Slovak University of Technology in Bratislava

Faculty of Informatics and Information Technologies

FIIT-5212-8290

Tomáš Drutarovský

DRIVER'S MICROSLEEP DETECTION

Bachelor thesis

Degree Course:	Informatics
Field of study:	9.2.1 Informatics
Place of development:	Institute of Applied Informatics, FIIT STU Bratislava
Supervisor:	Ing. Andrej Fogelton

May 2014

ANOTÁCIA

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: INFORMATIKA

Autor:	Tomáš Drutarovský
Bakalárska práca:	Detekcia mikrospánku vodiča
Vedúci bakalárskej práce:	Ing. Andrej Fogelton

máj, 2014

Bakalárska práca predstavuje problém detekcie mikrospánku vodiča. Podľa známych riešení danej problematiky vieme, že únava môže byť detegovaná prostredníctvom analýzy frekvencie žmurkania. Cieľom bakalárskej práce je navrhnúť spoľahlivú detekciu žmurknutia, ktorá by mohla byť použitá v systéme na detekciu mikrospánku.

Úvod je venovaný definovaniu problému mikrospánku. Jadro práce analyzuje dostupné riešenia v oblasti problematiky a predstavuje metódy na detekciu a sledovanie tváre a očí. Záver analýzy obsahuje zhodnotenie súčasného stavu oblasti a prehľad dostupných metód detekcie žmurknutia.

V práci navrhujeme opis stavu uzatvorenosti oka na základe SVM klasifikátora (angl. Support Vector Machine) a deskriptora vzorky oka – Vertikálnej Projekcie Intenzity (angl. Intensity Vertical Projection), SIFT deskriptora a vlastných gradientových deskriptorov. Ďalej sa sústreďujeme na detekciu žmurknutia za pomoci sekvenčnej analýzy snímkov, ktorá dosahuje lepšie výsledky ako detekcia optickým tokom z práce Divjak–Bischof (2009). Navrhované metódy boli otestované na dátach od rôznych subjektov za odlišných svetelných podmienok.

ANNOTATION

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: INFORMATICS

Author:	Tomáš Drutarovský
Bachelor thesis:	Driver's Microsleep Detection
Supervisor:	Ing. Andrej Fogelton

May, 2014

The bachelor thesis introduces a problem of driver's microsleep detection. According to the state-of-the-art, sleepiness can be detected using the analysis of the eye-blink frequency. The goal of the bachelor thesis is to propose a reliable eye-blink detection algorithm, which could be used in the microsleep detection system.

Introduction is devoted to the microsleep problem definition. The thesis core analyzes currently available solutions of this problem and it presents face and eye detection and tracking methods. Analysis conclusion describes eye-blink detection algorithms.

Afterwards a proposal of eye state description algorithms based on SVM (Support Vector Machine) and feature descriptors is presented – Intensity Vertical Projection, SIFT descriptor and own gradient descriptors. Further focus is put on our own eye-blink detection methods based on sequential frame analysis, which achieves better results than Divjak–Bischof optical flow detection method. The proposed methods were tested with datasets obtained from different subjects under different light conditions.

Declaration of Honor

I honestly declare, that I wrote this thesis independently under professional supervision of Ing. Andrej Fogelton with citated bibliography.

May, 2014 in Bratislava signature

Acknowledgement

I would like to sincerely thank Ing. Andrej Fogelton for his guidance, effort and his prompt and creative help during the work on this bachelor thesis. His encouragement in tough times and compliments in good times were very motivational to me. I want to thank my girlfriend, family and friends for their patience and support, which helped me to create the needed work atmosphere.

Contents

List of Abbreviations

ASM	-	Active Shape Model
BD	-	Blink Duration
CA	-	Condensation Algorithm
CCD	-	Charge-Coupled Device
CDF	-	Cumulative Distribution Function
CVS	-	Computer Vision Syndrome
FP	-	False Positive
GPF	-	General Projection Function
HSC	-	Horizontal Symmetry Calculation
IVP	-	Intensity Vertical Projection
KLT	-	Kanade-Lucas-Tomasi
NN	-	Nearest Neighbor
PERCLOS	-	Percentage of Closure
RGB	-	Red-Green-Blue
ROI	-	Region of Interest
SIFT	-	Scale-invariant Feature Transform
SVM	-	Support Vector Machine
TLD	-	Tracking-Learning-Detection
TP	-	True Positive

Chapter 1

Introduction

A human needs to sleep. Sleep is not an option, it is necessary and inevitable to regenerate the human body. The longer a man tries not to sleep, the more he or she feels tired and needs to rest. Due to fatigue and an effort of prolonging consciousness – *microsleep* can occur. Microsleep is a short and unintended episode of sleep, which can last from a fraction of a second up to thirty seconds [AASM 1991].

It is a result of sleep deprivation or a sleep debt. It can be also caused by a sleep disorder, medication usage, mental fatigue or other neurological disorders. Microsleep can happen anytime. However, it is the most dangerous when it happens during situations, which demand your focused attention like driving.

Microsleep usually occurs without driver's awareness and it is accompanied with a blank stare, a nodding head or prolonged eye closure. Its consequences are impairment of a driving performance and clear judgment, lack of driver's awareness, short-term memory deficit or increasing of reaction time. It can even happen with open eyes.

A driver is not capable of answering to external stimulation, what causes that he or she does not have to notice a red traffic light, an incoming curve, a road narrowing or an oncoming vehicle. Obviously, this could lead to serious traffic accidents.

There are many available solutions to avoid the microsleep. Some of them are cheap and simple, but also not so reliable. Others are quite accurate, but they require expensive technologies and complicated algorithms. Nevertheless, they do not guarantee real-time detection.

According to the report [Dinges et al. 1998], eyelid closure and eye-blink activity belong to the most reliable signs of fatigue and microsleep. Therefore, we want to focus on eye tracking and eye-blink analysis. We want to propose a solution, which would be simple and available enough and still sufficiently accurate and reliable. The driver could be tracked by the front camera of a *smartphone*. In the case of incoming microsleep, an alarm will sound. This simple action could prevent the driver from falling asleep and avoid an accident.

1.1 Traffic

Microsleep is one of the main factors causing fatal traffic accidents along with alcohol or other narcotics and overestimation of abilities (Figure 1.1). A study from Australia and

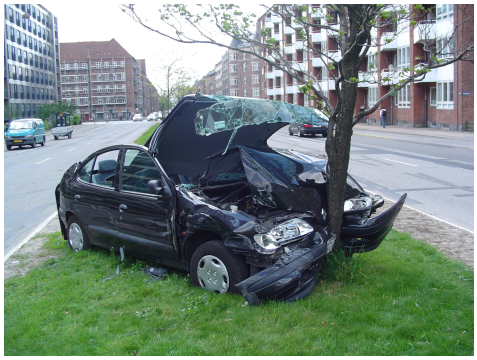


Figure 1.1: Microsleep is a serious traffic problem.[1]

New Zealand says that driving after 17 to 19 hours of vigilance equals to driving with blood alcohol concentration of 0.05% [Williamson – Feyer 2000]. Sleeping for 5 seconds at the speed of 130 km/h on a highway corresponds to 180 meters passed by a car. This could be a significantly long distance, which could lead to a fatal accident without awareness of the driver.

According to National Highway Traffic Safety Administration [NHTSA 2011] 2.5% of fatal accidents in the USA between years 2005 and 2009 were caused by microsleep, which resulted to more than 5000 fatalities. However, amount of microsleep crashes could have be much higher because it is very difficult to determine whether microsleep was the real cause or not and also what level of fatigue could lead to the crash. Study from VirginiaTech Transportation Institute [Klauer 2012] says that even more than fifth of accidents happen due to fatigue and drowsy driving.

Signs of an accident caused by microsleep:

- single car on a roadside,
- highway collision,
- driver did not try to brake or avoid an obstacle,
- driver was alone in the car,
- accident happened early in the morning.

1.2 Prevention

Microsleep is preceded by fatigue and drowsiness. An important part of its detection and prevention is to recognize signs, which signalize incoming microsleep:

- yawning,
- slow and frequent blinking,
- eye scratching,
- poor concentration,

[1] `http://upload.wikimedia.org/wikipedia/commons/e/e1/Car_crash_1.jpg` [last access: 16.11.2013]

- tired or sore eyes,
- restlessness and boredom,
- slow reactions,
- not remembering last kilometers of a road,
- feeling irritable,
- making fewer and larger steering corrections,
- missing road signs,
- loosing ability to stay in the lane,
- blurred vision.

A reduce ability of driver's own sleepiness judgment, fighting the fatigue and an effort to stay awake belong to the natural signs of falling asleep in the car, too.

Having one of these signs says it is a very high chance of microsleep and the driver should take a break.

Microsleep occurs mostly under these conditions:

- during long journey on a highway or a monotonous road,
- between 2am and 6am (when your circadian rhythm tells you to sleep),
- between 2pm and 4pm (especially after eating),
- after sleeping less than normal at night,
- after taking medication which can cause sleepiness,
- after night shifts or long hours at work.

According to [Akerstedt et al. 2001], driving at night or in the early morning is approximately five times more dangerous than driving during a daytime.

Prevention is certainly one of the most effective way of avoiding microsleep [Accidents 2001, Higgins – Bernie 2011]. First step of prevention is to realize that you are sleepy and need to rest. Microsleep often occurs when you overestimate your abilities and try to reach the goal destination as soon as possible. It is important to realize the necessity of a short break.

If you feel tired, take a break. Just few minutes of sleep can help you and bring a momentary regeneration to your body and mind. It is also called a *power nap*[2]. However, be aware of the fact that the nap lasting more than 30 minutes can bring you to deep sleep and it will be hard to wake up and concentrate back on driving.

Before you drive, have a good sleep. Most of adults need from 7 to 9 hours of daily sleep. When you drive, take a break regularly. Avoid medication which cause sleepiness and definitely avoid alcohol.

Moreover, positively affects an open car window or just a right air-condition adjustment (not cool, but not so warm either). The *Buddy system* (Figure 1.2) is also a reliable method. It means you are driving with a passenger who is talking to you, what supports you in staying awake.

[2] `http://www.sleepforall.com/microsleep.htm` [last access: 22.8.2013]

Figure 1.2: Talking to friends will help a driver to stay awake[3].

If you really do not want to stop the car and have a rest, try to avoid anything what demand your constant attention. Coffee works too, but be careful because caffeine takes about 30 minutes to affect.

Alongside all these advices and hints there are many prevention and detection systems available to avoid or detect microsleep. They are discussed in the next chapter.

Our goal is to propose a real-time application, which would detect fatigue and incoming microsleep of a driver in a vehicle. Nevertheless, nothing compares to good sleep and avoiding of driving sleepy.

1.3 Requirements

According to [Dinges et al. 1998], determination of eye-blink duration, eye-blink frequency and eye lid closure are one of the most reliable detection methods. Therefore, we decided to detect and track the driver's eyes with a camera to achieve the best results. To keep the solution affordable, we consider to use the front camera of a smartphone.

Smartphones are highly available to the most of drivers and they are easy to use. According to [Stern et al. 1984], eyelids are closed for 50 ms during the fastest eye-blink, therefore smartphone cameras with a 30 frames per second are sufficient for this matter.

We want to create a simple application which will use real-time and accurate algorithms for blink features' measuring. Obviously, we want to achieve a reasonable trade-off between the computational time and the detection rate. Detection should be fast enough to operate in the real-time and robust enough to alarm the driver in the case of microsleep.

Our solution should be non-intrusive so the driver would have minimal worries with smartphone control and maintenance. We should avoid false positive detections as much as possible so that the application would not annoy the driver. User interface should be simple to understand and easy to use.

We want to focus on the detection under good light conditions what in our case means daylight. However, we require to use algorithms which work with a grayscale image. Therefore, our solution should be easily extendible to detect microsleep even at night using an infrared camera or an additional infrared light projection.

[3] `http://www.automedia.com/Blog/post/Teen-Texting-Logic-Extreme-Mileage-Long-Commutes-Indy-500-Pace-Car-More.aspx` [last access: 16.11.2013]

Summarization of requirements for our microsleep detection smartphone application is as follows:

- real-time and accurate detection algorithms,
- high detection rate,
- low false positives,
- low computational requirements,
- algorithms working with grayscale images,
- non-intrusive and simple solution,
- simple graphical user interface.

Chapter 2

Microsleep Detection Systems

Efficient prevention certainly involves many commercial and non-commercial detection systems which detect fatigue, drowsiness and microsleep using different technologies. We can divide them into *non-eye based* systems and *eye based* systems. Unlike non-eye based systems, eye based detection systems use eye detection and eye tracking to determine, whether the driver is falling asleep.

In fact, most systems focus on fatigue detection instead of microsleep detection, because almost always it is fatigue what appears first. This is logical approach since the microsleep detection may not come on time. Therefore, it is necessary to identify fatigue before driver falls asleep and alert him or her properly.

2.1 Non-Eye Based Detection Systems

Non-eye based systems do not use sight organs to detect fatigue or microsleep. Many of them are commercial and popular, however they are not so accurate. They are not so reliable, because they rely on characteristics, which do not have to be clear signs of microsleep. This may cause an undesirable false microsleep identification. However, they may also reveal fatigue in a lot of cases.

We can divide these systems into two groups: *vehicle based* systems and *human monitoring* based systems.

2.1.1 Vehicle based systems

This type of detection systems is based on tracking of a driver and a car behavior on the road. There are several products on the market, which are developed by car companies and subsequently they are mounted in vehicles for an extra charge. These systems offer tracking of a driver's routine behavior, his or her driving habits and a driving pattern. They can determine a level of fatigue and alert the driver properly with a visual or an acoustic signal.

Volvo developed *Driver Alert Control (DAC)*[1], which determines the driver's fatigue level by monitoring of traffic lines and vehicle control. Traffic lines are tracked by a camera installed

[1] `https://www.media.volvocars.com/global/enhanced/en-gb/Media/Preview.aspx?mediaid=12130` [last access: 22.8.2013]

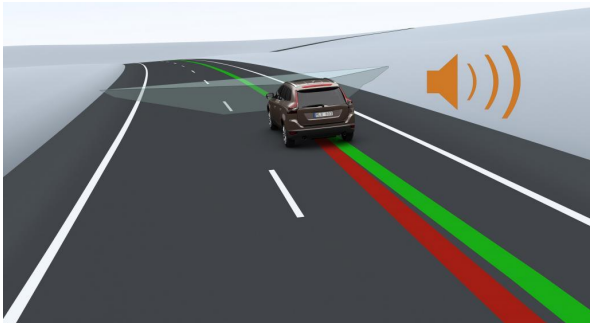


Figure 2.1: Volvo's Driver Alert Control tracks road lanes and notifies the driver in case of emergency[2].

behind the inner rear window. When a motion indicating sleepiness occurs, system evaluates the fatigue level and alerts the driver to take a break (Figure 2.1). Very similar system is also used in *Ford*'s *Driver Alert*[2].

Mercedes-Benz offers *Attention Assist*[3]. In the first 20 minutes of driving, system creates driver's profile including information about his or her driving style. Subsequently, the profile is constantly compared with actual data from sensors. System evaluates data of more than 70 factors, which determine whether the driver is drowsy or not.

Volkswagen developed *Lane Assist*[4] which helps to keep a vehicle in the current lane in case of microsleep. It detects a position of the vehicle on the roadway by a camera mounted behind the inner rear window. When the car is going out of the lane, the steering wheel is turned in an opposite direction. This causes that the car does not leave the lane and the driver remains safe. Volkswagen also offers a fatigue detection system similar to other systems.

Driver Drowsiness Detection[5] was made by *Bosch* company. System detects driver's drowsiness using driving behavior analysis. It contains a steering angle sensor, a camera with the lane assist, other additional information, e.g. vehicle speed, turn indicator, duration of travel. System evaluates all factors and calculates a warning index which can trigger an alert.

2.1.2 Human monitoring based systems

Human monitoring based systems are detection systems which monitor a human behavior, his or her physical state and other biological features to determine the level of fatigue and incoming microsleep. Most of them are products that are popular and available on the market, because of their relatively low price. However, we can mark them as less reliable detection systems due to their low detection quality.

A German company offers a product named *Stopsleep*[6]. It is an anti-sleep alarm which can be put on fingers. It measures the conductivity of the skin, which reflects the brain activity.

[2] `http://www.euroncap.com/rewards/ford_driver_alert.aspx` [last access: 22.8.2013]

[3] `http://www.euroncap.com/rewards/mercedes_benz_attention_assist.aspx` [last access: 22.8.2013]

[4] `http://www.volkswagen.co.uk/technology/proximity-sensing/lane-assist` [last access: 22.8.2013]

[5] `http://www.sae.org/events/gim/presentations/2012/sgambati.pdf` [last access: 22.8.2013]

[6] `http://www.stopsleep.de/` [last access: 22.8.2013]

Figure 2.2: Nap alarm can be put behind the ear and it sounds when the head drops[9].

This product is available at the cost of 150€, but the producer does not provide any technical data or detection rates.

Holux company from Taiwan developed a device called driver hypo-vigilance/fatigue detector[7]. The device is put on your seat belt and subsequently, it determines the fatigue level by heart rate analysis. After a limit is exceeded, an acoustic signal alerts about the incoming microsleep.

Maybe the most wide-spread and low-cost sleepiness detectors are called *personal nap alarms* or just *nap alarms* (Figure 2.2). The machine is put behind the driver's ear and it beeps when his or her head drops down. This happens due to an electronic position sensor. Nap alarm is a clever solution, but it is very hard to estimate its detection rate and accuracy. Alarms are sold on the internet for a few dollars[8].

A yawning measurement belongs to more complicated detection methods. Yawning detection in [Hariri et al. 2011] contains face detection and tracking, mouth contour detection with further mouth tracking and yawning detection based on changes in a mouth region. The face is detected by an algorithm using a skin color and a location of the face features like eyebrows, nose, eyes and mouth. This approach was tested on several videos in different light conditions and for different yawning types. The method is able to detect yawning with 80% detection rate.

Work presented in [Vural et al. 2007] uses a detection system, which targets on data analysis of a spontaneous behavior of a driver. It reveals relations between facial movements and a fatigue state or sleepiness. In addition, the system tracks head and steering movements to evaluate the precise behavior of the driver in moments just before microsleep. Testing was performed using camera detecting human faces during a virtual driving in a computer game between midnight and 3am. It shows up drivers yawn even less than normal in the last minute before microsleep.

