Francisco José Martínez Bolaños

Using Computer Vision System for Detecting Tiredness

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Information Technology

Thesis

15 May 2015

Helsinki Metropolia University of Applied Sciences

Author(s) Title Number of Pages Date	Francisco José Martínez Bolaños Using Computer Vision System for Detecting Tiredness 33 pages + 1 appendix 15 May 2015
Degree	Information Technology
Degree Programme	Bachelor of Engineering
Instructor(s)	Sakari Lukkarinen, Senior Lecturer

The purpose of the thesis is to demonstrate an application of computer vision systems which are useful nowadays, especially in face recognition. Computer vision systems is a big field with multiple applications. We can compare image processing from its beginning when a camera looks at an image and end with the final processed image with human beings looking at an object and later in both human and machines, using that image, processed in the machines, for recognition of reality.

The reality is perceiving in our minds, through eyes. This process is also happening in the virtual reality of the artificial neurons. It is necessary to develop algorithms for "teaching" the computer system how to see and to act, giving us useful service.
For face recognition the Haar Cascades is used with the OpenCV and EMGU libraries, both open source, with the Windows Visual Studio 2013.

Finally, we have to validate all, and if not good, to change it in brief all we must follow a research method. A confusion matrix will give us the accuracy in order to achieve the validation of the program
The results of face detection and recognitions were successful.

Keywords	Image processing, computer vision, Haar cascades, tiredness recognition.

Contents

Abbreviations

ANN	Artificial Neural Network
CCD	Charge Coupled Device
DIP	Digital Image Processing
ECG	Electrocardiography
EEG	Electroencephalography
Emgu CV	Cross platform .Net wrapper to the OpenCV image processing library
EOG	Electro Oculography
FC	Cardiac Frequency
FN	False Negative
FP	False Positive
GNU	Unix-like computer operating system free software which provides a collection of applications, libraries, and developer tools, plus a program to allocate resources and talk to the hardware (kernel).
HRV	Hearth Rate Variability
IPL	Image Processing Library.
K	Kelvin degrees (color temperature)
LBPH	Local Binary Pattern Histograms
MinGW	Minimalist GNU for Windows
MLL	Machine Learning Library.
MxN	Matrix where the index of the column or row has a numeric element which represents the pixel in the digital image in the rank 0 -255.
Opencv	Open source computer vision library for C and C++
OS	Operative System
PERCLOS	Opening and closing of eyes measurement
RAM	Random Access Memory
RGB	Red, Green and Blue the original colors
ROI	Region of Interest.
TN	True Negative
TP	True Negative
VB	Visual Basic

1 Introduction

In the European Union roads died about 28 000 persons in 2012 and 280.000 were seriously injured. One of the causes was fatigue [28] Another example was a sleeping air traffic controllers in central China and a Boeing 737 was planning in the limbo 12 minutes [40]. It is necessary to take action to avoid events like these.

In the field of Image Processing there are many applications which some years ago sounded like science fiction, for example the detection of fatigue in drivers or pilots.

The aim of the thesis is to use computer vision for the detection of fatigue. With the support of a video image, the amplitude of the eyes opening, the mouth as well as the movements of the head can be detected. This is not only useful to prevent accidents in driving vehicles, or in the piloting aircrafts, but also for medical studies or research. We also need a system that can learn from a dynamic video of a person in order to learn different situations and generate an alarm with the feedback of the video and in real time. This is possible with soft computing and with the Artificial Neuronal Network, a system based on the human neuronal network. Later the data can be collected and send to a central server in order to act if necessary or for future research.

.

2 Fatigue

The fatigue is a lack of energy and motivation and as we can read in the Mayo Clinic web site [1] we can opposite it the word to sleepiness (part of another research [11]) to fatigue. Sleepiness is coming when a person needs to sleep. The sleepiness and the apathy are symptoms of fatigue. There is two main kind of fatigue: Mental and physical fatigue. Physical fatigue is when our muscles cannot do things as easily as they used to. Climb stairs or carrying laden supermarket bags may be much harder than before. Physical fatigue is also known as muscle weakness, weakness, or lack of strength. In brief is this sensation of tiredness. Fatigue can be a response to physical exertion, emotional stress, boredom or when we need to sleep. When we have problem to concentrate in our tasks, not want to get out of bed in the morning, we are speaking of mental fatigue [2]. Mental fatigue often appears together with physical fatigue, but not always. People may feel sleepy, have a decreased level of consciousness, and in some cases show signs similar to that of an intoxicated state. Mental fatigue may be life threatening, especially when the sufferer has to perform some tasks, such as driving a vehicle or operating heavy machinery.

2.1 Physical measures

Based on facial expression there are the physical measures to detect the fatigue [35]. Among the aspects to identify fatigue are nod, yawn, and flicker eyes. We can analyze using video images. Similarly, the presence of fatigue can be validated by the measurement of the duration of eye closure during blinking, with an electro-oculography (EOG). We can also use the electroencephalogram (EEG) and cardiac frequency (FC), which include parameters such as heart rate variability (HRV), respiration, temperature of peripheral skin and blood pressure. [35]

2.2 Time measurements

The opening and closing of eyes, **PERCLOS** is defined as the measurement of the percentage of time the pupils of the eyes are 80% or more occluded over a specified time interval. Fatigue is related to lack of sleep, the diameter of the pupil, the rapid rotation of the eyeball as well as other factors. To define the average distance and the amount of time needed to close or fully open eyes, we can set the speed of opening and closing, of eye. In a sleepy person is

clearly different from an alert person. The eye opening degree is characterized by the shape thereof. It is observed that closing the eyes, they start being covered by the eyelids and shape becomes more elliptical. The cumulative duration of eye closure excluding the time devoted to the normal eye blinking is used to calculate the PERCLOS. The principle of measurement of PERCLOS, as we see in the figure 1, t1 to t4 are used to measure of the opening closing value. In the PERCLOS equation (figure 2) **f** is the final value and t1 and t2 correspond to the times when the eyes are closed from 80% to 20 %, and t2 and t4 time from 20% to 80% of opening [3].

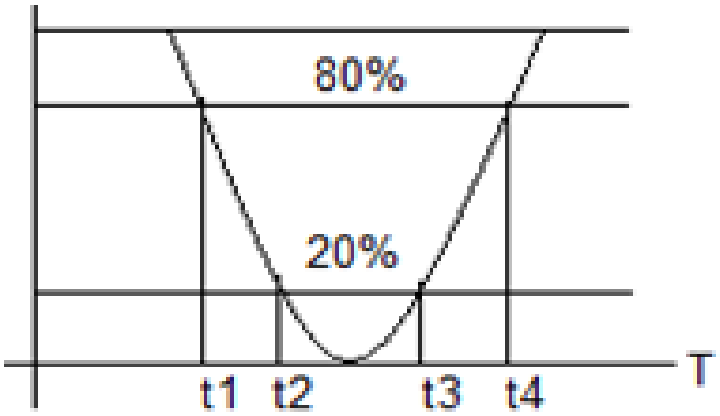


Figure 1. PERCLOS. Times t1 to t4 used for the opening closing. Adapted from [3]

$$f = \frac{t_3 - t_2}{t_4 - t_1} \times 100\,\%$$

Figure 2. PERCLOS equation [3]

We are going to focus in to use the next characteristics to define the fatigue with a no invasive method based on gesture recognition [25]:

- The grade of the eye opening
- The grade of the mouth opening.
- The nodding.
- The yawn.

Through a face analysis with opening and closure of the eyes, with repetitive yawns and the detection of nodding, it is possible to point an alarm over the state of fatigue of a driver or pilot. If we use a video camera as a sensor to get data of the scene where the driver is working, we can extract for each sequence (frames), through process image techniques the opening and closing parameters of the eyes, the nodding and the yawns, to point an automatic process of discrimination of the fatigue state of the driver or the pilot [4].

3 Computer vision

For detecting faces we use computer vision, the science and technology that can "see", and seeing in this case means that a machine is able to extract from an image some information that is necessary for solving some task. As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data [12].

3.1 Charge Couple Device (CCD)

A digital camera captures information through the CCD. Fundamentally, a Charge Coupled Device (CCD) is an integrated circuit etched onto a silicon surface forming light sensitive elements called pixels. The light incident on this surface generate charge that can be read by electronics and turned into a digital copy of the light patterns falling on the device. CCD´s come in a wide variety of sizes and types and are used in many applications from cell phone cameras to high-end scientific applications [14]. In the figure 3 we see the CCD inside of the camera.

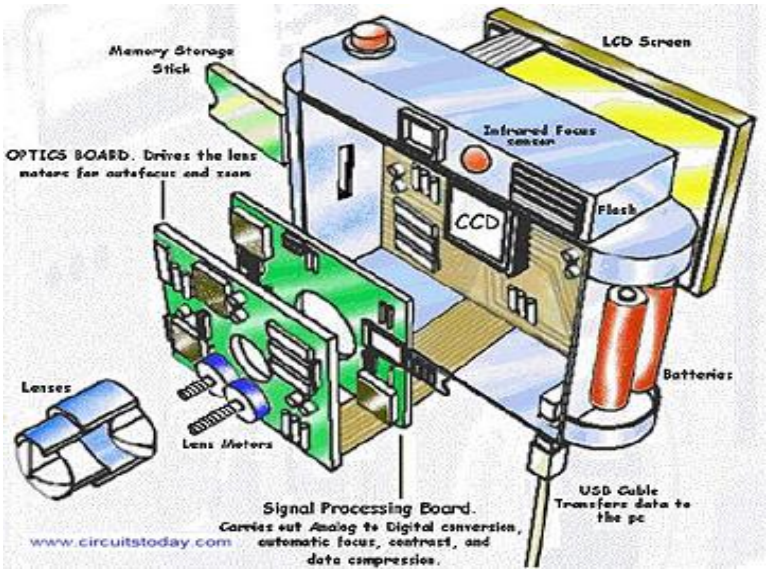


Figure 3. CCD in the hearth of a camera. Copied from [39]

Before reaching the pixel, the light passes through a filter that passes only photons [36] of the desired wavelength. Each pixel can only have a filter and is therefore only sensitive to a color.

