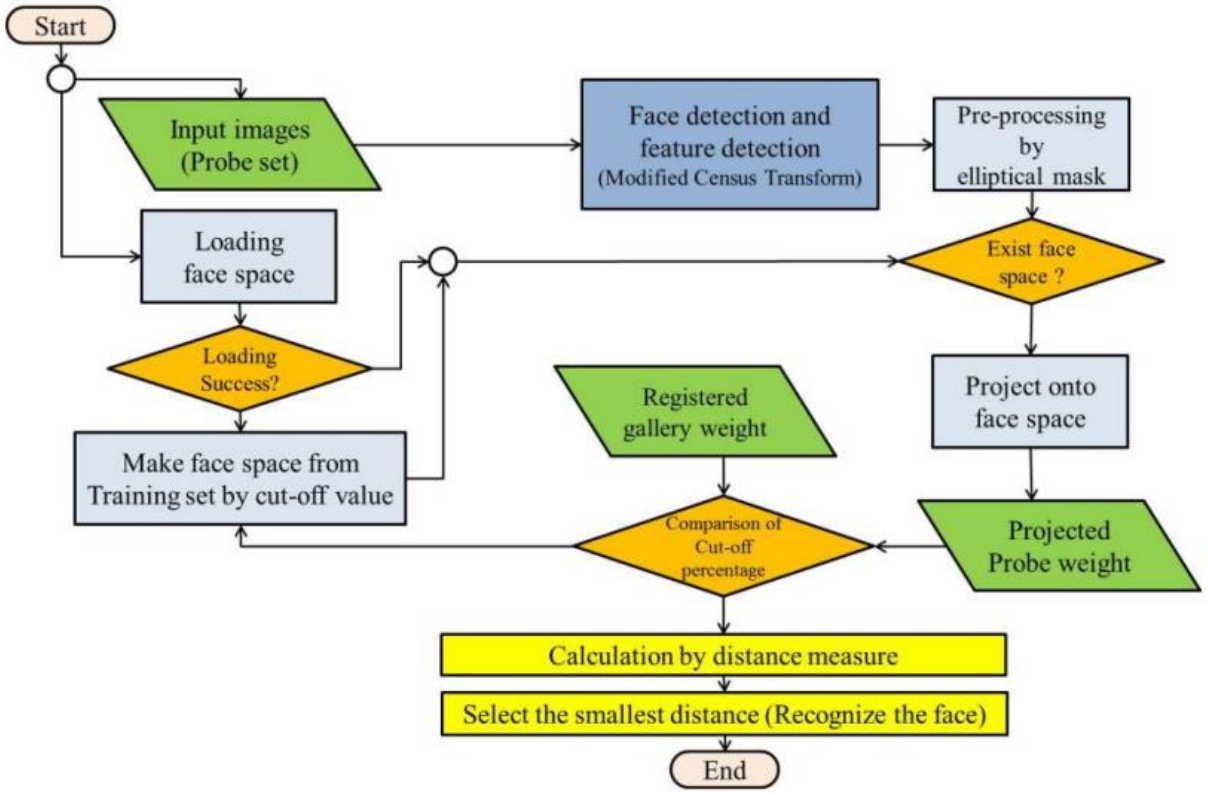


Fig.24. Structure of Proposed System

Since face recognition algorithm use floating point based calculation algorithm, performance degradation can be significant. In order to reduce these differences, integer number type was used with fixed point method which is a numerical analysis method to store fraction's part of the decimal and the integer number separately. Fixed-point method was used for face recognition algorithm to produce faster than existing algorithm with floating algorithm. We designed fixed decimal point algorithm is designed to perform faster than floating point algorithm. There is proved measurement of a runtime through simple C program. Integral fragmentary performance can be verified through profiling. In practice, performance can be evaluated through integral conversion which uses fixed point algorithm. Face image data was used for performance evaluation with 120 by 160 pixels. Evaluation unit compare each module during the process. Change of total arithmetic velocity depends on integral transform considerably. This fact is that because additional arithmetic is contained through integral transform, and performance improvement is guaranteed if it can be applied on smartphone environment as shown in Table.6. and Table.7. (smaller numbers faster).

Table.6.

7,000,000 Times of Profiling Repetition Results on (a) Intel ® Core™2 i5 2.6GHz and (b) Intel ® Pentium – 233

	Integer number		Floating number		Double number	
Test environment	(a)	(b)	(a)	(b)	(a)	(b)
Addition	62.1	274.722	89.1	270.63	255.1	280.31
Subtraction	78.2	267.69	87.6	269.54	277.6	282.80
Multiplication	113.3	274.09	447.2	3702.76	997.1	3702.03
Division	311.2	1379.03	490.5	4053.54	817.7	4022.88

The face recognition system is essential in multimodal biometrics system since it provides the only non-contact biometric features and user friendly information which can be processed by both human and computer. This is very important feature in case of communication problem caused by network failure which prohibits database access for smartphone related applications. Recently, face detection and recognition technology can be built into an embedded system for mobile devices.

Table.7.

Face Recognition Time in Desktop Computer and Smartphone Environment. 1200 by 1600 was not Experiment, because Face Space File Size is too Large To Load into Memory

Image size	1200 x 1600	120 x 160	30 x40
Face recognition time (msec)	649.06 (-)	17.79 (38.03 ms in smartphone)	0.54 (0.87 ms in smartphone)
Face space file size	1.551 GByte	15.001 MByte	0.938 MByte

However, this system should respond to numerous conditions including various pose and lighting which is challenging for face recognition system. Generally, a face recognition system contains two major functions which is face

detection and face recognition. We produce several face detection and recognition algorithms in smartphone environment for various biometric applications.

Our modifications of face images were done by three steps:

1. Forward transform of image by DWT;
2. Setting horizontal, diagonal and vertical details in frequency spectrum;
3. Image reconstruction by inverse DWT.

Experiment with wavelet transform demonstrated improvement of one sample per person face recognition using neural network methods - RBF network and MLP. These methods confirmed increase of recognition rate with extending the training sets with images modified by wavelet transform. Improvement above 10% was achieved for RBF network with adding three samples per person (1+3_train) into training sets. On the other hand, SVM method achieved very low face recognition accuracy.

After the all facial features are properly localized and represented by contour and middle points, the next step is to generate target expressions. Because the change of an expression involves moving detected feature points, there is a need to change texture information as well. Real expressions and direction of movements during the expression depends on strength of muscles contractions. We divided each face image into triangles according to direction of these contractions. Face features localization process and dividing into triangles (also called triangulation) is fully automated (unlike usual manual method described in using active shape models. Using active shape models produces very precise positions of facial features and facial boundaries. Result of triangulation is facial graph containing only triangles among detected points determining facial features.

Making use of rule based system, similar to system described, we generated different expressions from each training sample by moving location of points in the facial graph. Texture in each triangle containing moved points is then interpolated from original according new coordinates.

Summary of chapter II

In this chapter we considered relevant issues related to one sample per person problem in the area of face recognition. We focused mainly on recognition efficiency of several methods working with single and multiple samples per subject. We researched techniques for enlargement of the training set by new (artificial, virtual or nearly synthetic) samples, in order to improve recognition accuracy. Such samples can be generated in many ways – we concentrated on modifications of the original samples by noise. We examined the impact of these extensions on various methods.

I presented an empirical evaluation of the popular appearance based feature extraction algorithms within face recognition system based on image face deblurring. The tested algorithms used in this research (KFA, KPCA, LDA, PCA, LBP, GKPCA, GLDA, GPCA, PCKFA, PCKPCA, PCLDA and PCPCA) were evaluated using the databases, created using centralized sparse representation (CSR) and adaptive sparse domain selection with adaptive regularization (ASDS-AR).

My experiments suggest that when we associate the LBP gives better results than Phase Congruency. In addition, Centralized Sparse Representation (CSR) has proven its effectiveness in image face deblurring compared with Adaptive Sparse Domain Selection and Adaptive Regularization (ASDS-AR). However, among the methods tested, LBP 10x10w was judged the best achieving the lowest error rate compared to other methods.

So, I proposed face detection and recognition methods are integral transformed which can be calculated faster, and optimized for smartphone environment. Our experimental result shows that we can minimize the run-time by decreasing computational complexity while maintaining reasonable accuracy which is applicable for smartphone environment.

Chapter III. Development face recognition systems based on solving problems of current methods

1. Create face recognition systems on based PCI+LBP7X7 algorithm

Noise in face images can seriously affect the performance of face recognition systems. Each image capturing generates digital or analog noise of diverse intensity. The noise is also generated while transmitting and copying analog images. Noise generation is a natural property for image scanning systems. Herein we use noise for generating modified samples of original image. In our modifications, we use Gaussian noise [23].

Gaussian noise was generated using Gaussian distribution function:

$$p(x) = 12\pi\sigma^2 - \sqrt{e - (x-\mu)^2 2\sigma^2}$$

where μ is the mean value of the required distribution and σ^2 is a variance.

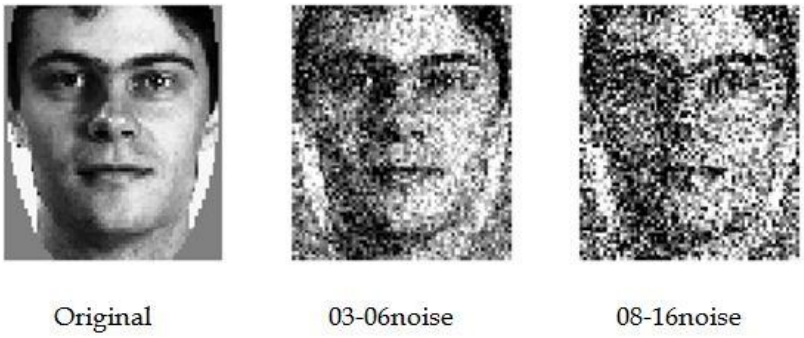


Fig.25. Gaussian noise

Gaussian noise was applied on each image with zero mean and in two random intervals of variance. Examples of images degraded by Gaussian noise can be seen in fig.25. The labels *03-06noise* and *08-16noise* mean that the variance of Gaussian noise is random between values 0.3 - 0.6 and 0.8 - 16, respectively. The same notation is used also in presented graphs and tables (Tab.3.1 and 3.2, fig.26 and 27). Noise parameters settings for our simulations were determined empirically. Training sets were created by noise modification of samples added to the original one (*1+1noise*, *1+2noise* and *1+3noise*).

Presented results for noise modifications shown in tab.8, 9 and fig.26, 27 are summarized as follows:

The improvement for RBF, MLP and SVM is clearly visible. In both noise modifications (*03-06noise*and *08-16noise*), the most significant increase in accuracy of test results is achieved by RBF network (about 80% for 1+3 training sets). Similarly to the tests in subchapter, adding more samples into training sets has a constant effect on the recognition results.

