Faculty of Mathematics and Information Science Warsaw University of Technology

Biometric fusion system for human recognition using face and voice

prepared by:
Kuśmierczyk Aleksander
Sławińska Martyna
Żaba Kornel

Version no. 1.2.1

Date: 05.12.2017

History of changes

Date	Author	Description	Version
14.10.2017	Kuśmierczyk Aleksander Sławińska Martyna Żaba Kornel	First version	1.0
19.10.2017	Kuśmierczyk Aleksander Sławińska Martyna Żaba Kornel	Glossary added	1.0.1
15.11.2017	Kuśmierczyk Aleksander Sławińska Martyna Żaba Kornel	Technical project	1.1
25.11.2017	Kuśmierczyk Aleksander Sławińska Martyna Żaba Kornel	Thesis' projects division	1.2
05.12.2017	Kuśmierczyk Aleksander Sławińska Martyna Żaba Kornel	Minimum Distance Classifier added	1.2.1

Table of Contents

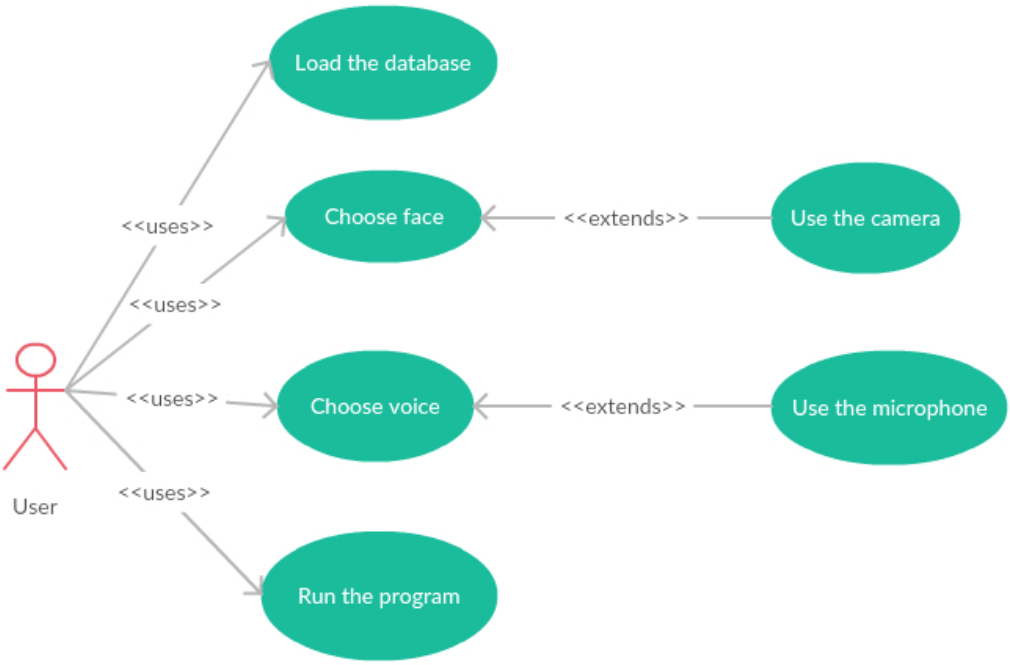
1 Specification

1.1 Executive summary

This system is designed for identification and/or verification of human. With the use of biometric algorithms based on extracting face and voice features, the program recognizes the user with the sufficient degree of compliance. The acquisition of the aforementioned data is done by the portable or built-in camera with a microphone.

1.2 Functional requirements

1. The system should display graphical user interface with options.
2. The system should allow the user to choose what to identify/verify first: face or voice.
3. The system should identify and/or verify the face:
   3.1. The system should use the camera for capturing the image of the face.
   3.2. The system should compare PV's biometric with the biometrics stored in the database.
4. The system should identify and/or verify the speech/voice:
   4.1. The system should use the microphone for recording the speech.
   4.2. The user should say the specific word in order for the system to identify his/her voice.
   4.3. The system should compare PV's biometric with the biometrics stored in the database.
5. After collecting PV's face and speech data, the system should compare collected results with the database and give an answer whether the PV already belongs to the database collection (verifying stage).



Actor	Name	Description	System response
User	Load the database	Load the database for machine learning purposes	The system loads the database from location pointed by the user.
	Choose face	Choose face for identification	The program starts the camera
	Choose voice	Choose voice for identification	The program starts the microphone.
	Run the program	Run the program which compares the PV's data with the database collection	The system returns the result of recognition.
	Use the camera	Use the camera for capturing the face of PV	The system captures the face of PV with camera.
	Use the microphone	Use the microphone for recording the speech of PV	The system records the voice of PV with microphone.

User stories

1. Attempt of PV verification

1.1 The user starts the program, chooses which biometric should be used in recognition process as first. Then the user uses the appropriate hardware to capture the required input or uses the previously prepared data for the program to use in order to recognize the PV. The user reads the results of the verification.

1.3 Non-functional requirements

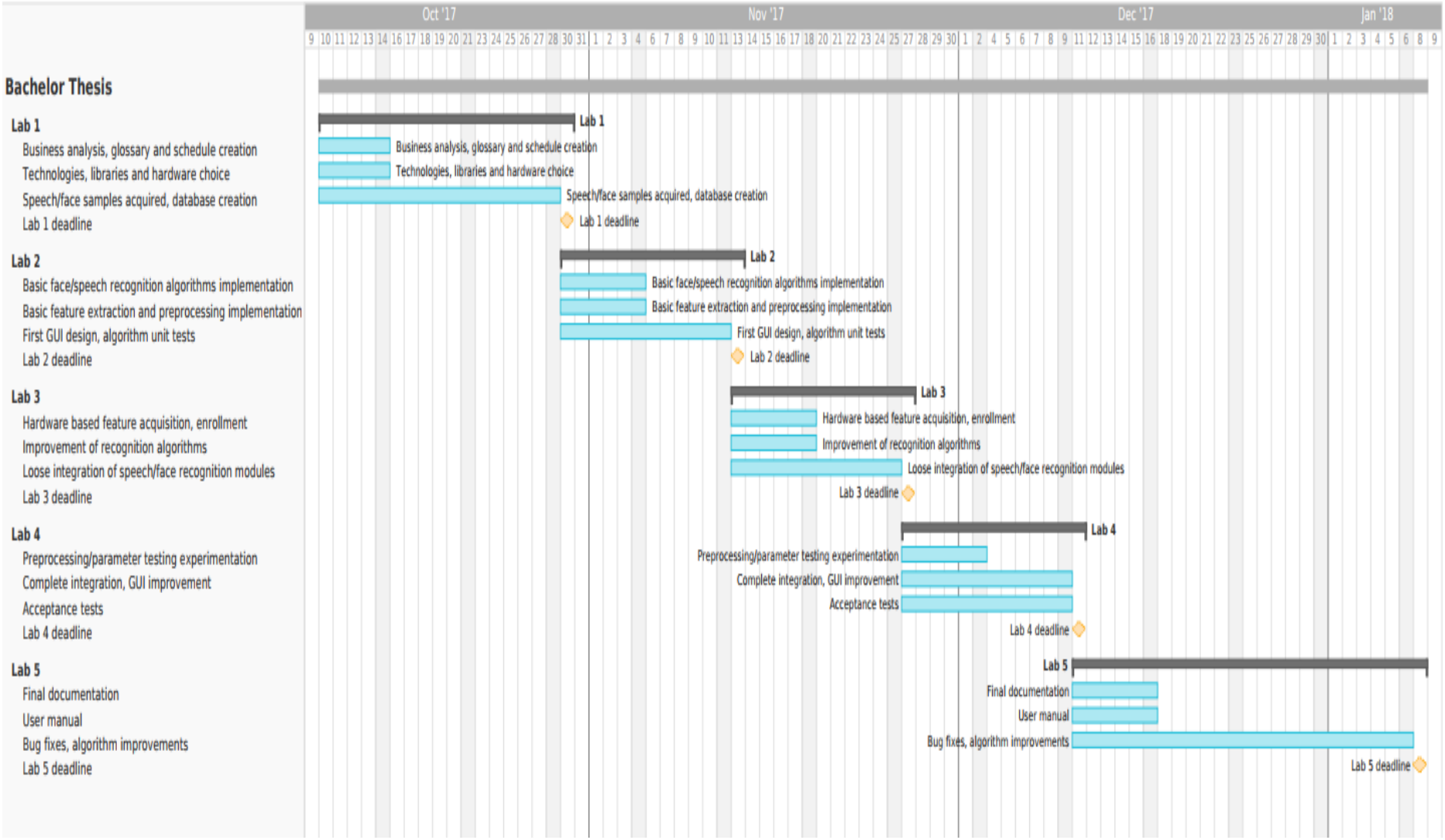
1. The system does not have any login options.
2. The system allows identifying one face/voice at the time.
3. The system should support the Windows operating system.
4. The system should work in an offline mode.
5. The system response time depends on the size of the database.
6. The system's hardware connects by USB cable: camera and microphone, or is built-in the workstation.