[7] `http://www.holux.com/hcEN/en/products/products_content.jsp?pno=413` [last access: 22.8.2013]

[8] `http://napzapper.com/` [last access: 22.8.2013]

[9] `http://napzapper.com/images/VV-GH-03.jpg` [last access: 17.11.2013]

2.2 Eye Based Detection Systems

Eye based detection systems are focused on driver's eye detection and tracking. Many of them measures a value called percentage of closure or *PERCLOS*. PERCLOS is the percentage of time when eyes were 80% to 100% closed over a one minute interval. The term was first used in 1994 in the research [Wierwille et al. 1994] and it is considered to be a reliable and accurate determination of the driver's fatigue level [Dinges 1998]. Work in [Kozak et al. 2005] presented a finding that sleep deprived drivers have higher PERCLOS values, longer reaction time and higher tendency of making mistakes. These measures were carried out during a three-hour simulator driving test of fully rested as well as sleep deprived drivers.

Smart Eye AB company from Sweden introduced *Smart Eye Pro 5.10*[10]. Smart Eye is a 3D Remote Eye Tracking system that provides real-time monitoring of head position and angles, gaze direction, eyelid openness and pupil size. This solution promises fast calibration, automatic initialization and available data output. However, it uses eight cameras set up from different angles, what is quite costly. Similar commercial solution is called *EyeAlert*[11].

In work of [Smith et al. 2003], the authors presented a system for analyzing of a driver visual attention. Their method starts with detection of facial features like lips, sides of the face and even a yawning using color characteristics. Eyes are tracked using the combination of three strategies. First, an image intensity of the eye region is used to locate the pupil. Second, skin color information is used to find eyes, if necessary. Third, eyes can be tracked by an affine tracker, which computes affine transformation using the location of the eye in previous frames. After that, the system determines the bounding box of the face to calculate the head rotation so that eye-blink and eye closure can be determined. The method was tested under various daylight conditions. The system does not belong to real-time algorithms.

In [Brandt et al. 2004], a visual driver surveillance system based on eye-blink detection was introduced. Blinks are detected using an analysis of the optical flow in the eye region. The system is also not considered as real-time.

In [Ji et al. 2004], authors developed an online non-intrusive monitoring prototype for driver's fatigue detection. For the estimation of the fatigue level several software implementations were used: real-time eye tracking, eyelid movement parameters computation, eye-gaze estimation, facial-pose determination, facial expression analysis. Eye tracker is based on the combination of the *Kalman filter* and *mean shift tracking*. Kalman filter predicts a bright pupil position, but the trace can be lost after sudden head movement or unwanted face rotation. Therefore, it is supported with the mean shift tracker, which works with an intensity distribution of the eye region. However, the mean shift tracker does not have a capability to correct itself, hence the combination. System uses two cameras with an infrared illuminator to brighten up the driver's face. Therefore, the prototype is based on a grayscale image.

A similar solution was presented in [Bergasa et al. 2006]. Their prototype of a computer vision system for driver's vigilance monitoring achieves high detection performance in the PERCLOS estimation. The system measures ocular parameters by fitting of two ellipses using the modification of the algebraic distance algorithm for conics approximation [Fitzgib-

[10] `http://www.smarteye.se/productseye-trackers/smart-eye-pro-2-6-cameras` [last access: 22.8.2013]

[11] `http://www.eyealert.com/` [last access: 22.8.2013]

bon – Fisher 1995]. This algorithm is available in *OpenCV*[12] library. Other tested parameters are eye closure duration, blink frequency, nodding frequency, face position, and fixed gaze. Monitoring was tested on ten video sequences of real driving situation mostly at night and it achieved 93.12% correctness of PERCLOS results.

Work of [D'Orazio et al. 2007] proposed a solution for driver's sleepiness monitoring which does not consider skin-segmentation or frame background. It uses *Hough transform* and eye validation algorithm based on neural networks to detect right eyes' positions. The second step of the method is the behavior analysis. During initial moments of observation, a normal-behavior model is built to identify driver's normal act behind the steering wheel. This model is characterized by head movements and eye-blink data. As we can see, the solution does not require any face detection.

Authors in [Flores et al. 2008] came up with a non-intrusive system which detects the sleepiness of a driver using graysscale images. A face is detected using *Viola – Jones algorithm* [Viola – Jones 2001], which creates an imaginary rectangle around a face. The face is tracked using neural network-based tracker in conjunction with *The Condensation Algorithm (CA)*. CA was proposed in [Isard – Blake 1998] and it is a method for active contours tracking using a stochastic approach. This tracker detects the precise position of the face using a previous frame. Eye are detected using face anthropometric properties based on face database and further they are tracked by CA tracker. For eye state determination, *Support Vector Machine (SVM)* is used [Cortes – Vapnik 1995, Cristianini – Shawe-Taylor 2000]. SVM is avialable in *LIBSVM*[13] library. The system estimates the sleepiness by the amount of consecutive frames with closed eyes.

In papers [Garcia et al. 2010] and later in [Garcia et al. 2012], authors proposed a vision-based drowsiness detector, which can detect a driver in a realistic driving simulator. This detector is based on an infrared stereo camera (Figure 2.3). The camera is mounted behind the steering wheel so it can track the driver's face in frontal orientation to the camera. The detector constantly tracks eyes and estimates the PERCLOS. For the best estimation, PERCLOS values were compared to results of several psychological experiments. First, face and eyes are detected using Viola – Jones detector and detection failures are then corrected using Kalman filter. Subsequently, the sequence of filters is used to improve the frame quality so the PERCLOS can be estimated more accurately. PERCLOS is calculated from the ratio between the iris height in the frame and the nominal value assigned during a ten-second calibration at the start of the tracking. This detector reaches high recall (90.68%) and low false positive rates using their own database consisting of 25 hours of driving records. However, the system uses an infrared stereo camera what makes it an expensive solution with high hardware requirements.

Authors in [Lenskiy – Lee 2012] proposed another system based on measuring of eye-blink rate and eye closure duration. This method uses skin-color segmentation for finding faces in a frame and *Circular Hough transform* for iris detection [D'Orazio et al. 2004].

2.2.1 Blink waveform fatigue estimation

The work of [Suzuki et al. 2006] describes a system, which uses eye-blink information to determine the level of consciousness of a driver according to the blink waveform. The system

[12] `http://opencv.org/` [last access: 6.9.2013]

[13] `http://www.csie.ntu.edu.tw/~cjlin/libsvm//` [last access: 7.9.2013]

Figure 2.3: Garcia's detector consist of infrared stereo camera mounted behind the steeering wheel [Garcia et al. 2012].

uses an infrared pulsed light projection and a camera mounted in the inner rear mirror to capture the driver in various light conditions. Face and eyes are detected using three-layered neural network with four-direction edge characteristic detection method. The authors assume that neural network learning was performed on sample images with more than 80% visibility of an upper lip and both eyes. The learning process was performed more than 60000 times.

Subsequently, eyelids are detected as follows (Figure 2.4): First, an image is divided into several vertical sections. For each section, candidates for upper and lower eyelids are defined as the maximal and minimal differential values of the gray level distribution. These candidates are grouped in five sections, two of them are chosen to represent the upper and lower eyelid. All five sections are then used to calculate the *eye gap* – an average of distances between eyelid candidates. In fact, the eye gap is the degree of eye openness. This method is considered as robust, but authors admit that detection results can be affected by the varied shape of a human eye.

Using the eye gap, we can estimate the eye openness over time, i.e a blink waveform. Measures in this paper show that the eye gap decreases rapidly when the eye-blink starts. After the eye reaches the minimal value (eye is considered as closed), the eye gap starts to increase gradually. However, blink duration and blinking frequency also vary for different drivers. Therefore, the system can not use a method of the default threshold limit to classify eye-blinks. To determine the right start and end point of blinking, a zero-crossing of the blinking waveform is used (Figure 2.5). The method uses second order derivative of the waveform to assign zero-cross points.

Authors present a new and more robust method for the estimation of the consciousness level – *3-factor method*. The presumption of the driver's drowsiness starts with the calculation of these three ratios or factors, which are retrieved from the blink waveform:

- *Long blink ratio* – ratio of eye-blinks longer than a threshold to the total blink count.
- *Closure rate* – ratio of the total time when eyes were closed to the total time measured.
- *Blink rate* – number of eye-blinks per one minute.

Each of these three factors has its own correlation with the fatigue level of the driver. Thus, authors decided to estimate Y – the weighted sum of these factors (Equation 2.1), which indicates a strong relation to the consciousness of the driver. Value Y is calculated through

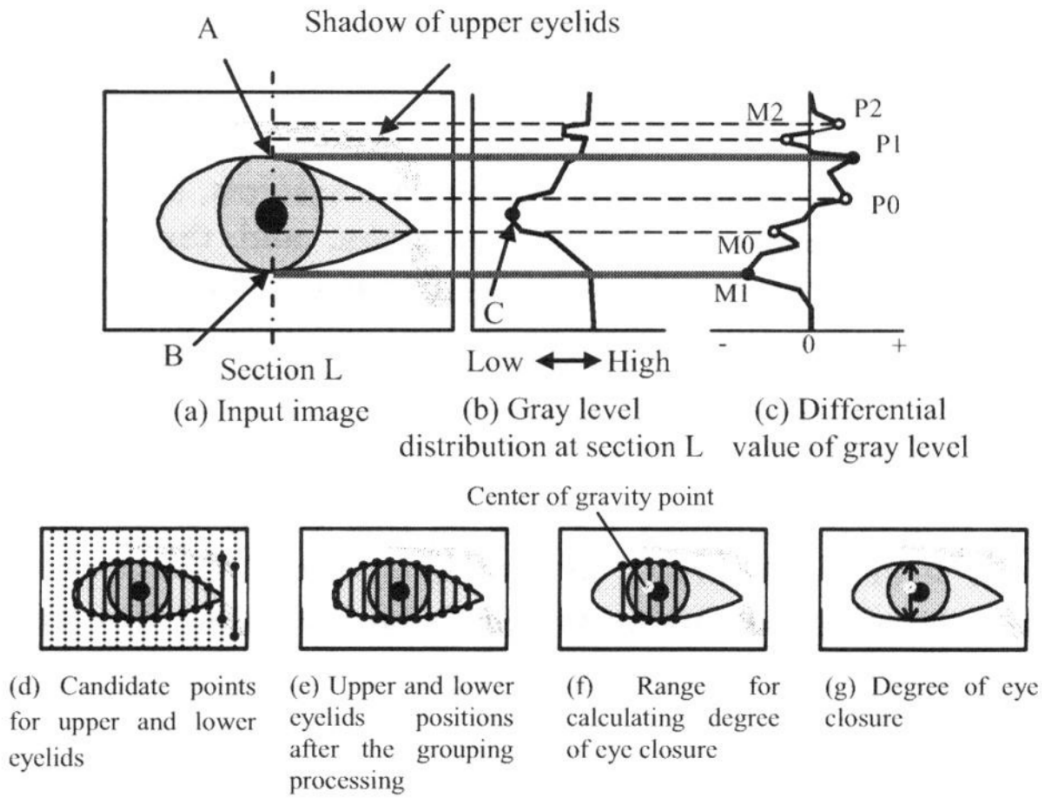


Figure 2.4: Eyelid extraction using the maximal and minimal values of the gray level distribution and sections grouping [Suzuki et al. 2006].

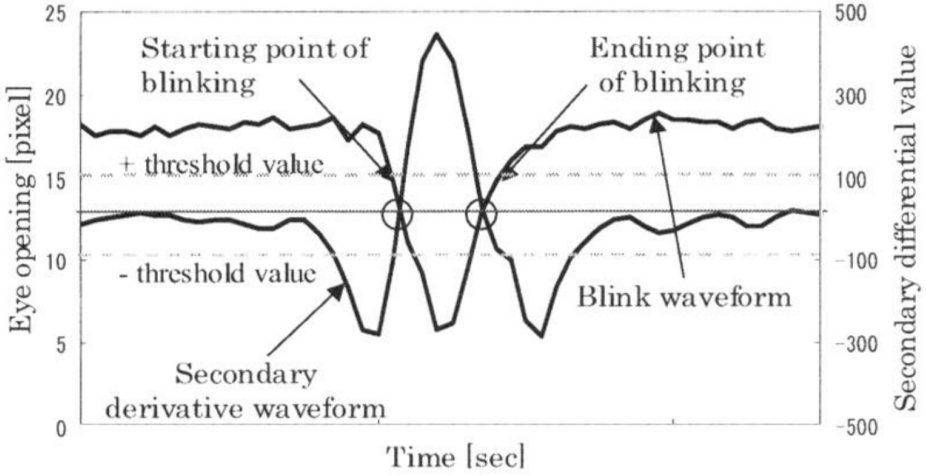


Figure 2.5: Zero-crossing of the blinking waveform with second order derivative of this waveform. We can see desired start and end points of the blink [Suzuki et al. 2006].

Table 2.1: Distribution of consciousness states according to Z_1 and Z_2 values with observed impressions [Suzuki et al. 2006].

State	Range	Facial impressions	Other impressions
Normal	$Z_1 < 0$	fast and constant blinks quick eye movement wide opened eyes	right posture no actions other than driving
Sleepy	$Z_1 \geq 0, \quad Z_2 < 0$	slow blinks narrow blinks	yawn frequent mumbling frequent posture change
Very Sleepy	$Z_1 \geq 0, \quad Z_2 \geq 0$	long blinks	shaking and bowing head

the multiple regression analysis, for each individual extra.

$$Y = w_1 \cdot L + w_2 \cdot C + w_3 \cdot B \tag{2.1}$$

where w_1, w_2 and w_3 are unique weights of each individual, L is the Long blink ratio, C is Closure rate and B is Blink rate.

However, the human body and its physiology can change with time so Y can reflect some inaccuracies and also can contain a noise. Thus, the authors use weighted moving average Z (Equation 2.2) of some consecutive Y values. Tests of Z values prove that it is a more accurate parameter for determination of the fatigue level.

$$Z = \sum_{k=0}^{n} a_k \cdot Y_{i-k} + b \tag{2.2}$$

Y_{i-k} value represents Y at time $i - k$. Z as the result can be considered to be a status of the driver.

Weights estimation was calculated using discriminant analysis proposed by the authors. Moreover, this method distinguishes two variables Z_1 and Z_2 (Equation 2.3), that divides consciousness into three states (Table 2.1).

$$\begin{aligned} Z_1 &= \sum_{k=0}^{n} a_k \cdot Y_{i-k} + b \\ Z_2 &= \sum_{k=0}^{n} c_k \cdot Y_{i-k} + d \end{aligned} \tag{2.3}$$

The authors further analyzed the system for $n = 1, 2$ and 4. The best results were achieved for $n = 4$, when the highest correlation between presumed value and the state of drivers was reached. The method was experimentally verified on fifteen subjects and results were compared to a visual observation of the consciousness level. However, authors did not present any data related to the computational time or the effectiveness of the solution.

2.2.2 Horizontal symmetry calculation

The automatic drowsy driver monitoring and accident prevention system presented in [Danisman et al. 2010] uses the pupil analysis based on *Horizontal Symmetry Calculation* (HSC).

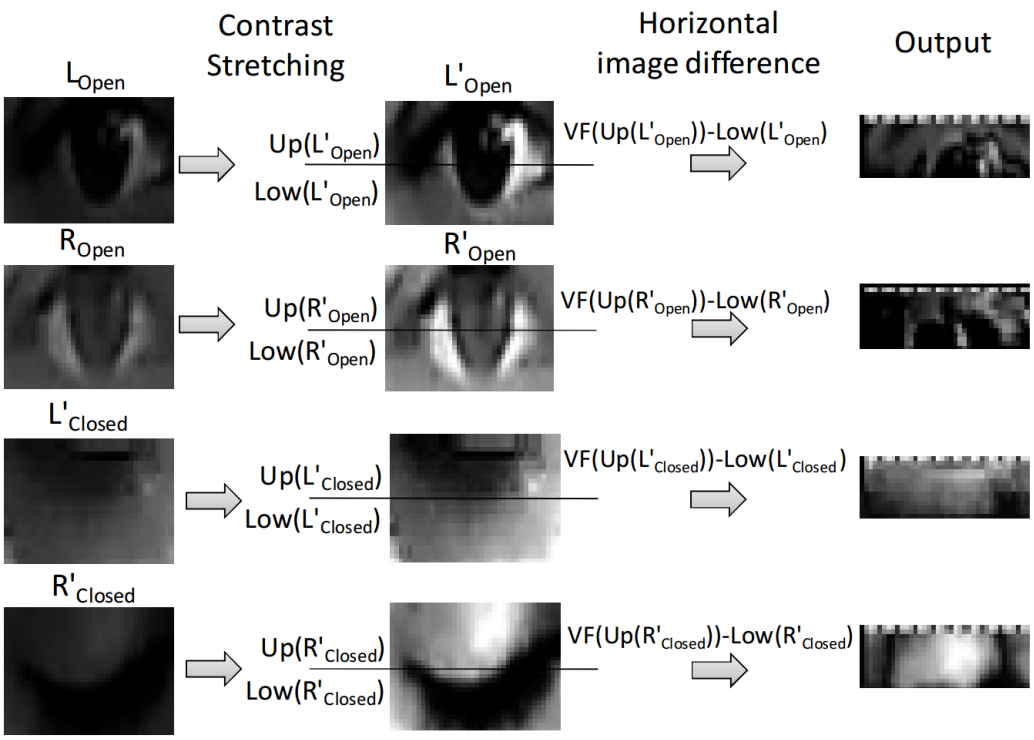


Figure 2.6: Horizontal Symmetry Calculation to determine the eye state. The method uses the difference image of the upper and lower eyelid half in the eye image [Danisman et al. 2010].

The system receives input color frames from a video camera and continuously measures the eye-blink duration of a driver. Detection starts with a face detection using Viola – Jones algorithm. Subsequently, the neural network-based detector is used for the precise eye pupil position estimation. This algorithm is available in *STASM*[14] library – C++ software library for finding face features in faces. STASM is presented as a variation of Active Shape Model mentioned in the work of [Cootes et al. 1995]. After pupil detection, the head rotation angle is calculated using the vertical position of both pupils. If eye detection in the next frame fails, the angle helps to determine the right face and eye position.