The CCD becomes a mosaic of pixels (as we see in the figure 4 with a representation of one pixel) respectively sensitive to red, green and blue. As is logical in the pixel in which collects information from one color, red for example, cannot capture the information of the other colors. The information of color pixels that are not sensitive is deduced by interpolation from the neighboring pixels of that color. Due to the optimal interpolation that it is never real, the images captured with the CCD mosaic give a degree of fuzziness which makes them low quality. One solution to this problem, which is employed in the domestic video cameras and digital photography, is increase the percentage sensitive to green pixels (the human eye is much more sensitive to this color. We can see that in the representation of the pixel with two green squares) so that greens are much less blurring than red or blue and the overall image definition wins [13].

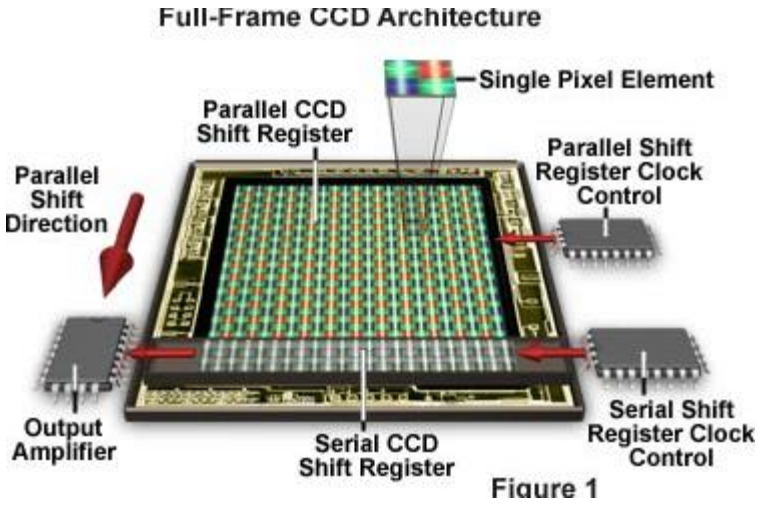


Figure 4. CCD architecture. Copied from [38]

3.2 Digital image

Digital Image is a function of bi-dimensional intensity F(x, y), where x and y are the coordinates, and F it is the intensity of grey. A digital image can be represented with a matrix of **M**x**N** where the index of the column or row has a numeric element which represents the pixel in the digital image in the rank (0 -255). 0 is black and the 255 is white. Figure 5 is a picture of an image and its two-dimensional representation, which is

done with small elements called pixels. They represent the colors, the heights and opacities.

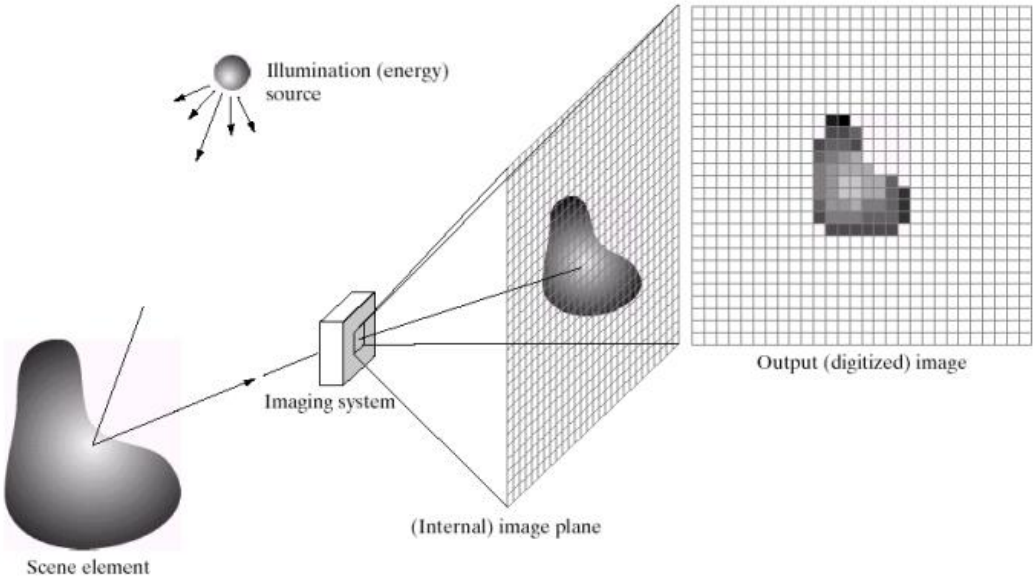


Figure 5 Digital Image. Copied from [23]

When an image is digitalized we get the next elements and properties:

- **Pixel:** The representation of the intensity in a point of the image. (x, y)
- **Adjacency**: Two pixels in adjacency when they shared frontiers and squares.
- **Neighbor pixels**: The pixels are neighbors if they are in adjacency. If they share a frontier they are direct neighbors. If they share a square they are indirect neighbors
- **Neighborhood in a pixel** [22]: A neighborhood in a pixel **p, denoted** as $\vee \mathbf{p}$ is a submatrix $\mathbf{M_{ji}}$ of a size |x|, with J e I odd integers, inside the matrix $\mathbf{I_{MN}}$

$$\vee \boldsymbol{p} = \left\{\boldsymbol{p}: \boldsymbol{p} \in \boldsymbol{M_{ji}}; \boldsymbol{M_{ji}} \boldsymbol{C}\ \boldsymbol{I_{ji}}; \boldsymbol{J} = \mathbf{1} = \{\mathbf{3}, \mathbf{5}, \mathbf{9}\}\right\} \text{ [12]}$$

3.3 Digital Image Processing

Digital Image Processing (DIP) is when manipulate a digital image using a computer [22]. The input is digital image and with the help of some algorithms we get an output of another image. For example when we take an image with a digital camera and download it in our computer and then we use a software in order to edit the image to get another enhanced one. There different edition programs like Adobe Photoshop for photos or Avid, or Final Cut for video. The figure 5 is the DIP.

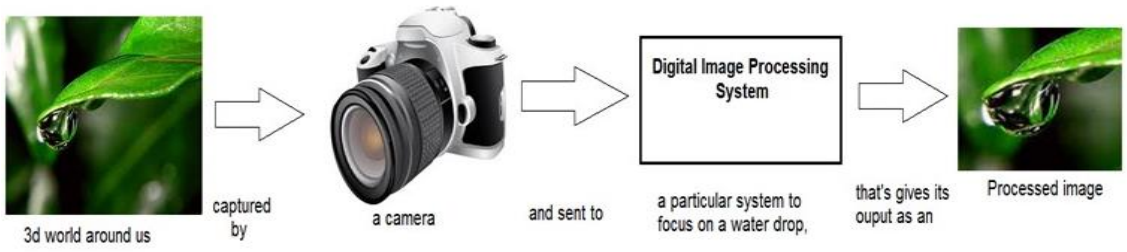


Figure 6. An image being processed. Copied from [27]

We have seen how a camera works and the elements of a digital image. The next step is to recognize a face [19]. It is no easy task. There is an image with millions of pixels and we have to get an algorithm that can detect a face (later also the regions of the eyes and the mount) and quickly. We need an algorithm. The algorithm has to give us every face detected, its situation in the image and to include a box where the eyes, mouth and nose will be. For example the Viola-Jones, which is based on a boosting algorithm to take care of selecting images among the thousands. The Viola-Jones algorithm is based on the Haar descriptors [17].

3.4 Boosting

The classifiers were based on the algorithm of boosting by Freund and Schapire.
Boosting is a machine-learning algorithm based on the idea of converting weak learners into strong ones.
Intuitively, for a learned classifier to be effective and accurate in its predictions it should meet three conditions:

1. it should have been trained on "enough" training examples;

2. it should provide a good fit to those training examples (usually meaning that it should have low training error); and

3. it should be "simple.

With the descriptors and a database of faces and not faces positive faces and negative faces, we use a function for training and learning. Here we have the AdaBoost (Adaptive Boosting). AdaBoost combines a number of weak learners to form a strong learner, in order to achieve better separation between classes.
There are two kinds of learner in a machine learning weak and strong. Weak classifier is a classifier which is only slightly correlated with the true. The strong learner will be well correlated with the true classification. [18]

3.5 Haar like-wavelets

The Haar like-wavelets are single wavelength square waves, as we see in the figure 7, one high interval and one low interval. In two dimensions, a wavelet is a pair of adjacent rectangles, one light and one dark.

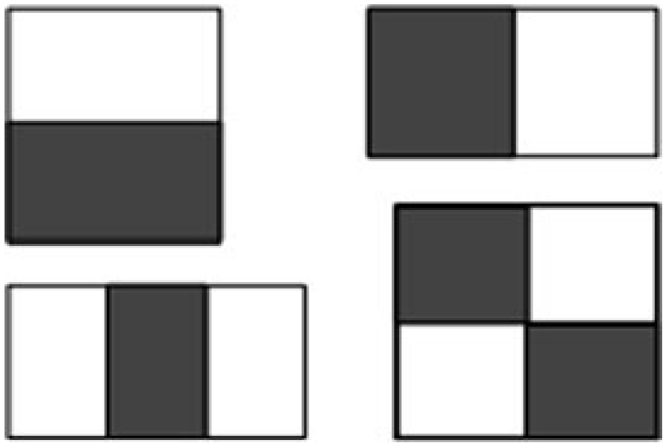


Figure 7. The Haar wavelet-like features. Copied from [20]

When we have extracted features from the images, we have to associate every feature with an identifier: the feature descriptor. There are three kind of descriptors:

1 Descriptor two rectangles: Difference between the sum of the pixels of the two rectangular regions of the same size and shape that are horizontally or vertically adjacent.

2 Descriptor three rectangles: A rectangle with three rectangles inside, the sum of the pixels between two outer rectangles and subtracted from the sum of the pixels in a central to these rectangle.