Below is the table that describes the URPS requirements of the program.

Requirements	Nr wymagania	Opis
Usability	1	The software is user friendly, as the only requirement for the use of the program is to load the database and to capture the image or to record the speech.
Reliability	2	The system is a desktop application, after manual start-up it is ready to work.
	3	The program handles the incorrect data types in the database, and shuts down if a hardware malfunction occurs.
Performance	4	The system returns the verification result with the delay basing on the database size.
	5	The program verifies the PV using two biometric features, thus achieves the sufficiently good verification correctness.
Supportability	6	In order to use the application, the user is required to have Windows operating system.
	7	The program requires a built-in or a USB-connected camera with a microphone.
	8	In order for the program to function correctly, it requires a database consisting of a large enough samples of face images and voice recordings.

1.4 Project schedule

DEADLINE	MILESTONE	LAB DATE
2017-10-14	Business analysis, glossary and schedule creation	LAB 1 30.10.2017: Business analysis, Glossary, Supplementary spec, Schedule
2017-10-14	Technologies, libraries and hardware choice	
2017-10-28	Speech/face samples acquired, database creation	
2017-11-04	Basic face/speech recognition algorithms implementation	LAB 2 13.11.2017: Technical project, Choice of development model
2017-11-04	Basic feature extraction and pre-processing implementation	
2017-11-11	First GUI design, algorithm unit tests	
2017-11-18	Hardware based feature acquisition, enrolment	LAB 3 27.11.2017: Source code of non-integrated modules, unit tests
2017-11-18	Improvement of recognition algorithms	
2017-11-25	Loose Integration of speech/face recognition modules.	
2017-12-02	Pre-processing/parameter testing experimentation	LAB 4 11.12.2017: Integrated modules, smoke test, acceptance tests
2017-12-09	Complete integration, GUI improvement	
2017-12-09	Acceptance tests	
2017-12-16	Final documentation	LAB 8.01.2018: Working v.1.0, user manual, code/docs, acceptance tests results, list of changes
2017-12-16	User manual	
2018-01-06	Bug fixes, algorithm improvements	
2018-01-06	Others	


Oct '17
Nov '17
Dec '17
Jan '18
Bachelor Thesis
Lab 1
Business analysis, glossary and schedule creation
Technologies, libraries and hardware choice
Speech/face samples acquired, database creation
Lab 1 deadline
Lab 2
Basic face/speech recognition algorithms implementation
Basic feature extraction and preprocessing implementation
First GUI design, algorithm unit tests
Lab 2 deadline
Lab 3
Hardware based feature acquisition, enrollment
Improvement of recognition algorithms
Loose integration of speech/face recognition modules
Lab 3 deadline
Lab 4
Preprocessing/parameter testing experimentation
Complete integration, GUI improvement
Acceptance tests
Lab 4 deadline
Lab 5
Final documentation
User manual
Bug fixes, algorithm improvements
Lab 5 deadline

1.5 Glossary

Biometric – feature owned by human, which allows verification of an individual

Enrolment – process of acquisition of data of the new PV

Face features – face biometric

Feature acquisition – process of collecting data

Feature extraction – process of extracting biometric information from data

Identification – the automatic identification of living individuals by using their physiological and behavioural characteristics

Offline mode – the mode of program, which does not require internet connection

PV – person being verified

Recognition - refers to the automated recognition of individuals based on their biological and behavioural traits

Verification - an identity authentication process used to confirm a claimed identity through uniquely identifiable biological traits

Voice features – voice biometric

CDF - Cumulative Distribution Function

DTW - Dynamic Time Warping

FT - Fourier Transform

MFCC - Mel Frequency Cepstral Coefficients

DCT - Discrete Cosine Transform

2 Technical project

2.1 Selected technology

The technology we have chosen for our project is the **C# .NET 4.6** due to the familiarity with this framework of our group. Furthermore, the availability of the online sources of information for this technology allowed us to search for the solutions of encountered implementation problems in the web. Throughout the work it has proven to be stable and provided useful functionalities.

For the development and management of the database used in this project we have chosen **MSSQL Server 2012**. This platform has proven to be stable and using the 2012 version allowed us to find multiple solutions to the problems we have encountered during the work over the project. Besides, our group is familiar to it's structure due to the previous experience with this program.

For capturing of images and the recording of sound we have chosen to use external program - **FFmpeg** as it provided us with easy to adapt commands for the aforementioned activities. As hardware solution we have chosen Logitech HD Webcam C270 as it provides the necessary video and audio data required by the project.

2.2 Development model

The goal of our project is to provide the method of identification and/or verification of a person, the study over this subject is still being developed, thus the model we had to adapt for its creation had to be flexible and allow us for frequent theoretical discussion. The development technique we have chosen is the **iterative waterfall model** with the elements of extreme programming. Such choice is based on the need of flexibility for implementation and theoretical discussion, which this model provides, as well as the constant assessment of the chosen methods and their adaptation. All the methods implemented are tested afterwards and the results are discussed with the supervisor in order to gain more understanding over them and to establish further direction for the project. Elements of extreme programming we have adapted to our model were the periodic replacement of the roles of people in the team. Such decision is made in order to allow all the members to follow and to understand the theoretical reasons behind the decisions over the methods of human recognition. Furthermore, this supports better quality of the code produced and allowed for frequent discussion over the problems encountered during the development process.

2.3 System architecture

System architecture is composed of layered modules, each representing different responsibility.

List of modules:

Common - module responsible for database access, as well as containing utility classes used by multiple of below modules.