Results for generating new face samples (modifications of original face samples by Gaussian noise – lower variance).

Table.8.

Results of 0.3 - 0.6 noise for trains in real time

Methods	1_train	l+1_train 03-06noise	l+2_train 03-06noise	l+3_train 03-06noise
PCA	72.26	72.16	72.16	72.16
2D-PCA	73.41	73.18	73.24	73.24
PCA+SVM	79.97	78.73	79.03	78.43
SVM	66.32	67.44	74.76	74.84
MLP	61.69	65.99	68.39	71.48
RBF	66.41	79.19	79.73	79.87
LBP-5x5	83.02	83.02	83.02	83.02
LBP-7x7	85.29	85.29	85.29	85.29
LBP-7x7w	**85.81**	**85.81**	**85.81**	**85.81**

Results for generating new face samples (modifications of original face samples by Gaussian noise – higher variance)

The results of PCA and 2D PCA methods are only slightly affected when adding additional images with different amount of noise to the training set. The results with the noise images added are approximately 1% worse than the original recognition rate with 1 img./subj. Reason for this effect can be probably found in the fact that the transformation matrix computed from the training sample with added noise represents the variances in the space worse than after computing it from original images only.

Table.9.

Results of 0.8 - 16noise for trains in real time

Methods	1_train	1+1_train 08-16noise	l+2_train 08-16noise	l+3_train 08-16noise
PCA	72,26	72,16	72,16	72,16
2D-PCA	73,41	73,24	73,13	73,07
PCA+SVM	79,97	78,90	77,83	77,79
SVM	66,32	74,84	74,87	74,81
MLP	61,69	64,83	65,57	65,63
RBF	66,41	77,1	79,3	80,07
LBP-5x5	83,02	83,02	83,02	83,02
LBP-7x7	85,29	85,29	85,29	85,29
LBP-7x7w	**85,81**	**85,81**	**85,81**	**85,81**

Adding samples to training set is also very uneconomical from the point of view of PCA methods since the time needed to compute the transform matrix grows.

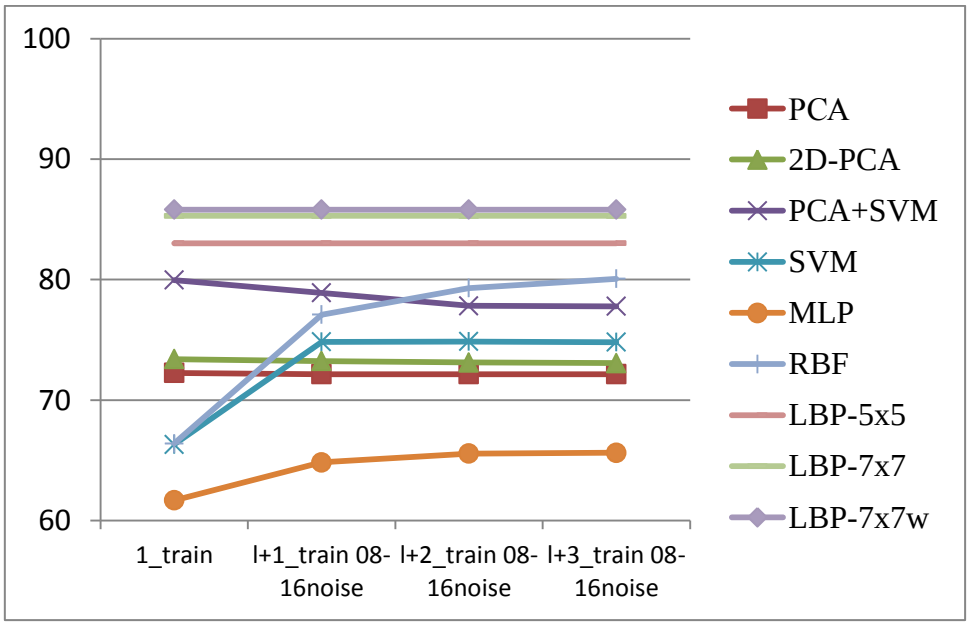


Fig.26. Counting the transform matrix of 03-06noise

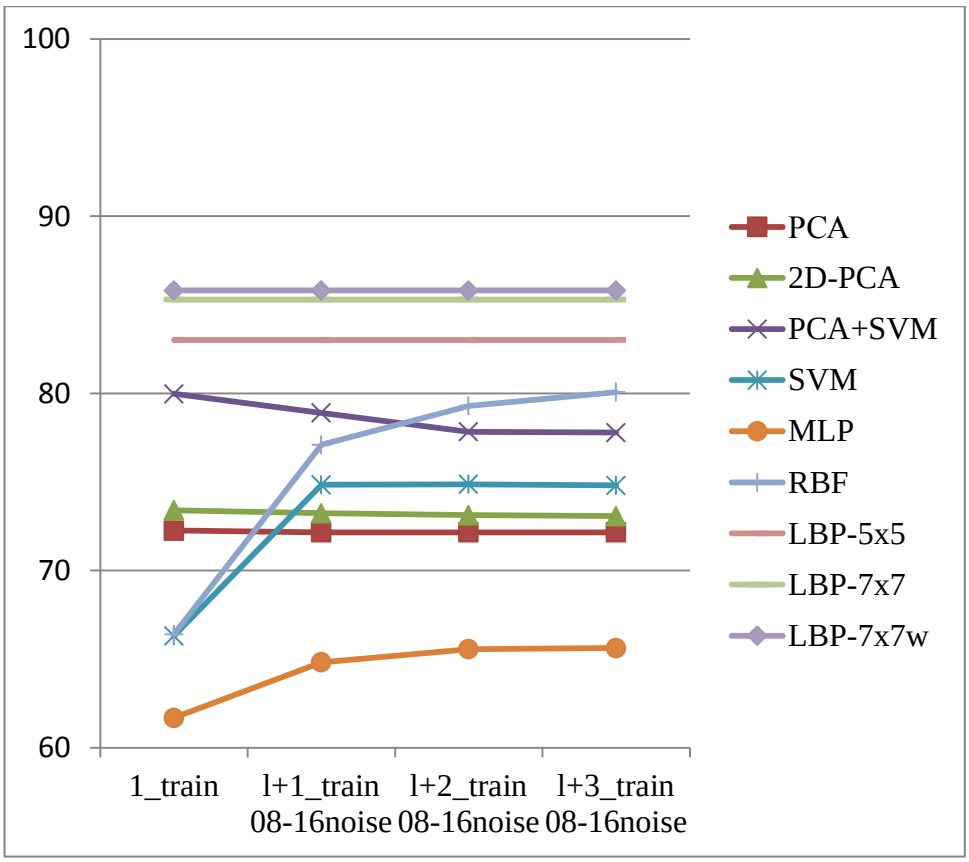


Fig.27. Counting the transform matrix of 08-16noise

PCA+SVM. The effect observed with PCA can be observed also with PCA+SVM method. Adding the noise images to the training set leads to worse results than with the original training set for about 1% for every scenario. The SVM classification model is influenced by the features extracted from noisy samples, but this accuracy drop is not dramatic.

LBP. The results of LBP methods are not influenced with the noisy samples at all. This has two reasons:

- By LBP method no model or transformation is calculated from the training images, so there cannot be such global effect to the recognition results as with PCA or SVM.

- The histograms of LBP patterns in noisy images change rapidly so the distance between the noisy image and the original image of the same person is higher than the distance between two original images of different persons. The consequence is that the minimal distances between the testing and training images do not change and the results are the same as without the noisy images in training

set. See Table 3.2 for illustration of the distances between original and noisy images.

Methods such as PCA+SVM or LBP achieved recognition results above 80% for single sample per person in the training set. Experimental results for PCA and 2D PCA show only negligible influence of adding modified samples. We can conclude that the use of modified samples for PCA and 2D PCA has no added value, especially when samples are modified by Gaussian noise only.

PCA+SVM (two-stage method with PCA for feature extraction and SVM for classification) achieved very good results even for 1 img./subj. Adding any modified images to the training set did not improve the recognition rates, but the results were still one of the best from the compared methods.

Our experiments show that LBP is one of the most efficient state-of-the-art methods in face recognition. Adding noise modified images to the training set does not have any effect on the recognition rates of LBP – unlike other methods that use the training sample to compute models or transformation matrices. This is caused by the nature of the method, where the histogram of LBP patterns of the noisy image differs too much from the original images. This can be also a disadvantage, when the images in the test set are corrupted with noise. On the other hand, adding images with transformed face expression helps and the system is more resistant to expression change in the images [25].

2. Create the algorithm of program for face recognition

This system was designed with comparing and using together algorithms LBP 7x7 and PCI. Keywords of face: eyes, nose, mouth and structure of face were saved in specialized files .xml. Program takes key-parameters and computes.

Registration chapter (block schema.1):

1. Entering x and y points of key-points of face: facex, facey;

2. Entering starting value for dividing x and y:

nosex = 0;

leftxynisbat = 0;

rightxynisbat = 0;

nosexynisbat = 0;

mouthxynisbat = 0;

3. Reclamation values in coincidence files .xml:

face, eye, mouth, nose.

Writing every points and their color x and y with helping methods Block-filters through integral images in files .xml:

$$(X_1\ Y_1\ X_2\ Y_2\ \text{color\_code})$$

For example:

3 9 14 2 2.

Here:

$X_1 = 3$;

$Y_1 = 9$;

$X_2 = 14$;

$Y_2 = 2$;

Color_code = 2.