Consequently, pupils are analyzed using HSC to determine whether the eyes are *Open* or *Closed*. First, *normalization* (often called contrast stretching) is performed to improve the image quality. Then pupil region is divided in two halves using the axial symmetry around the line crossing centers of both pupils (Figure 2.6). Created halves represents the upper and lower eyelid regions. These halves are fold over. If the eye is open, then the folded fragment preserves its circular shape symmetry, unlike the closed eye. Therefore, difference between upper half and lower half is estimated (Equation 2.4), where I_{dif} is the horizontal symmetry of the eye, I' is the normalized image, $VF\left(Up\left(I'\right)\right)$ is the vertically flipped upper half of the image and $Low\left(I'\right)$ is the lower half of the image.

$$I_{dif} = VF\left(Up\left(I'\right)\right) - Low\left(I'\right) \tag{2.4}$$

After that, the cumulative sum I_{sum} of the difference image I_{dif} for the whole image is

[14] `http://www.milbo.users.sonic.net/stasm/` [last access: 28.8.2013]

calculated (Equation 2.5).

$$I_{sum} = \sum_{i=0,i=j}^{width,height} I_{dif}\,(i,j) \tag{2.5}$$

The eye state I_{state} is determined according to the threshold limit T and the I_{sum} value (Equation 2.6).

$$I_{state} = \left\{ \begin{array}{ll} Open & I_{sum} < T \\ Closed & I_{sum} \geq T \end{array} \right\} \tag{2.6}$$

If the I_{sum} reaches higher values, a brighter output image of the eyes is created, what results in the Closed eye state. Authors selected default threshold limit $T = 12000$, what achieves the lowest false positives according to their own experiments.

Authors defined three drowsiness states according to [Caffier et al. 2005] based on the *blink duration* (BD) as follows:

- *Awake* with BD < 400 ms,
- *Drowsy* with BD > 400 ms and BD < 800 ms,
- *Sleeping* with BD > 800 ms.

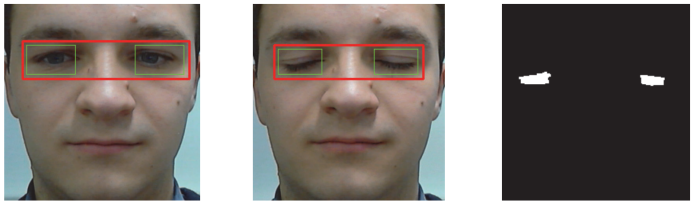
This means that a car going 90 km/h passes only 20 meters until an alarm signal is given to the driver in the *Sleeping* state. The mentioned algorithm was tested on *ZJU*[15] database and it achieved 94.8% accuracy for eye-blink detection.

2.2.3 Pixel difference blink estimation

Work presented in [Kurylyak et al. 2012] solves the problem of another non-intrusive vision based system which detects eye-blink to determine the fatigue level of a driver. The specialty of this approach is that it does not detect face, but it starts right away with eye detection using Viola – Jones algorithm. To speed up the eye-tracking algorithm, the authors assume that person does not move significantly away from the camera during the tracking. Therefore, eyes are repeatedly detected only if an important movement within the eye region is noticed. Otherwise, the eye region from the previous frame is used. Even if a small movement is detected, new eye regions are considered from the enlarged area from the previous frame. Thus, the system avoids processing of a whole frame to increase the performance. Eye blink determination starts, when the difference between the current and previous frame is estimated for the each eye (Figure 2.7). If the number of different pixels is higher than a default threshold, the current frame is considered to be the frame when blink appeared. Boolean value $F(t)$ (Equation 2.7) determines, whether the blink was found or not. If the $F(t) = 1$, the blink appeared.

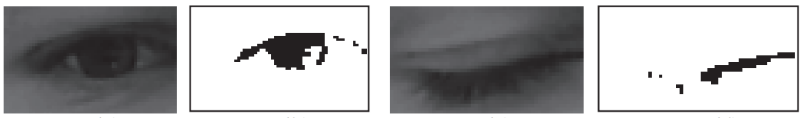
$$F\,(t) = \left\{ \begin{array}{ll} 1, & \left(\frac{\sum mask_r(t)}{\mathrm{CARD}(mask_r(t))} > T \quad \text{and} \quad \frac{\sum mask_l(t)}{\mathrm{CARD}(mask_l(t))} > \frac{1}{2}T\right) \quad or \\ & \left(\frac{\sum mask_r(t)}{\mathrm{CARD}(mask_r(t))} > \frac{1}{2}T \quad \text{and} \quad \frac{\sum mask_l(t)}{\mathrm{CARD}(mask_l(t))} > T\right) \\ 0, & \text{otherwise} \end{array} \right\} \tag{2.7}$$

[15] `http://www.cs.zju.edu.cn/~gpan/database/db_blink.html` [last access: 27.8.2013]



(a) (b) (c)

Figure 2.7: Difference between first frame with open eyes (a) and second frame with closed eye (b) forms the difference image (c). Red rectangle are the eye region and green rectangles represents detected eyes [Kurylyak et al. 2012].



(a) (b) (c) (d)

Figure 2.8: Grayscale image of an open eye (a) is converted to a binary image (b). Grayscale image of a closed eye (c) is converted to a binary image (d). Degree of eye openness can be determined from the binary images [Kurylyak et al. 2012].

Values $mask_r\,(t)$ and $mask_l\,(t)$ are the difference masks between the current and the previous frame of the right and the left eye, $CARD$ is the cardinality of the input mask, T is the default threshold. T is estimated as a minimal number of pixels, which must differ in a frame to classify blink. In this paper, the predefined threshold value T is determined experimentally as 0.15. As we can see, the number of different pixels of one eye is compared to the half of the threshold $(\frac{1}{2}T)$ in any case. This approach considers two things: First, a non-uniform change of the illumination for each eye in consecutive frames is not rejected. Second, a voluntary blink is excluded from further analysis, when one eye is open but another is closed (eyes do not blink simultaneously).

The analysis of the frame difference indicates a change that happened. However, it does not say anything about the direction of the change, whether the eye was opening or closing. To determine the movement direction, the system uses analysis of consecutive frames with vertical and horizontal projections. Vertical (horizontal) projection is the sum of pixels in each row (column) of an image. First, the difference image is converted to a grayscale and the Otsu's algorithm [Otsu 1979] is used to threshold the image to a binary image. Then vertical projection with its maximal value is used to determine a degree of eye openness (Figure 2.8).

Authors presented that data should be further analyzed to avoid detecting a voluntary blink. Thus, corrections are performed:

- detection of eye closure longer than one second could not mean a spontaneous closure, therefore an alarm is given,
- detection of two consecutive blinks with short repeat period is considered as one blink,
- detection of more than two following eye blinks is considered as voluntary, so they are eliminated from the further analysis.

Subsequently, the fatigue level can be determined according to involuntary eye-blink infor-

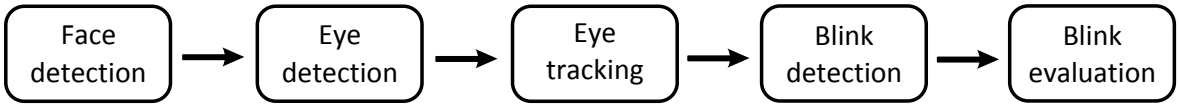


Figure 2.9: Workflow diagram of proposed application.

mation, e.g. ratio between the closed eyes and open eyes in a particular period of time can be measured. The detector can detect the driver successfully from the one meter distance and the horizontal head rotation up to 30 degrees. Even though the system was not tested in real driving situations, it can be certainly used in vehicles. It was tested with five subjects in a daytime and even at night. It achieved an average accuracy of 94.48% for the test set of 5 videos of people captured in daytime and nighttime conditions.

2.3 Summary

Currently, there are many microsleep and fatigue detection systems. Some of them are non-eye based – they are focused on detection through monitoring of driver's and car behavior. Usually, they are already available in vehicles – *Driver Alert Control (DAC)* by *Volvo*, *Attention Assist* by *Mercedes-Benz* or *Lane Assist* by *Volkswagen*.

Other systems are focused on monitoring of human biological features connected to incoming microsleep. Commercially available are *Stopsleep* or *Holux's* hypo-vigilance/fatigue detector. Systems specialized on facial features tracking were presented in [Vural et al. 2007, Hariri et al. 2011].

The most important type of detection in this field is eye-based. Eye-based detection systems detect fatigue and microsleep using eye or pupil tracking with following eye-blink frequency measuring or PERCLOS estimation. Many of them use infrared cameras or light projections and work with a grayscale image – [Ji et al. 2004, Bergasa et al. 2006, Suzuki et al. 2006, Garcia et al. 2010, Kurylyak et al. 2012], other use skin-color segmentation or different detection methods and they work with a colored image – [Smith et al. 2003, D'Orazio et al. 2007, Danisman et al. 2010, Lenskiy – Lee 2012].

During the analysis of this field we did not find any microsleep detection system, which would use a smartphone. Therefore, we want to propose a simple and available eye based detection system, which would use a smartphone with its front camera and real-time algorithms based on a grayscale image. We decided to focus on an eye-blink detection and analysis, which would lead to an effective and reliable microsleep detection.

By analyzing the afore mentioned methods it was derived that fatigue or microsleep detection can be decomposed into the workflow depicted in Fig 2.9. All parts of the fatigue detection are essential. To use fast and accurate methods, it is necessary to analyze particular steps individually. Thus, in the Chapter 3, different approaches of face and eye detection and tracking algorithm are considered and Chapter 4 provides eye-blink detection algorithms.

Chapter 3

Face and Eye Detection and Tracking

Eye based microsleep detection requires constant monitoring of human face and eyes. Therefore, it is important to use real-time algorithms. Successful eye-blink detection is preceded by face and eye detection and eye tracking. In this chapter we analyze the chosen algorithms, which offer such solutions.

3.1 Viola – Jones Face Detection

Viola – Jones object detection framework [Viola – Jones 2001; 2004] is one of the most popular breakthroughs in object detection. It uses a cascade of boosted simple classifiers working with Haar-like features [Papageorgiou et al. 1998]. The method is suitable for a fast detection of pre-learned objects, but it is mostly used for face detection. The algorithm presented by Viola and Jones is implemented in OpenCV[1] library. It can be also used to detect parts of a face, e.g. eyes.

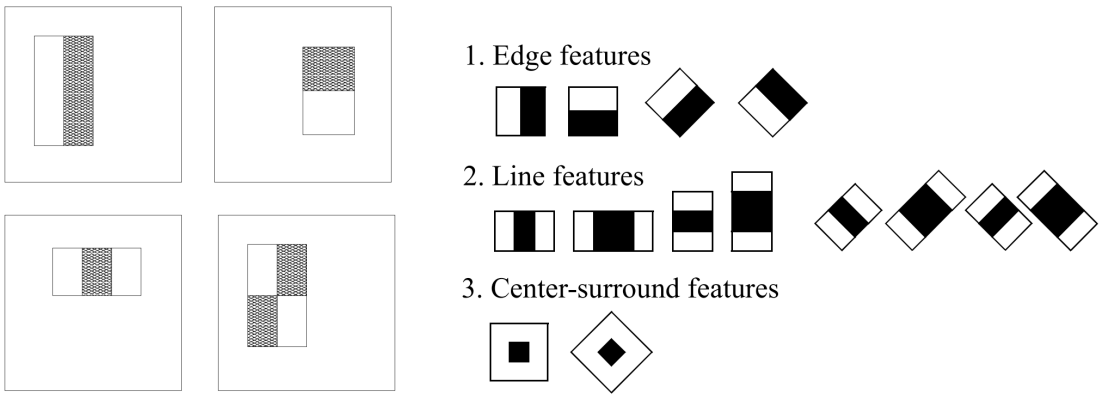
The proposed algorithm uses a sliding-window approach, which is moving over the image. For each sub-window a Haar-like (Figure 3.1) feature is computed. After that, the current feature is compared to the set of learned features in the cascade structure to reject or accept the object.

The authors came up with fast and accurate detection algorithm. They mentioned three improvements, which noticeably speed up the detection algorithm:

- *Integral Image* – increases the speed of feature computing.
- *Learning method based on AdaBoost* – speeds up re-sorting of the classifier set.
- *Cascade structure* – greatly increases speed of feature classifying.

The proposed algorithm is feature-based, because it considers features instead of pixels to achieve better performance. It uses Haar-like features (Figure 3.1a): two-rectangles, three-rectangles and four-rectangles. These features represent the difference between the sum of pixels of specific inner rectangles. In work of [Lienhart – Maydt 2002], the authors proposed a novel set of Haar-like features to use in Viola – Jones algorithm. New rectangles are rotated by 45 degrees (Figure 3.1b). This improvement decreases average false alarm rate by 10%.

[1] `http://docs.opencv.org/modules/objdetect/doc/cascade_classification.html` [last access: 25.8.2013]



(a) Examples of original rectangles proposed by Viola and Jones [Viola – Jones 2001].

(b) A novel set presented by Lienhart and Maydt, which contains rotated rectangles [Lienhart – Maydt 2002].

Figure 3.1: Haar-like features.

For the rectangles calculation the *integral image* is used, which results in significant speed improvement. The integral image at positions x, y contains the sum of the pixels located left and above of the x, y, inclusive (Equation 3.1), where $ii\,(x, y)$ is the integral image and $i\,(x, y)$ is the original image.

$$ii\,(x, y) = \sum_{x' \leq x, y' \leq y} i\,(x', y') \tag{3.1}$$

The integral image can be calculated using only a few operations at the current region. Once it is estimated, all Haar-like features can be computed in constant time. Moreover, using the recurrent equations, the integral image can be calculated only in one pass over the original image.

The paper further proposes a learning method based on feature selection and classifier training using AdaBoost. AdaBoost or Adaptive Boosting was first introduced by Freund and Schapire in [Freund – Schapire 1997]. The method uses sub-window 24×24 pixels. Each sub-window contains many rectangles, which slow the whole process. Many of them can be excluded from further analysis. Thus, AdaBoost is an effective method that selects only critical and important features from a rectangle. It is based on modification of weak classifiers into one strong classifier.

One of the major speed improvements of Viola – Jones algorithm is *Cascade classifier*. A cascade rejects lots of negative sub-windows. Vast majority of sub-windows are unimportant, because they represent non-objects, e.g. the background. Cascade uses set of simple classifiers to sort sub-windows in the first step. After that, series of more complex classifiers can be used to analyze their perspective sub-windows. Single stages of the cascade are made of classifiers using AdaBoost. In work, a complete cascade has 38 stages with more than 6000 features. In average, just 10 stages where used in the test on 507 faces (75 million sub-windows). System was compared to the Rowley's implementation of neural network-based face detector [Rowley et al. 1996] and it achieved 15 times faster detection.

Many approaches have been presented to detect face and eyes using AdaBoost algorithm since Viola and Jones. Eye detection improved by 2D cascaded AdaBoost classifier framework based on the work in [Niu et al. 2006]. Other face and eye detection using learning

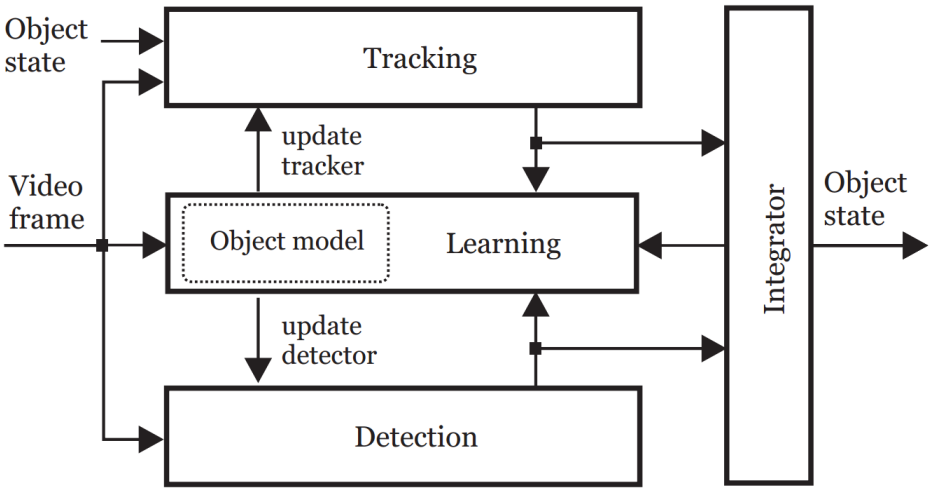


Figure 3.2: The block diagram representing the framework architecture [Kalal et al. 2012].

method to extract features based on recursive nonparametric discriminant analysis was proposed in [Wang – Ji 2007]. Work in [Choi et al. 2011] mentioned face and eye detection using AdaBoost training with Modified Census Transform (MCT) [Froba – Ernst 2004] based features.

3.2 TLD Object Tracking

A novel tracking framework TLD – Tracking-Learning-Detection was proposed in [Kalal et al. 2009] and later with improvements in [Kalal et al. 2012]. The framework includes three parts (Figure 3.2):

- *tracking* – frame-to-frame tracker tracks an object while it is visible.
- *detection* – object detector tries to localize an object so the tracker could be reinitialized, if needed.
- *learning* – P-N learning part estimates detection errors and it tries to update the detector so that it will not make same mistakes in the future.

The authors proposed Face-TLD [Kalal et al. 2010b], what is a human face tracker in an unconstrained video based on TLD.

3.2.1 Detection

Detector scans the current frame using the sliding-window and it decides whether the object occurs in the frame – it is a binary classifier. Considering all possible object scales, there are many possible patches to be scanned. Therefore, patches need to be classified fast. The system uses three stages which rejects or accepts patches containing three stages:

- *patch variance* – this stage rejects about 50% of patches using gray-value variance filter.
- *ensemble classifier* – in this stage patches pass through a pixel comparison.

- *nearest neighbor (NN)* – remaining patches are classified using the NN classifier.

Detector uses an *Object model* – a data structure representing an object, which contains a collection of positive and negative patches $M = \left\{p_1^+, p_2^+, dots, p_m^+, p_1^-, p_2^-, dots, p_n^-\right\}$, where p_1^+ is the first positive patch and p_1^- is the first negative patch. Detector further uses a model to measure the similarity of the current patch and surrounding patches in the model.

3.2.2 Tracking

The tracker is based on a *Median-Flow tracker* [Kalal et al. 2010c] proposed by the same authors. It is frame-to-frame object tracker, which provides tracking object movement until it is visible. Naturally, failure probability of the tracker is quite high.

The tracker determines number of points that are shifted from the bounding box according to the previous frame. The bounding box is an imaginary rectangle which surrounds the desired object. The authors use the *Lucas – Kanade tracker* [Tomasi – Kanade 1991] and its pyramidal implementation [Bouguet 2000]. The Median-Flow tracker fails when the object moves out of the frame or when fast movement occurs. To determine the trace loss, median displacement is used. The tracker is considered to be failed if $\text{median}\,|d_i - d_m| > 10$ pixels, where d_i is the displacement of the single point and d_m is the median displacement. Such a displacement causes that the bounding box does not appear. Therefore, it is necessary to find the object as soon as it reveals itself again. TLD uses *Integrator* which merges Detector and Tracker and it combines their bounding boxes. The result of this conjunction is an output bounding box. The authors realized that this method can achieve real-time and long-term tracking.