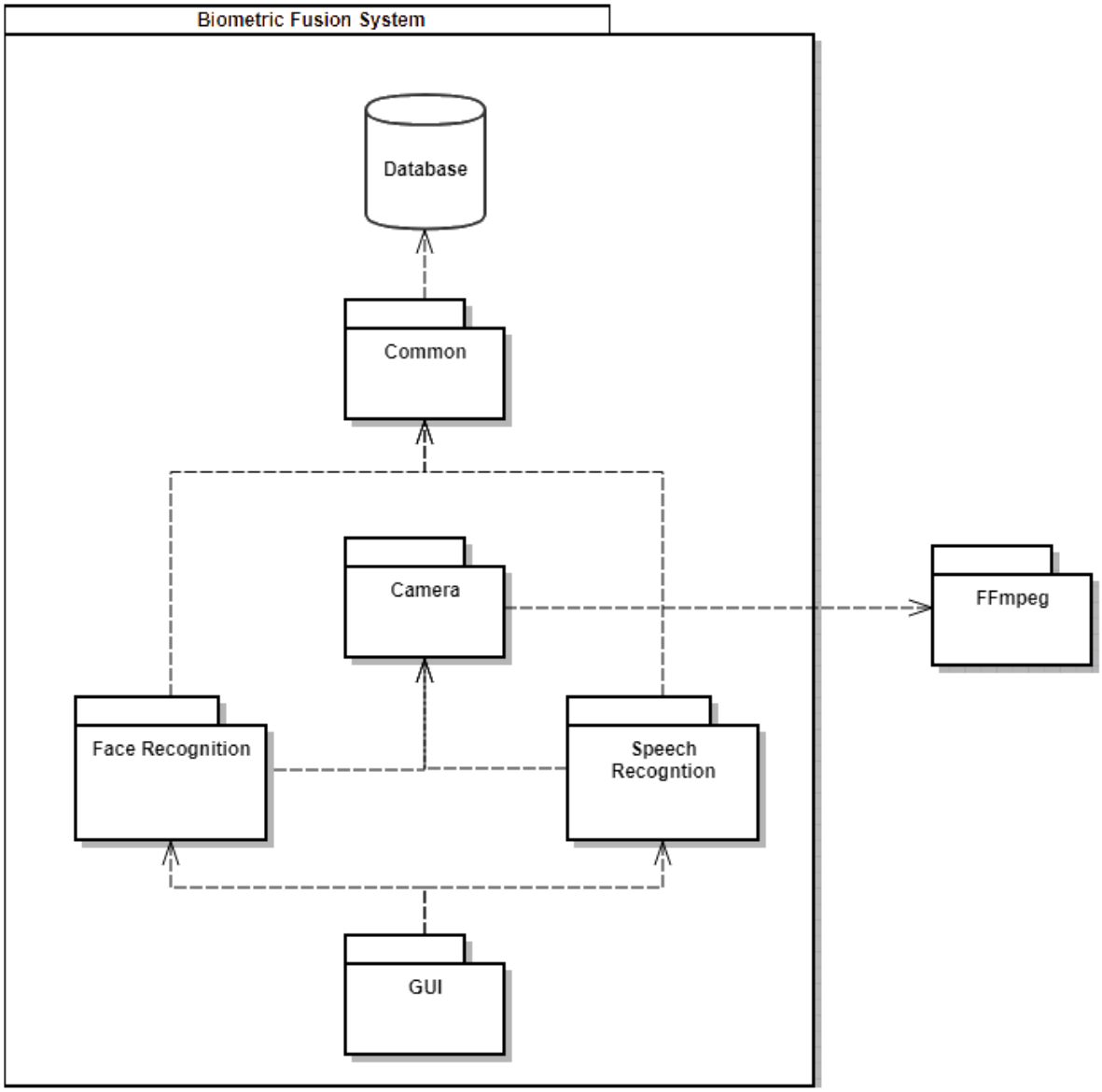
Camera - module responsible for acquisition of the face images and speech recordings. Depends on the scriptable, external video and audio recording program **FFmpeg**.

FaceRecognition - module containing algorithms for face image preprocessing, feature extraction and recognition.

SpeechRecognition - module containing algorithms for speech recording preprocessing, feature extraction and recognition.

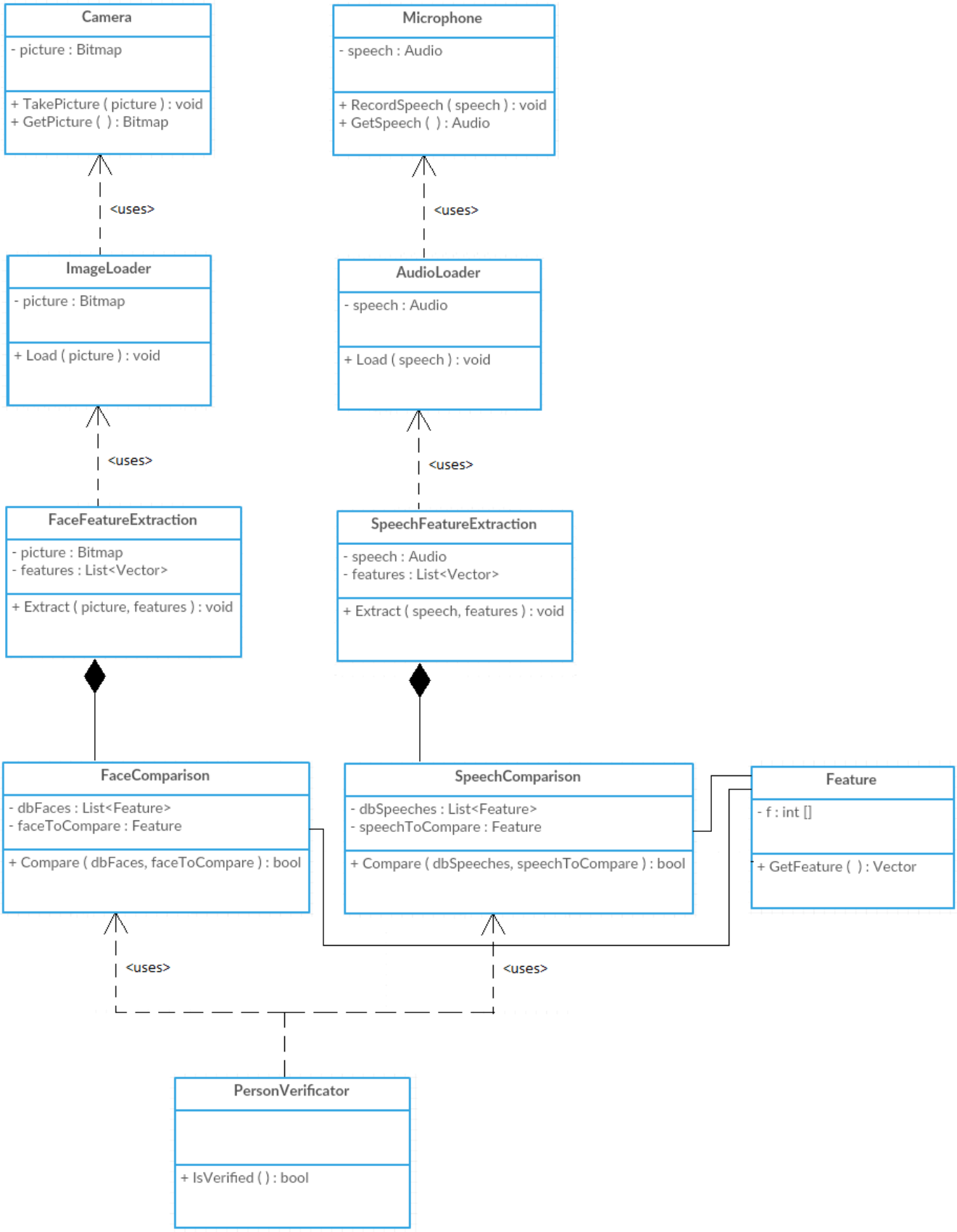
Gui - module defining user interface and interaction. Depends on the .NET desktop application framework **Windows Presentation Foundation.**

Package diagram of system modules:



2.4 Class diagram

The below picture represents the class diagram of the whole system.