4. Capturing images are saved in files file-format .bmp and .txt base on user's name.

5. Capturing window: 560, 420.

6. Selected face with rectangle size 100x100 and calculated distance between rectangle end face X, Y.

xns = f.rect.X – yuzx

yns = f.rect.Y – yuzy

7. Also, calculated X and Y for left and right eye, nose, mouth.

8. For differing of left and right eyes:

xns< nosex bo'lsa, leftxynisbat = (xns / yns)

xns> nosex bo'lsa, rightxynisbat = (xns / yns)

9. Name, group and all results are entered to the database yuz.mdp.

name>3

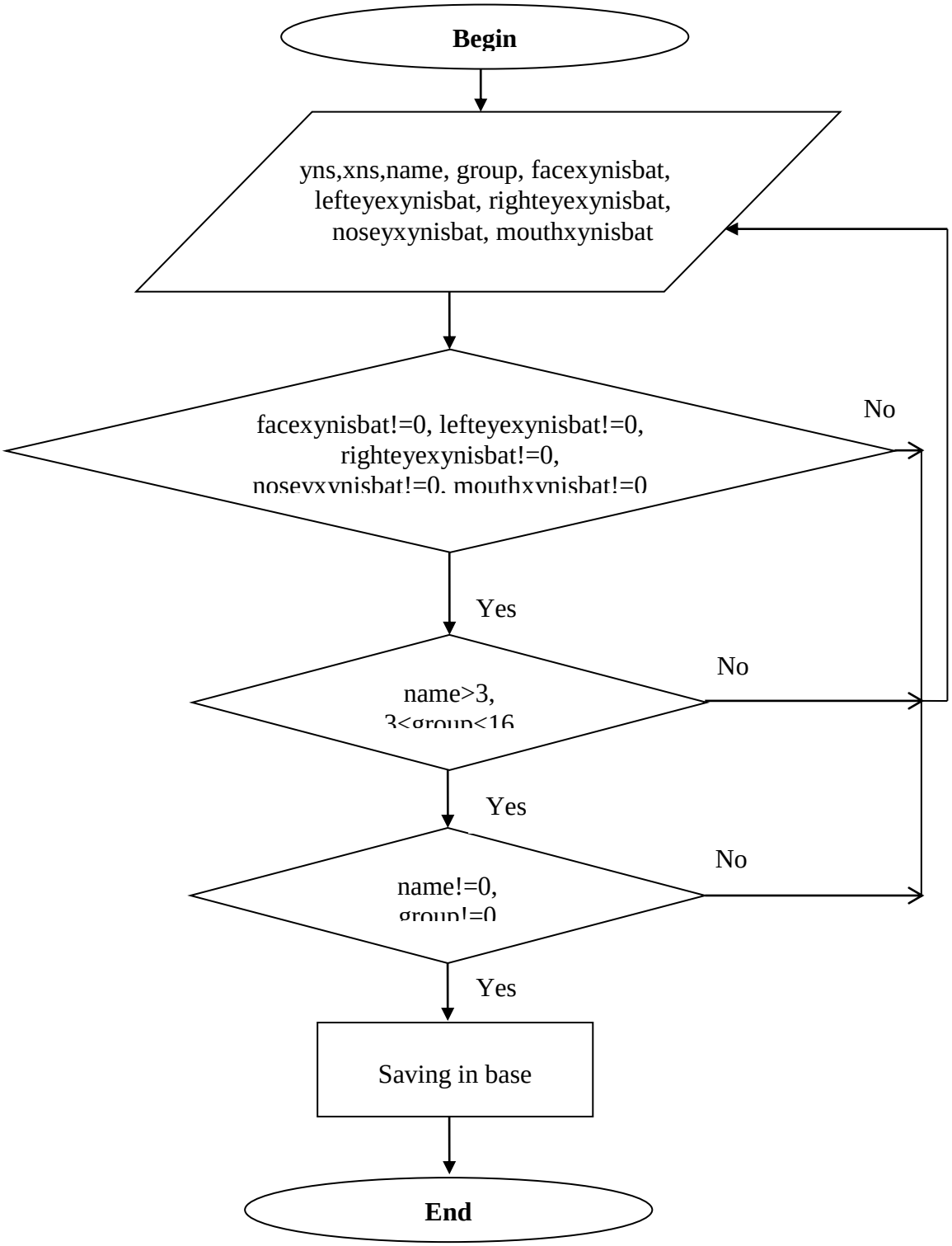
leftxynisbat!=0;

rightxynisbat!=0;

nosexynisbat!=0;

mouthxynisbat!=0

10. All images in folder TrainedFaces, names in file TrainedFaces.txt and other information in database are saved



Block schema.1. Block-schema of registration chapter

Entering chapter (block schema.2**):**

1. Parameters face, eye, mouth, nose are took.

2. Define a user base on entering image. System are recognized and saved user's information to the base.

3. You can enter degree of security:

Low: This degree of security includes: size of face, left eye, group and name;

Medium: This degree of security includes: size of face, left and right eye, name, group;

High: This degree of security includes: size of face, left and right eye, nose, name, group;

Very high: This degree of security includes: size of face, left and right eye, nose, mouth name, group;

Chapter admin:

1. It takes password from data-base, table admin for entering to the window admin.

2. You can take all information about user base on the entering user's name, also: their images and registration time. May be delete the user by button delete

Similarity chapter:

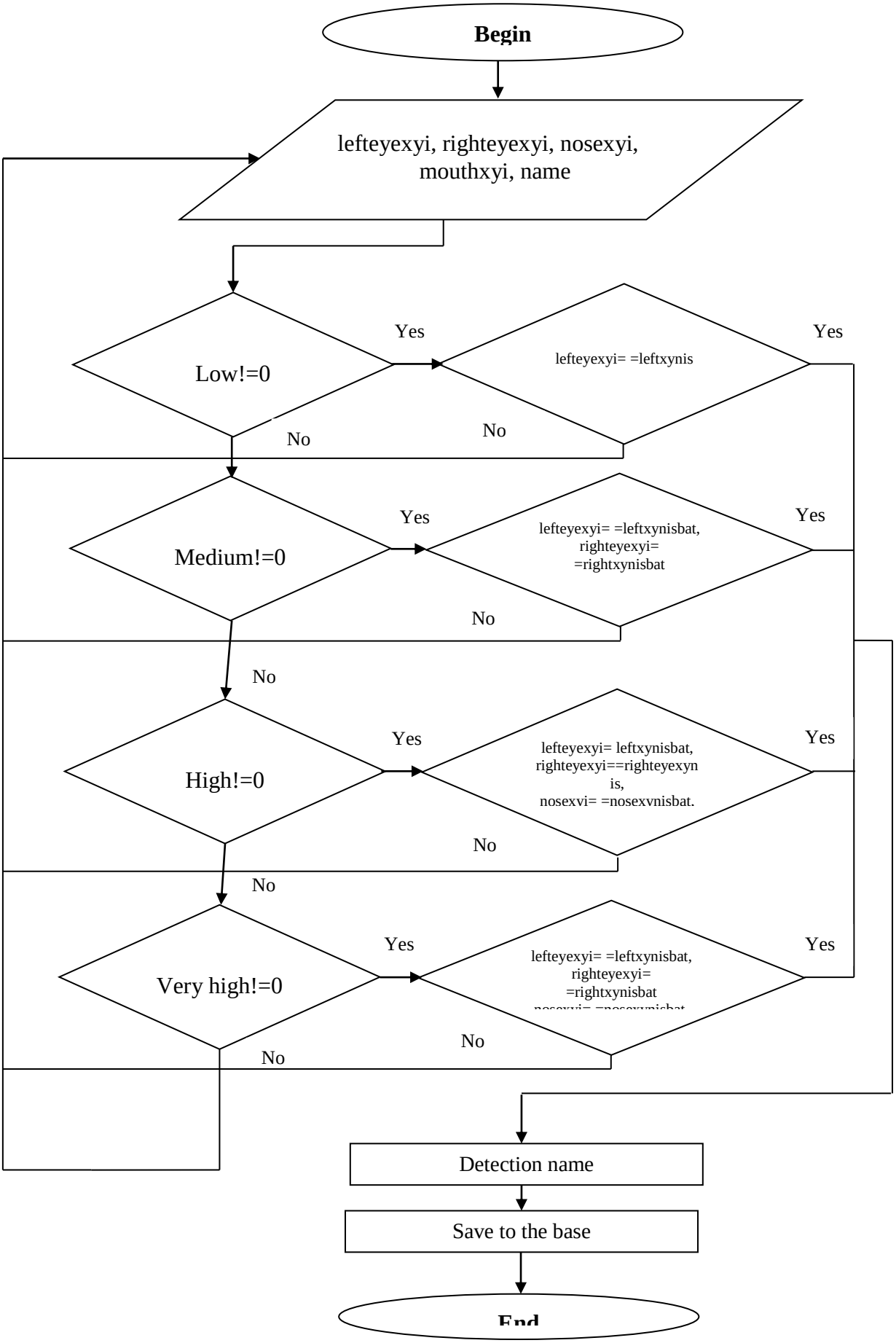
You can see degree of similarity in present "%":

$$\frac{xns}{xns2}100\%,$$

Here xns – resent value;

xns_2 – current value.

The proposed face recognition system consists of two phases which are the enrollment and recognition/verification phases as depicted in block schema 1 and block schema 2. It consists of several modules which are Image Registration, Face Detection, Recognition, Checking the system and Comparing similarity of images. In image processing session, the image acquisition, feature extraction and data normalization are performed.



Block schema.2. Algorithm block-schema of entering chapter

3. Development the algorithm of program for face recognition

As mentioned above, the detection algorithm moves a ‘window’ through a snapshot, and a classification procedure is executed for every window. This can take 500-1000 ms on "slow" computers (Silverlight, C#). When an object is detected, recognized and performed. Detection procedure detects object only in positions which are adjacent to previously detected object position. Detection takes much less time than object registration and it can be performed more frequently (with higher FPS).

The Process of face detection and recognition:

- Initialize the Haar Cascade that is used as a “face library”;
- Convert image to grey scale and some other tweaks;
- First pass try to find a face with settings for speed;
- If no face is found in first pass, go through again with more detail, slower search;
- Once all the “faces” have been found by files .xml, we need to go through and combine overlapping rectangles into singe faces;
- For my uses, I then pick the largest face in the photo as my cropping bounds.

Recognition of object classifier uses together LBP 7x7 and PCI algorithm. Our experiments show that LBP 7x7 is one of the most efficient state-of-the-art methods in face recognition. Adding noise modified images to the training set does not have any effect on the recognition rates of LBP 7x7 – unlike other methods that use the training sample to compute models or transformation matrices. This is caused by the nature of the method, where the histogram of LBP 7x7 patterns of the noisy image differs too much from the original images. This can be also a disadvantage, when the images in the test set are corrupted with noise. On the other hand, adding images with transformed face expression helps and the system is more resistant to expression change in the images.

The training algorithm is launched offline and it takes about a week to train a face detector. As an input it has a list of positive (i.e. face) and negative (i.e.

background, non-face) examples of objects. Consequently, it generates a list of weak classifiers which are grouped into stages.

This our software consist of two modules(fig.28):

- Face detection and recognition module;

- Admin module.

Face detection and recognition is main module. It can register new user and enter current user.

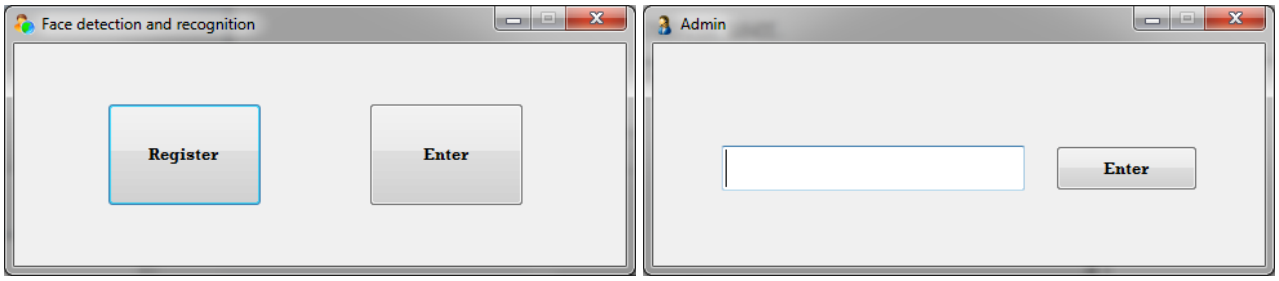


Fig.28. Main form of software

Registration module: We can register to the system on the base key-points of the face, group number and student name. Registration can be following: borderline of face – red, eye – blue, nose – green and mouth – yellow, so name and group – not free. You can see on figure.3.5.