3 Descriptor four rectangles: Difference between the diagonal pairs of rec tangles involved.

The figure 8 show us different kind of descriptors in the faces

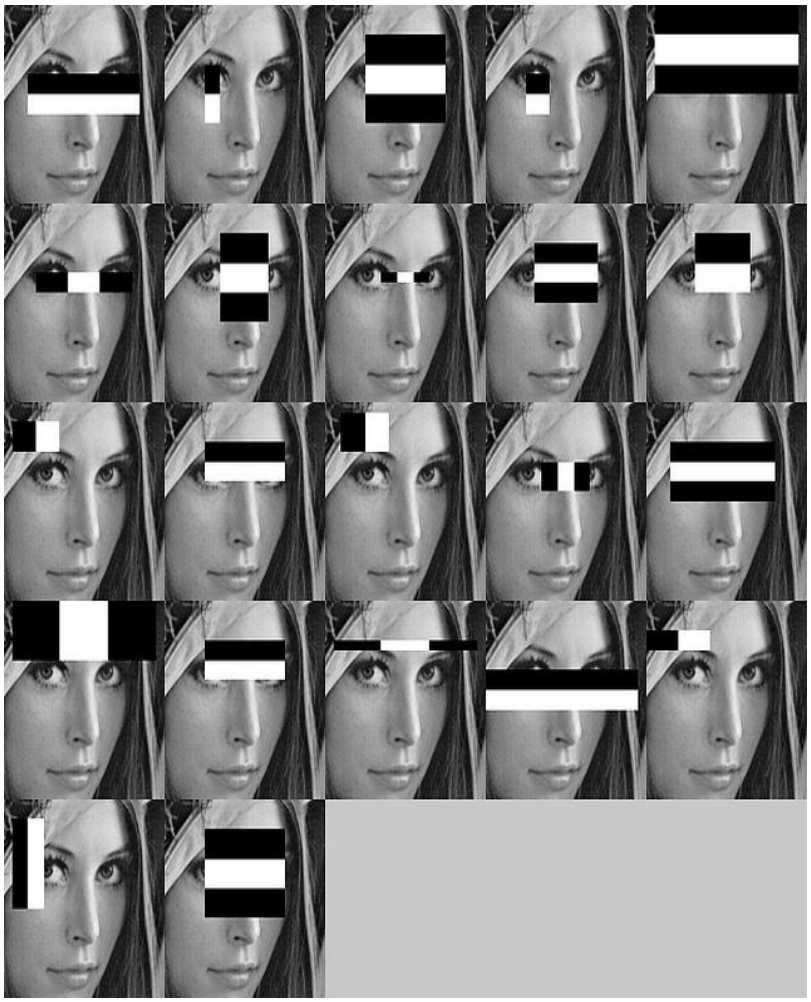


Figure 8. The rectangles features in the different regions of the face. Copied from [33]

If any one of these filters fails to pass an image region, that region is immediately classified as "Not Face." When a filter passes an image region, will go to the next filter in the chain. Image regions that pass through all filters in the chain are classified as "Face".

3.6 Integral Image

To determine the presence or absence of hundreds of Haar features at every image location and at several scales efficiently, Viola and Jones used a technique called an *Integral Image*. In general, "integrating" means adding small units together to calculate the integral image beginning with the position (0, 0). The down pixel of the right has the sum of all intensity of the image [19]

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x, y).$$

The figure 9 shows the sum of all pixels values in the colored region of the top left (x,y)

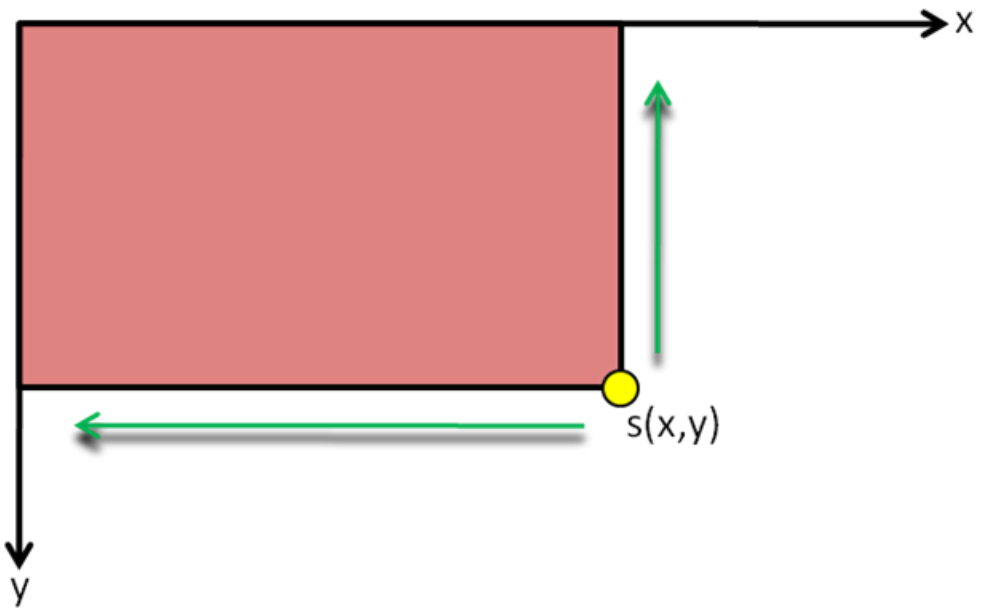


Figure 9. Integral Image. Copied from [19]

An integral value of the image ii(x, y) will be equal to the pixel valor i(x, y) added to all pixels of the image which are up to the left and up of the position (x, y)

$s(x, y) = s(x, y - 1) + i(x, y)$

$ii(x, y) = ii(x - 1, y) + s(x, y)$

In this case, the small units are pixel values. The integral value for each pixel is the sum of all the pixels above it and to its left. Starting at the top left and traversing to the right and down, the entire image can be integrated with a few integer operations per pixel. For it we create a Summed Area Table. If we have the point (x, y), the value will be the colored area [10]

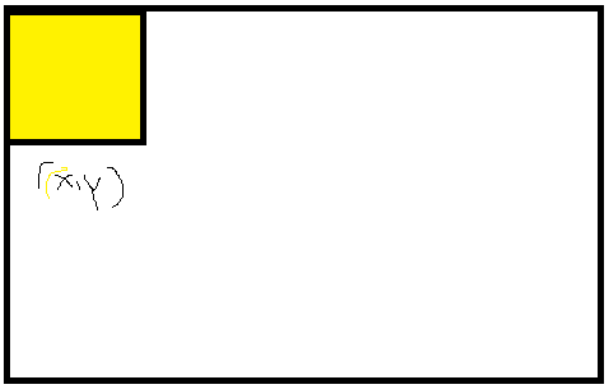


Figure 10. The point value (x, y) Adapted from [8]

After the integration the value at each pixel location contains the sum of all pixel values within a rectangular region at the top left of the image. To find the average pixel value in this rectangle we divide the value at (x, y) by the rectangle area.

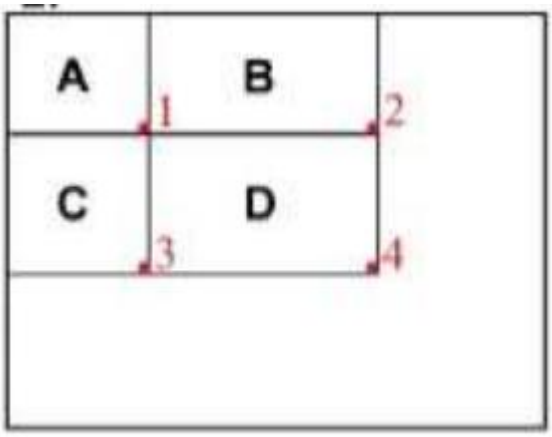


Figure 11 the values A+B+C+D. Adapted from [8]

If we want to know the summed values for other rectangles, for example in the figure 11, if we want to know the rectangle D = A+B+C - (A+B)-(A+C) +A//

- A+B+C+D will be the Integral Image´s value at location 4
- A+B is the value at location 2
- A+C is the value at location 3
- A is the value at location 1

With three integral operations we can find the sum of pixel values for any rectangle in the original image:

(X4, y4) - (X2, y2) - (x3, y3) + (x1, y1).

3.7 Cascades

Viola and Jones use a cascade of very simple classifiers that run one after another. Each cascade classifier is trained with the boosting algorithm AdaBoost. The first classifiers are simple and allow to reject many of the no-faces while accepting a very high percentage of faces. Positive results from the first classifiers triggers the evaluation of a second, more complex, classifier and so on. The detection is fast because the cascade target those areas where there most probability a face is. Chain classifiers will be more complex and will have lower false positive rates. It will be necessary to train the weak learner threshold to minimize the errors. In the figure 12 we can see the classifier cascade where the image regions are classified as face or not face.

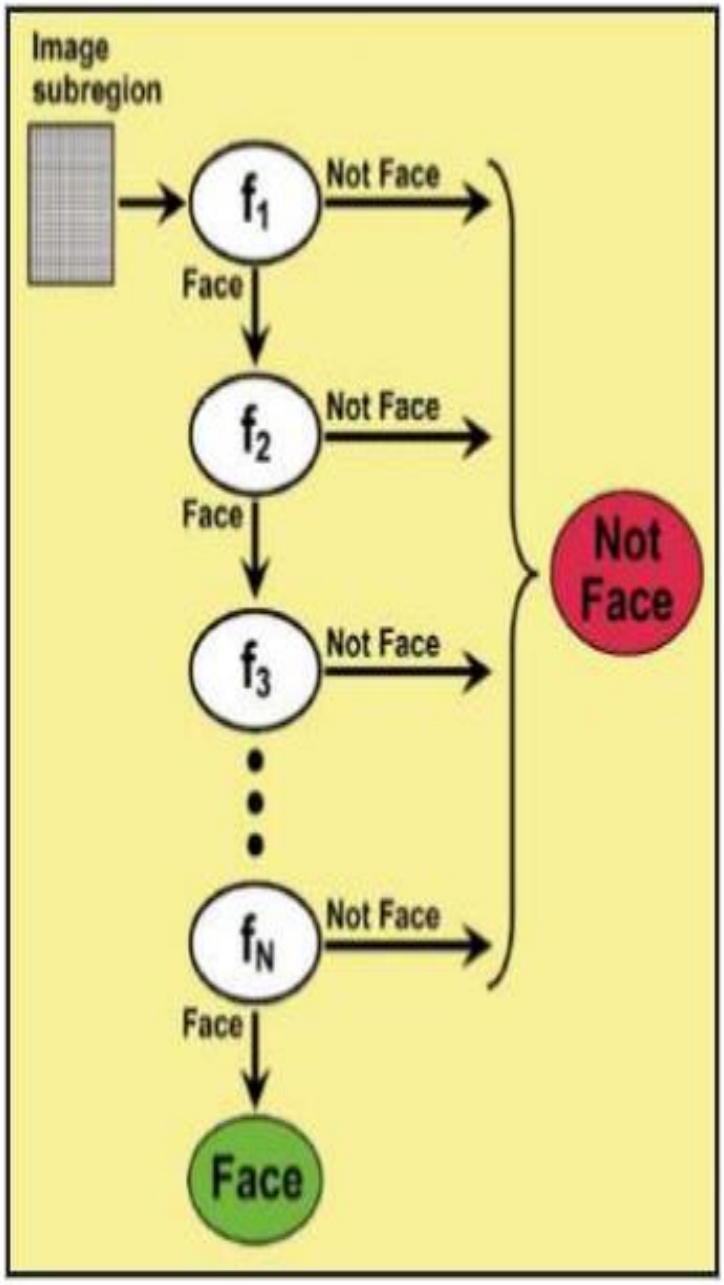


Figure 12. The classifier cascade: a chain of filters. Copied from [19]

When we get the strong classifiers, we have to validate them. We can use a confusion matrix [16]. The next step is to reduce the three channels of the image, the red, green and blue (RGB) in a channel in the greyscale. Grayscale images are distinct from one-bit bitonal black-and-white images, which in the context of computer imaging are images with only two colors, black and white, and we have to apply a thresholding.