Camera
- picture : Bitmap
+ TakePicture (picture) : void
+ GetPicture ( ) : Bitmap
Microphone
- speech : Audio
+ RecordSpeech (speech) : void
+ GetSpeech ( ) : Audio
<uses>
<uses>
ImageLoader
- picture : Bitmap
+ Load (picture) : void
AudioLoader
- speech : Audio
+ Load (speech) : void
<uses>
<uses>
FaceFeatureExtraction
- picture : Bitmap
- features : List<Vector>
+ Extract (picture, features) : void
SpeechFeatureExtraction
- speech : Audio
- features : List<Vector>
+ Extract (speech, features) : void
FaceComparison
- dbFaces : List<Feature>
- faceToCompare : Feature
+ Compare (dbFaces, faceToCompare) : bool
SpeechComparison
- dbSpeeches : List<Feature>
- speechToCompare : Feature
+ Compare (dbSpeeches, speechToCompare) : bool
Feature
- f : int []
+ GetFeature ( ) : Vector
<uses>
<uses>
PersonVerificator
+ IsVerified ( ) : bool

3 Thesis division into projects

3.1 General division into projects

Classes description of *Camera* project:

- **WavFile** - class containing the structure of wav. files: Header, Left channel and Right channel
- **WavHeader** - class containing the information of the wav. file stored in the header of the file.
- **WavReader** - class which handles reading of the speech input data.

Classes description of *Common* project:

- **DbConnection** - class containing the information necessary for establishing connection with the database: Server, Data Source, Initial Catalog, User ID and Password to the corresponding user.
- **FaceRepository** - class responsible for the storing and recovering face biometric data from the database.
- **SpeechRepository** - class responsible for the storing and recovering of speech biometric data from the database.
- **Person** - class containing the information of the person whose data is to be used for the verification/identification process.
- **IVerifier** - Interface defining the function which verifies the input against the data stored in the database.
- **FourierTransform** - class which applies the ***FT*** onto the samples using the implementation in the MathNet package.

Classes description of *FaceRecognition* project:

- **INormalization** - interface that stores definition of a function Normalize which takes a bitmap as an argument and normalizes it.
- **Histogram** - class that contains a function GetHistogram which returns the three channel color histogram of a bitmap.
- **HistogramEqualization** - class that implements the interface INormalization, the function Normalize is implemented such that it returns the bitmap after histogram equalization (bitmap with greater contrast).
- **GrayscaleConverter** - class that implements INormalization interface, the function Normalize converts the image into the grayscale image.
- **GaborFilter** - class with functions for applying Gabor filter to the bitmap which is an edge detection filter.
- **HistogramFeatureExtraction** - class for creating the face feature vectors basing on the color histograms and calculating moments from those histograms.
- **GaborFilterMagnitudes** - another class for creating the face feature vectors basing on the Gabor filter and calculating mean, standard deviation and skewness using Gabor filter functions.
- **FaceFeatureExtractor** - class that utilizes **HistogramFeatureExtraction** and **GaborFilterMagnitudes**, and combines the final face feature vector from feature vectors calculated in those two classes.
- **MinimumDistanceClassifier** - class used for the verification of the input against the database.

Classes description of *SpeechRecognition* project:

- **Frame** - class containing partial list of samples from given *.wav* file.
- **FrameMaker** - class dividing stream of samples from given .wav file into frames of equal length given length of frame and interval between frames in ms.
- **IWindow** - interface defining function applying a window function to a given frame.
- **GaussWindow** - class applying Gauss window to a given frame, implements **IWindow** interface.

- **HammingWindow** - class applying Hamming window to a given frame, implements **IWindow** interface.
- **Periodogram** - class calculating power spectrum coefficients of a given frame.
- **MelConverter** - class converting given frequency in Hertz to Mel scale and back.
- **MelFilterbank** - class generating filters for given range of frequencies and sample rate and computing a set of ***MFCC*** for generated set of filters and spectral density coefficients of given frame.
- **SpeechFeatureExtractor** - class extracting a feature vector from a given list of frames. Utilizes a **IWindow** implementation, **Periodogram** and **MelFilterbank** to generate a combined feature vector of ***MFCC*** of each frame.
- **DynamicTimeWarping** - class used for the verification of the input against the database using the ***DTW*** algorithm with a given threshold.

Classes description of *UnitTests* project:

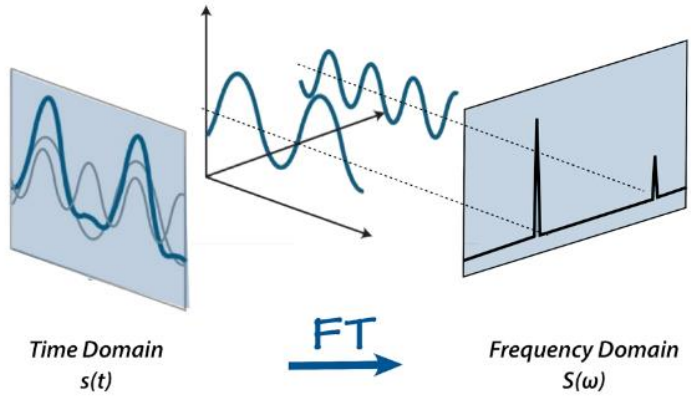
- **DynamicTimeWarpingTest** - set of tests checking correctness of ***DTW***.
- **FrameMakerTest** - set of tests checking correctness of frame generation from the audio recording.
- **GaborFilterTest** - set of tests checking correctness of the Gabor filter algorithm.
- **HistogramTest** - set of tests checking correctness of histogram creation.
- **MelConverterTest** - set of tests checking correctness of conversions between Mel and Hertz.
- **MelFilterbankTest** - set of tests checking correctness of ***MFCC*** generation algorithm.
- **PeriodogramTest** - set of tests checking correctness of periodogram coefficients calculation.
- **WindowFunctionTest** - set of tests checking correctness of calculation of Hamming window and Gauss window.

Classes description of *AlgorithmTests* project:

- **SpeechAlgorithmTest** - set of tests checking for successful identification of an input .wav file against set of template .wav files. Each file consists of a recording of a single word '*kurczak*'.
  Files: "aleks_k.wav", "aleks_k2.wav", "aleks_k3.wav",
  "aleks_k4.wav", "kornel_k.wav", "martyna_k.wav"
  Each test checks if different combination of "aleks" template and input recording will be correctly verified.
- **FaceAlgorithmTest** - set of tests checking for successful identification of an input .bmp file against set of template .bmp files. Each file contains a picture of a face.
  We have two pictures of person1 (input and template), one picture of person2 and one picture of person3. We compare the similarities between person1 (input) and
  - person1 (template),
  - person2 picture,
  - person3 picture.

3.2 Classes description in Common project

FourierTransform
Fourier Transform is mapping the signal from time domain to spectral domain.