Data base of registration:

login	gurux	face	leftxynisbat	rightxynisba	nosexynisba	mouthxynis	vaqt
Islomov Sh Z	302-12Axu	face14.bmp	0,703125	0,203456	0,693129	0,54542	17.05.2014 16:15:36
Mavlonov O	302-12Axu	face13.bmp	0,835352	0,335865	0,703125	0,54542	11.05.2014 16:20:52
Xidirov B	302-12Axu	face12.bmp	0,783124	0,283565	0,773126	0,44201	10.05.2014 13:26:52
Ibrohimov A A	302-12Axu	face11.bmp	0,843113	0,443115	0,783124	0,75421	09.05.2014 11:30:52
Pardayev R	302-12Axu	face10.bmp	0,793128	0,593128	0,783199	0,12464	07.05.2014 10:50:52
Ro'zimetov A A	302-12Axu	face09.bmp	0,903144	0,203147	0,793128	0,78542	07.05.2014 18:46:52
Otaxonov A	302-12Axu	face08.bmp	0,693129	0,223128	0,835352	0,45764	07.05.2014 14:32:52
Jumayev S S	302-12Axu	face07.bmp	0,773126	0,233122	0,843113	0,86421	03.05.2014 15:14:52
Teshabayev O	302-12Axu	face06.bmp	0,783199	0,343194	0,883156	0,74212	01.05.2014 10:11:52
Bekmirzayev C	302-12Axu	face05.bmp	0,883156	0,193158	0,893112	0,74211	30.04.2014 16:21:21
Isoqov E	302-12Axu	face04.bmp	0,893112	0,213178	0,903144	0,50000	18.04.2014 16:21:21

Entering module: this module consists of degree of the security and face detection. Users are recognised when their key-pints matchs with being in the current base. There are four types of degree of the security: low, medium, high and very high.

Low degree: This degree of security includes: size of face, left eye, group and name;

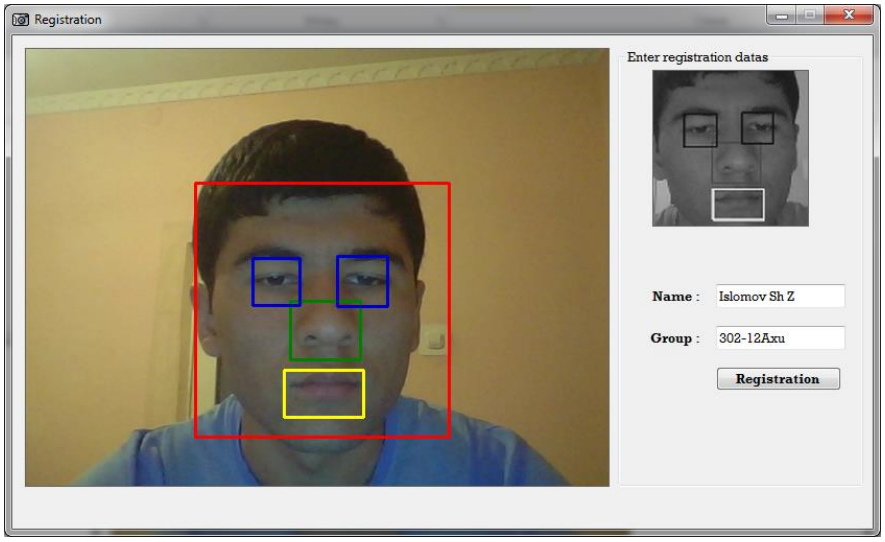


Fig.29. Registration module

Medium degree: This degree of security includes: size of face, left and right eye, name, group;

High degree: This degree of security includes: size of face, left and right eye, nose, name, group;

Very high degree: This degree of security includes: size of face, left and right eye, nose, mouth name, group.

Data base of entering:

login	vaqt
Islomov Sh Z	17.05.2014 16:15:36
Ibrohimov A A	11.05.2014 16:20:52
Mavlonov O	10.05.2014 13:26:52
Ro'zimetov A A	09.05.2014 11:30:52
Otaxonov A	07.05.2014 10:50:52
Jumayev S S	07.05.2014 18:46:52
Teshabayev O	07.05.2014 14:32:52
Bekmirzayev O N	03.05.2014 15:14:52
Isoqov E	01.05.2014 10:11:52

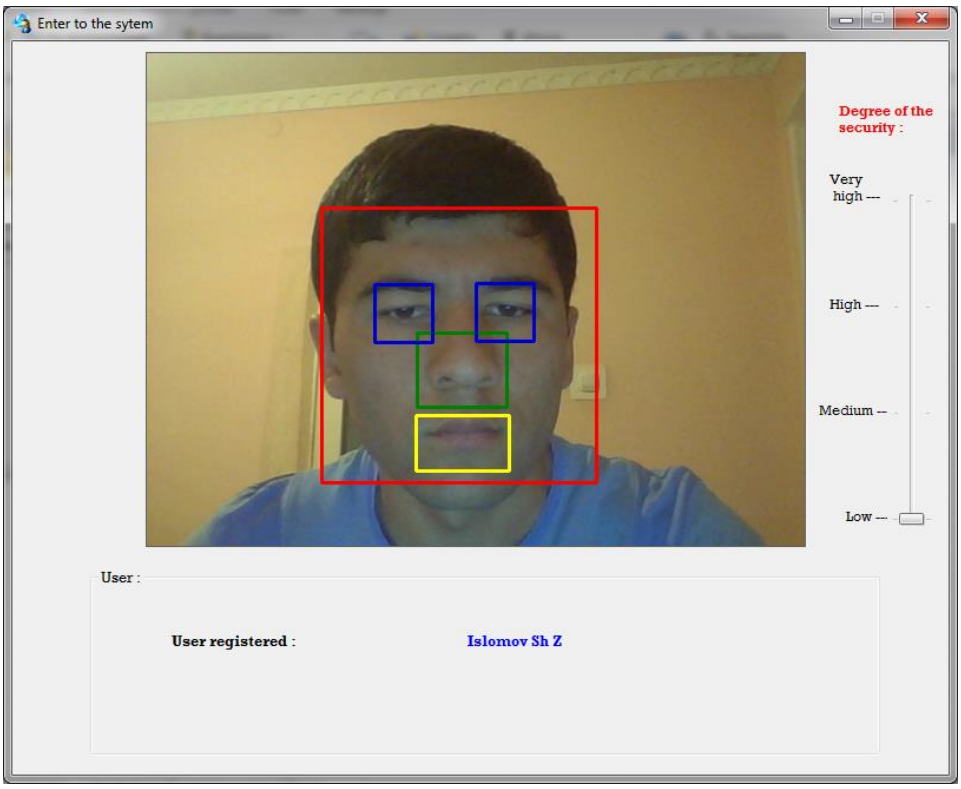


Fig.30. Entiring module

We know what admin controls and manages all the system by the politic of a security. Admin module consist two small modules(fig.31):

- Check the system;
- Degree of similarity.

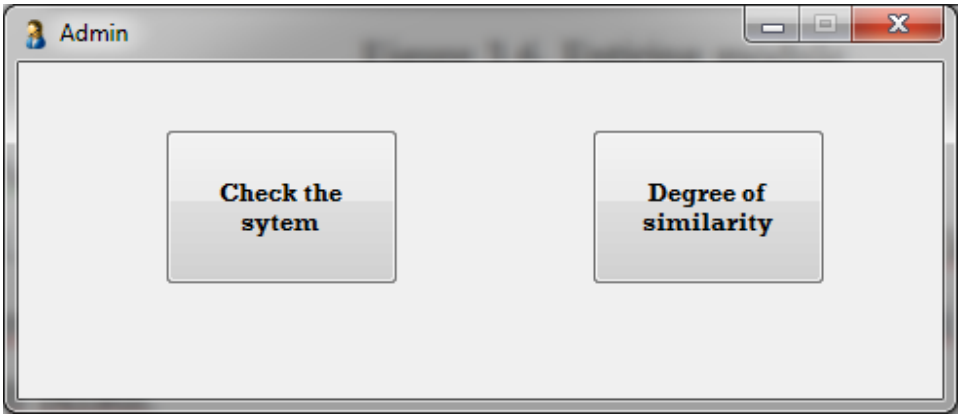


Fig.31. Admin module

Check the system module. We can take all information (name, registration time and image) about user base on inputting user's name (fig.32).

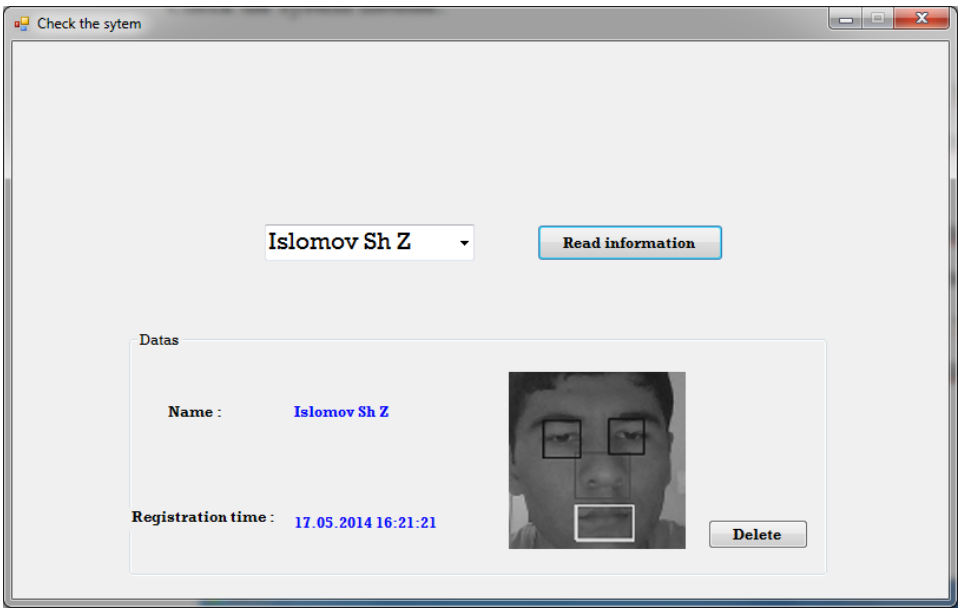


Fig.32. Check the system module

Degree of similarity module: We can see that how degree of the similarity is.