3.2.3 Learning

The main contribution of the work in [Kalal et al. 2012] is the improvement of the learning part of TLD. P-N learning is used to monitor and evaluate the detector, identify its errors and correct the detector to the avoid same errors in the future. Learning uses *P-experts* and *N-experts*. P-experts identify the false negatives and N-experts identify the false positives. The experts contribution is that they extend the training set with positive and negative training examples. Training set is used to train the classifier (Figure 3.3). The task of P-experts is to generalize the detector and obtain new possible object variations. This expert exploits the fact that objects move along a trajectory. It tries to find reliable parts of the trajectory which results in new positive samples. N-expert's task is to obtain examples of background or obstacles or other negative elements. The main idea is that the tracked object occupies only one place in the frame so many negative elements can be discarded very quickly. The P-N learning is widely analyzed in [Kalal et al. 2010a].

TLD has open implementations available[2][3][4]. Therefore, it can be freely used to track the driver – his or her face or eyes.

[2] `http://gnebehay.github.io/OpenTLD/` [last access: 28.8.2013]
[3] `https://github.com/zk00006/OpenTLD` [last access: 28.8.2013]
[4] `http://libccv.org/doc/doc-tld/` [last access: 28.8.2013]

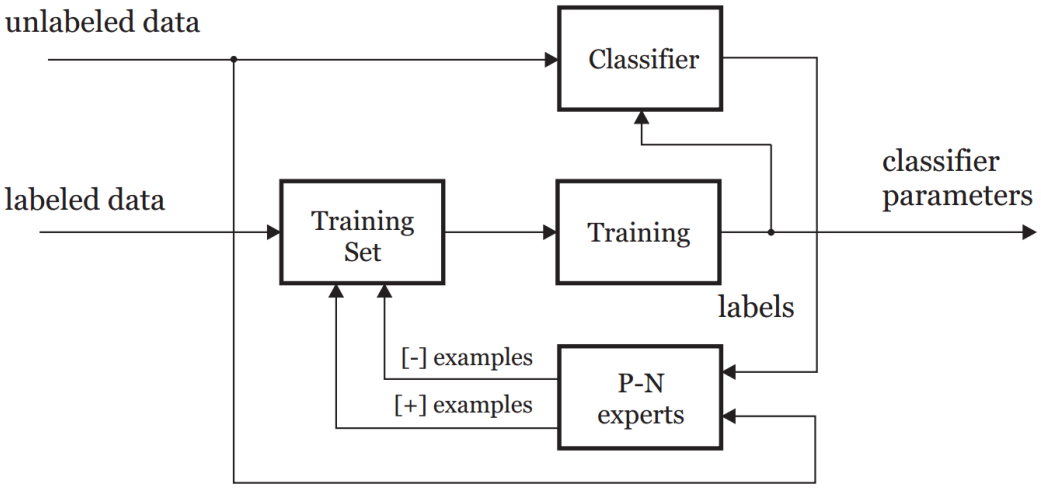


Figure 3.3: The block diagram of the P-N learning process. Training process starts with inserting the labeled data to the training set. This set then trains the classifier using the supervised learning. The classifier classifies the unlabeled set that is then evaluated by P-N experts. Experts revise the set and add examples to the training set [Kalal et al. 2012].

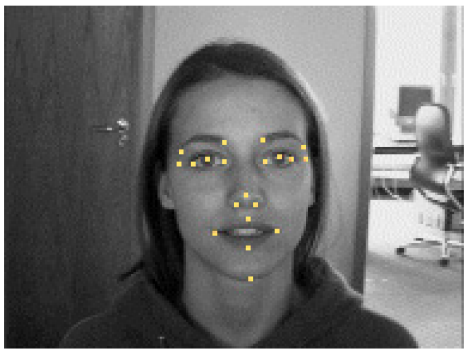


Figure 3.4: The face with correctly detected and highlighted facial landmarks [Milborrow 2007].

3.3 Active Shape Model Face Detection

One of the face detection methods is Active Shape Model (ASM) proposed in [Cootes et al. 1995]. ASM is used to localize facial features of a human face. This approach is implemented in C++ *STASM*[5] library, which provides an improved variation of ASM presented in [Milborrow 2007, Milborrow – Nicolls 2008].

ASM is first trained on a set with manually labeled facial landmarks marked before training. A landmark is a distinguishable face feature, e.g. right pupil (Figure 3.4). The two main ideas of landmarks finding are:

- Attempt to identify every facial landmark independently from other landmarks.
- Correct the position of founded landmarks based on some predefined relations.

According to these two points, ASM consists of two sub-models: *shape model* and *profile model*.

[5] `http://www.milbo.users.sonic.net/stasm/` [last access: 28.8.2013]

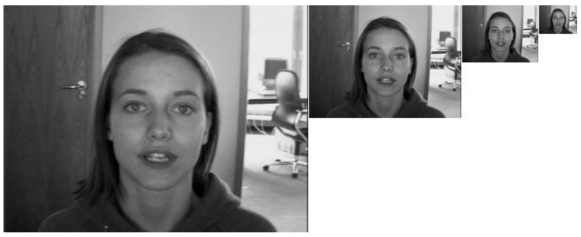


Figure 3.5: Image pyramid – each image is a quarter from previous one [Milborrow 2007].

The profile model characterizes neighborhood of each landmark – it works locally. It is used to determine how the area around a landmark should look like. During the training this area is processed and the model is created. After that, when the detector tries to estimate a landmark position, an actual landmark neighborhood is compared to the profile model. The result is that landmark point is set to the most fitting position. This new position is also called the *suggested shape*.

The shape model tries to correct the global shape of the landmarks – it works globally. Since the profile model can be inaccurate, it can disorder points globally. The shape model's task is to organize landmarks naturally so the result is the lifelike face. This model consider shape distortions and it generates a shape $\hat{x}$ using an average shape from the training (Equation 3.2),

$$\hat{x} = \bar{x} + \Phi\vec{b} \tag{3.2}$$

where $\bar{x}$ is the mean shape, Φ is a matrix of selected eigenvectors and $\vec{b}$ is a parameter vector.

Searching in ASM is provided by *image pyramids*. Image pyramid is down-scaled version of the previous image (Figure 3.5). The detector searches in each pyramid level starting from the smallest image. Then it continues to upper images with best face shape from the previous level. Each level uses the same shape model, but the profile model is special. ASM was used in [Yang et al. 2012] where the authors presented an eye tracker based on a conjunction of ASM and KLT tracker based on [Shi – Tomasi 1994]. Further, the tracker is used to track eyelids and then to determine the fatigue level. Facial landmarks extraction was also presented in [Uřičář et al. 2013] where the authors use Deformable Part Models to form a face shape and Structured Output SVM [Tsochantaridis et al. 2005] for learning features parameters of the detection. This solution provides an open source library called Flandmark[6].

ASM based detector is a good choice for global face detection and for facial feature estimation. However, we want to focus on eye region or eye pupil detection. Thus, detection of other features could decrease the performance of the detector.

3.4 Pupil Localization

In work [Ciesla – Koziol 2012], the authors compared three approaches used for pupil localization. All algorithms were implemented and tested using static images and also web-camera images. Algorithms are considered to be real-time.

[6] `http://cmp.felk.cvut.cz/~uricamic/flandmark/` [last access: 29.8.2013]

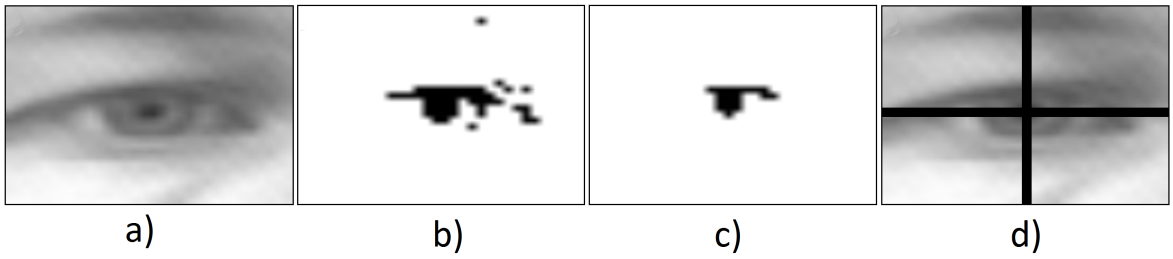


Figure 3.6: Original input image (a) is binarized using CDF (b). Afterwards, it is filtered using erosion (c). The pupil center location can be then estimated (d) [Asadifard – Shanbezadeh 2010].

3.4.1 Cumulative Distribution Function

The first approach called Cumulative Distribution Function (CDF) was proposed in [Asadifard – Shanbezadeh 2010]. The method tries to localize the center of a pupil using CDF, but it demands the frontal position of a human face.

After the face is detected using Viola – Jones algorithm, the method divides the face region into two vertical halves – top and bottom. The top half is then divided into left and right horizontal halves. Each half is considered to be a region of interest (ROI) and it contains one eye (Figure 3.6a). Working with ROI speeds up the computation. ROI is filtered using CDF (Equation 3.3), where $P(w)$ is a probability of existence of the pixel with intensity w.

$$\mathrm{CDF}\,(r) = \sum_{w=0}^{r} P\,(w) \tag{3.3}$$

The filter changes each pixel intensity $l\,(x, y)$ according to the threshold 0.05, which was found experimentally (Equation 3.4).

$$l'\,(x, y) = \left\{ \begin{array}{cc} 255, & \mathrm{CDF}\,(l\,(x, y)) \leq 0.05 \\ 0, & otherwise \end{array} \right\} \tag{3.4}$$

The result is a binarized filtered image (Figure 3.6b). Afterwards, an erosion morphological operation is applied to eliminate singular white pixels (Figure 3.6c). The pupil center is then estimated by comparison with the original image (Figure 3.6d).

3.4.2 Projection Functions

Another approach was presented in [Zhou – Geng 2004]. The authors use vertical and horizontal projection functions of the pixel intensities to determine the pupil location within the eye region.

The input image of the eye region can be divided into several subsections, which are bordered accroding to the significant change in vertical or horizontal projection intensity (Figure 3.7). As we can see in the horizontal projection direction, there are four rapid intensity changes. x_1 and x_2 represent corners of eyelids – these are not so important for further analysis. However, x_3 and x_4 are important because these points border the iris in a horizontal way. Same changes can be watched in the vertical projection direction. The iris is vertically bordered

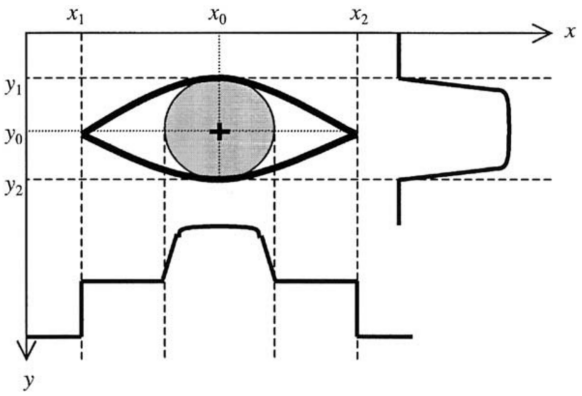


Figure 3.7: Relation of projection functions to the pupil location. Axis x represents horizontal projection with five significant points. Point x_0 is the center of the pupil in the horizontal direction. Axis y means vertical projection and y_0 is the center of the pupil in the vertical direction [Zhou – Geng 2004].

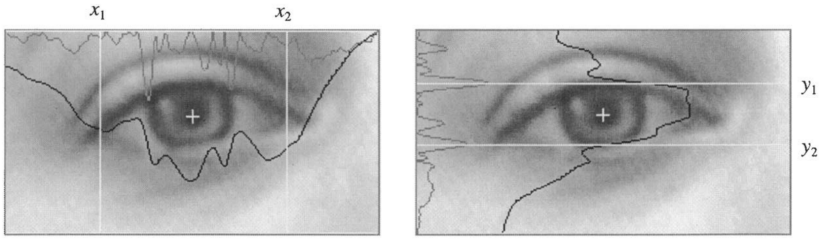


Figure 3.8: Usage of image projection function to locate eyelid corners in the horizontal direction (x_1, x_2) and in the vertical direction (y_1, y_2). Darker curve represents a projection function and brighter curve means the first derivative of the projection function [Zhou – Geng 2004].

by y_1 and y_2. Therefore, the center of the pupil (x_0, y_0) can be computed geometrically (Equation 3.5).

$$x_0 = \frac{x_3 + x_4}{2}, \quad y_0 = \frac{y_1 + y_2}{2} \tag{3.5}$$

The solution depends on usage of the most effective projection function (Figure 3.8). In this paper authors chose *General Projection Function (GPF)*, which combines *Integral Projection Function (IPF)* and *Variance Projection Function (VPF)*. GPF combination is presented in Equation 3.6, where $\mathrm{GPF}_h(y)$ is horizontal GPF and $\mathrm{GPF}_v(x)$ is vertical GPF of $I(x, y)$ intensity of a pixel at the location (x, y).

$$\begin{aligned} \mathrm{GPF}_h(y) &= (1-\alpha)\,\mathrm{IPF}_h{}'(y) + \alpha \mathrm{VPF}_h(y) \\ \mathrm{GPF}_v(x) &= (1-\alpha)\,\mathrm{IPF}_v{}'(x) + \alpha \mathrm{VPF}_v(x) \end{aligned} \tag{3.6}$$

Contribution of both projection functions IPF and VPF is controlled with a parameter α, where $0 \leq \alpha \leq 1$. The best result was achieved for $\alpha = 0.6$. Projection functions are quite popular. An eye detection system based on color characteristics and a projection function was proposed in [Kumar et al. 2002].

Figure 3.9: Input eye image (a) and eye edges detected using Canny algorithm (b) [Asteriadis et al. 2006].

3.4.3 Edges Analysis

Edges analysis method was proposed in [Asteriadis et al. 2006]. An input image is filtered using Gaussian blur filter and then it is processed by edge detection algorithm developed by Canny [Canny 1986]. The output is a binary image (Figure 3.9). Afterwards, vertical and horizontal lines sharing the highest number of points with edges are considered. However, this method is not as accurate as the previous methods.

3.5 Summary

We described successful the Viola – Jones algorithm, which uses a cascade of classifiers based on Haar-like features. Then we looked at the problem of real-time and long-term tracking provided by TLD tracker and facial landmarks recognition problem solved by Active Shape Model. Other solutions of pupil localization were proposed using CDF algorithm, Projection functions and Edge analysis. Real-time eye tracking offers us good conditions to use effective eye-blink detection algorithm to estimate eye-blink information. The chosen algorithms of this problem are discussed further in Chapter 4.

Chapter 4

Eye-blink detection

When it comes to microsleep detection, eye-blink detection is essential. Eye-blink and eye closure information have a strong relation to fatigue and microsleep detection [Dinges et al. 1998]. It is known that blink frequency [Stern et al. 1994] and blink duration [Häkkänen et al. 1998, Schleicher et al. 2008] can be very good signs of incoming microsleep. In this chapter we analyze some of the most used methods and algorithms, which recognize eye-blink.

4.1 Correlation Score

In the paper [Grauman et al. 2001] authors present a real-time vision based system, which automatically detects eye-blink and measures the duration of eye-blink to control a computer. The system is proposed to help disabled people. The idea is to control mouse clicks with long voluntary blinks and to distinguish them from involuntary blinks.

Eyes are detected using the correlation coefficient over time. The system does not need any manual initialization, because it can be initialized all by itself using initial blinks. This paper offers the real-time and non-intrusive system, which does not require face detection or an additional light source. It can work continuously at 30 frames per second.

During first moments of working, an eye location is automatically obtained from the motion analysis of the blink i.e. *blinklike* motion. First, a bi-directional image $[D]_{i,j}$ (Equation 4.1) is created, which expresses the difference of brightness for all pixels (i, j) of two consecutive frames.

$$[D]_{i,j} = \left| \left([F_t]_{i,j} - [F_{t-1}]_{i,j} \right) \right| \tag{4.1}$$

F_t is a current frame and F_{t-1} is a previous frame. Subsequently, thresholding method forms a binary image, which indicates regions of motion, e.g. blink. The image is further processed through the erosion, which removes all captured redundant motions (Figure 4.1). Several filters are then used to remove pairs of non-blink representatives. Weighted Mahalanobis distance further eliminates all other non-blink motion pairs. In this part also an open eye template is acquired.

After a successful eye localization, the system learns how an open eye appears so it can be used in the next phase of blink detection, when the eye closure is estimated. Tracking

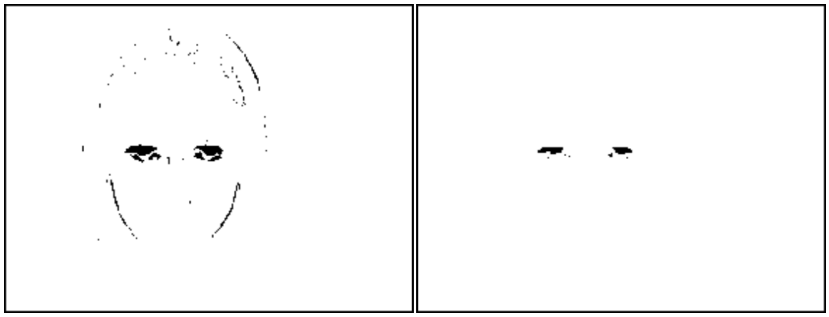


Figure 4.1: Left picture is the original binary image representing a motion change in two consecutive frames. Right picture is a binary image processed through the erosion to remove redundant motion pixels [Grauman et al. 2001].

is performed using a simple algorithm based on normalized correlation coefficient $R(x, y)$ (Equation 4.2).

$$R\left(x,y\right)=\frac{\sum_{y'=0}^{h}\sum_{x'=0}^{w}T_0\left(x',y'\right)I_0\left(x+x',y+y'\right)}{\sqrt{\sum_{y'=0}^{h}\sum_{x'=0}^{w}T_0(x',y')^2\sum_{y'=0}^{h}\sum_{x'=0}^{w}I_0(x+x',y+y')^2}} \tag{4.2}$$

$T_0\left(x',y'\right)=T\left(x',y'\right)-\bar{T}$, $I_0\left(x+x',y+y'\right)=I\left(x+x',y+y'\right)-\bar{I}\left(x,y\right)$, $T\left(x',y'\right)$ and $I\left(x',y'\right)$ are the brightness of the pixels (x, y) in the template and source image, respectively. $\bar{T}$ is the average of pixels in the template raster and $\bar{I}(x, y)$ is the average of pixels in the current searching window. $R(x, y)$ is a value between -1 and 1 and it measures similarity between the open eye template and a small region around the eyes from the previous frame. This value is verified 30 times per second so it is essential to critically evaluate whether the trace is lost. A default threshold $F = 0.55$ is set according to tests. If the correlation value $R(x, y)$ falls under the threshold F, the system starts the process of reinitialization.