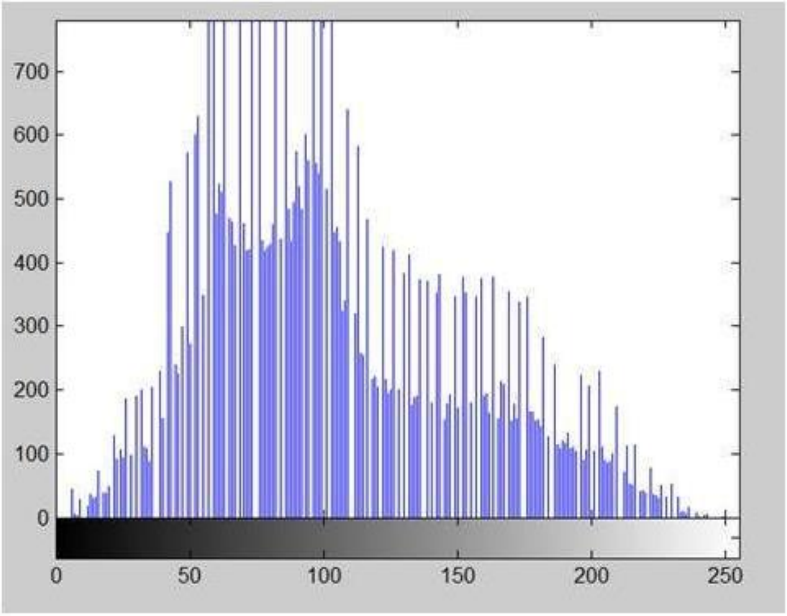


The formula for Fourier Transform calculation is the following:

$$x_k = \sum_{n=0}^{N-1} x_n \cdot e^{-i2\pi kn/N} = \sum_{n=0}^{N-1} x_n \cdot [cos(2\pi kn/N) - i \cdot sin(2\pi kn/N)]$$

3.3 Classes description in FaceRecognition project

Histogram
For every channel, distribution of intensities is calculated.
Sample grayscale image histogram:



HistogramEqualization
Histogram equalization calculates the ***CDF*** for the color channel values, which is also the image's accumulated normalized histogram and applies to the picture. Such operation increases the global contrast for the image, as the intensities of the color level are better distributed on the histogram.
CDF is calculated with the following formula:

$$cdf_x(i) = \sum_{j=0}^{i} p_x(j)$$

where $p_x(i)$ is the probability of an occurrence of a pixel of level i in the image and is calculated with the following formula:

$$p_x(i) = p(x = i) = \frac{n_i}{n}, 0 \leq i < L$$

where L is the number of color levels in the channel, n is the total number of pixels in the image and n_i is the number of occurrences of the color level i.
In order to apply this method back to the original picture the final function is used:

$$y = \lfloor cdf(i) * ((max\{x\} - min\{x\}) * min\{x\}) \rfloor$$

where y is the new pixel color level for the channel x from the original color level i

The result of using HistogramEqualization implemented in the project:

before after

GrayscaleConverter
In order to transform the image into grayscale image, the function Normalize loops through every pixel of the image and sets the value for each channel of given pixel to the average value of values of all channels of that pixel.

The pseudocode is the following (actions done for each pixel):
channelValue = (pixel.Red + pixel.Green + pixel.Blue) / 3
pixel.Red = channelValue
pixel.Green = channelValue
pixel.Blue = channelValue

GaborFilter
Gabor filter is an edge detection filter. The edges of an image are detected from different orientations, and next they are placed on top of one another to get the final image with detected edges.

The formula used to calculate Gabor filter is the following:

$$G_{\sigma_y,\sigma_x,f_i,\theta_k}(x,y) = e^{-[\frac{x_{\theta_k}^2}{\sigma_x^2} + \frac{y_{\theta_k}^2}{\sigma_y^2}]} \cdot cos(2\pi f_{ix\theta_k} + \phi)$$

where

$$x_{\theta_k} = x \cdot cos\theta_k + y \cdot sin\theta_k$$

$$y_{\theta_k} = y \cdot cos\theta_k - x \cdot sin\theta_k$$

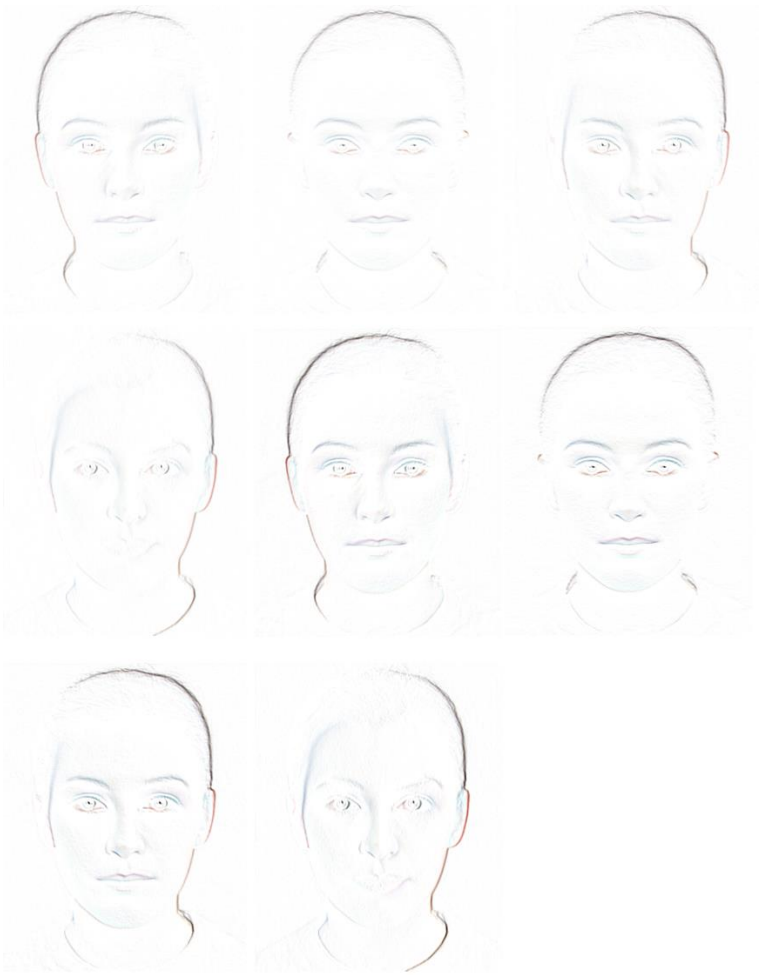
$$\phi = \pi/2$$

$$\theta_k = \frac{k\pi}{n}$$

The next step is to calculate the kernel by applying the formula to particular elements of the kernel:

$$kernel[x, y] = e^{-[\frac{x_{\theta_k}^2}{\sigma_x^2} + \frac{y_{\theta_k}^2}{\sigma_y^2}]} \cdot cos(2 \cdot \pi \cdot \frac{x_{\theta_k}}{waveLength} + \phi)$$

Finally, applying the kernel to the image produces the following images:



After placing them one on top of another, we obtain final image after Gabor filtering:

HistogramFeatureExtraction
Color moment generating formula used in HistogramFeatureExtraction:

$$M_k^1 = \frac{1}{XY}\sum_{x=1}^{X}\sum_{y=1}^{Y} f_k(x,y) \quad \text{for the first order}$$

$$M_k^h = (\frac{1}{XY}\sum_{x=1}^{X}\sum_{y=1}^{Y}(f_k(x,y) - M_k^1)^h)^{\frac{1}{h}} \quad \text{for any subsequent order defined by h}$$

k - represents the k-th color component.
h - order of the moment
$f_k(x, y)$ - the color value of the k-th color component in the pixel at the coordinates of (x,y)
The moments generated by the aforementioned functions are: mean for the first order, variance for the second order and skewness for the third order, all efficiently describe the color distributions of images[http://shonen.naun.org/multimedia/NAUN/bio/bio-2.pdf]

GaborFilterMagnitudes
This class calculates the Gabor filter magnitudes which are used as a part of face feature vector.