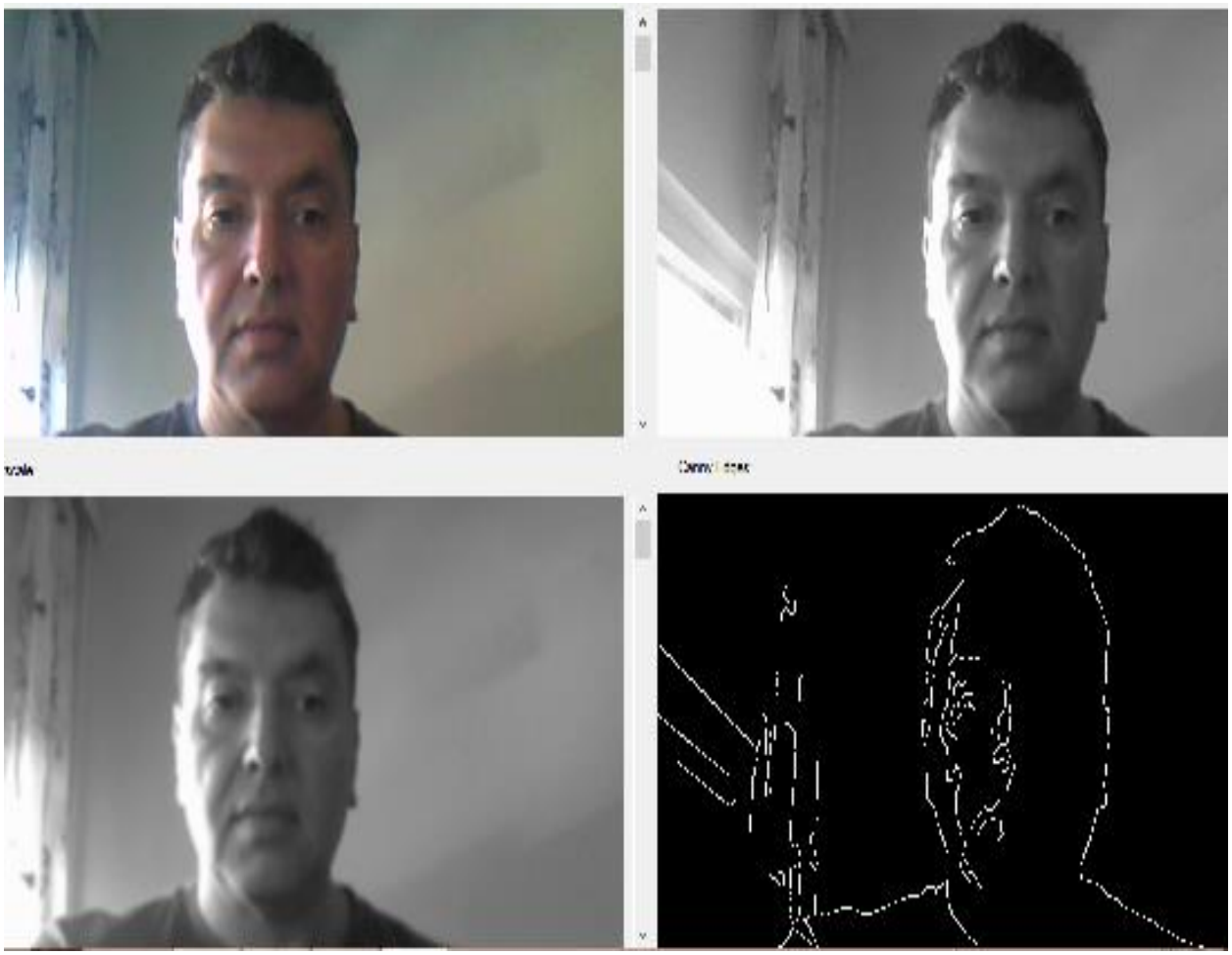


Figure 13. Using the grey-scale

As we can see in figure 13, the light over the face is not the same on every eye. The thresholding in each eye will be different because the illumination.

3.8 Detection Algorithms

3.8.1 Eyes

Starting from the recognition faces we have got previously, we will use the function:

- cvHaarDetectObjects, [26]

The algorithm for the eye detection is:

1. To apply the Haar classifier to separate the eyes.
2. To separate the RGB components from the eyes region and to convert them into a greyscale with the equation

   0.2989*R+0.5870*G+0.1140*B

And the trained classifiers:

- haarcascade righteye 2splits.xml
- haarcascade lefteye 2splits.xml

For false positives, if is necessary we implement the function cvHaarDetectObjects with the parameter "min_neighbors" for the right detection. Once we have got it, the next step is to use the Haar classifiers in order to achieve the Perclos (the open and closure of the eyes. We can implement it with an initial qualification of the opening and closure of the eye, for it we include the following:

- **yl** the left most
- **yr** the right most
- **xs** top center point
- **xi** bottom center point.

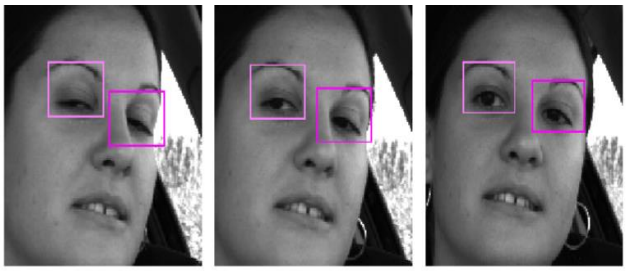


Figure 14. An eye detection. Copied from [30]

The distance between |xs – xi| discriminates the relationship opening-closing of each eye. We can see the eyes detection in the figure 14. Figure 15 illustrates a diagram state of fatigue recognition with the eyes.

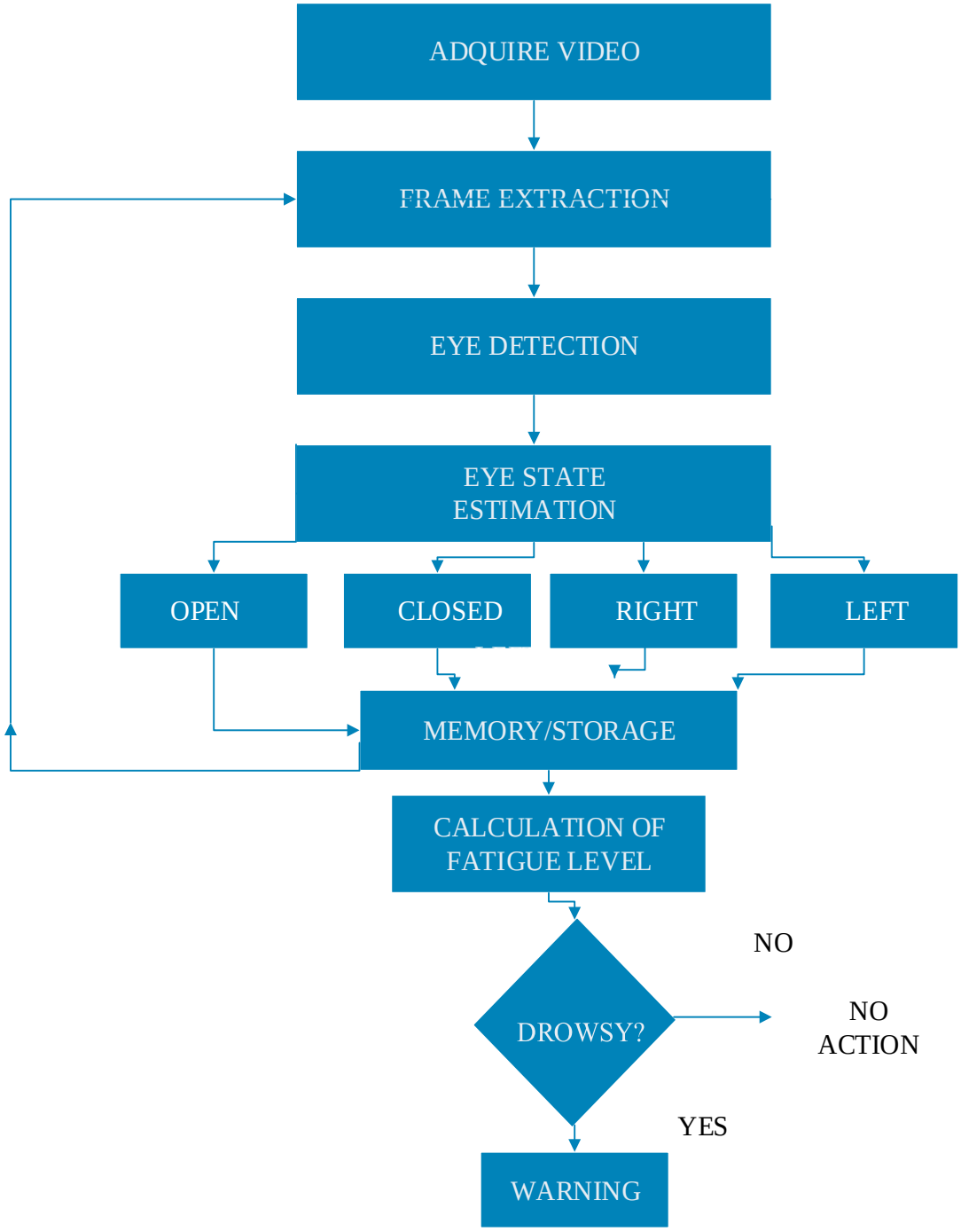


Figure 15. The diagram state of fatigue recognition with the eyes. Modified from [9]