When the eye is being closed, it stops to resemble the open eye template, what can be very useful during tracking. The system follows correlation scores obtained from the tracker. According to the changing correlation of the open eye and its open eye template, an eye-blink waveform can be established. Experiments were also carried out on the correlation of the closed eye and its closed eye template to compare and confirm wavelet results (Figure 4.2). However, the closed eye template is not used in the system. Subsequently, the correlation score is binarized to achieve the partition of the wavelet in two states: Open and Closed. This happens according to the threshold $O = 0.85$, which was set experimentally. O represents the minimal correlation score needed to classify an eye as open. Using thresholds O and F, each frame can have even three final states: Open, Closed, Unfound. The blink duration can be further used to determine the way of user's interaction with a computer.

The system works with a grayscale image of 320×240 dimension. It was tested on 15 subjects, all were distant 60-120 centimeters from the camera. The subjects were told to blink naturally and frequently, they could move slightly. After manual re-count, 98% of eye-blinks were successfully detected, only few false negatives. Therefore, total detection rate of 95.6% was estimated. The same authors later proposed an extension of this system, which uses also eyebrow detection to control a computer [Grauman et al. 2003]. Robust reimplementation of the mentioned system was provided in [Chau – Betke 2005], where the authors improved the algorithms to achieve better results using common web-camera instead of a CCD camera. The system works with a grayscale image of 320×240 dimension. It was tested on 15 sub-

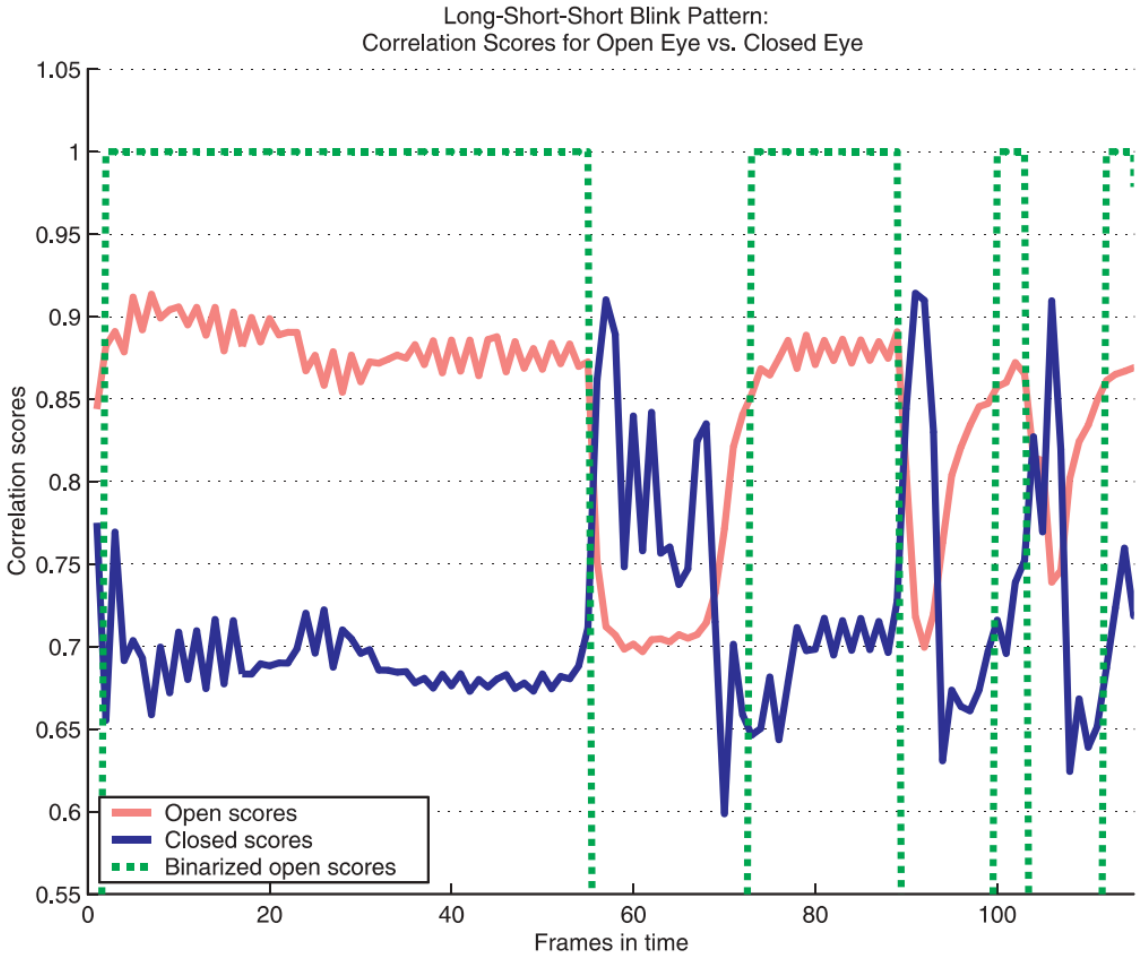


Figure 4.2: Correlation scores over time determine the openness of the eye. Binarized scores are reached according to the threshold $O = 0.85$, what is minimal value to declare an eye as open. Blink duration can be clearly calculated from gaps between the binarized correlation scores. [Grauman et al. 2001].

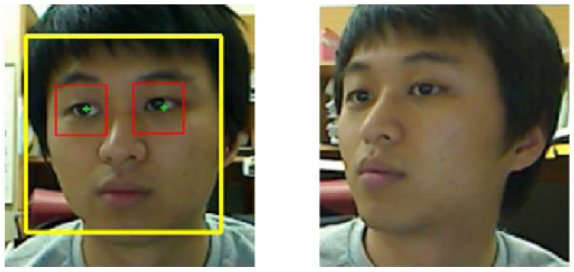


Figure 4.3: The picture (a) shows successful detection of face and eyes using AdaBoost detector. In picture (b) we can see absence of detection due to face rotation [Lee et al. 2010].

jects, all were distant 60-120 centimeters from the camera. The subjects were told to blink naturally and frequently, they could move slightly. After manual re-count, 98% of eye-blinks were successfully detected, only few false negatives. Therefore, total detection rate of 95.6% was estimated. The same authors later proposed an extension of this system, which uses also eyebrow detection to control a computer [Grauman et al. 2003]. Robust reimplementation of mentioned system was provided in [Chau – Betke 2005], where the authors improve algorithms to achieve better results using common web-camera instead of CCD camera.

A similar approach was presented in [Królak – Strumiłło 2012]. The method also uses Viola – Jones face detector to detect face and correlation scores to determine eye-blinks. However, eyes are detected using geometrical dependencies of the face. The authors use the fact that eyes are 0.4 of the imaginary line from the top of the head.

4.2 Eye Region Pixel Amount

Work in [Lee et al. 2010] presents an approach, which measures eye-blink using eye region characteristics. The method consists of three components: usage of AdaBoost face and eye detector in conjunction with KLT-based detection method, eye closure estimation based on two features and finally, eye state determination using the SVM classifier. However, the biggest contribution of this work is the method, which can detect eye-blink even at different face positions and rotations.

The method starts with capturing of an RGB image from a camera. After that, a face and eyes are detected using AdaBoost algorithm proposed in [Lienhart – Maydt 2002]. Nevertheless, this method is not so confident when face is rotated or turned (Figure 4.3). Therefore, KLT feature tracker [Bouguet 2000, Tomasi – Kanade 1991] is used to compensate the failure of AdaBoost. KLT tracker works using the correlation computing between its features by measuring mean square error.

After the image is captured, its quality needs to be enhanced. Few filters were tested in this work, but authors decided to use an illumination normalization method. First, the original image of an eye region (Figure 4.4a) is used to obtain an illumination component using the 31×31 median filter (Figure 4.4b). The size of the median filter is set according to experiments in the eye detection field. Subsequently, the image is inverted (Figure 4.4c) and then the illumination-compensated RGB image is acquired (Figure 4.4d). The picture is binarized (Figure 4.4e) according to the automatically determined threshold.

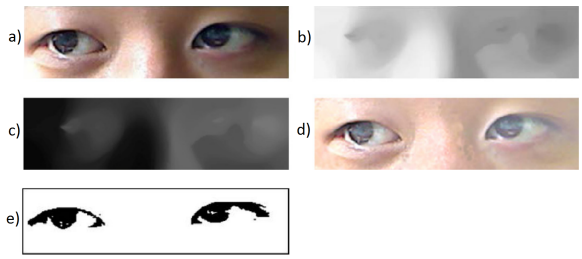


Figure 4.4: Image processing – original image (a) is filtered using median filter (b) and then it is inverted (c). Illumination-compensated RGB image (d) is the binarized (e) [Lee et al. 2010].

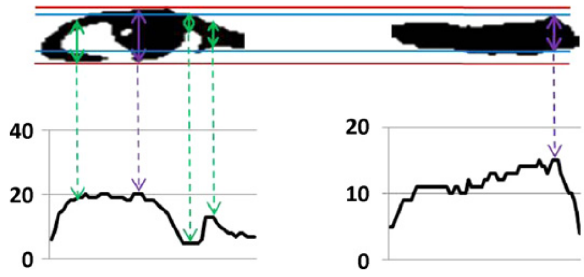


Figure 4.5: Arrow signifies the highest intensity projection value [Lee et al. 2010].

The work presents two measured features $F1$ and $F2$. $F1$ is the cumulative difference of black pixels in the eye region estimated from the binary image (Figure 4.4d) in successive frames. The authors assume that number of black pixels in a closed eye image is higher than in an open eye image. Although, there is a problem with estimation when the tracking subject is moving closer to the camera and back, because it causes eye enlargement and pixel increase. To avoid this problem, the method uses the cumulative difference counting with an adaptive threshold. This threshold changes due to the accumulated difference over time.

The second feature $F2$ represents a ratio of an eye height to an eye width. To calculate $F2$, a binarized eye region is processed through erosion and dilation filters. Then using a maximal vertical projection of black pixels, we can exactly estimate whether the eye is open or closed. An open eye has greater value of this ratio, because it has higher maximal projection value (arrow in Figure 4.5).

For the correct eye openness estimation, the authors use the features $F1$ and $F2$ as input values to *SVM classifier* [Cortes – Vapnik 1995]. SVM or Support Vector Machine is a method, which can classify and recognize patterns according to its input. In this paper three SVM classifiers are used for three different rotation angles to determine the eye state of the subject.

This approach was tested on three datasets: ZJU[1], Talking Face Video[2] and own dataset consisting of 5 videos with 1500 frames and 83 blinks on 640×480 frame resolution at 15 fps. The method achieved precision of 95.14% at eye detection in rotated faces and 94.1% at eye-blink detection in datasets.

[1] `http://www.cs.zju.edu.cn/~gpan/database/db_blink.html` [last access: 27.8.2013]

[2] `http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html` [last access: 28.8.2013]

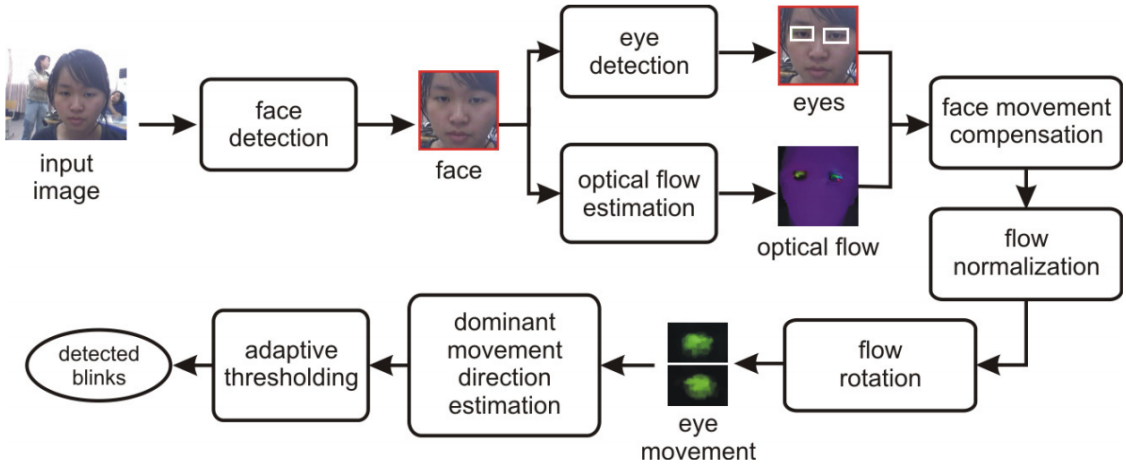


Figure 4.6: Eye-blink detection workflow diagram [Divjak – Bischof 2009].

4.3 Optical Flow Based Detection

The prototype of computer vision system based on optical flow measurement was presented in [Divjak – Bischof 2009]. The system measures eye-blink features to prevent computer users from Computer Vision Syndrome (CVS) [Yan et al. 2008]. The subject is monitored with a conventional web-camera. The prototype uses AdaBoost detector [Lienhart – Maydt 2002] to detect face and eyes. If the face is not frontal directed, then Lucas-Kanade algorithm [Tomasi – Kanade 1991, Bouguet 2000] is used to reconstruct the face region. Blink detector is based on the work presented in [Heishman – Duric 2007] with certain improvements. For the blink estimation, the authors proposed the following steps also depicted in Figure 4.6:

- computing of the optical flow of face region proposed in [Zach et al. 2007],
- compensation for face movement removal,
- optical flow normalization,
- rotation of flow vectors,
- estimation of the main flow direction,
- raw blinks extraction using adaptive threshold.

The system measures three blink-related data every second:

- blink frequency,
- average blink duration,
- average eye closure duration.

These variables are compared to previous data and can be used to predict eye fatigue. The authors guarantee the system to work real-time at 20-30 fps. The systems was compared on ZJU, Talking Face Video and own database (10 videos, 3 subjects, 150 blinks). The authors claim that their approach can detect blink with accuracy of 90%.

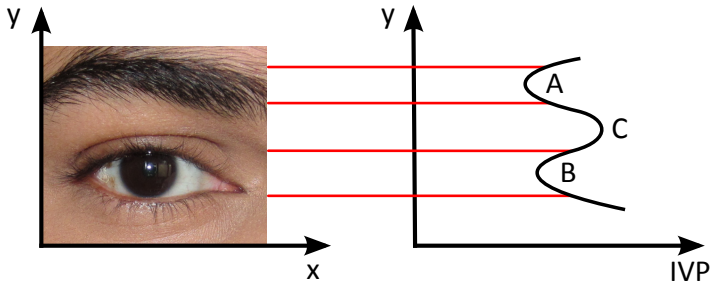


Figure 4.7: Point A is first local minimum of IVP function and it represents eyebrow. Point B is second local minimum of IVP function and it represents iris. Point C is local maximum of IVP function and it represents the skin area between an eyebrow and an iris. Curve is rotated by 90 degrees to visualize relation between IVP and the eye frame better [Dinh et al. 2012].

4.4 Vertical Intensity Projection

The authors in [Dinh et al. 2012] presented a method of eye-blink detection using *intensity vertical projection (IVP)*. IVP is the total pixel intensity in a frame row. The proposed method works with a color image.

The method uses the fact that an iris has a lower intensity vertical projection value than other regions. As we can see in Figure 4.7, IVP function has two local minimum considering one eye. the first minimum (point A) belongs to the eyebrow. The second and more important minimum (point B) represents the center of an iris or a pupil. On the other hand, IVP function has often one local maximum (point C), which means the brightest area between an eyebrow and an iris. When blink occurs, the IVP curve is changing. It is obvious that IVP of points around point B is increasing – B area is narrowing and C is widening. This observation is used to determine a blink.

The method uses the fact that an iris has a lower intensity vertical projection value than other regions. As we can see in Figure 4.7, IVP function has two local minimum considering one eye. the first minimum (point A) belongs to the eyebrow. The second and more important minimum (point B) represents the center of an iris or a pupil. On the other hand, IVP function has often one local maximum (point C), which means the brightest area between an eyebrow and iris. When blink occurs, the IVP curve is changing. It is obvious that IVP of points around point B is increasing – B area is narrowing and C is widening. This observation is used to determine a blink.

The authors define *Eye opening* parameter (Equation 4.3), where $|EF|$ is a distance between point E and F and $|DE|$ is a distance between points D and E (Figure 4.8). Thus, eye opening is a ratio of the iris area to the skin area. When the eye is closing, the eye opening parameter is decreasing because the iris is disappearing and vice-versa. This parameter also allows us to determine whether the eye is open or closed using suitable threshold.

$$Eye\ opening = \frac{|EF|}{|DE|} \tag{4.3}$$

The authors further implemented this solution based on the already mentioned method and Viola – Jones detection algorithm to detect the face. It is essential to be aware of that the proposed method can also be used in grayscale image systems.

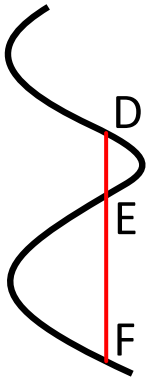


Figure 4.8: IVP function of one eye. Distance between point D and E is the skin area between an eyebrow and an iris and the distance between point E and F represents the iris [Dinh et al. 2012].

4.5 Summary

In this chapter, we analyzed methods of eye-blink detection. The first method was built to help disabled people and it is based on the correlation score of eye samples. Another method called *eye region pixel amount* uses pixel intensities in the eye region to reveal a closed eye. It processes an input frame with several filters and it uses SVM classifier to determine one of two possible eye states. For eye-blink detection, optical flow detection method calculates blink-like movements and Vertical Intensity Projection compares projection function characteristics. We decided to propose and test own eye-blink detection algorithms in Chapter 5, which are inspired by the previously analyzed methods.

Chapter 5

Proposed Detection Algorithms

Blink detection is a challenging problem. In our work, we consider two main branches of blink detection algorithms.

The first group of detection methods belongs to a *static* blink detection, which uses the fact, that we can determine the current eye state (open or closed) for each eye frame from the video sequence independently. Based on these methods, we can analyze captured eye states to reveal blinks.

Another group of blink detection methods is based on the *sequential* frame analysis, which considers moves within the eye region from a sequence of consecutive frames. According to series of expected moves, we can discover significant blink-like behavior.