The magnitudes (mean, standard deviation, skewness) are calculated from the following formulas:

$$\mu = \frac{1}{XY}\cdot\sum_{x=1}^{X}\sum_{y=1}^{Y} imageAfterGaborFiltering(x,y)$$

$$std = \sqrt{\frac{1}{XY}\cdot\sum_{x=1}^{X}\sum_{y=1}^{Y} ||imageAfterGaborFiltering(x,y)| - \mu|^2}$$

$$skew = \frac{1}{XY}\cdot\sum_{x=1}^{X}\sum_{y=1}^{Y} (\frac{imageAfterGaborFiltering(x,y) - \mu}{std})^3$$

FaceFeatureExtractor

It combines the feature vectors obtained from the classes HistogramFeatureExtraction and GaborFilterMagnitudes into one face feature vector.

Minimum Distance Classifiers
This method analyses the numerical face feature vectors and calculates the Euclidean distance between them in order to further use the calculated distance in determining how similar the given vectors are.

The formula for calculating the Euclidean distance:

$$\mathrm{d}(\mathbf{q},\mathbf{p}) = \sqrt{(q_1-p_1)^2+(q_2-p_2)^2+\cdots+(q_n-p_n)^2}$$

3.4 Classes description in SpeechRecognition project

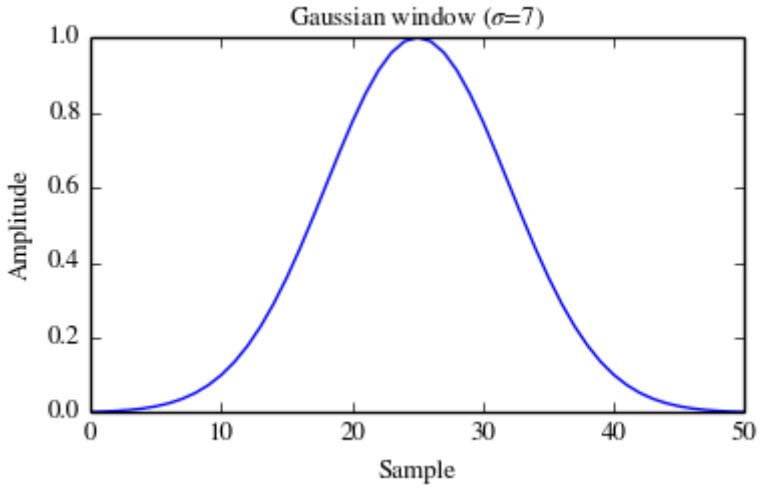
FrameMaker
FrameMaker divides a stream of sampled voice recording into frames of equal length.
Length of frame in samples is obtained by the following formula:

$F_{len} = t_{len} \cdot S_r t_(len)$ = frame length in seconds S_r = sample rate of recording

Interval between frames is obtained by the following formula:

$F_{int} = t_{int} \cdot S_r t_(int)$ = frame interval in seconds S_r = sample rate of recording

GaussWindow
Gauss Window goes over each frame of the audio recording, amplifies the middle of the frame and weakens the beginning and end of the frame.



HammingWindow
Hamming Window goes over each frame of the audio recording, amplifies the middle of the frame and weakens the beginning and end of the frame.

The formulas used:

$$w(n) = \alpha - \beta \cdot cos(\frac{2\pi n}{N-1})$$

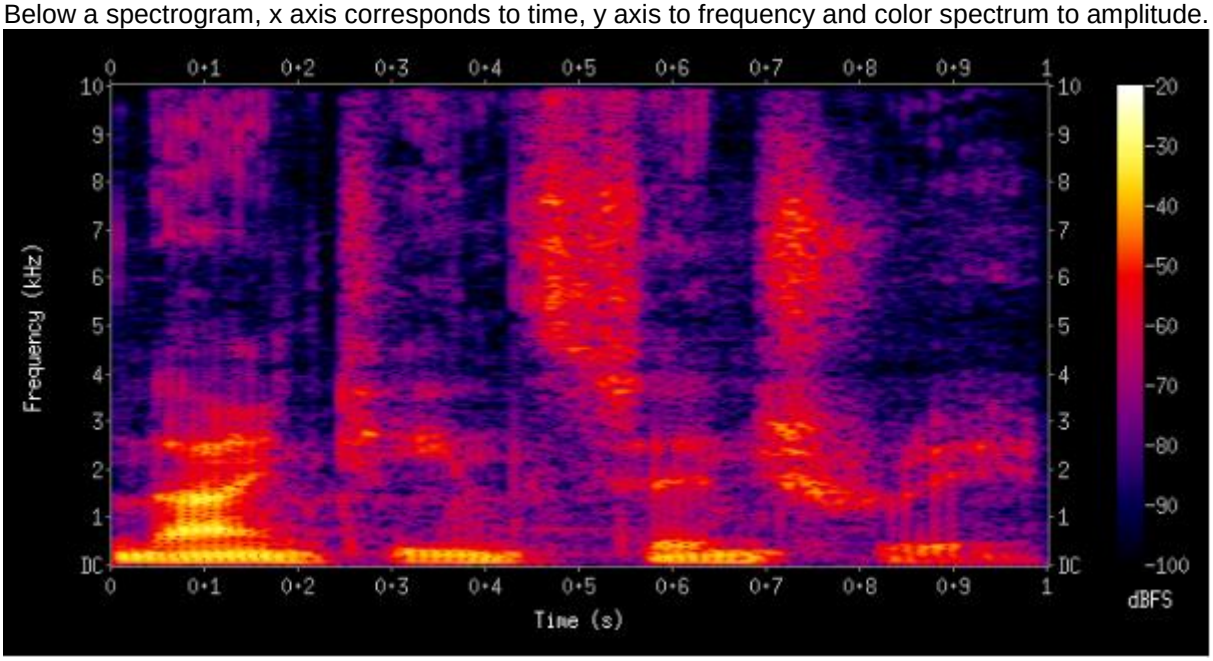
n=sample in a frame N=number of samples in a frame

α = 0.53836
β = 0.46164

Periodogram
Periodogram applies ***FFT*** to a *n* long Frame, mapping the signal from time to spectral domain, resulting in *n* complex coefficients. Afterwards spectral density estimates, which describe strength of signal given the frequency, are calculated from first half of ***FFT*** coefficients.
Spectral density estimation of a single coefficient is given by the following formula:

$$P(c_i) = \frac{|c_i|^2}{N} c_i$$ = *i*-th coefficient of ***FFT*** N = amount of samples in a frame

Below a spectrogram, x axis corresponds to time, y axis to frequency and color spectrum to amplitude.