Summary of chapter III

In this chapter, we have analyzed various current PCA based on LPB7x7 based and other algorithms for face recognition. This analysis is vital in developing new robust algorithms for face recognition. Among various PCA and LBP 7x7 algorithms analyzed, the best result was found when Manual face localization was used on ORL and SHEFFIELD database consisting of 100 components. The face recognition rate in this case was approximately 100%. The next best was about 90% face recognition rate using PCA-LBP 7x7 on ORL database.

We developed software using combining methods PCA and LBP 7x7. The created software may be recognize and detect slowly but it has very high probability.

Also I used degree of security – low, medium, high and very high in software.

CONCLUSION

We can speak as the conclusion following:

- Studied problems and errors of using face recognition and detection technologies in information and communication systems and designed events preventing;
- Analyzed and compared of face recognition algorithms and methods for solving problems and selected the most good algorithm;
- Experimented face recognition algorithms by adding of modification methods which will applicable for creating face recognition system;
- Developed face recognition systems based on solving problems of current methods;
- Create face recognition systems on the based PCI+LBP7X7 algorithm;
- Developed software base on finally algorithm and this system will prevent all detects which shown in above.

REFERENCES

1. I.A.Karimov. "On measures for further development of the National Information and Communication System of the Republic of Uzbekistan". June 27, 2013. № DP 1989.
2. I.A.Karimov. The global financial and economic crisis, ways and measures to overcome it in the conditions of Uzbekistan. - T. Uzbekistan. 2009.
3. S. Parupati, R. Bakkannagiri, S. Sankar and V. Kulathumani, Collaborative acquisition of multi-view face images for real-time face recognition using a wireless camera network, ICDSC. 2011. 312p.
4. V. Kulathumani, S. Parupati, A. Ross and R. Jillela, Collaborative face recognition using a network of embedded cameras, Springer-Verlag, 2011. 540p.
5. R.Chellappa, C.L.Wilson, S.Sirohey, Human and machine recognition of faces, Proceedings of IEEE (2010). 705-740p.
6. B.Gokberk, L.Akarun, E.Alpaydin. Feature selection for pose invariant face recognition. 2012. 150p.
7. B.Gokberk, M.O.Irfanoglu, L.Akarun, E.Alpaydin. Optimal gabor kernel location selection for face recognition. Proceedings of the IEEE International Conference on Image Processing. 2009. 750p.
8. N.Kruger. An algorithm for the learning of weights in discrimination functions using a priori constraint. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2010. 764–768pp.
9. Z.Lihong, W.Ye, and T.Hongfeng, Face Recognition based on Independent Component Analysis in Proc. 2011 Chinese Control and Decision Conference. May 2011. 426-429pp.
10. J. Lei, C. Lu, and Z. Pan, “Enhancement of components in ICA for face recognition” in Proc. 9th International Conf. on Software Engineering Research, Management and Applications. August 2011. 33 – 38 pp.
11. P. Chandra, M. Chandra, M. Kumar, B. Latha “Multi scale feature extraction and enhancement using SVD towards secure face

recognition system," in Proc. International Conf. on Signal Processing, Communication, Computing and Networking Technologies. July 2011. 64 – 69pp.

12. A. Pervaiz, "Real time face recognition system based on EBGM framework," in Proc. 12th International Conf. on Computer Modelling and Simulation. March 2010. 262 – 266pp.

13. B. Weyrauch, B. Heisele, J. Huang, and V. Blanz, "Component-based face recognition with 3d morphable models," in Proc. CVPRW'04 Conf. on Computer Vision and Pattern Recognition Workshop. June 2004. 85p.

14. C. Wang, L. Lan, Y. Zhang, and M. Gu, "Face recognition based on principle component analysis and support vector machine," in Proc. 3rd International Workshop on Intelligent Systems and Applications. May 2012. 400-450pp.

15. Y. Choi, T. Tokumoto, M. Lee, and S. Ozawa, "Incremental two-dimensional two-directional principal component analysis (I(2D)2PCA) for face recognition," in Proc. IEEE International Conf. on Acoustics, Speech and Signal Processing. May 2011. 1493 – 1496 pp.

16. Z. Lin, Z. Wenrui, S. Li, and F. Zhijun, "Infrared face recognition based on compressive sensing and PCA," in Proc. IEEE International Conf. on Computer Science and Automation Engineering. June 2011. 51 – 54 pp.

17. G. Sun, L. Zhang, and H. Sun, "Face recognition based on symmetrical weighted PCA," in Proc. International Conf. on Computer Science and Service System. August 2011. 2249 – 2252 pp.

18. Sh.Z.Islomov, "Axborotni himoyalash tizimida tasvirlarni tanib olishning katakdagi piksellarni ifodalash algoritmini qo'llash". Republican Seminar: "Information security in communication, information and telecommunication technologies. Problems and Solutions". Scientific Engineering and Marketing Research "Unicon.uz". October, 2013.

19. Sh.Z.Islomov, “Axborotni himoyalash tizimida tasvirlarni tanib olishning onlayn va oflayn usullarida neyron tarmoqlarni qo’llash”. Republican Seminar: “Information security in communication, information and telecommunication technologies. Problems and Solutions”. Scientific Engineering and Marketing Research “Unicon.uz”. October, 2013.

20. Sh.Z.Islomov, “Models of faces representations and localization accuracy”. Republican scientifically-technical conference: “Prospects for effective development of information technologies and systems of telecommunication”, Tashkent. March, 2014.

21. Sh.Z.Islomov, “Optimization genetic model for hausdor distance-based face localization”. Republican scientifically-technical conference: “Prospects for effective development of information technologies and systems of telecommunication”, Tashkent. March, 2014.

22. M.M.Karimov, Sh.Z.Islomov, S.M.Sagatova, “Robot tizimlarida tasvirlarni tanib olish algoritmlarini tahlillash”. Chemical technical world magazine. June, 2014.

23. M.M.Karimov, Sh.Z.Islomov, “Analyses of face recognition methods and modifications by adding Gaussian-Noise”. TUIT news. June, 2014.

24. http://www.lex.uz/law_collection. “The global financial and economic crisis, ways and measures to overcome it in the conditions of Uzbekistan”.

25. http://en.wikipedia.org/wiki/Facial_recognition_system. “Facial recognition system”.

26. http://www.facedetection.com/facedetection/publications.htm. “Genetic Model Optimization for Hausdorff Distance-Based Face Localization”.

27. http://www.face-rec.org/algorithms/#PCA. “Face Recognition by Independent Component Analysis”.

APPENDIX

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using System.IO;

namespace Face_Recognition
{
    public partial class Asosiy : Form
    {

        public Asosiy()
        {
            InitializeComponent();

        }

        private void button1_Click(object sender, EventArgs e)
        {
            Ruyxat rx = new Ruyxat();
            rx.ShowDialog();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Kirish kr = new Kirish();
            kr.ShowDialog();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Admin ad = new Admin();
            ad.ShowDialog();
        }

        private void Asosiy_Load(object sender, EventArgs e)
        {
            Admin ad = new Admin();
            ad.Show();
        }

    }
}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using System.IO;
using System.Data.OleDb;


namespace Face_Recognition
{
    public partial class Ruyxat : Form
    {
        Image<Bgr, Byte> currentFrame;
```

```
        Capture grabber;
        HaarCascade face;
        HaarCascade eye;
        HaarCascade mouth;
        HaarCascade nose;

        MCvFont font = new MCvFont(FONT.CV_FONT_HERSHEY_TRIPLEX, 0.5d, 0.5d);
        Image<Gray, byte> result, TrainedFace = null;
        Image<Gray, byte> gray = null;
        List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>();
        List<string> labels = new List<string>();
        List<string> NamePersons = new List<string>();
        int ContTrain, NumLabels, t;
        string name, names = null;
        int yuzx, yuzy;
        float burunx = 0;
        float leftxynisbat = 0;
        float rightxynisbat = 0;
        float nosexynisbat = 0;
        float mouthxynisbat = 0;


        public Ruyxat()
        {
            InitializeComponent();

            face = new HaarCascade("haarcascade_frontalface_default.xml");
            eye = new HaarCascade("haarcascade_eye.xml");
            mouth = new HaarCascade("haarcascade_mcs_mouth.xml");
            nose = new HaarCascade("haarcascade_mcs_nose.xml");
            try
            {

                string Labelsinfo = File.ReadAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt");
                string[] Labels = Labelsinfo.Split('%');
                NumLabels = Convert.ToInt16(Labels[0]);
                ContTrain = NumLabels;
                string LoadFaces;

                for (int tf = 1; tf < NumLabels + 1; tf++)
                {
                    LoadFaces = "face" + tf + ".bmp";
                    trainingImages.Add(new Image<Gray, byte>(Application.StartupPath +
"/TrainedFaces/" + LoadFaces));
                    labels.Add(Labels[tf]);
                }

            }
            catch (Exception e)
            {

            }

        }




        void FrameGrabber(object sender, EventArgs e)
        {

            NamePersons.Add("");
```

```
            //Get the current frame form capture device
            currentFrame = grabber.QueryFrame().Resize(560, 420,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

            //Convert it to Grayscale
            gray = currentFrame.Convert<Gray, Byte>();

            //Face Detector
            MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
          face,
          1.2,
          10,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(20, 20));

            MCvAvgComp[][] eyesDetected = gray.DetectHaarCascade(
          eye,
          1.2,
          20,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(2, 2));


            MCvAvgComp[][] mouthDetected = gray.DetectHaarCascade(
          mouth,
          1.2,
          100,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(2, 2));

            MCvAvgComp[][] noseDetected = gray.DetectHaarCascade(
          nose,
          1.2,
          10,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(2, 2));


            foreach (MCvAvgComp f in noseDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
                currentFrame.Draw(f.rect, new Bgr(Color.Green), 2);

                if (trainingImages.ToArray().Length != 0)
                {
                    float xns = (f.rect.X - yuzx);
                    float yns = (f.rect.Y - yuzy);
                    burunx = (f.rect.X - yuzx);

                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));
                    nosexynisbat = (xns / yns);

                }


                NamePersons[t - 1] = name;
                NamePersons.Add("");


            }
```

```
            foreach (MCvAvgComp f in mouthDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
                currentFrame.Draw(f.rect, new Bgr(Color.Yellow), 2);

                if (trainingImages.ToArray().Length != 0)
                {
                    ////TermCriteria for face recognition with numbers of trained images
like maxIteration
                    //MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

                    ////Eigen face recognizer
                    //EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 3000, ref termCrit);
                    //name = recognizer.Recognize(result);
                    float xns = (f.rect.X - yuzx);
                    float yns = (f.rect.Y - yuzy);

                    //Draw the label for each face detected and recognized
                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));
                    mouthxynisbat = (xns / yns);
                }