3.8.2 Mouth

The International Journal of Vehicular Technology provides the mouth detection and also the nodding algorithm. Yawning is almost similar to surprise, where the mouth opens in both cases. The algorithm for detecting mouth:

1. Detect facial edge using gradient edge detector.
2. Compute vertical projection on the half lower face edge.by, where symbolizes the image gradient and represent the row and the column, respectively. This step aims to detect the right and the left mouth region boundaries. [37]
3. Compute horizontal projection to resulting gradient region according to obtain the upper and lower limits of the mouth and then the mouth localized region.[5]

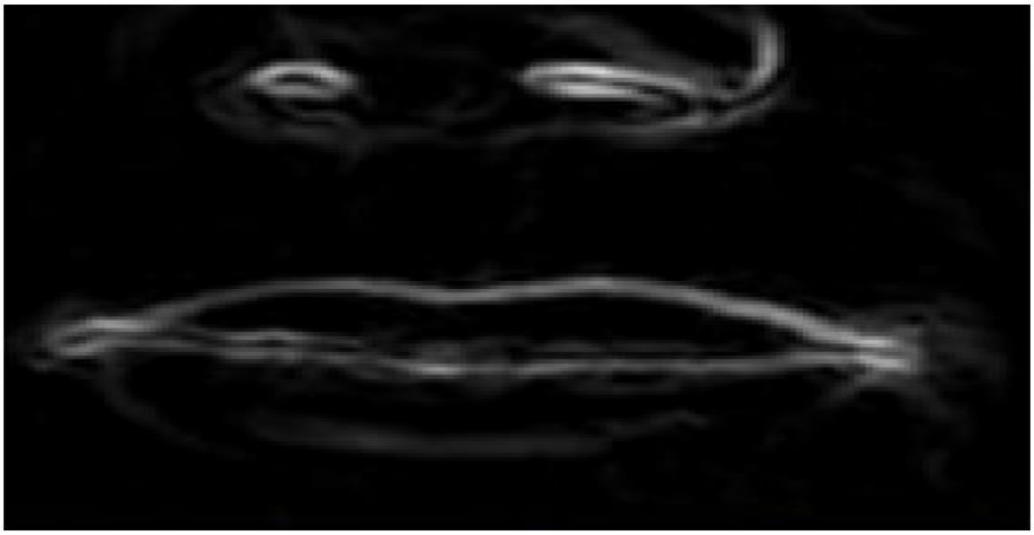


Figure 16. Left and right mouth boundaries using the gradient edge detector copied from [5]

3.8.3 Nodding

Nodding and head-shaking are relative movements. However the difference is nodding is in the high coordinate and the head-shaking in the wide one. [25]
The algorithm for detecting nodding is:

1. To take as a reference point the initial position of the head the distance in the middle of the eyes in the first 20 frames.
2. To calculate the distance of nodding and head-shaking such as the difference between (Pi [0]) less the frame of 20 frames before (pf [-20]).
3. If the difference (|Pi [0] - |Pf [20]) is bigger than threshold, then there is the nodding...
4. Update the value of the stored frames replacing the older frame by the next until the current frame.

We have done the detection of the face, the steps of eyes movements, yawn and nodding. It is necessary to do the alarms for detecting the fatigue. There are eight states which then can be seen in the figure 17.
We can asset that there are three outputs:

1. Good, no necessary action to do.
2. Alarm
3. Danger

Figure 17 show us a diagram of fatigue with three outputs, in this case the alarms.

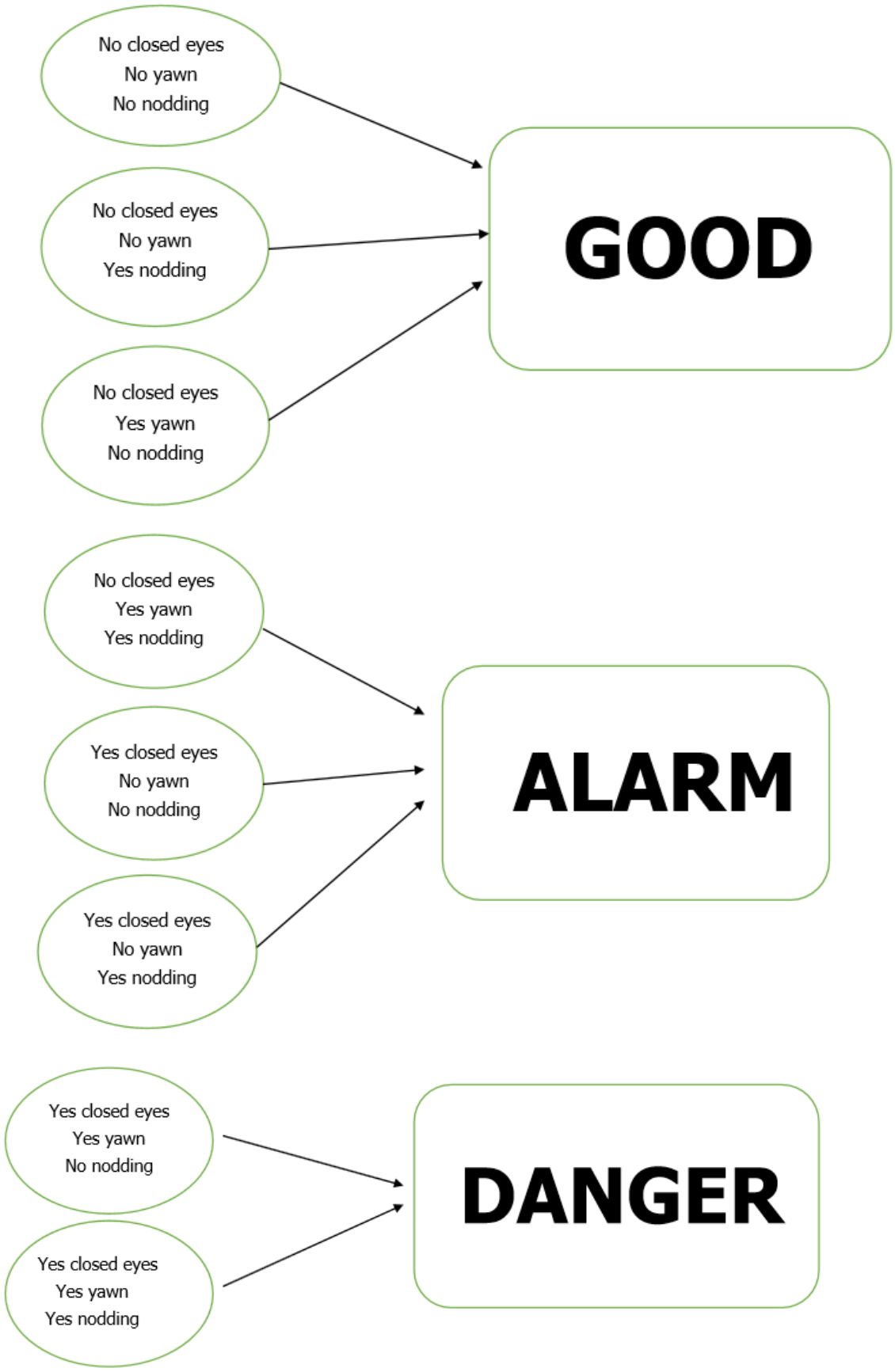


Figure 17 diagram of fatigue alarms

So far we have an image processed method, but this method has the handicap that is necessary to calculate from frame to frame, and generate the alarms from them, however we need a system that can learn from a dynamic video of a person in order to learn different situations and generate an alarm with the feedback of the video and in real time. We can get it with the soft computing [21]. One of the elements of soft computing is the Artificial Neural Network, an electronic model based on the neural structure of the brain. The ANN can also "learn". There are two kinds of learning: Supervised and Unsupervised. Supervised gives us the right input and output, and the ANN adjusts the weights. It will indicate the data to the right output as in a conventional system of learning.
Unsupervised learning only the stimuli is given. The ANN makes the calculation for adjusting the weights using the stimuli and the output is calculated by the net.
This kind of learning does not use previous information and is also called automatic learning. The learning has to evaluate the concepts itself. [32]

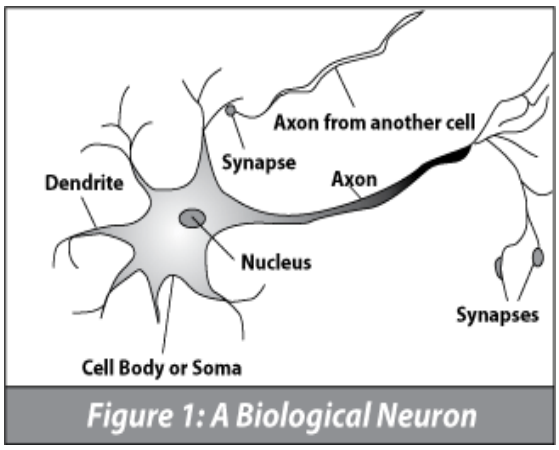
3.9 Artificial Neural Network (ANN)

There are three elements to consider

1. An artificial neuron is a simple dispositive to calculate from an external input vector or another neurons, which give an output. [29]
2. Neuronal networks are systems like the human being brains. They are processors: the neurons and they are connected and they communicate themselves by weighted connections, the synaptic weights. [31]
3. The neuro-fuzzy system is a combination of artificial neuron and fuzzy systems (fuzzy logic) used for managing the fuzzy information, an approach to computing based on "degrees of truth" rather than the usual "true or false" in Boolean logic. The fuzzy system seems closer to the way our brain works. [24]

   The human brain has millions of neurons connected among themselves. When the information arrives to the neurons, it is processed for giving an answer to different stimuli. All neurons have three parts:

   1. A cellular body
   2. A door gate the Dendrite
   3. An out gate the Axon [29]



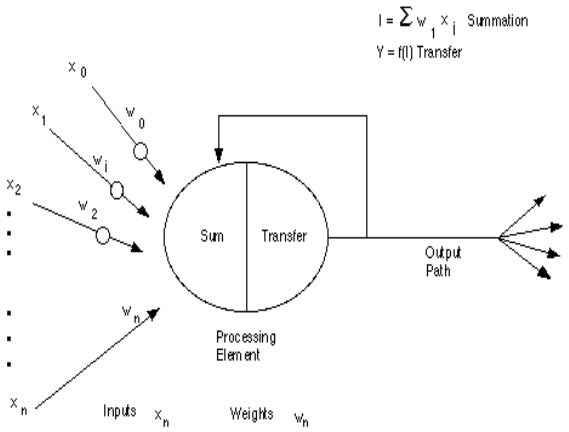


Figure 18. Human neuron copied from [31].Figure 19. Artificial neuron copied from [32].