MelConverter
Mel Converter converts the Mel units into Hertz, and Hertz into Mel.
Formula for converting Mel into Hertz:

$$M^{-1}(m) = 700 \cdot (e^{m/1125} - 1)$$

Formula for converting Hertz into Mel:

$$M(f) = 1125 \cdot ln(1 + \frac{f}{700})$$

MelFilterbank
Mel filterbank computes a set of ***MFCC*** for a generated set of filters and spectral density coefficients of given frame.
1. Generation of filters:

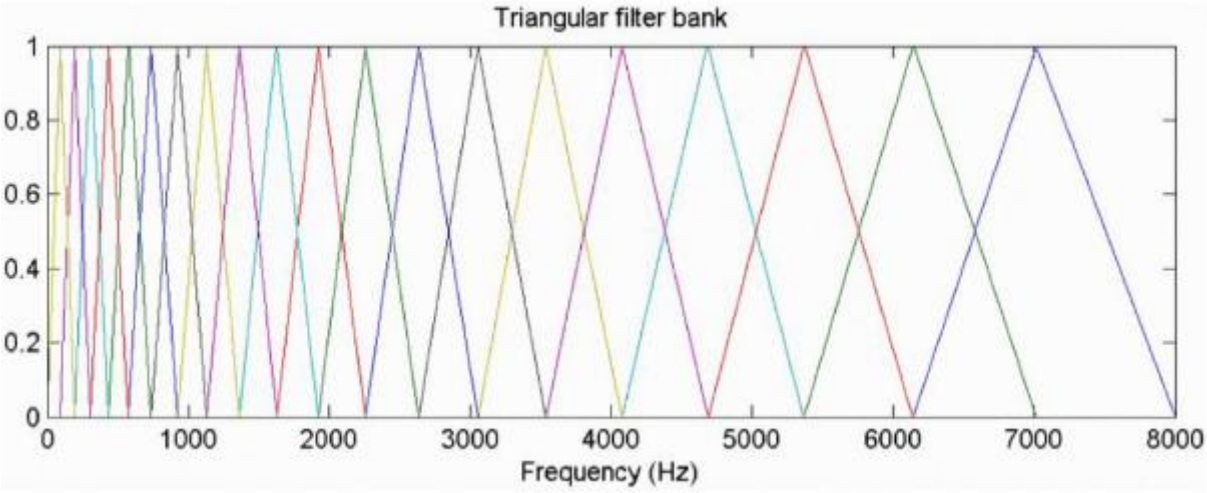
- For a given frequency range in Mel scale and n amount of filters, *n + 2* linearly spaced filter intervals are generated
- Filter intervals are converted back to Hertz scale.
- Corresponding to the number of coefficients of ***FFT*** used in **Periodogram,** *n + 2* filter intervals are generated.

$$F_i = \lfloor \frac{(2 \cdot L + 1) \cdot f_i}{S_r} \rfloor$$

L - number of coefficients of ***FFT*** f_i - *i*-th filter interval in Hertz scale
S_r - sample rate of the recording

- Finally, a filterbank of *n* vectors of triangular filters is computed according to the below formula:
  Formula for obtaining *k*-th filter*:*
  For *i* from *0* to *max(F)* where *F* is set of ordered filter intervals:

$$Filter_k(i) = \begin{cases} 0 & i < F_{k-1} \vee i > F_{k+1} \\ \frac{i - F_{k-1}}{F_k - F_{k-1}} & F_{k-1} \leq i \leq F_k \\ \frac{F_{k+1} - i}{F_{k+1} - F_k} & F_k \leq i \leq F_{k+1} \end{cases}$$



2. Generation of ***MFCC***
   - For each filter in a filterbank, the energy in the filter is obtained by the dot product between a filter vector and a vector of spectral estimates of a single frame.
   - Because humans perceive loudness in logarithmic scale, logarithm of filter energies is taken.
     $E_k = \ln(Filter_k \circ P_{frame}) P_{frame}$ - spectral density estimates vector
     E_k - energy of *k*-th filter

Finally, a ***DCT*** of energies of a single frame is computed, resulting in a set of *n* real valued ***Mel Frequency Cepstral Coefficients***.

SpeechFeatureExtractor
Speech Feature Extractor follows the steps below to obtain a feature vector for a single voice recording.
1. Framing of the signal:
   - Speech Feature Extractor feeds a vector of signal samples into the **FrameMaker**, obtaining back a set of equally spaced intersecting frames.
2. Windowing the signal:
   - In preparation for ***FFT****,* to each frame a window (**Gauss** or **Hamming**) function is applied.
3. Obtaining spectral estimates by **Periodogram**:
   - Each frame is subjected to ***FFT*** and a set of spectral density estimates is obtained.
4. Creation of a **Mel filterbank**:
   - For given length of ***FFT*** and sample rate, a filterbank of triangular filters is created.
   - For each frame, its set of spectral density estimates is filtered through the filterbank, obtaining a set of energies left in each filterbank.

- A ***DCT*** of logarithms of energies is computed, resulting in a set of ***Mel Frequency Cepstral Coefficients***.

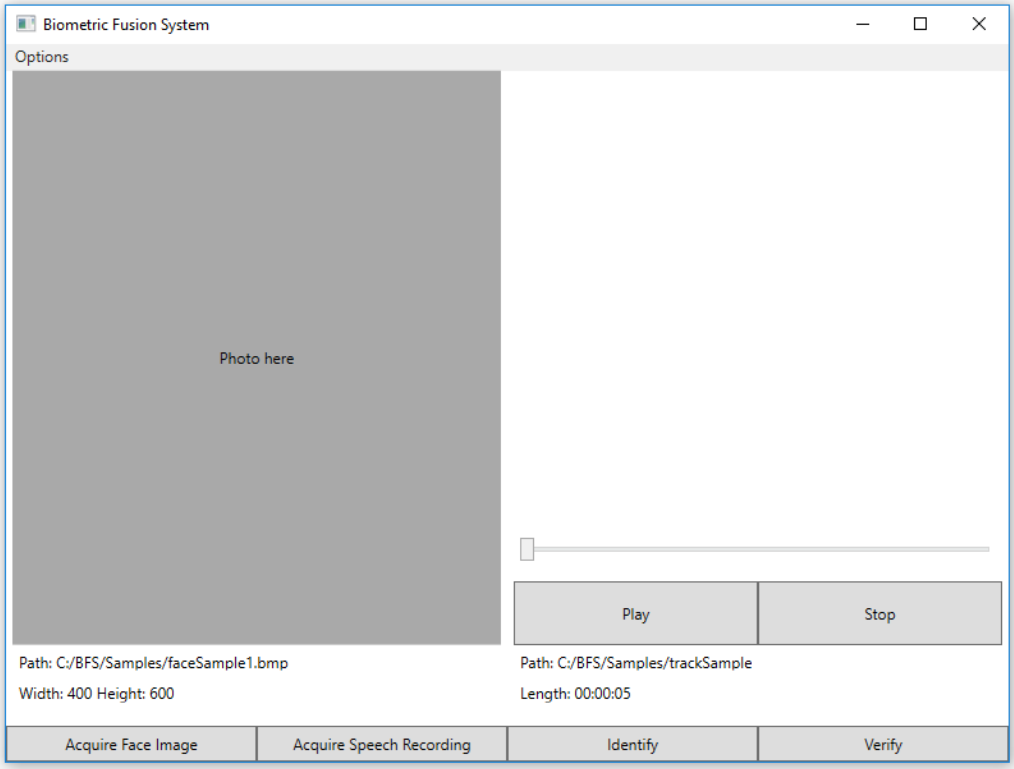
5. Combining the ***MFCC*** sets:
Sets of ***MFCC*** of each frame are concatenated, resulting in a feature vector of the recording.