                NamePersons[t - 1] = name;
                NamePersons.Add("");


            }
            foreach (MCvAvgComp f in eyesDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
                currentFrame.Draw(f.rect, new Bgr(Color.MediumBlue), 2);

                if (trainingImages.ToArray().Length != 0 && burunx != 0)
                {
                    //TermCriteria for face recognition with numbers of trained images
like maxIteration
                    MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

                    ////Eigen face recognizer
                    //EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 3000, ref termCrit);
                    //name = recognizer.Recognize(result);
                    float xns = (f.rect.X - yuzx);
                    float yns = (f.rect.Y - yuzy);

                    //Draw the label for each face detected and recognized
                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));

                    if (xns < burunx)
                    {

                        leftxynisbat = (xns / yns);
                    }
                    if (xns > burunx)
```

```
                    {
                        rightxynisbat = (xns / yns);
                    }

                }

                NamePersons[t - 1] = name;
                NamePersons.Add("");
            }


            //Action for each element detected
            foreach (MCvAvgComp f in facesDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
                currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

                if (trainingImages.ToArray().Length != 0)
                {

                    ////TermCriteria for face recognition with numbers of trained images
like maxIteration
                    //MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

                    ////Eigen face recognizer
                    //EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 3000, ref termCrit);
                    //name = recognizer.Recognize(result);

                    //Draw the label for each face detected and recognized
                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));

                    yuzx = f.rect.X;
                    yuzy = f.rect.Y;
                }
                NamePersons[t - 1] = name;
                NamePersons.Add("");


            }
            t = 0;

            //Names concatenation of persons recognized
            for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)
            {
                names = names + NamePersons[nnn] + ", ";
            }
            //Show the faces procesed and recognized
            imageBoxFrameGrabber.Image = currentFrame;

            names = "";
            //Clear the list(vector) of names
            NamePersons.Clear();

        }

        private void Ruyxat_Load(object sender, EventArgs e)
        {
            //camerani yoqish
            grabber = new Capture();
            grabber.QueryFrame();
```

```
            Application.Idle += new EventHandler(FrameGrabber);
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            bool loginbuyicha = true;
            OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string login = "";
            string queryString = "SELECT*from main;";
            OleDbCommand command1 = new OleDbCommand(queryString, con1);
            con1.Open();
            OleDbDataReader reader = command1.ExecuteReader();
            while (reader.Read())
            {
                object logini = reader["login"];
                login = String.Format("{0}", logini);
                if (login == textBox1.Text)
                {
                    loginbuyicha = false;
                }
            }
            reader.Close();

            try
            {
                if
(loginbuyicha&&leftxynisbat!=0&&rightxynisbat!=0&&nosexynisbat!=0&&mouthxynisbat!=0)
                {
                    ContTrain = ContTrain + 1;
                    gray = grabber.QueryGrayFrame().Resize(320, 240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

                    MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
                    face,
                    1.2,
                    10,
                    Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
                    new Size(20, 20));


                    foreach (MCvAvgComp f in facesDetected[0])
                    {
                        TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();
                        break;
                    }


                    TrainedFace = result.Resize(150, 150,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                    trainingImages.Add(TrainedFace);
                    labels.Add(textBox1.Text);


                    imageBox1.Image = TrainedFace;


                    File.WriteAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", trainingImages.ToArray().Length.ToString() + "%");

                    string yuzsaqlash="";
                    for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                    {
```

```
                        trainingImages.ToArray()[i - 1].Save(Application.StartupPath +
"/TrainedFaces/face" + i + ".bmp");
                        File.AppendAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt", labels.ToArray()[i - 1] + "%");
                        yuzsaqlash = "face"+i+".bmp";
                    }

                    OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                    OleDbCommand command = new OleDbCommand();
                    command.Connection = con;
                    command.CommandText = "INSERT INTO
main(login,gurux,face,leftxynisbat,rightxynisbat,nosexynisbat,mouthxynisbat,vaqt)
values('" + textBox1.Text + "','" + textBox2.Text + "','" + yuzsaqlash + "','" +
leftxynisbat + "','" + rightxynisbat + "','" + nosexynisbat + "','" + mouthxynisbat +
"','" + DateTime.Now.ToString() + "')";
                    con.Open();
                    command.ExecuteNonQuery();
                    con.Close();
                    MessageBox.Show(textBox1.Text + " Registired to the system!",
"Information of the system", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    textBox1.Text = "";
                    textBox2.Text = "";

                }
                else
                {
                    MessageBox.Show("Error in entering parameters!", "System
information", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    textBox1.Text = "";
                    textBox2.Text = "";

                }
            }
            catch
            {
            }
        }

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }

        private void imageBoxFrameGrabber_Click(object sender, EventArgs e)
        {

        }

        private void imageBoxFrameGrabber_Click_1(object sender, EventArgs e)
        {

        }

        private void textBox2_TextChanged(object sender, EventArgs e)
        {

        }
```

```
    }
}

using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;
using System.IO;
using System.Data.OleDb;

namespace Face_Recognition
{
    public partial class Kirish : Form
    {
        Image<Bgr, Byte> currentFrame;
        Capture grabber;
        HaarCascade face;
        HaarCascade eye;
        HaarCascade mouth;
        HaarCascade nose;
        MCvFont font = new MCvFont(FONT.CV_FONT_HERSHEY_TRIPLEX, 0.5d, 0.5d);
        Image<Gray, byte> result, TrainedFace = null;
        Image<Gray, byte> gray = null;
        List<Image<Gray, byte>> trainingImages = new List<Image<Gray, byte>>();
        List<string> labels = new List<string>();
        List<string> NamePersons = new List<string>();
        int ContTrain, NumLabels, t;
        string name, names = null;
        int yuzx,yuzy;
        float burunx = 0;
        float leftxynisbat = 0;
        float rightxynisbat = 0;
        float nosexynisbat = 0;
        float mouthxynisbat = 0;
        bool chapkuzyozish = false;
        bool ungkuzyozish = false;
        bool ogizyozish = false;
        bool burunyozish = false;
        bool yuzyozish = false;
        bool aniqlandi = true;

        public Kirish()
        {
            InitializeComponent();

            face = new HaarCascade("haarcascade_frontalface_default.xml");
            eye = new HaarCascade("haarcascade_eye.xml");
            mouth = new HaarCascade("haarcascade_mcs_mouth.xml");
            nose = new HaarCascade("haarcascade_mcs_nose.xml");
            try
            {

                string Labelsinfo = File.ReadAllText(Application.StartupPath +
"/TrainedFaces/TrainedLabels.txt");
                string[] Labels = Labelsinfo.Split('%');
                NumLabels = Convert.ToInt16(Labels[0]);
                ContTrain = NumLabels;
                string LoadFaces;

                for (int tf = 1; tf < NumLabels + 1; tf++)
                {
```

```
                    LoadFaces = "face" + tf + ".bmp";
                    trainingImages.Add(new Image<Gray, byte>(Application.StartupPath +
"/TrainedFaces/" + LoadFaces));
                    labels.Add(Labels[tf]);
                }

            }
            catch (Exception e)
            {

            }

        }


        void FrameGrabber(object sender, EventArgs e)
        {
            NamePersons.Add("");

            //Get the current frame form capture device
            currentFrame = grabber.QueryFrame().Resize(560, 420,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

            //Convert it to Grayscale
            gray = currentFrame.Convert<Gray, Byte>();



            //Face Detector
            MCvAvgComp[][] facesDetected = gray.DetectHaarCascade(
          face,
          1.2,
          10,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(20, 20));

            MCvAvgComp[][] eyesDetected = gray.DetectHaarCascade(
          eye,
          1.2,
          20,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(2, 2));


            MCvAvgComp[][] mouthDetected = gray.DetectHaarCascade(
          mouth,
          1.2,
          100,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(2, 2));

            MCvAvgComp[][] noseDetected = gray.DetectHaarCascade(
          nose,
          1.2,
          10,
          Emgu.CV.CvEnum.HAAR_DETECTION_TYPE.DO_CANNY_PRUNING,
          new Size(2, 2));


            foreach (MCvAvgComp f in noseDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
```

```
                currentFrame.Draw(f.rect, new Bgr(Color.Green), 2);

                if (trainingImages.ToArray().Length != 0)
                {
                    float xns = (f.rect.X - yuzx);
                    float yns = (f.rect.Y - yuzy);
                    burunx = (f.rect.X - yuzx);

                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));
                    nosexynisbat = (xns / yns);
                }
                if (burunyozish)
                {
                    TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();

                    TrainedFace = result.Resize(50, 55,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                    trainingImages.Add(TrainedFace);
                    for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                    {
                        trainingImages.ToArray()[i - 1].Save(Application.StartupPath +
"/TrainedFaces/" + label4.Text + "/burun" + ".bmp");
                    }
                    burunyozish = false;
                }
                NamePersons[t - 1] = name;
                NamePersons.Add("");
            }
            foreach (MCvAvgComp f in mouthDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
                currentFrame.Draw(f.rect, new Bgr(Color.Yellow), 2);

                if (trainingImages.ToArray().Length != 0)
                {
                    ////TermCriteria for face recognition with numbers of trained images
like maxIteration
                    //MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

                    ////Eigen face recognizer
                    //EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 3000, ref termCrit);
                    //name = recognizer.Recognize(result);
                    float xns = (f.rect.X - yuzx);
                    float yns = (f.rect.Y - yuzy);
                    mouthxynisbat = xns / yns;
                    //Draw the label for each face detected and recognized
                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));
                    if (ogizyozish)
                    {
                        TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();

                        TrainedFace = result.Resize(80, 50,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                        trainingImages.Add(TrainedFace);
                        for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                        {
                            trainingImages.ToArray()[i - 1].Save(Application.StartupPath
+ "/TrainedFaces/" + label4.Text + "/ogiz" + ".bmp");
                        }
```

```
                    ogizyozish = false;

                }

            }

            NamePersons[t - 1] = name;
            NamePersons.Add("");


        }
        foreach (MCvAvgComp f in eyesDetected[0])
        {
            t = t + 1;
            result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
            //draw the face detected in the 0th (gray) channel with blue color
            currentFrame.Draw(f.rect, new Bgr(Color.MediumBlue), 2);

            if (trainingImages.ToArray().Length != 0&&burunx!=0)
            {
                //TermCriteria for face recognition with numbers of trained images
like maxIteration
                MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

                ////Eigen face recognizer
                //EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 3000, ref termCrit);
                //name = recognizer.Recognize(result);
                float xns = (f.rect.X - yuzx);
                float yns = (f.rect.Y - yuzy);

                //Draw the label for each face detected and recognized
                currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));

                if (xns<burunx)
                {
                    leftxynisbat = (xns / yns);
                    if (chapkuzyozish)
                    {
                        TrainedFace = currentFrame.Copy(f.rect).Convert<Gray,
byte>();