Almost all the axons are connected with the dendrites of other neurons, the synapse, as we can see in the biological neuron picture. A signal of input with some power arrives to the neuron and depending of this the neuron emits an answer signal, the synapse. To a neuron can arrive millions of input signals, with different power or different weight. The behaviorism of the neuron can be represented mathematically with a list of input signals which are multiplied (X) by their weights and later they are added (+), (using the properties of ORing and ANDing):

Fi (Xj) = Sj * Wj

- Fi (t) It is the synaptic potential of the neuron i in the moment t
- Sj input from j (information)
- Wj synaptic weight.[32]

The result is the activation level of the neuron who depends on the level of the output which will be send to every connected neuron. An (ANN) is a system with processors, neurons, connected in parallel among them. [29] The ANNs learn by experience like persons. The neurons can be united in structural units, the layers, and finally the united layers can make a neural net. [15]

There are three layers in ANNS, as we can see in the figure 20: the input of the data, the hidden where the data are processed (for example where the algorithms are working) and the output with the final result.

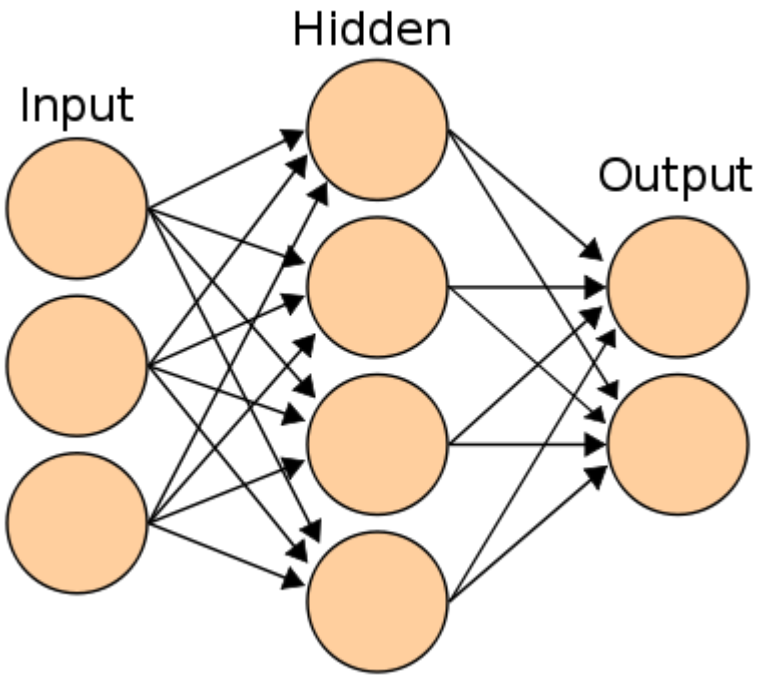


Figure 20. ANN. Copied from [31]

The data are shared among the layers. When a neuron gets a result sends to all the neurons of the next layer. Every result will be weighted in every neuron by the synaptic weight, the strength or amplitude of the connection between two nodes. For example if a large signal in an input neuron corresponds with a large output, then the synaptic weight between these two neurons will increase. It is necessary to adjust the weights, a process called "learning training". Thanks to a learning algorithm the network can calculate the weights. To the beginning the result will not be good, but with the next training, and adjusting the weight, the results will be accurate.

We can train ANN giving the face measures of the face and the outputs the fatigue stages (Good, Alarm and Danger).

4 Implementation

4.1 Hardware

The following hardware was used in this project: The PC was an ACER Aspire 5560 with a processor AMD A4-3300M APU Graphics 1.90 GHz, with a RAM of 6 GB, and the OS was Windows 8.1 Pro. The camera was a 1, 3 HD Webcam.

4.2 Software

From the previous early 2000 years to nowadays, the codes with Opencv have been implemented, which is an open source computer vision library for C and C++. On its web site there is instructions for the installation [6]. There are versions for Linux, Mac OS and Windows for the processing image and computational vision tasks. It is possible to use with different compilers: Microsoft Visual C++, MinGW (Code::Blocks), Eclipse. It is useful also with Image Processing Library (IPL) of Intel, and adds detection of faces, the movement, and 3D.

The algorithms combined with the IPL are flexible structures thanks to the Intel Architecture. This is an advantage for the computational efficiency and real-time applications. There are over 500 functions for different areas such as vision, medical imaging, security, camera calibration, stereo vision, and robotics (for Machine Learning, the MLL library). On the web site of OpenCV there is repository of books where we can access to the information necessary for doing or implementing a project. [7]

IplImage is a structure of OpenCV (originally of the Intel Imaging Processing Library IPL) for building and processing images and basically with these elements in its structure:

- **int width** is the width of the image in pixels
- **int height** is the height of the image in pixels
- **int depth** is the depth in bits
- **int nChannels** is the number of channels. 1 for Greyscale and three for RGB
- **char *imageData** is a pointer of the adjustment of the image (depending the depth)
- **int widthStep** is a number of bits in every line of the image (in the adjustment of imageData).

Today in OpenCV there is a new structure Mat instead of IplImage. [26]

Combined with OpenCV is Emgu CV, as in its site web defines as "a cross platform .Net wrapper to the OpenCV image processing library. Allowing OpenCV functions to be called from .NET compatible languages such as C#, VB, VC++, IronPython etc. The wrapper can be compiled by Visual Studio, Xamarin Studio and Unity, it can run on Windows, Linux, Mac OS X, iOS, Android and Windows Phone". This library is useful for creating a database we want to use for research and we have to use it combined with the algorithms of the libraries of OpenCV. Now there is a beta version for Android, and commercials versions for IOS (IPhone, IPad, and IPod Touch). From the code (appendix), these elements are relevant in the structure:

Image<Bgr, Byte> currentFrame:
Current image acquired from web-cam for display.

Image<Gray, byte> gray_frame = null:
GreyScale of current image acquired from webcam for processing

Classifier_Train Eigen_Recog = new Classifier_Train();
Classifier with default training location

Image<Gray, byte> result, TrainedFace = null;
Used to store the result image and trained face

Capture grabber;
The capture variable

Eigen_Recog.IsTrained
Load the trained faces

gray_frame = currentFrame.Convert<Gray, Byte>();
Convert it to Grey-Scale

Rectangle[] facesDetected = Face.DetectMul
tiScale(gray_frame, 1.2, 10 new Size(50, 50), Size.Empty);

Face detector. [34]

4.3 Training and results

It is necessary to train the software in order to get accurate results. At the camera of pc were showed different faces and later were labelled. The interface has there recognizer type: Eigen, Fisher and Local Binary Pattern Histogram (LBPH). For training the Eigen was used in the initial recognition. Later when the database was robust the LBPH was used.

The test subjects were done in the same room with same illumination. The color temperature was 4500 k (Kelvin degrees). At the store process was necessary the faces were without motionless due to store in the database. It was necessary to evaluate how the system worked. It used a confusion matrix. A confusion matrix or contingence table tested the algorithm and the image processing. Basically there is a table in a matrix format that displays the frequency distribution of the variables, in each column the prediction for each class, and in the row the actual class classification. For the detection of the faces, in a model of binary outcomes, we can compare the prediction and actual outcome, predicted 1 and actual 0, there are four possibilities:

1. True positive: predicted 1 and actual 1
2. False positive: predicted 1 and actual 0
3. True negative: predicted 0 and actual 0
4. False negative: predicted 0 and actual 1

	TN 0	**FN 0**
	FP 11	**TP 115**

Table of the confusion matrix

Accuracy is the total positive plus total negatives divided by the sum of the total population (all the faces in the database).

Accuracy $\frac{\mathbf{TP+TN}}{\mathbf{all\ faces}} = 1$

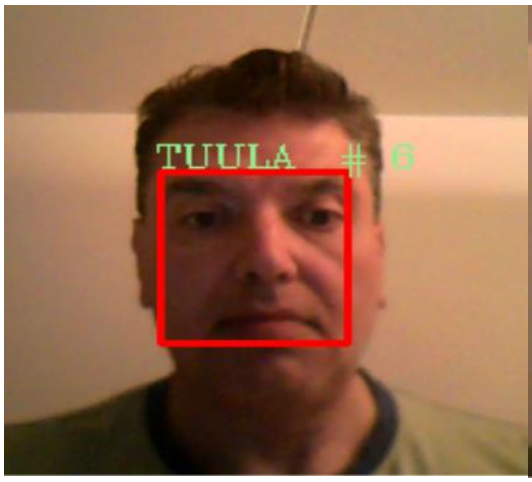


Figure 21 False positive (FN)

Figure 22 True Positive (TP)

The pictures were taken in different distances and with the same rate of temperature of colour. The figure 22 is a successful detection and labelled as FRANCISCO # 1.

5 Conclusion

The aim of this project was to use the computing vision system to detect tiredness.
The steps of face recognition worked successfully but some non-faces were detected, although the error rate was very low, overall due to the light and the use a small database.

Due to time limitations I was not able to complete the part of eyes recognition and the R.O.I. of the mouth. For the recognition of tiredness it necessary to have a big database in order to train the classifiers for a low rate of error.

On the other hand I was able to understand the learning structure of the algorithms based on the Viola Jones detectors and the Haar classifiers.
The thesis is explains how to arrive to the recognition thanks to a local feature analysis with the algorithms of Viola Jones.

The neural network is another widely studied technique for small databases, but requires many images per person to train the network to achieve results close to exactness. In the future the aim is to get stress detection and an application for medical resources. If in theory and in practice fatigue is detected, perhaps stress could be detected training the Haar cascade.