DynamicTimeWarping
Dynamic Time Warping is used for comparing two sequences of elements that differ in length. It returns the sum of distances between elements of those two sequences.
Pseudocode:

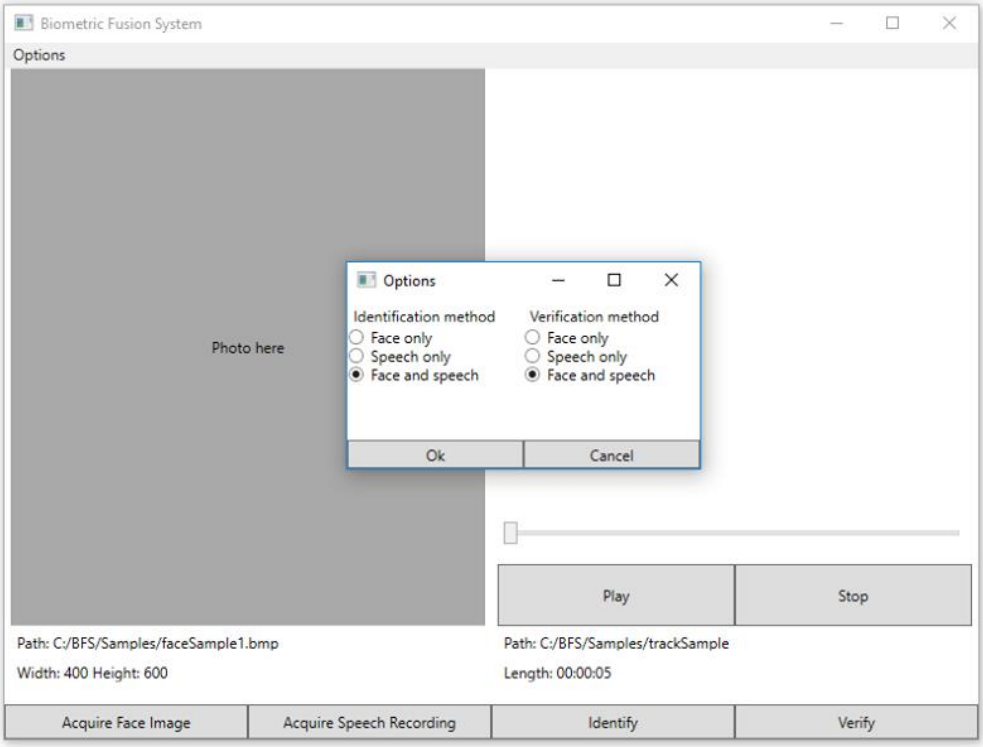
```
int DTWDistance(s: array [1..n], t: array [1..m]) {
  DTW := array [0..n, 0..m]

  for i := 1 to n
     DTW[i, 0] := infinity
  for i := 1 to m
     DTW[0, i] := infinity
  DTW[0, 0] := 0

  for i := 1 to n
     for j := 1 to m
        cost := d(s[i], t[j])
        DTW[i, j] := cost + minimum(DTW[i-1, j  ],    // insertion
                                    DTW[i  , j-1],    // deletion
                                    DTW[i-1, j-1])    // match

  return DTW[n, m]
}
```

3.5 Graphical User Interface

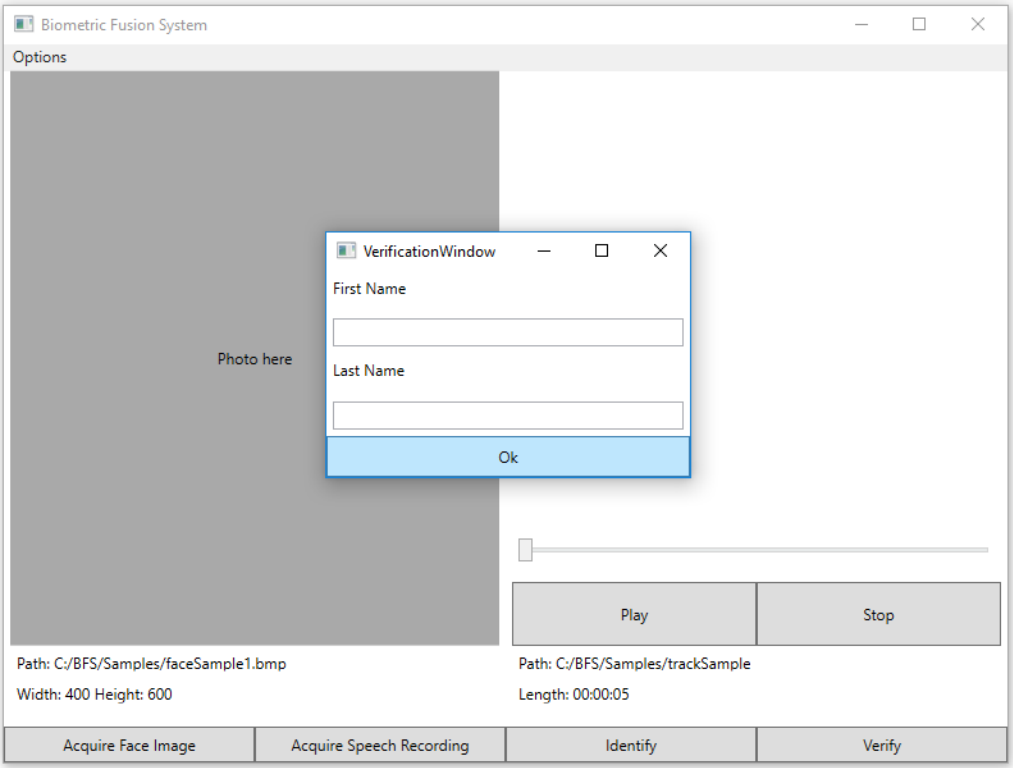
The main window of an application:



The options window:



The verification window:


Biometric Fusion System
Options
Photo here
VerificationWindow
First Name
Last Name
Ok
Play
Stop
Path: C:/BFS/Samples/faceSample1.bmp
Width: 400 Height: 600
Path: C:/BFS/Samples/trackSample
Length: 00:00:05
Acquire Face Image
Acquire Speech Recording
Identify
Verify

4 Bibliography

Choraś R. “Image Feature Extraction Techniques and Their Applications for CBIR and Biometrics Systems”, International Journal of biology and biomedical engineering issue 1, vol. 1, 2007

Rouhi R., Amiri M., Irannejad B. “A review on feature extraction techniques in face recognition”, Signal & Image Processing: An International Journal (SIPIJ) Vol.3, No.6, December 2012

Muda L., Begam M, Elamvazuthi I. “Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques “, Journal of computing, volume 2. Issue 3. March 2010 ISSN 2151-9617

Chakraborty K., Talele A., Upadhya S., “Voice Recognition Using MFCC Algorithm”, International Journal of Innovative Research in Advanced Engineering (IJIRAE) Volume 1 Issue 10 November 2014, ISSN: 2349-2163

Furui S., “Digital Speech Processing”, edition 2, 2001

Internet sources:

“Mel Frequency Cepstral Coefficient (MFCC) tutorial link: http://practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/ (accessed at 25.11.2017)