                        TrainedFace = result.Resize(50, 50,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                        trainingImages.Add(TrainedFace);
                        for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                        {
                            trainingImages.ToArray()[i -
1].Save(Application.StartupPath + "/TrainedFaces/" + label4.Text + "/chapkuz" + ".bmp");
                        }
                        chapkuzyozish = false;

                    }
                }
                if (xns>burunx)
                {
                    rightxynisbat = (xns / yns);

                    if (ungkuzyozish)
                    {
                        TrainedFace = currentFrame.Copy(f.rect).Convert<Gray,
byte>();
```

```
                            TrainedFace = result.Resize(50, 50,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                            trainingImages.Add(TrainedFace);
                            for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                            {
                                trainingImages.ToArray()[i -
1].Save(Application.StartupPath + "/TrainedFaces/" + label4.Text + "/ungkuz" + ".bmp");
                            }
                            ungkuzyozish = false;

                        }
                    }

                }

                NamePersons[t - 1] = name;
                NamePersons.Add("");
            }

            //Action for each element detected
            foreach (MCvAvgComp f in facesDetected[0])
            {
                t = t + 1;
                result = currentFrame.Copy(f.rect).Convert<Gray, byte>().Resize(100, 100,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                //draw the face detected in the 0th (gray) channel with blue color
                currentFrame.Draw(f.rect, new Bgr(Color.Red), 2);

                if (trainingImages.ToArray().Length != 0)
                {

                    ////TermCriteria for face recognition with numbers of trained images
like maxIteration
                    //MCvTermCriteria termCrit = new MCvTermCriteria(ContTrain, 0.001);

                    ////Eigen face recognizer
                    //EigenObjectRecognizer recognizer = new
EigenObjectRecognizer(trainingImages.ToArray(), labels.ToArray(), 3000, ref termCrit);
                    //name = recognizer.Recognize(result);

                    //Draw the label for each face detected and recognized
                    currentFrame.Draw("", ref font, new Point(f.rect.X - 2, f.rect.Y -
2), new Bgr(Color.LightGreen));

                    yuzx = f.rect.X;
                    yuzy = f.rect.Y;

                    if (yuzyozish)
                    {
                        TrainedFace = currentFrame.Copy(f.rect).Convert<Gray, byte>();

                        TrainedFace = result.Resize(150, 150,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
                        trainingImages.Add(TrainedFace);
                        for (int i = 1; i < trainingImages.ToArray().Length + 1; i++)
                        {
                            trainingImages.ToArray()[i - 1].Save(Application.StartupPath
+ "/TrainedFaces/"+label4.Text+"/yuz" + ".bmp");
                        }
                        yuzyozish = false;

                    }

                    if (trackBar1.Value == 0 && aniqlandi)
                    {
```

```
                        OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                        string lefteyexy = "";
                        string queryString = "SELECT*from main;";
                        OleDbCommand command1 = new OleDbCommand(queryString, con1);
                        con1.Open();
                        OleDbDataReader reader = command1.ExecuteReader();
                        while (reader.Read())
                        {
                            object logini = reader["login"];
                            string login = String.Format("{0}", logini);
                            object lefteyexyi = reader["leftxynisbat"];
                            lefteyexy = String.Format("{0}", lefteyexyi);
                            if (Convert.ToSingle(lefteyexy) + 0.01 > leftxynisbat &&
Convert.ToSingle(lefteyexy) - 0.01 < leftxynisbat)
                            {
                                label4.Text = login;
                                chapkuzyozish = true;
                                ungkuzyozish = true;
                                ogizyozish = true;
                                burunyozish = true;
                                yuzyozish = true;
                                aniqlandi = false;
                                Directory.CreateDirectory(Application.StartupPath +
"/TrainedFaces/" + label4.Text);
                                OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                                OleDbCommand command = new OleDbCommand();
                                command.Connection = con;
                                command.CommandText = "INSERT INTO kirish(login,vaqt)
values('" + label4.Text + "','" + DateTime.Now.ToString() + "')";
                                con.Open();
                                command.ExecuteNonQuery();
                                con.Close();
                            }
                        }
                        reader.Close();
                    }

                    if (trackBar1.Value == 1 && aniqlandi)
                    {

                        OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                        string lefteyexy = "";
                        string righteyexy = "";
                        string queryString = "SELECT*from main;";
                        OleDbCommand command1 = new OleDbCommand(queryString, con1);
                        con1.Open();
                        OleDbDataReader reader = command1.ExecuteReader();
                        while (reader.Read())
                        {
                            object logini = reader["login"];
                            string login = String.Format("{0}", logini);
                            object lefteyexyi = reader["leftxynisbat"];
                            lefteyexy = String.Format("{0}", lefteyexyi);
                            object righteyexyi = reader["rightxynisbat"];
                            righteyexy = String.Format("{0}", righteyexyi);

                            if (Convert.ToSingle(lefteyexy) + 0.05 > leftxynisbat &&
Convert.ToSingle(lefteyexy) - 0.05 < leftxynisbat && Convert.ToSingle(righteyexy) + 0.05
> rightxynisbat && Convert.ToSingle(righteyexy) - 0.05 < rightxynisbat)
                            {
                                label4.Text = login;
                                chapkuzyozish = true;
```

```
                                ungkuzyozish = true;
                                ogizyozish = true;
                                burunyozish = true;
                                yuzyozish = true;
                                aniqlandi = false;
                                Directory.CreateDirectory(Application.StartupPath +
"/TrainedFaces/" + label4.Text);
                                OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                                OleDbCommand command = new OleDbCommand();
                                command.Connection = con;
                                command.CommandText = "INSERT INTO kirish(login,vaqt)
values('" + label4.Text + "','" + DateTime.Now.ToString() + "')";
                                con.Open();
                                command.ExecuteNonQuery();
                                con.Close();
                            }
                        }
                        reader.Close();
                    }


                    if (trackBar1.Value == 2 && aniqlandi)
                    {

                        OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                        string lefteyexy = "";
                        string righteyexy = "";
                        string nosexy = "";
                        string queryString = "SELECT*from main;";
                        OleDbCommand command1 = new OleDbCommand(queryString, con1);
                        con1.Open();
                        OleDbDataReader reader = command1.ExecuteReader();
                        while (reader.Read())
                        {
                            object logini = reader["login"];
                            string login = String.Format("{0}", logini);
                            object lefteyexyi = reader["leftxynisbat"];
                            lefteyexy = String.Format("{0}", lefteyexyi);
                            object righteyexyi = reader["rightxynisbat"];
                            righteyexy = String.Format("{0}", righteyexyi);
                            object nosexyi = reader["nosexynisbat"];
                            nosexy = String.Format("{0}", nosexyi);

                            if (Convert.ToSingle(lefteyexy) + 0.05 > leftxynisbat &&
Convert.ToSingle(lefteyexy) - 0.05 < leftxynisbat && Convert.ToSingle(righteyexy) + 0.05
> rightxynisbat && Convert.ToSingle(righteyexy) - 0.05 < rightxynisbat &&
Convert.ToSingle(nosexy) + 0.05 > nosexynisbat && Convert.ToSingle(nosexy) - 0.05 <
nosexynisbat)
                            {
                                label4.Text = login;
                                chapkuzyozish = true;
                                ungkuzyozish = true;
                                ogizyozish = true;
                                burunyozish = true;
                                yuzyozish = true;
                                aniqlandi = false;
                                Directory.CreateDirectory(Application.StartupPath +
"/TrainedFaces/" + label4.Text);
                                OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                                OleDbCommand command = new OleDbCommand();
                                command.Connection = con;
```

```
                                command.CommandText = "INSERT INTO kirish(login,vaqt)
values('" + label4.Text + "','" + DateTime.Now.ToString() + "')";
                                con.Open();
                                command.ExecuteNonQuery();
                                con.Close();
                            }
                        }
                        reader.Close();
                    }

                    if (trackBar1.Value == 3 && aniqlandi)
                    {
                        OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                        string lefteyexy = "";
                        string righteyexy = "";
                        string nosexy = "";
                        string mouthxy = "";
                        string queryString = "SELECT*from main;";
                        OleDbCommand command1 = new OleDbCommand(queryString, con1);
                        con1.Open();
                        OleDbDataReader reader = command1.ExecuteReader();
                        while (reader.Read())
                        {
                            object logini = reader["login"];
                            string login = String.Format("{0}", logini);
                            object lefteyexyi = reader["leftxynisbat"];
                            lefteyexy = String.Format("{0}", lefteyexyi);
                            object righteyexyi = reader["rightxynisbat"];
                            righteyexy = String.Format("{0}", righteyexyi);
                            object nosexyi = reader["nosexynisbat"];
                            nosexy = String.Format("{0}", nosexyi);
                            object mouthxyi = reader["mouthxynisbat"];
                            mouthxy = String.Format("{0}", mouthxyi);

                            if (Convert.ToSingle(lefteyexy) + 0.05 > leftxynisbat &&
Convert.ToSingle(lefteyexy) - 0.05 < leftxynisbat && Convert.ToSingle(righteyexy) + 0.05
> rightxynisbat && Convert.ToSingle(righteyexy) - 0.05 < rightxynisbat &&
Convert.ToSingle(nosexy) + 0.05 > nosexynisbat && Convert.ToSingle(nosexy) - 0.05 <
nosexynisbat && Convert.ToSingle(mouthxy) + 0.05 > mouthxynisbat &&
Convert.ToSingle(mouthxy) - 0.05 < mouthxynisbat)
                            {
                                label4.Text = login;
                                chapkuzyozish = true;
                                ungkuzyozish = true;
                                ogizyozish = true;
                                burunyozish = true;
                                yuzyozish = true;
                                aniqlandi = false;
                                Directory.CreateDirectory(Application.StartupPath +
"/TrainedFaces/" + label4.Text);
                                OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                                OleDbCommand command = new OleDbCommand();
                                command.Connection = con;
                                command.CommandText = "INSERT INTO kirish(login,vaqt)
values('" + label4.Text + "','" + DateTime.Now.ToString() + "')";
                                con.Open();
                                command.ExecuteNonQuery();
                                con.Close();
                            }
                        }
                        reader.Close();
                    }
                }
```

```
                NamePersons[t - 1] = name;
                NamePersons.Add("");


            }
            t = 0;

            //Names concatenation of persons recognized
            for (int nnn = 0; nnn < facesDetected[0].Length; nnn++)
            {
                names = NamePersons[nnn];
            }
            //Show the faces procesed and recognized
            imageBoxFrameGrabber.Image = currentFrame;

            names = "_ Mavjud emas _";
            //Clear the list(vector) of names
            NamePersons.Clear();