5.1 Static Blink Detection

Static blink detection determines the current eye state from each frame separately. We can decide, whether the eye is open or closed according to a suitable image descriptor and SVM classifier. Before the SVM returns binary response, it has to be trained on static positive and negative samples represented by the image descriptor. The trained SVM can be then used for the real-time state prediction.

In this section, we describe several methods for a descriptor construction. The first implemented and tested method is based on the intensity projection. The second used method is a well-known SIFT descriptor [Lowe 2004], which uses image gradient information. Inspired by SIFT, we propose another gradient descriptors, which use gradient orientations and magnitudes to describe the eye sample.

5.1.1 Intensity Vertical Projection

Our *Intensity Vertical Projection (IVP)* descriptor was developed by following the work presented in [Dinh et al. 2012]. An open and a closed eye differ in distribution of the IVP. A vertical projection is the average row intensity counted as a sum of pixel intensities for each frame row and divided by the number of frame columns (Equation 5.1). IVP values form an IVP function for the whole frame, which we consider as the image descriptor. The

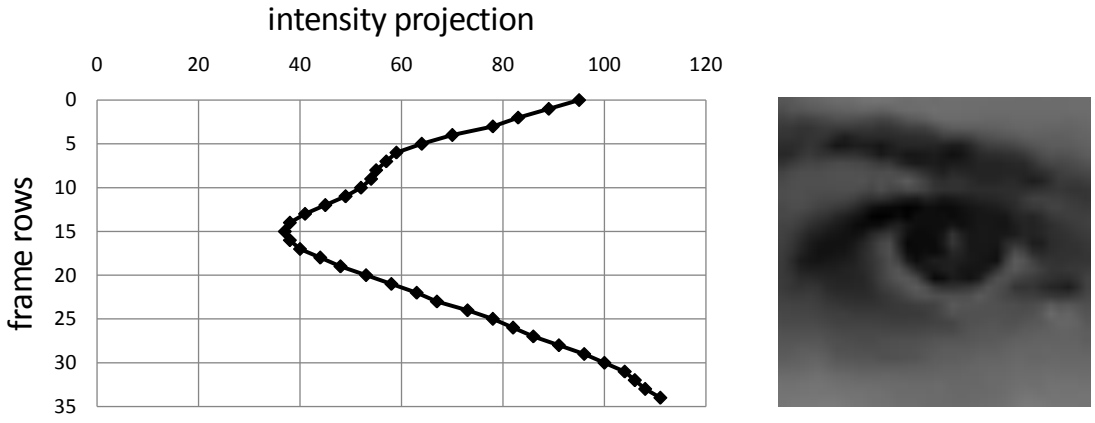


Figure 5.1: IVP function of the open eye has only one local minimum at the row 15, because the sample consists of many dark pixels in the iris area.

IVP functions of both eye states differ in their peaks' distribution and size. In general, the open eye sample has:

- higher amount of dark pixels,
- visible iris – lower IVP values around the eye center,
- minor and darker region between an eyebrow and eyelashes,
- smaller differences in the vertical image cross-section.

$$IVP(x) = \frac{\sum_{y=0}^{cols-1} i(x,y)}{cols} \tag{5.1}$$

In addition, the area between the eyebrow and eyelashes is responsible for strong separation of the eyebrow and the iris peak. All these features contribute to the descriptor construction and SVM classification. Eq. 5.2 (Eq. 5.3) represent the IVP descriptor functions of the eye samples from Figure 5.1 and Figure 5.2, respectively. Different courses of IVP functions contribute to the significant SVM discrimination.

$$\begin{aligned}\bar{O} = [&95, 89, 83, 78, 70, 64, 59, 57, 55, 54, 52, 49, 45, 41, 38, 37, 38, 40, 44, 48,\\ &53, 58, 63, 67, 73, 78, 82, 86, 91, 96, 100, 104, 106, 108, 111]\end{aligned} \tag{5.2}$$

$$\begin{aligned}\bar{C} = [&72, 65, 61, 60, 54, 52, 53, 56, 61, 65, 68, 72, 73, 73, 70, 67, 65, 62, 57, 52,\\ &49, 49, 50, 53, 57, 64, 73, 81, 89, 96, 102, 106, 108, 111]\end{aligned} \tag{5.3}$$

Implementation

The key parameter of our IVP implementation is the size of the descriptor. Eye samples can have different number of frame rows – the descriptor size varies between 18 to 36 elements.

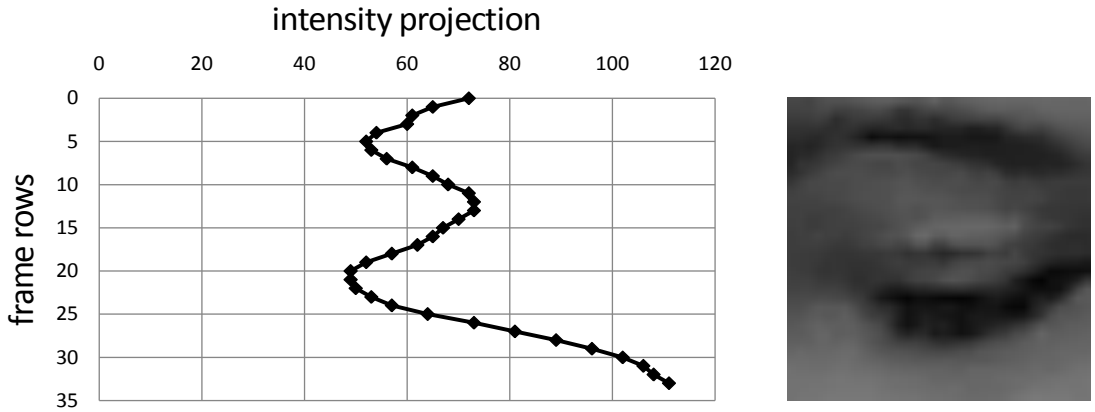


Figure 5.2: IVP function of the closed eye has two local minima (near the row 5 and 20). As we can see, a bright region between the eyebrow and eyelashes divides function on these local minima, which represents the main difference comparing to the open eye sample. However, the IVP is potentially dependent on the eyebrow's presence.

Therefore, we can not send them directly to the SVM training (SVM requires equal size of all input samples). We perform function resampling to achieve an equal input vector size.

Before the SVM training, each input vector is normalized by scaling to fall in between 0 and 1 [Hsu et al. 2003] according to its minimal and maximal projection value. Examples of normalized vectors $\bar{O}$ and $\bar{C}$ are shown in Equation 5.4 and Equation 5.5. These vectors are also normalized to the size of 30 elements. We use the standard SVM algorithm (regularized support vector classification) with a linear kernel type.

$$\begin{aligned}\bar{O}_{norm} = [&0.78, 0.69, 0.600.50, 0.390.31, 0.27, 0.24, 0.22, 0.18, 0.13,\\ &0.06, 0.01, 0.00, 0.02, 0.07, 0.13, 0.20, 0.28, 0.36, 0.43,\\ &0.52, 0.59, 0.65, 0.73, 0.81, 0.7, 0.92, 0.95, 1.00]\end{aligned} \tag{5.4}$$

$$\begin{aligned}\bar{C}_{norm} = [&0.37, 0.25, 0.19, 0.14, 0.06, 0.06, 0.10, 0.19, 0.26, 0.32, 0.38,\\ &0.39, 0.36, 0.30, 0.26, 0.21, 0.12, 0.04, 0.00, 0.01, 0.05,\\ &0.12, 0.23, 0.40, 0.54, 0.68, 0.80, 0.89, 0.94, 1.00]\end{aligned} \tag{5.5}$$

A disadvantage of the IVP method is that there are many various of eye types. Every subject can have different eyebrow, eyelashes or an eye closure appearance, what can cause unpredictable IVP behavior. Moreover, the SVM can reach lower success ratio due to absence of eyebrows.

5.1.2 SIFT descriptor

SIFT feature descriptor is widely used in the computer vision [Lowe 2004]. In our work, we use SIFT to describe eye states in order to demonstrate gradient characteristics of eye samples. We locate one keypoint in the image center (we consider that an eye center or

an iris is located in the image center) with the default orientation value of 0 degrees and keypoint size set to the one eight of the column number – $size = cols/8$. This keypoint size was chosen empirically, so the SIFT descriptor overlays the whole eye sample.

Before the SIFT is calculated, the input eye frame is filtered using 3×3 Gaussian filter to smooth unimportant edges. An output SIFT descriptor consists of 128 elements – binned gradient orientations, which are invariant to the scale transformations. The descriptor is further normalized by scaling to fall in between 0 and 1 values and then it can be used to train the SVM. SVM train parameters were set similarly as in the IVP function testing.

5.1.3 Gradient descriptors

To achieve better results than SIFT descriptor, we developed our own descriptor, which is dependent on the image gradient information. Our goal was to construct a gradient descriptor with better adaptation for the eye state determination problem. One of the most significant characteristics for the human discrimination of eye states is an ability to recognize a shape. A shape is often described by gradient orientations. We want to use that fact and build a feature descriptor which could help a computer to discriminate different eye states using gradient orientations. We propose the gradient calculation and orientation sorting for each eye sample. This work was presented in [Drutarovsky 2014].

Gradient calculation

The first step of descriptor construction is the gradient calculation from the image. We assume that the input eye sample contains nothing but an eye aligned to the center of the sample. This could be achieved using Viola – Jones eye detector. After successful detection, we align rectangles according to the pupils using the modified gradient pupil locator [Timm – Barth 2011], which offers open source implementation[1]. In the case of the closed eye sample, we assume that eyelashes or eyelids' link is located in the center of the sample.

Our method uses *Sobel* operator for the edge detection, because it is partly rotation invariant. Sobel operator achieves better results for computation of diagonal edges than the edge detector which uses $[-1, 0, 1]$ kernel and an image preprocessed with Gaussian filter.

An input image is decomposed into derivatives – horizontal image gradients dx and vertical image gradients dy. Using these two images we can compute gradient magnitude m for each pixel as a maximum of absolute values from both images (Equation 5.6). We can also compute gradient orientation α using an arctangent function (Equation 5.7). This approach was inspired by the work presented in [Arbelaez et al. 2011].

$$m_{x,y} = max(abs(dx_{x,y}), abs(dy_{x,y})) \tag{5.6}$$

$$\alpha_{x,y} = \arctan\left(\frac{dx_{x,y}}{dy_{x,y}}\right) \tag{5.7}$$

[1] `https://github.com/trishume/eyeLike` [last access: 4.4.2014]

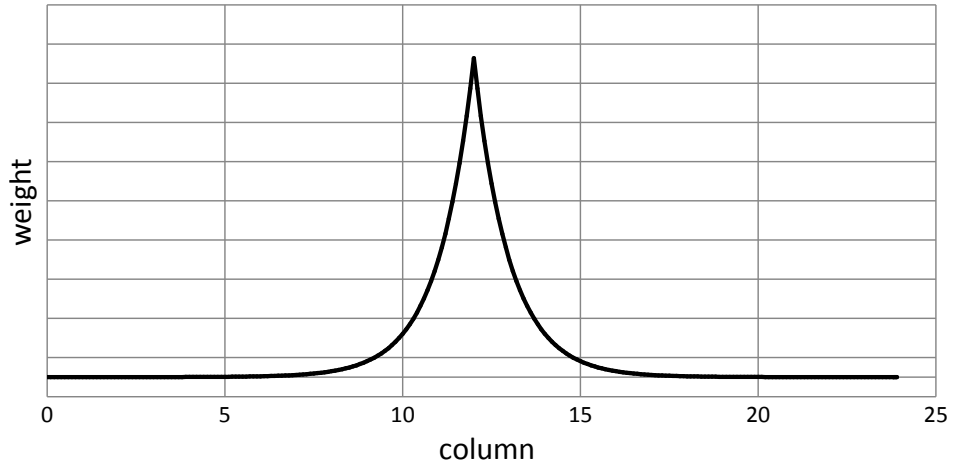


Figure 5.3: The graph represents a relation between sample columns and the function exponent. Weight map gives the highest weight to the pixels located in the eye samples center. According to our modified gradient pupil locator, we assume that the eye center is located in the image center.

Weight map

In our gradient descriptor, we use a *weight map* to adjust weights of values which are added to the orientation bins (Figure 5.3). This map is used to control the weight of image pixels so the pixel in the center of the image (x_c, y_c) has higher weight than the pixel at the image border. Each point in the eye sample has own weight $w_{x,y}$. In our algorithm, the weight function is calculated as an exponential function dependent on the particular pixel position in the frame and using a constant value C (Equation 5.8). Exponent values were chosen by empirical tests. This approach is inspired by the FREAK descriptor [Alahi et al. 2012] which uses knowledge about human's retina and vision sharpness.

$$w_{x,y} = e^{n_{x,y}}, n_{x,y} = C\left(1 - \sqrt{\frac{(x-x_c)^2 + (y-y_c)^2}{(x_c)^2 + (y_c)^2}}\right) \tag{5.8}$$

Orientation function

After these operations, we have two essential values for each image pixel so we can sort gradient orientations into 360 bins (Algorithm 1). Each gradient orientation is dispersed from -10 to 10 degrees and classified into bins to avoid errors in orientation calculation. We consider only gradients with higher magnitude values to avoid non-significant gradients. In our work, we tested two approaches – with and without gradient weighting. No weighting causes that the appropriate bin is increased by 1. Weighting increases appropriate bin by corresponding value from a weight map. A difference between weighted and non-weighted orientation sorting can be seen in Figure 5.4 and Figure 5.5.

The next step in the gradient descriptor construction is the processing of the 360-binned orientation function. An input orientation function has a lot of local minima and maxima, therefore we smooth the function. Each bin is smoothed using an average value from the four closest bins. We smooth the function until it has 4 extremes exactly – two minima and two

Algorithm 1 Sorting into orientation function.

INPUT: gradient magnitudes and angles
OUTPUT: binned orientation function

```
1:  procedure SORTING(magnitudes, angles)
2:      function ← 0                                  ▷ initialize output vector with zeros
3:
4:      for x ← 0, rows − 1 do
5:          for y ← 0, cols − 1 do
6:              if magnitudes(x, y) > threshold then
7:                  if angles(x, y) < 0 then          ▷ angles preprocessing
8:                      temp ← angles(x, y) + 360
9:                  else
10:                     temp ← angles(x, y)
11:                 end if
12:                 weight ← weight_map(x, y)         ▷ get specific weight
13:                 for i ← temp − 10, temp + 10 do   ▷ ±10 degrees dispersion
14:                     function(i) ← function(i) + weight   ▷ increase specific bin
15:                 end for
16:             end if
17:         end for
18:     end for
19:
20:     return function
21: end procedure
```

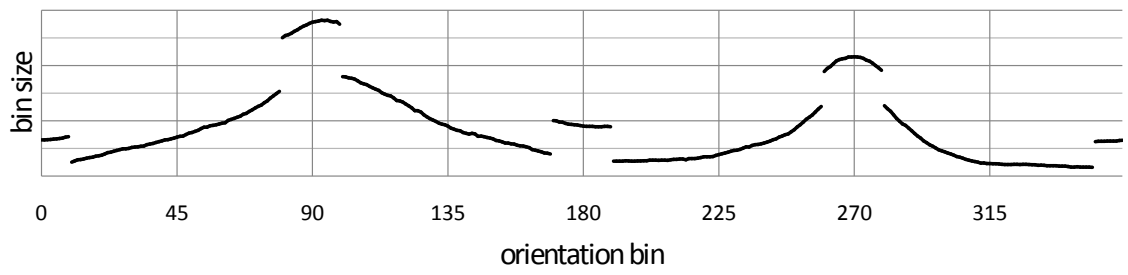


Figure 5.4: An average orientation function of 150 open eye samples (from SmallEye dataset) with bin dispersion and without weighting. A non-weighted function considers all gradients to be equal for bin sorting.

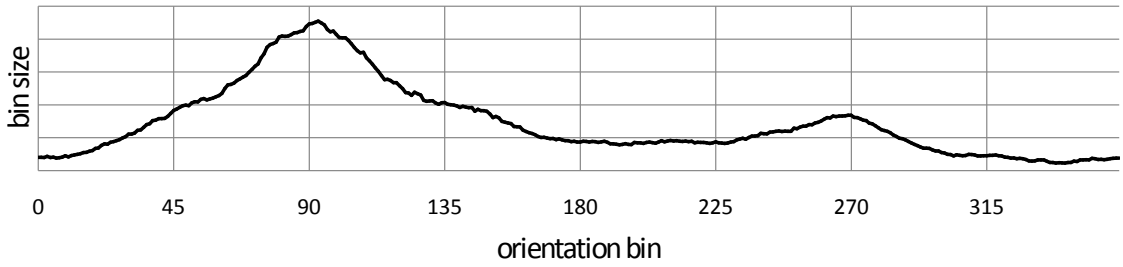


Figure 5.5: An average orientation function of 150 open eye samples (same as in Figure 5.4) with bin dispersion and with weighting. Weighting affects the contribution of significant gradient around an iris.

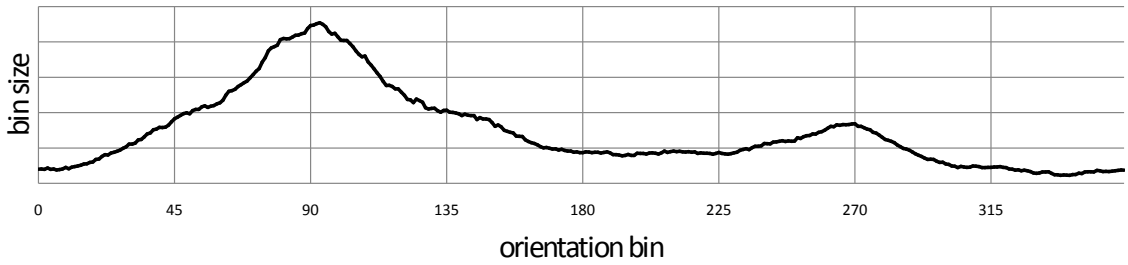


Figure 5.6: An orientation function of open eye samples. The function is averaged from 200 samples from SmallEye dataset. As we can see, the function has two local maxima and the first maximum has a higher value.

maxima. As we consider aligned eye samples, these extremes are often around 0, 90, 180, 270 degrees. We use the fact that open eyes (Figure 5.6) have a maximum with the highest value around 90 degrees, but closed eyes have a higher maximum around 270 degrees bin (Figure 5.7). That is because an open eye has more gradients oriented vertically up near the eye center. Moreover, an open eye sample has bigger disperse of the maximum peak caused by higher amount of horizontal orientations around the iris.

The smoothed function is then aligned to the first local maximum. Rotation invariance can be guaranteed by shifting bin values according to the rotation angle. This angle can be computed as an angle between a line connecting both irises and a horizontal frame line.