References

1 Fatigue Definition - Symptoms - Mayo Clinic. 2015. Fatigue Definition - Symptoms - Mayo Clinic. [online].
URL: http://www.mayoclinic.org/symptoms/fatigue/basics/definition/sym-20050894
Accessed 15 January 2015

2 Mental Fatigue. 2015.[online].
URL:http://www.betterhealthusa.com/public/235.cfm.
Accessed 18 April 2015.

3 2015. Real-Time Eye, Gaze, and Face Pose Tracking for Monitoring Driver Vigilance Qiang Ji and Xiaojie Yang. [online]
URL: http://www.ecse.rpi.edu/~qji/rti.pdf.
Accessed 18 April 2015.

4 Drowsy Driver Detection Through Facial Movement Analysis: 2015.[online].
URL: http://mplab.ucsd.edu/wordpress/wp-content/uploads/vesra.pdf.
[Accessed 05 March 2015].

5 Driver's Fatigue Detection Based on Yawning Extraction. 2015. [online].
URL:http://www.hindawi.com/journals/ijvt/2014/678786/.
Accessed 23 April 2015.

6 Installation in Windows — OpenCV 2.4.11.0 documentation. 2015. Installation in Windows — OpenCV 2.4.11.0 documentation. [online]
URL:http://docs.opencv.org/doc/tutorials/introduction/windows_install/windows_install.html.
Accessed 16 April 2015.

7 Books | OpenCV. 2015. Books | OpenCV. [online].
URL: http://opencv.org/books.html.
Accessed 18 April 2015

8 DETECCIÓN, LOCALIZACIÓN E IDENTIFICACIÓN BIOMÉTRICA DE CARAS EN IMÁGENES: MÉTODOS Y EVALUACIÓN EN EL MARCO NIIST-MBGG, Andrei Bogdan Dragolici.[online].
URL:http://atvs.ii.uam.es/files/2010_0602AndreiDragolici.pdf.
Accessed 18 April 2015.

9 Vision-based Real-time Driver Fatigue Detection System for Efficient Vehicle Control, D.Jayanthi, M.Bommy 2015. [online].
URL: http://www.ijeat.org/attachments/File/v2i1/A0808102112.pdf.
Accessed 18 April 2015.

10 A detector tree of boosted classifiers…Rainer Lienhart, Luong Liang, and Alexander Kuranov [online].
URL:http://www.lienhart.de/Prof._Dr._Rainer_Lienhart/Source_Code_files/ICME2003.pdf.
Accessed 18 April 2015.

11 Sleepiness and Head Movements, Johannes VAN DEN BERG 2015 [online].
URL: https://www.jniosh.site/old/niih/en/indu_hel/2006/pdf/indhealth_44_4_564.pdf.
Accessed 10 March 2015.

12 Recuperación de imágenes 2015. [online].
URL:http://dspace.ups.edu.ec/bitstream/123456789/1710/15/UPS-CT002312.pdf.
Accessed 23 April 2015.

13 CCD. 2015. CCD. [online].
URL:http://www.cnb.csic.es/~fotonica/Photonic_en/Review/ccd1.htm.
Accessed 13 May 2015.

14 Kodak engineer had revolutionary idea: the first digital camera - seattlepi.com. 2015. *Kodak engineer had revolutionary idea: the first digital camera - seattlepi.com*. [online]
URL: http://www.seattlepi.com/business/article/Kodak-engineer-had-revolutionary-idea-the-first-1182624.php.
Accessed 11 March 2015.

15 From Curve Fitting to Machine Learning: Achim Zielesny: Free Download & Streaming: Internet Archive. 2015. *from Curve Fitting to Machine Learning: Achim Zielesny: Free Download & Streaming: Internet Archive*. [online].
URL: https://archive.org/details/From_Curve_Fitting_to_Machine_Learning.
Accessed 11 March 2015.

16 Deep Data Mining Blog: Calculate Confusion Matrix Using SQL. 2015. *Deep Data Mining Blog: Calculate Confusion Matrix Using SQL*. [online].
URL: http://www.deep-data-mining.com/2014/02/calculate-confusion-matrix-using-sql.html.
Accessed 13 March 2015.

17 Hessian-Laplace Feature Detector and Haar Descriptor for Image Matching, Akshay Bhatia 2015. [online].
URL: https://www.site.uottawa.ca/~laganier/publications/thesis/akshay-thesis.pdf.
Accessed 13 March 2015.

18 BOOSTING (ADABOOST ALGORITHM), Eric Emer 2015. [online].
URL: http://math.mit.edu/~rothvoss/18.304.3PM/Presentations/1-Eric-Boosting304FinalRpdf.pdf.
Accessed 1 March 2015.

19 How Face Detection Works [online].
URL: http://www.laurence.com.ar/artes/comun/Haar.pdf.
Accessed 19 March 2015.

20 Face Detection (Face Recognition Techniques) Part 1. 2015. Face Detection (Face Recognition Techniques) Part 1. [online].
URL:http://what-when-how.com/face-recognition/face-detection-face-recognition-techniques-part-1/.
Accessed 23 April 2015.

21 What is Soft Computing? Models of Decision and Optimization (MODO) Research Group.Departamento de Ciencias de la Computación e Inteligencia Artificial E.T.S. de Ingenierías Informática y de Telecomunicación Universidad de Granada. E-18071 Granada [online].
URL: http://modo.ugr.es/en/soft_computing.
Accessed 18 April 2015.

22 Digital Image Processing .Rafael C. Gonzalez, Richard E. Woods, 2nd ed. Upper Saddle River, New Jersey 07458: USA. Tom Robbins; 2002. .

23 Digital Image Processing: Introduction, Brian Mac Namee 2015. [online].
URL http://www.sci.utah.edu/~gerig/CS6640-F2012/Materials/ImageProcessing1-Introduction-Bryan-Mac-Namee.pdf.
Accessed 16 April 2015.

24 What is fuzzy logic? - Definition from WhatIs.com. 2015. What is fuzzy logic? - Definition from WhatIs.com. [online].
URL: http://whatis.techtarget.com/definition/fuzzy-logic .
Accessed 18 April 2015.

25 Real-time Detection of Nodding and Head-shaking by Directly Detecting and Tracking the "Between-Eyes" Shinjiro Kawato and Jun Ohya2015. [online].
URL: http://www.math.hcmuns.edu.vn/~ptbao/LVTN/2003/eyes/131000840610.pdf.
Accessed 23 March 2015.

26 Basic Structures — OpenCV 2.4.11.0 documentation. 2015. Basic Structures — OpenCV 2.4.11.0 documentation. [online].
URL: http://docs.opencv.org/modules/core/doc/basic_structures.html.
Accessed 18 April 2015.

27 Digital Image Processing. 2015. *Digital Image Processing*. [online].
URL: http://www.tutorialspoint.com/dip/.
Accessed 09 April 2015.

28 European Commission. Road safety: EU reports lowest ever number of road deaths and takes first step towards an injuries strategy. . [online].
URL: http://europa.eu/rapid/press-release_IP-13-236_en.html .
Accessed 12 April 2015.

29 Artificial Neural Networks for Beginners. Carlos Gershenson. 2015. [online].
URL: http://webcache.googleusercontent.com/search?q=cache:06zN5phnPkIJ:https://datajobs.com/data-science-repo/Neural-Net-%5BCarlos-Gershenson%5D.pdf+&cd=10&hl=es&ct=clnk&gl=fi.
Accessed 14 April 2015

30 Detección de fatiga en conductores mediante fusión de sistemas ADAS 2015, Iván García Daza. [online].
URL: http://dspace.uah.es/dspace/bitstream/handle/10017/16621/main.pdf?sequence=1&isAllowed=y.
Accessed 15 April 2015

31 An overview of Neural Network. 2015. N E U R A L P O W E R. [online].
URL: http://www.neuralpower.com/technology.
Accessed 15 April 2015.

32 What are Artificial Neural Networks? 2015.[online].
URL: http://www.psych.utoronto.ca/users/reingold/courses/ai/cache/neural2.html.
Accessed 15 April 2015.

33 Adam Harvey Explains Viola-Jones Face Detection. 2015. *Adam Harvey Explains Viola-Jones Face Detection*. [online].
URL http://makematics.com/research/viola-jones/.
Accessed 16 April 2015.

34 Tutorial - Emgu CV: OpenCV in .NET (C#, VB, C++ and more). 2015 Tutorial - Emgu CV: OpenCV in .NET (C#, VB, C++ and more). [online].
URL: http://www.emgu.com/wiki/index.php/Tutorial.
Accessed 16 April 2015.

35 Biopac. Data Acquisition Systems | Data Loggers, Amplifiers, Transducers, Electrodes for Life Science Research and Education| BIOPAC. 2015. Data Acquisition Systems | Data Loggers, Amplifiers, Transducers, Electrodes for Life Science Research and Education| BIOPAC. [online].
URL: http://www.biopac.com/.
Accessed 19 April 2015.

36 What is a Photon? Andrew Zimmerman Jones. About Education. [online].
URL: http://physics.about.com/od/lightoptics/f/photon.htm
Accessed 14 May 2015.

37 A Face Recognition Method Based on Local Feature Analysis [online].
URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.119.9344&rep=rep1&type=pdf
Accessed 23 April 2015

38 CCD Cameras: Elissa Croghan. 2015. CCD Cameras: Elissa Croghan. [online].
URL: http://www.dapep.org/DAPT/EM-Wiki/ccd-camera-2.html.
Accessed 23 April 2015.

39 Working of digital camera-block diagram, parameters, colour filtering. 2015. Working of digital camera-block diagram, parameters, colour filtering. [online].
URL: http://www.circuitstoday.com/working-of-digital-cameras.
Accessed 23 April 2015.

40 Sleeping Chinese air traffic controllers leave plane in limbo | World news | The Guardian. 2015. Sleeping Chinese air traffic controllers leave plane in limbo | World news | The Guardian. [online].
URL: http://www.theguardian.com/world/2014/aug/19/sleeping-chinese-air-traffic-controller-plane.
Accessed 27 April 2015.