        }


        private void Kirish_Load(object sender, EventArgs e)
        {
            grabber = new Capture();

            grabber.QueryFrame();
            Application.Idle += new EventHandler(FrameGrabber);
        }



        private void button1_Click_1(object sender, EventArgs e)
        {

        }

        private void imageBoxFrameGrabber_Click(object sender, EventArgs e)
        {

        }

        private void trackBar1_Scroll(object sender, EventArgs e)
        {

        }

        private void label5_Click(object sender, EventArgs e)
        {

        }

        private void imageBoxFrameGrabber_Click_1(object sender, EventArgs e)
        {

        }

    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;

namespace Face_Recognition
{
    public partial class Admin : Form
    {
        public Admin()
        {
            InitializeComponent();
        }

        private void Admin_Load(object sender, EventArgs e)
        {
            button2.Visible = false;
            button3.Visible = false;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string parol = "";
            string queryString = "SELECT*from admin;";
            OleDbCommand command = new OleDbCommand(queryString, con);
            con.Open();
            OleDbDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                object paroli = reader["parol"];
                parol = String.Format("{0}", paroli);
            }
            reader.Close();
            if (textBox1.Text == parol)
            {
                textBox1.Visible = false;
                button1.Visible = false;
                button2.Visible = true;
                button3.Visible = true;
            }
            else
            {
                MessageBox.Show("Administratorga kirish uchun kalit so\'z xato
kiritildi.", "Tizim bo\'yicha xabar", MessageBoxButtons.OK, MessageBoxIcon.Error);
                textBox1.Text = "";
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Tizimni_tekshirish tiztek = new Tizimni_tekshirish();
            tiztek.ShowDialog();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Uxshashlik uxshash = new Uxshashlik();
            uxshash.ShowDialog();
        }
```

```
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;


namespace Face_Recognition
{
    public partial class Tizimni_tekshirish : Form
    {
        public Tizimni_tekshirish()
        {
            InitializeComponent();
        }
        int comuchir = 0;
        private void Tizimni_tekshirish_Load(object sender, EventArgs e)
        {
            OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string login = "";
            string queryString = "SELECT*from main;";
            OleDbCommand command1 = new OleDbCommand(queryString, con1);
            con1.Open();
            OleDbDataReader reader = command1.ExecuteReader();
            while (reader.Read())
            {
                object logini = reader["login"];
                login = String.Format("{0}", logini);
                this.comboBox1.Items.AddRange(new object[] {
            login});
                comuchir++;
            }
            reader.Close();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            DialogResult rs = MessageBox.Show("Do you want delete all datas?", "Query
window", MessageBoxButtons.OKCancel, MessageBoxIcon.Question);
            if (rs == DialogResult.OK)
            {
                OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                OleDbCommand command = new OleDbCommand();
                command.Connection = con;
                command.CommandText = "Delete from main where [login]='" + label2.Text +
"'";
                con.Open();
                command.ExecuteNonQuery();
                con.Close();


                OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
                OleDbCommand command1 = new OleDbCommand();
                command1.Connection = con1;
```

```
                command1.CommandText = "Delete from kirish where [login]='" + label2.Text
+ "'";
                con1.Open();
                command1.ExecuteNonQuery();
                con1.Close();


                pictureBox1.Image.Dispose();
                pictureBox1.Image = null;
                //this.Refresh();
                File.Delete(Application.StartupPath + "/TrainedFaces/" + rasm);
                File.Delete(Application.StartupPath + "/TrainedFaces/" + label2.Text +
"/chapkuz.bmp");
                File.Delete(Application.StartupPath + "/TrainedFaces/" + label2.Text +
"/ungkuz.bmp");
                File.Delete(Application.StartupPath + "/TrainedFaces/" + label2.Text +
"/burun.bmp");
                File.Delete(Application.StartupPath + "/TrainedFaces/" + label2.Text +
"/ogiz.bmp");
                File.Delete(Application.StartupPath + "/TrainedFaces/" + label2.Text +
"/yuz.bmp");
                Directory.Delete(Application.StartupPath + "/TrainedFaces/"+label2);
                pictureBox1.Image = System.Drawing.Image.FromFile(Application.StartupPath
+ "/TrainedFaces/" + "oddiy.bmp");
                label2.Text = "_ _ _ _ _ _";
                label4.Text = "_ _ _ _ _ _";
                this.comboBox1.Items.RemoveAt(comuchir - 1);

            }
        }
        string rasm = "";
        private void button2_Click(object sender, EventArgs e)
        {
            OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string login = comboBox1.Text;
            string vaqt = "";

            string queryString = "SELECT*from main where login = '" + login + "';";
            OleDbCommand command = new OleDbCommand(queryString, con);
            con.Open();
            OleDbDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                object vaqti = reader["vaqt"];
                vaqt = String.Format("{0}", vaqti);

                object rasmi = reader["face"];
                rasm = String.Format("{0}", rasmi);
                pictureBox1.Image = System.Drawing.Image.FromFile(Application.StartupPath
+ "/TrainedFaces/" + rasm);
                label2.Text = login;
                label4.Text = vaqt;
            }
            reader.Close();

        }
    }
}

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.OleDb;
using System.IO;

namespace Face_Recognition
{
    public partial class Uxshashlik : Form
    {
        public Uxshashlik()
        {
            InitializeComponent();
        }
        int comuchir = 0;
        private void Uxshashlik_Load(object sender, EventArgs e)
        {
            OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string login = "";
            string queryString = "SELECT*from kirish;";
            OleDbCommand command1 = new OleDbCommand(queryString, con1);
            con1.Open();
            OleDbDataReader reader = command1.ExecuteReader();
            while (reader.Read())
            {
                object logini = reader["login"];
                login = String.Format("{0}", logini);
                this.comboBox1.Items.AddRange(new object[] {
            login});
                comuchir++;
            }
            reader.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            OleDbConnection con = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string login = comboBox1.Text;
            string yangivaqt = "";
            string eskivaqt = "";
            string eskirasm = "";

            string yangichapkuz = "";
            string eskichapkuz = "";

            string yangiungkuz = "";
            string eskiungkuz = "";

            string yangiburun = "";
            string eskiburun = "";

            string yangiogiz = "";
            string eskiogiz = "";

            string queryString = "SELECT*from kirish where login = '" + login + "';";
            OleDbCommand command = new OleDbCommand(queryString, con);
            con.Open();
            OleDbDataReader reader = command.ExecuteReader();
            while (reader.Read())
            {
                object yangivaqti = reader["vaqt"];
                yangivaqt = String.Format("{0}", yangivaqti);
```

```
                object yangichapkuzi = reader["leftxynisbat"];
                yangichapkuz = String.Format("{0}", yangichapkuzi);


                object yangiungkuzi = reader["rightxynisbat"];
                yangiungkuz = String.Format("{0}", yangiungkuzi);

                object yangiburuni = reader["nosexynisbat"];
                yangiburun = String.Format("{0}", yangiburuni);

                object yangiogizi = reader["mouthxynisbat"];
                yangiogiz = String.Format("{0}", yangiogizi);

                yangirasmpic.Image =
System.Drawing.Image.FromFile(Application.StartupPath + "/TrainedFaces/" +login+
"/yuz.bmp");
                chapkuzpic.Image = System.Drawing.Image.FromFile(Application.StartupPath
+ "/TrainedFaces/" + login + "/chapkuz.bmp");
                ungkuzpic.Image = System.Drawing.Image.FromFile(Application.StartupPath +
"/TrainedFaces/" + login + "/ungkuz.bmp");
                burunpic.Image = System.Drawing.Image.FromFile(Application.StartupPath +
"/TrainedFaces/" + login + "/burun.bmp");
                ogizpic.Image = System.Drawing.Image.FromFile(Application.StartupPath +
"/TrainedFaces/" + login + "/ogiz.bmp");

                label10.Text = yangivaqt;
            }
            reader.Close();

            OleDbConnection con1 = new
OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source = yuz.mdb");
            string queryString1 = "SELECT*from main where login = '" + login + "';";
            OleDbCommand command1 = new OleDbCommand(queryString1, con1);
            con1.Open();
            OleDbDataReader reader1 = command1.ExecuteReader();
            while (reader1.Read())
            {
                object eskivaqti = reader1["vaqt"];
                eskivaqt = String.Format("{0}", eskivaqti);

                object eskirasmi = reader1["face"];
                eskirasm = String.Format("{0}", eskirasmi);

                object eskichapkuzi = reader1["leftxynisbat"];
                eskichapkuz = String.Format("{0}", eskichapkuzi);

                object eskiungkuzi = reader1["rightxynisbat"];
                eskiungkuz = String.Format("{0}", eskiungkuzi);

                object eskiburuni = reader1["nosexynisbat"];
                eskiburun = String.Format("{0}", eskiburuni);

                object eskiogizi = reader1["mouthxynisbat"];
                eskiogiz = String.Format("{0}", eskiogizi);


                if (Convert.ToSingle(yangichapkuz) > Convert.ToSingle(eskichapkuz))
                {
                    label7.Text = (Convert.ToInt32((Convert.ToSingle(eskichapkuz) /
Convert.ToSingle(yangichapkuz))*100)).ToString() +"%";
                }
                else
                {
                    label7.Text = (Convert.ToInt32(
(Convert.ToSingle(yangichapkuz)/Convert.ToSingle(eskichapkuz))*100)).ToString() + "%";
```

```
                }


                if (Convert.ToSingle(yangiungkuz) > Convert.ToSingle(eskiungkuz))
                {
                    label8.Text = (Convert.ToInt32((Convert.ToSingle(eskiungkuz) /
Convert.ToSingle(yangiungkuz)) * 100)).ToString() + "%";
                }
                else
                {
                    label8.Text = (Convert.ToInt32((Convert.ToSingle(yangiungkuz) /
Convert.ToSingle(eskiungkuz)) * 100)).ToString() + "%";
                }



                if (Convert.ToSingle(yangiburun) > Convert.ToSingle(eskiburun))
                {
                    label9.Text = (Convert.ToInt32((Convert.ToSingle(eskiburun) /
Convert.ToSingle(yangiburun)) * 100)).ToString() + "%";
                }
                else
                {
                    label9.Text = (Convert.ToInt32((Convert.ToSingle(yangiburun) /
Convert.ToSingle(eskiburun)) * 100)).ToString() + "%";
                }


                if (Convert.ToSingle(yangiogiz) > Convert.ToSingle(eskiogiz))
                {
                    label6.Text = (Convert.ToInt32((Convert.ToSingle(eskiogiz) /
Convert.ToSingle(yangiogiz)) * 100)).ToString() + "%";
                }
                else
                {
                    label6.Text = (Convert.ToInt32((Convert.ToSingle(yangiogiz) /
Convert.ToSingle(eskiogiz)) * 100)).ToString() + "%";
                }

                eskirasmpic.Image = System.Drawing.Image.FromFile(Application.StartupPath
+ "/TrainedFaces/" + eskirasm);

                label5.Text = eskivaqt;
            }
            reader1.Close();

        }
    }
}
```