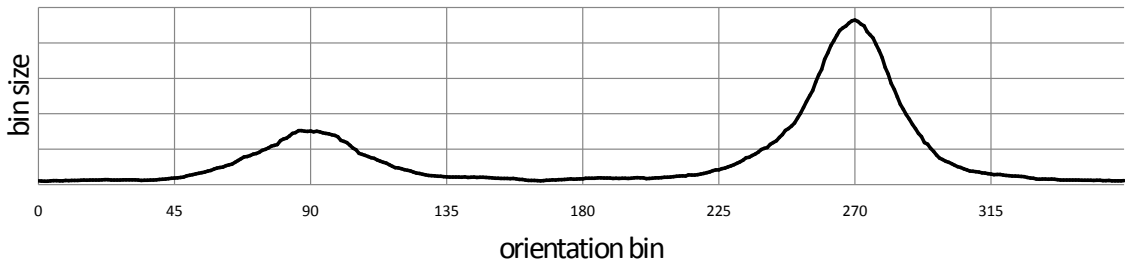


Figure 5.7: An orientation function of closed eye samples. The function is averaged from 200 samples from SmallEye dataset. The function has also two local maxima, but the second maximum has a higher value.

An orientations function with 4 extremes is used to construct the final gradient descriptor (Algorithm 2). Our proposed descriptor consists of 8 numbers – four pairs. Each pair represents one extreme or peak and consists of two numbers $rate_{x,y}$ and $distance_{x,y}$. $rate_{x,y}$ represents a rate of the peak height among other peaks. This means that the summed size of the all four peaks gives 1. $distance_{x,y}$ represents the number of bins located between the considered peak and the closest right peak. A descriptor $\bar{O}_w$ represents a descriptor of an averaged vector of 200 open eye samples (Equation 5.9).

Algorithm 2 Descriptor construction from the orientation function.

INPUT: binned orientation function
OUTPUT: gradient descriptor with 8-elements

```
1:  procedure CONSTRUCTION(function)
2:      descriptor ← 0                          ▷ initialize the output descriptor with zeros
3:      repeat
4:          SMOOTH(function)                    ▷ average each value considering 4 nearest neighbors
5:      until GET_PEAKS_NUMBER(function) = 4
6:
7:      ALIGN_FUNCTION(function)                ▷ align to the maximum with the highest value
8:
9:      for i ← 0, GET_PEAKS(function) do
10:         descriptor(2 * i) ← GET_RATE(function)                    ▷ peak height rate
11:         descriptor(2 * i + 1) ← GET_DISTANCE(function)  ▷ peak distance from first
    right neighbor
12:     end for
13:
14:     return descriptor                       ▷ function returns the gradient descriptor with 8-elements
15: end procedure
```

$$\bar{O}_w = [0.59, 0.30, 0.10, 0.18, 0.24, 0.22, 0.07, 0.30] \tag{5.9}$$

5.1.4 Evaluation and discussion

Static detection algorithms were tested with own dataset – *SmallEye*. Our dataset consists of 300 open eye samples and 300 closed eye samples (640×480) of the six individuals sitting in front of the camera without eyeglasses. Eye samples were acquired using Viola – Jones eye detector and our gradient pupil locator at the average resolution of 24×24 pixels. A crucial point of the testing is SVM classifier. SVM was trained on other 100 open eye samples (negative samples) and 100 closed eye samples (positive samples) from the dataset. The testing was performed on other 200 positive samples and 200 negative samples.

Intensity Vertical Projection was tested using several descriptor sizes. Best results were achieved for the descriptor size of 30 elements, because SmallEye samples have average row number around 30 rows.

Our gradient descriptors are dependent on the iris location (for the open eye) and eyelashes (for the closed eye), therefore we consider that these features are located in the middle of the eye frame.

Table 5.1: Testing of static detection algorithms was performed on SmallEye dataset. We show true positive rate (correct closed eye estimation) and false positive rate (wrong closed eye estimation) for different descriptor sizes. The best results were achieved for SIFT method, however our weight gradient method reaches comparably good results.

Method	Descriptor size	TP	FP
IVP	15	80.0%	23.0%
IVP	20	80.3%	28.0%
IVP	25	79.5%	33.2%
IVP	30	81.3%	19.2%
IVP	35	81.3%	31.0%
SIFT	128	89.8%	13.2%
No weighted	8	67.5%	26.5%
Weighted	8	86.3%	18.0%

Test results show that the gradient weighting of center pixels prefers significant gradients. A weighting method achieves better results than a non-weighting method, because it considers eye center gradients with higher weights. The non-weighted method does not emphasize desired eye features. However, the best results were achieved using SIFT method – 89.8% on closed eye samples and only 13.2% of false positives. Although our gradient method does not achieve the performance of SIFT descriptor, we use only 8 descriptor elements, SIFT uses 128 elements. Test results can be seen in Table 5.1.

Static methods and SVM are not so confident because they do not achieve full performance. Our gradient descriptors are dependent on the precise eye alignment in the frame, therefore it can cause detection problems with wrongly located eyes. SVM can decide only between an open and a closed eye, therefore static detection usage can also cause difficulties in precise blink determination. We decided to move along and try blink detection using sequential frame analysis.

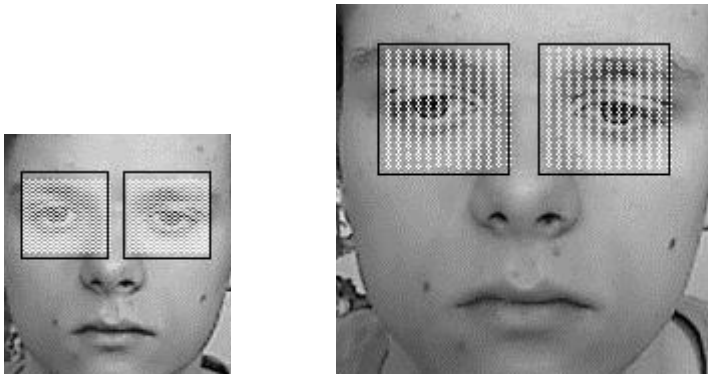
5.2 Sequential Blink Detection

A second type of blink detection is a sequential method, which uses analysis of consecutive frames to estimate possible blink occurrence. In contrast to static detection methods, this approach can not determine the actual eye state (open or closed) in every point of the input video sequence. However, the sequential analysis uses feature moves within the eye region to estimate blink-like behavior.

A sequential method is based on the *KLT Tracker* [Tomasi – Kanade 1991, Bouguet 2000], which is a well-known feature tracker. KLT uses the intensity and spatial characteristics of input features to find new features' positions according to their optical flow.

Our proposed method tracks both eyes separately, which supports the control of blink moves for each eye. After eye detection, features are placed in eye regions and they are tracked using an optical flow calculation. Every new feature set is processed to detect errors and outliers.

Subsequently, we estimate eye centers and eye rectangles. Features are then sorted into cells, which help to consider moves in different parts of the eye region. These moves are



a) b)

Figure 5.8: Initial feature grids. Figure (a) represents the grid of an eye rectangle with width of 44 pixels – grid has 306 features. Figure (b) represents the grid of an eye rectangle with width of 73 pixels – grid has 262 features.

used to identify blink-like behavior. We propose two methods for blink determination based on sequential analysis – *Cell method* and *Move bins method*. Both methods use the following preprocessing.

5.2.1 Feature placement

The first step of eye tracking is the feature placement in eye regions. We detect eyes' positions using the Viola – Jones detection algorithm [Viola – Jones 2001]. After the face detection, we detect both eyes to obtain eye rectangles. Consequently, we fill them with features regularly (grid of points to track), with spacing adaptive to the rectangle size (Figure 5.8). An adaptive feature grid is suitable for different eye sizes or distances from the camera. New feature placing is performed during the reinitialization. Therefore, each reinitialization can return various size of the feature vector, so we need to consider initial points number for further analysis.

5.2.2 Feature tracking

Feature tracking is performed by KLT tracker, which tries to find a new location for every input feature point. Nevertheless, tracking is not one hundred-percent sure. Hence, we need to verify new features after every tracking iteration. First, we verify features using the optical flow control, which removes points from future iterations when its optical flow is not found. Second verification is a major shift control. Feature points can move unnaturally with major shift across the frame. Therefore, it is highly probable that such points flow is wrongly estimated. We also remove these points from future iterations. The final feature control is the outlier control. Each point is verified according to its distance from the eye center. Points with higher distance than the default threshold are considered as outliers. An outlier is removed from further analysis, but it is not removed from future iterations – an outlier point can return to its previous location in the next frames (their exclusion could decrease tracking precision).

5.2.3 Eye center and bounding rectangle

To identify outliers, we need to consider one reference point – an *eye center*. This point is calculated as the average point (average of x-coordinates and y-coordinates) from all feature points for each eye, separately. Based on the both eye centers, we can count eye distance using the regular Euclidean distance. An eye distance is directly proportional to the subject distance from the camera. This key parameter helps us to set suitable blink thresholds.

An estimation of the eye rectangle (bounding box) is a more complex problem. We can not use the initial rectangle size during the tracking, because a subject can move towards and backwards the camera, what leads to the rectangle resizing. Therefore, we use histogram of point distances from the eye center (Algorithm 3). After each tracking iteration, set of features is sorted into bins according to their distance. Optimal rectangle dimensions and position is then estimated using the *distance gap* – a sequence of three bins, which carry small amount of feature points. The distance gap is located after the bin holding the global maximum (Figure 5.9).

Algorithm 3 Eye rectangle estimation according to feature points.

INPUT: feature points and eye center point
OUTPUT: rectangle bounding eye (non-outlier feature points)

```
1:  procedure GET_RECTANGLE(points, center)
2:      hist ← 0                                  ▷ initialize the distance histogram with zeros
3:      for i ← 0, points − 1 do
4:          distance ← EUCLIDEAN(points(i), center)
5:          hist(distance) ← hist(distance) + 1
6:      end for
7:      max ← FIND_MAX_BIN(histogram)
8:
9:      for i ← max, histogram − 1 do
10:         if hist(i) < T and hist(i + 1) < T and hist(i + 2) < T then
11:             distance_gap ← i
12:             break
13:         end if
14:     end for
15:     rectangle ← CREATE_RECT(points, center, distance_gap)
16:
17:     return rectangle                          ▷ final bounding box of considered points
18: end procedure
```

We can say, that the distance gap is a boundary separating outliers from the considered points. The obtained rectangle radius and the eye center is then used to construct a final eye rectangle. As it was mentioned before, outliers are not removed from the future tracking iterations, they are only excluded from the further analysis in the current iteration.

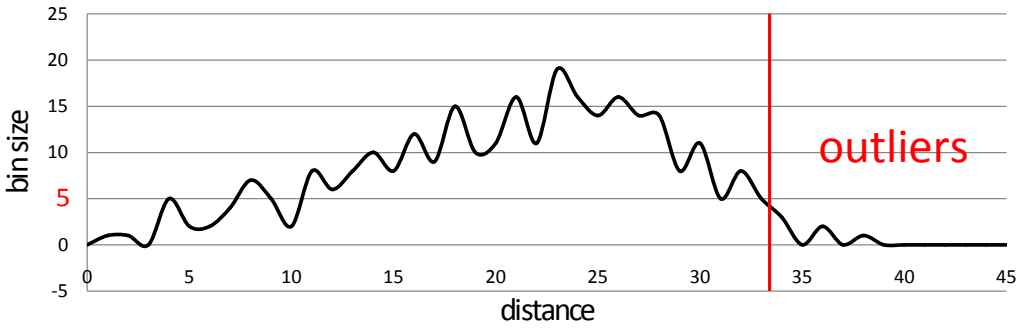


Figure 5.9: Histogram of point distances from the eye center. A vertical line represents a separator of outliers from considered points. Threshold of the distance gap used in our work is set to a 5-point bin.

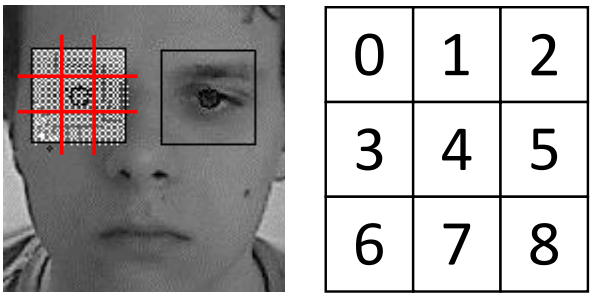


Figure 5.10: An eye rectangle is divided into 3×3 cells. Each cell group points from its area, what leads to the observation of noticeable changes in different eye parts. We are mostly interested in cells 1 and 4.

5.2.4 Cell method

The first proposed sequential method for blink determination is *Cell method*. The method usage for an eye-blink status decision can be seen in Algorithm 4. After getting the best eye rectangle, the eye (feature points) is divided into 9 parts – 3×3 cells (Figure 5.10). For each cell, we calculate an average vertical point move (across the y-coordinate). Therefore, we can estimate an approximate move of points in different eye cells. Using the sequential analysis, we can calculate a tendency of move changes using the moves within the current and the previous frame (Equation 5.10).

$$change(i) = current(i) - previous(i),\ i \in \langle 0, 8 \rangle \tag{5.10}$$

In 3×3 cells of the eye rectangle, a blink appears as a significant vertical move in the middle cells (numbers 3, 4 or 5), but only minor vertical move in the upper cells (number 0, 1 or 2). Our method uses the statistical variance of these 6 cells to evaluate the diversity of moves in the cells. If the variance of the current frame is high enough, it indicates a move in the eye region, which is not natural for the global moves (up or down shifting). On the other hand, it is the sign of eye-blink or other eye feature movements.

To mark and separate these unwanted eye feature moves (eyebrow or iris movement), we consider the blink as the conjugation of two moves – down move and up move. Down move of an eyelid (comparison to down head move can be seen in Figure 5.11) is accompanied

Algorithm 4 Cell method usage for an eye-blink status decision.

INPUT: current and previous moves vector, current state of the state machine, distance between eye centers, FPS count
OUTPUT: eye-blink status – boolean value

```
1:  procedure CELL_METHOD_STATUS(current, previous, state, distance, fps)
2:      for i ← 0, 5 do
3:          change(i) ← current(i) − previous(i)
4:      end for
5:
6:      threshold ← distance ∗ const/fps                    ▷ variance threshold
7:      upper ← change(1)
8:      lower ← change(4)
9:
10:     if state = 0 then
11:         if move in positive direction and GET_VARIANCE(change) > threshold then
12:             state ← 1                                   ▷ down eye-lid move occurred
13:         end if
14:     else if state = 1 then
15:         if move in negative direction and GET_VARIANCE(change) > threshold then
16:             state ← 0                                   ▷ up eye-lid move occurred
17:             return true                                 ▷ blink occurred
18:         end if
19:     end if
20:
21:     return false                                        ▷ blink did not occur
22: end procedure
```

a) Down head move

80%	100%	63%
66%	86%	66%

b) Down blink move

49%	100%	-8%
270%	301%	71%

varianceA = 0.013 * varianceB

Figure 5.11: Move changes in the first 6 cells of the eye rectangle during down head move (a) are accompanied with lower variance value (all feature points move almost equally) than during down blink move (b) (cell 4 shows higher positive change than cell 1).

a) Up head move

-133%	100%	-80%
-98%	-96%	-67%

b) Up blink move

-161%	-100%	-221%
-83%	-940%	-683%

varianceA = 0.006 * varianceB

Figure 5.12: Move changes in the first 6 cells of the eye rectangle during up head move (a) are accompanied with lower variance value (all feature points move almost equally) than during up blink move (b) (cell 4 shows higher negative change than cell 1).

by the positive change in the cell 4 (y-coordinate rises), which is more remarkable than the move in the cell 1. On the contrary, up move of an eyelid (comparison to up head move can be seen in Figure 5.12) is accompanied by the negative change in the cell 4 (y-coordinate falls), which is also more remarkable than the move in the cell 1.

Moves participating in blinks are chosen by the *variance threshold*, which chooses higher variance values – moves, which represent significant inner changes. The threshold was set empirically according to the tests (Equation 5.11). Its value is dependent on these parameters:

- Eye distance:
  - Distance between both eyes is directly proportional to the subject distance from the camera.
  - Eye size affects the change intensity during the blink.
- FPS (frames per second):
  - Frame rate of the input video sequence also influences the change intensity.
  - More frames per second causes, that variance of cell moves is smaller.
- Constant:
  - Constant value was chosen according to tests.

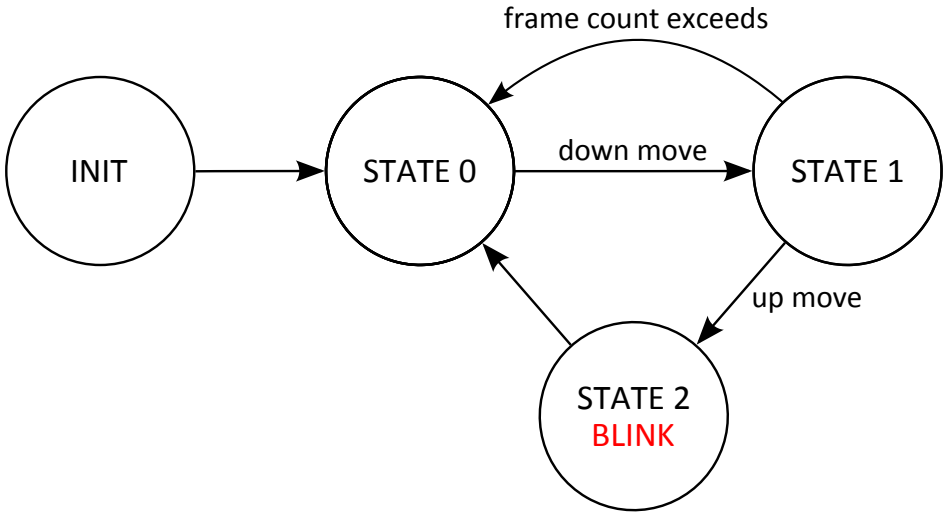


Figure 5.13: After machine initialization, the state machine is in State 0. After down blink move, the state changes into State 1. After up blink move, the state change into State 2 – blink occurred. State 1 can also change, when machine is in a current state for too long. Frame count threshold of the state machines is dependent on the frame rate of the video sequence.

$$T = dist \times \frac{1}{fps} \times const \tag{5.11}$$

The conjunction of eye moves and variance threshold reveals blinks. Down blink move and up blink move are assigned to states of the state machine (Figure 5.13).