Appendix 1: Source Code of Face Recognition

```
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 Emgu.CV.UI;
using Emgu.CV;
using Emgu.CV.Structure;
using Emgu.CV.CvEnum;

using System.IO;
using System.Xml;
using System.Runtime.InteropServices;
using System.Threading;
using System.Security.Principal;
using System.Threading.Tasks;
using Microsoft.Win32.SafeHandles;

namespace Face_Recognition
{
 public partial class Form1 : Form
 {
  #region variables
  Image<Bgr, Byte> currentFrame;
  Image<Gray, byte> result, TrainedFace = null;
  Image<Gray, byte> gray_frame = null;
  Capture grabber;
```

```
  public CascadeClassifier Face = new CascadeClassifier(Applica-
tion.StartupPath     +     "/Cascades/haarcascade_frontalface_de-
fault.xml")

  MCvFont font = new MCvFont(FONT.CV_FONT_HERSHEY_COMPLEX, 0.5,
0.5); //Our fount for writing within the frame

  int NumLabels;

  Classifier_Train Eigen_Recog = new Classifier_Train();

  #endregion

  public Form1()
  {
   InitializeComponent();

   if (Eigen_Recog.IsTrained)
   {
    message_bar.Text = "Training Data loaded";
   }
   else
   {
    message_bar.Text = "No training data found, please train pro-
gram using Train menu option";
   }
   initialise_capture();

  }
  private void trainToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {
   stop_capture();

   Training_Form TF = new Training_Form(this);
   TF.Show();
```

```
  }
  public void retrain()
  {

   Eigen_Recog = new Classifier_Train();
   if (Eigen_Recog.IsTrained)
   {
    message_bar.Text = "Training Data loaded";
   }
   else
   {
    message_bar.Text = "No training data found, please train pro-
gram using Train menu option";
   }
  }

  public void initialise_capture()
  {
   grabber = new Capture();
   grabber.QueryFrame();
   if (parrellelToolStripMenuItem.Checked)
   {
    Application.Idle += new EventHandler(FrameGrabber_Parrellel);
   }
   else
   {
    Application.Idle += new EventHandler(FrameGrabber_Standard);
   }
  }
  private void stop_capture()
  {
   if (parrellelToolStripMenuItem.Checked)
   {
    Application.Idle -= new EventHandler(FrameGrabber_Parrellel);
   }
   else
```

```
   {
    Application.Idle -= new EventHandler(FrameGrabber_Standard);
   }
   if(grabber!= null)
   {
   grabber.Dispose();
   }
  }

  void FrameGrabber_Standard(object sender, EventArgs e)
  {
      currentFrame   =   grabber.QueryFrame().Resize(320,   240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

   if (currentFrame != null)
   {
    gray_frame = currentFrame.Convert<Gray, Byte>();
    Rectangle[] facesDetected = Face.DetectMultiScale(gray_frame,
1.2, 10, new Size(50, 50), Size.Empty);
    for (int i = 0; i < facesDetected.Length; i++)// (Rectangle
face_found in facesDetected)
    {
          facesDetected[i].X  +=  (int)(facesDetected[i].Height  *
0.15);
     facesDetected[i].Y += (int)(facesDetected[i].Width * 0.22);
     facesDetected[i].Height  -=  (int)(facesDetected[i].Height  *
0.3);
     facesDetected[i].Width  -=  (int)(facesDetected[i].Width  *
0.35);

     result  =  currentFrame.Copy(facesDetected[i]).Convert<Gray,
byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);
     result._EqualizeHist();
     currentFrame.Draw(facesDetected[i], new Bgr(Color.Red), 2);

     if (Eigen_Recog.IsTrained)
```

```
      {
       string name = Eigen_Recog.Recognise(result);
       int match_value = (int)Eigen_Recog.Get_Eigen_Distance;

       currentFrame.Draw(name + " ", ref font, new Point(fac-
esDetected[i].X - 2, facesDetected[i].Y - 2), new Bgr(Color.Light-
Green));
       ADD_Face_Found(result, name, match_value);
      }
     }

     image_PICBX.Image = currentFrame.ToBitmap();
    }
   }

   void FrameGrabber_Parrellel(object sender, EventArgs e)
   {
    currentFrame    =    grabber.QueryFrame().Resize(320,    240,
Emgu.CV.CvEnum.INTER.CV_INTER_CUBIC);

    if (currentFrame != null)
    {
     gray_frame = currentFrame.Convert<Gray, Byte>();

     Rectangle[] facesDetected = Face.DetectMultiScale(gray_frame,
1.2, 10, new Size(50, 50), Size.Empty);


     Parallel.For(0, facesDetected.Length, i =>
      {
       try
       {
        facesDetected[i].X   +=   (int)(facesDetected[i].Height   *
0.15);
        facesDetected[i].Y   +=   (int)(facesDetected[i].Width   *
0.22);
```

```
        facesDetected[i].Height -= (int)(facesDetected[i].Height *
0.3);
        facesDetected[i].Width  -=  (int)(facesDetected[i].Width  *
0.35);

        result     =     currentFrame.Copy(facesDetected[i]).Con-
vert<Gray, byte>().Resize(100, 100, Emgu.CV.CvEnum.INTER.CV_IN-
TER_CUBIC);
        result._EqualizeHist();

        currentFrame.Draw(facesDetected[i],  new  Bgr(Color.Red),
2);

        if (Eigen_Recog.IsTrained)
        {
         string name = Eigen_Recog.Recognise(result);
         int match_value = (int)Eigen_Recog.Get_Eigen_Distance;


         currentFrame.Draw(name + " ", ref font, new Point(fac-
esDetected[i].X - 2, facesDetected[i].Y - 2), new Bgr(Color.Light-
Green));
         ADD_Face_Found(result, name, match_value);
        }

       }
       catch
       {


       }
      });

     image_PICBX.Image = currentFrame.ToBitmap();
    }
   }
```

```
int faces_count = 0;
int faces_panel_Y = 0;
int faces_panel_X = 0;

void Clear_Faces_Found()
{
 this.Faces_Found_Panel.Controls.Clear();
 faces_count = 0;
 faces_panel_Y = 0;
 faces_panel_X = 0;
}
void ADD_Face_Found(
                  Image<Gray, Byte> img_found,
                  string name_person,
                  int match_value)
{
 PictureBox PI = new PictureBox();
 PI.Location = new Point(faces_panel_X, faces_panel_Y);
 PI.Height = 80;
 PI.Width = 80;
 PI.SizeMode = PictureBoxSizeMode.StretchImage;
 PI.Image = img_found.ToBitmap();
 Label LB = new Label();
 LB.Text = name_person + " " + match_value.ToString();
 LB.Location = new Point(faces_panel_X, faces_panel_Y + 80);

 LB.Height = 15;

 this.Faces_Found_Panel.Controls.Add(PI);
 this.Faces_Found_Panel.Controls.Add(LB);
 faces_count++;
 if (faces_count == 2)
 {
  faces_panel_X = 0;
  faces_panel_Y += 100;
```

```
     faces_count = 0;
    }
    else faces_panel_X += 85;

    if (Faces_Found_Panel.Controls.Count > 10)
    {
     Clear_Faces_Found();
    }
   }

  private void exitToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {
   this.Dispose();
  }
  private void singleToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {
   parrellelToolStripMenuItem.Checked = false;
   singleToolStripMenuItem.Checked = true;
   Application.Idle -= new EventHandler(FrameGrabber_Parrellel);
   Application.Idle += new EventHandler(FrameGrabber_Standard);
  }
  private void parrellelToolStripMenuItem_Click(object sender,
EventArgs e)
  {
   parrellelToolStripMenuItem.Checked = true;
   singleToolStripMenuItem.Checked = false;
   Application.Idle -= new EventHandler(FrameGrabber_Standard);
   Application.Idle += new EventHandler(FrameGrabber_Parrellel);

  }
  private void saveToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {
   SaveFileDialog SF = new SaveFileDialog();
```

```
   switch (Eigen_Recog.Recognizer_Type)
   {
    case ("EMGU.CV.LBPHFaceRecognizer"):
     SF.Filter = "LBPHFaceRecognizer File (*.LBPH)|*.LBPH";
     break;
    case ("EMGU.CV.FisherFaceRecognizer"):
     SF.Filter = "FisherFaceRecognizer File (*.FFR)|*.FFR";
     break;
    case ("EMGU.CV.EigenFaceRecognizer"):
     SF.Filter = "EigenFaceRecognizer File (*.EFR)|*.EFR";
     break;
   }
   if (SF.ShowDialog() == DialogResult.OK)
   {
    Eigen_Recog.Save_Eigen_Recogniser(SF.FileName);
   }
  }
  private void loadToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {
   OpenFileDialog OF = new OpenFileDialog();
   OF.Filter         =         "EigenFaceRecognizer         File
(*.EFR)|*.EFR|LBPHFaceRecognizer   File   (*.LBPH)|*.LBPH|Fisher-
FaceRecognizer File (*.FFR)|*.FFR";
   if (OF.ShowDialog() == DialogResult.OK)
   {
    Eigen_Recog.Load_Eigen_Recogniser(OF.FileName);
   }
  }


  private void Eigne_threshold_txtbxChanged(object sender, Even-
tArgs e)
  {
   try
```

```
   {
    Eigen_Recog.Set_Eigen_Threshold        =        Math.Abs(Con-
vert.ToInt32(Eigne_threshold_txtbx.Text));
    message_bar.Text = "Eigen Threshold Set";
   }
   catch
   {
    message_bar.Text = "Error in Threshold input please use int";
   }
  }

  private void eigenToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {

   fisherToolStripMenuItem.Checked = false;
   lBPHToolStripMenuItem.Checked = false;

   Eigen_Recog.Recognizer_Type = "EMGU.CV.EigenFaceRecognizer";
   Eigen_Recog.Retrain();
  }

  private void fisherToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {

   lBPHToolStripMenuItem.Checked = false;
   eigenToolStripMenuItem.Checked = false;

   Eigen_Recog.Recognizer_Type = "EMGU.CV.FisherFaceRecognizer";
   Eigen_Recog.Retrain();
  }

  private void lBPHToolStripMenuItem_Click(object sender, Even-
tArgs e)
  {
```

```
  fisherToolStripMenuItem.Checked = false;
  eigenToolStripMenuItem.Checked = false;

  Eigen_Recog.Recognizer_Type = "EMGU.CV.LBPHFaceRecognizer";
  Eigen_Recog.Retrain();
 }
}
```