5.2.5 Move bins method

The second proposed sequential method for blink determination is *Move bins method*. This method uses the fact, that blink moves in the eye region bring series of vertically oriented moves with various relative movement. On the other hand, head moves are accompanied with similar features' movement. We count vertical shift for each considered feature as the difference between their y-coordinates. After that, each move is sorted into corresponding bin according to its orientation (positive moves belong to other bins than negative moves). Therefore, we can separate down blink moves from up blink moves.

For both sorted bin vectors (up vector and down vector), we count the number of nonzero bins. Comparison of these two numbers reveals the size and direction of the blink-like moves. Down eyelid movement causes that bins of the down vector are filled with higher values than bins of the up vector (Figure 5.14). On the other hand, up eyelid movement fills the up bins more than down bins (Figure 5.15).

Blink is naturally compound of down eyelid move followed by an up eyelid move. Therefore, we consider the sequence of these two moves to reveal blink occurrence. We use conjunction of a state machine and experimentally found threshold to determine such sequence. The state machine consists of the following states:

- STATE 0:
  - Initial machine state.

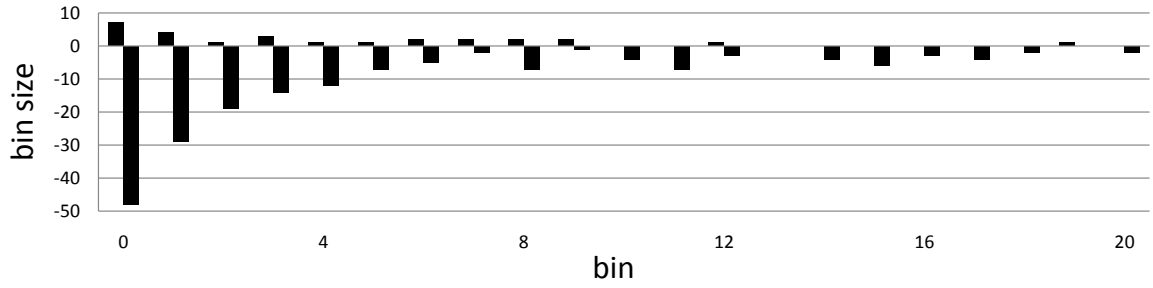


Figure 5.14: Comparison of up and down vector during the down eyelid movement. Down vector bins (values below zero) are more fully than up vector bins, because more features move vertically down than up.

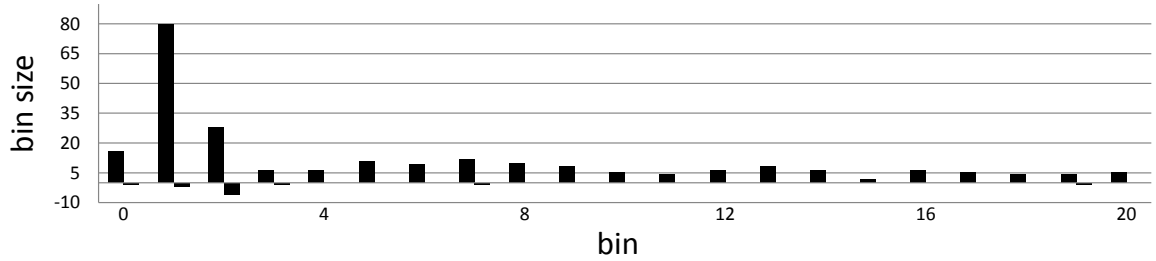
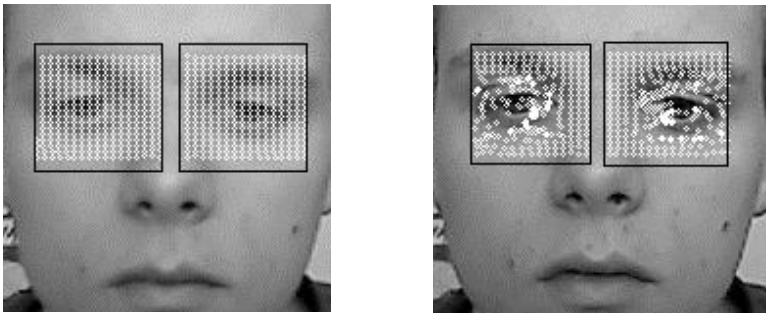


Figure 5.15: Comparison of up and down vector during the up eyelid movement. Up vector bins (values above zero) are more fully than down vector bins, because more features move vertically up than down.

- STATE 1:
  - This state is set to current, when down move occurs.
  - Typically, a blink starts with one significant down move.
- STATE 2:
  - The state is set to current, when up move occurs.
- STATE 3:
  - This state is set to current, when up move occurs again (up move expected at more than one frame gives better results – lower false positives).
- STATE 4:
  - The state is set, when blink occurred.
  - A sequence of blink moves ends and the state machine can be set to the STATE 0.

Furthermore, states can be also changed when blink frame count exceeds a default threshold.

Although a number of nonzero bins in the move vectors is a very reliable sign of inner eye moves, this method is not so confident as proposed cell method, because it is hard to find the suitable threshold for consideration of blink-like moves' separation.



a) b)

Figure 5.16: In figure (a), we can see initial feature points. Figure (b) represents the state of feature points after 3 blinks. We can see that many points are snapped to corners and edges of eye features. Therefore, reinitialization is necessary to obtain a new regular grid.

5.2.6 Reinitialization

A crucial point of our sequential algorithms is feature reinitialization. Wrongly assigned eye features can easily decrease the true positive and increase the false positive rate. An irregular feature grid causes, that sequential analysis can be negatively affected (Figure 5.16). Therefore, reinitialization is necessary for the correct working of cell sorting and move counting.

Hence, our algorithms reinitialize (achieve its initial state) after several events:

- large number of lost feature points,
- after blink occurrence,
- over time, after constant number of frames.

Reinitialization restores all parameters to its initial values. However, it is necessary to preserve the current state and blink frames count, because the reinitialization can occur during the important machine state – a blink in progress could be ignored.

5.2.7 Evaluation and discussion

Our methods based on sequential analysis of consecutive frames were tested on two datasets. Our dataset consists of 8 videos with 4 individuals. The dataset is quite challenging, because it contains one video with individual wearing glasses. In addition, videos were recorded under different conditions with faces mostly oriented directly to the camera, but natural face movements and other intensive non-blink moves should be expected, too. The dataset contains over 82600 frames (640×480) and 353 blinks. All videos were recorded using Logitech C905 webcam with standard 30 FPS frame acquisition.

The second dataset *Talking Face Video (Talking)* is free to use[2] and it contains 5000 frames (720×576) with 61 eye-blinks. A tested subject is a man taking conversation during the record. His face is mostly oriented directly to the camera and slightly turned aside (Figure 5.17), what causes minor problems with precise eye detection.

[2] `http://www-prima.inrialpes.fr/FGnet/data/01-TalkingFace/talking_face.html` [last access: 7.4.2014]

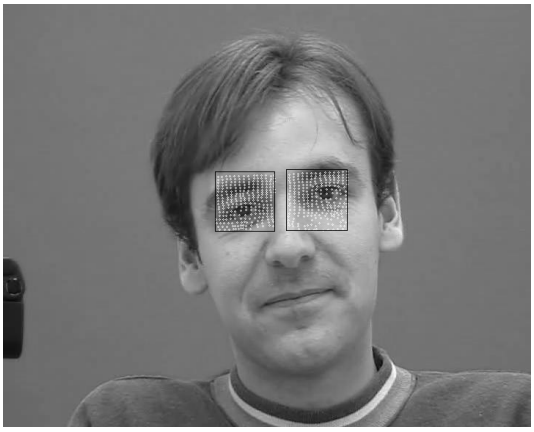


Figure 5.17: The subject participating in the video record has the face slightly turned aside.

Table 5.2: Results of our sequential blink detection methods on both datasets. TP represents true positive count (successful detected blinks) and FP stands for false positive (wrongly assigned blinks). We compared our method to optical flow method proposed in [Divjak – Bischof 2009].

Method	Dataset	TP	FP
Move bins	Own	82.1%	28.3%
Cell	Own	85.0	18.9%
Move bins	Talking	70.5%	17.3%
Cell	Talking	96.7%	9.2%
Divjak & Bischof	Talking	95.0%	19.0%

Using our cell sequential detection method, we achieved the detection rate of 96.7% on Talking Face Video. We failed to detect only two blinks, which happened during the downward sight, therefore the change of the blink moves is not so significant (Table 5.2). We compare our cell method to work presented in [Divjak – Bischof 2009], which uses optical flow blink determination. We achieved better results with higher true positive rate and false positive rate, too.

The move bins method does not reach so a high precision as the cell method, because it is harder to find suitable thresholds for algorithm working. Furthermore, the move bins method has bigger problems with sudden face movements and unnatural face mimicry.

Blink detection using our cell method is stable during almost every natural subject's behavior. It is reliable even in decent face movements, what makes the solution suitable for future use in microsleep detection applications. We present cell the method as our best solution for the problem of eye-blink detection.

5.3 Summary

This chapter summed up all detection algorithms, which were tested within this bachelor thesis. We looked into the problem of static detection using the conjugation of image descriptor and SVM classifier. We proposed own gradient descriptors, which were compared to well-known SIFT descriptor and own implementation of intensity projection descriptor.

We achieved comparable results to best SIFT method.

The second part of this chapter was devoted to the sequential analysis methods. We presented Cell method and Move bins method, which reached even better results on Talking Face Video dataset than Divjak and Bischof optical flow method.

Chapter 6

Conclusion

The bachelor thesis deals with the problem of microsleep detection using the eye-blink analysis. The first part of bachelor thesis was aimed at the state-of-the-art analysis. We looked at the available commercial and non-commercial solutions with confident detection rates. Subsequently, we analyzed microsleep detection systems proposed in several research papers. The stress was put on face and eye detection and tracking systems discussion, because they belong to primary parts of blink detections.

Our aim was to build an application, which could monitor a driver to predict incoming microsleep. We were not able to develop a solution with eye-blink frequency and duration analysis to obtain level of driver's sleepiness. However, we aimed at the construction of a reliable eye-blink detector, which is a crucial part of microsleep detector. A proposition and testing of own blink detection methods was presented in the second part of the bachelor thesis.

We proposed own gradient descriptor based on weighting of the significant eye gradients. Gradient descriptor with SVM achieves comparable results to SIFT descriptor. Furthermore, we implemented and tested Intensity Vertical Projection method, which is our own modification of the method described in [Dinh et al. 2012].

Another group of the proposed blink detection methods was based on sequential analysis of consecutive video frames. We developed and tested a sequential blink detector, which achieved better results on Talking Face Video dataset than the solution proposed in [Divjak – Bischof 2009]. Our solution works stable under natural conditions and behaves confident even during face movements. We consider our solution as very courageous and perspective for future usage in microsleep detectors or other eye-blink based applications.

We build two own datasets for the purpose of testing in this bachelor thesis. Own SmallEye dataset consists of static open and closed eye samples. Another dataset consists of challenging video records of 8 individuals.

Bibliography

AASM. The International Classification of Sleep Disorders: Diagnostic and Coding Manual. *Annals of Internal Medicine*. 1991, 115, 5, pages 401. doi: 10.7326/0003-4819-115-5-413_1. Available from: http://dx.doi.org/10.7326/0003-4819-115-5-413_1. AASM - American Academy of Sleep Medicine.

ACCIDENTS, T. R. S. F. T. P. O. Driver Fatigue and Road Accidents, February 2001. Available from: http://www.rospa.com/roadsafety/info/fatigue.pdf.

AKERSTEDT, T. – KECKLUND, G. – HORTE, L. G. Night driving, season, and the risk of highway accidents. *SLEEP*. 2001, 24, 4, pages 401 – 406. Available from: http://www.ncbi.nlm.nih.gov/pubmed/11403524.

ALAHI, A. – VANDERGHEYNST, P. – ORTIZ, R. FREAK: Fast Retina Keypoint. *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2012, 0, pages 510–517. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2012.6247715.

ARBELAEZ, P. et al. Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.* May 2011, 33, 5, pages 898–916. ISSN 0162-8828. doi: 10.1109/TPAMI.2010.161. Available from: http://dx.doi.org/10.1109/TPAMI.2010.161.

ASADIFARD, M. – SHANBEZADEH, J. Automatic Adaptive Center of Pupil Detection Using Face Detection and CDF Analysis. In *Proceedings of the International MultiConference of Engineers and Computer Science 2010 Vol I*, IMECS 2010, march 2010. ISBN 978-988-17012-8-2.

ASTERIADIS, S. et al. An Eye Detection Algorithm Using Pixel to Edge Information. In *An Eye Detection Algorithm Using Pixel to Edge Information*. European Signal Processing Conf. (EUSIPCO 2006), Florence, Italy, 4-8 September, 2006., 2006. Available from: http://www.image.ece.ntua.gr/publications.php.

BERGASA, L. et al. Real-time system for monitoring driver vigilance. *Intelligent Transportation Systems, IEEE Transactions on*. 2006, 7, 1, pages 63–77. ISSN 1524-9050. doi: 10.1109/TITS.2006.869598.

BOUGUET, J.-Y. Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2000. Available from: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.185.585.

BRANDT, T. – STEMMER, R. – RAKOTONIRAINY, A. Affordable visual driver monitoring system for fatigue and monotony. In *Systems, Man and Cybernetics, 2004*

IEEE International Conference on, volume 7, pages 6451–6456 vol.7, 2004. doi: 10.1109/ICSMC.2004.1401415.

CAFFIER, P. P. – ERDMANN, U. – ULLSPERGER, P. The spontaneous eye-blink as sleepiness indicator in patients with obstructive sleep apnoea syndrome-a pilot study. *Sleep Medicine*. 2005, 6, 2, pages 155 – 162. ISSN 1389-9457. doi: http://dx.doi.org/10.1016/j.sleep.2004.11.013. Available from: `http://www.sciencedirect.com/science/article/pii/S1389945704002230`.

CANNY, J. A Computational Approach to Edge Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 1986, PAMI-8, 6, pages 679–698. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851.

CHAU, M. – BETKE, M. Real Time Eye Tracking and Blink Detection with USB Cameras. Technical report, Boston University Computer Science, Boston, MA 02215, USA, May 2005.

CHOI, I. – HAN, S. – KIM, D. Eye Detection and Eye Blink Detection Using AdaBoost Learning and Grouping. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–4, 2011. doi: 10.1109/ICCCN.2011.6005896.

CIESLA, M. – KOZIOL, P. Eye Pupil Location Using Webcam. *CoRR*. 2012, abs/1202.6517.

COOTES, T. F. et al. Active shape models-their training and application. *Comput. Vis. Image Underst.* January 1995, 61, 1, pages 38–59. ISSN 1077-3142. doi: 10.1006/cviu.1995.1004. Available from: `http://dx.doi.org/10.1006/cviu.1995.1004`.

CORTES, C. – VAPNIK, V. Support-vector networks. *Machine Learning*. 1995, 20, 3, pages 273–297. ISSN 0885-6125. doi: 10.1007/BF00994018. Available from: `http://dx.doi.org/10.1007/BF00994018`.

CRISTIANINI, N. – SHAWE-TAYLOR, J. *An introduction to support Vector Machines: and other kernel-based learning methods*. New York, NY, USA : Cambridge University Press, 2000. ISBN 0-521-78019-5.

DANISMAN, T. et al. Drowsy driver detection system using eye blink patterns. In *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, pages 230–233, 2010. doi: 10.1109/ICMWI.2010.5648121.

DINGES, D. F. et al. Evaluation of Techniques for Ocular Measurement as an Index of Fatigue and the Basis for Alertness Management. Final report, April 1998.

DINGES, D. F. PERCLOS: A Valid Psychophysiological Measure of Alertness As Assessed by Psychomotor Vigilance. Technical Report FHWA-MCRT-98-006, Office of Motor Carrier Research and Standards, 1998.

DINH, H. – JOVANOV, E. – ADHAMI, R. Eye Blink Detection Using Intensity Vertical Projection. In *Eye Blink Detection Using Intensity Vertical Projection*, Huntsville, Alabama, USA, 2012. Dept. Electrical and Computer Engineering, University of Alabama in Huntsville. Available from: `http://www.iiis.org/CDs2012/CD2012SCI/IMETI_2012/PapersPdf/FA112HK.pdf`.

DIVJAK, M. – BISCHOF, H. Eye Blink Based Fatigue Detection for Prevention of Computer Vision Syndrome. In *IAPR Conference on Machine Vision Applications (MVA 2009)*, pages 350–353, May 2009. Available from: http://www.mva-org.jp/Proceedings/2009CD/papers/10-04.pdf.

D'ORAZIO, T. et al. A new algorithm for ball recognition using circle Hough transform and neural classifier. *Pattern Recognition*. March 2004, 37, 3, pages 393–408. ISSN 0031-3203. doi: 10.1016/s0031-3203(03)00228-0. Available from: http://dx.doi.org/10.1016/s0031-3203(03)00228-0.

D'ORAZIO, T. et al. A visual approach for driver inattention detection. *Pattern Recogn.* August 2007, 40, 8, pages 2341–2355. ISSN 0031-3203. doi: 10.1016/j.patcog.2007.01.018. Available from: http://dx.doi.org/10.1016/j.patcog.2007.01.018.

DRUTAROVSKY, T. Eye-Blink Detection Using Gradient Orientations. In *Proceedings of 10th Student Research Conference in Informatics and Information Technologies*, volume 2, pages 523–528. Publisher STU, April 2014. ISBN 978-80-227-4152-1.

FITZGIBBON, A. W. – FISHER, R. B. A buyer's guide to conic fitting. In *Proceedings of the 6th British conference on Machine vision (Vol. 2)*, BMVC '95, pages 513–522, Surrey, UK, UK, 1995. BMVA Press. Available from: http://dl.acm.org/citation.cfm?id=243124.243148. ISBN 0-9521898-2-8.

FLORES, M. – ARMINGOL, J. – ESCALERA, A. Real-time drowsiness detection system for an intelligent vehicle. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 637–642, 2008. doi: 10.1109/IVS.2008.4621125.

FREUND, Y. – SCHAPIRE, R. E. A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*. 1997, 55, 1, pages 119 – 139. ISSN 0022-0000. doi: http://dx.doi.org/10.1006/jcss.1997.1504. Available from: http://www.sciencedirect.com/science/article/pii/S002200009791504X.

FROBA, B. – ERNST, A. Face detection with the modified census transform. In *Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on*, pages 91–96, 2004. doi: 10.1109/AFGR.2004.1301514.

GARCIA, I. et al. Vision-based drowsiness detector for a realistic driving simulator. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 887–894, 2010. doi: 10.1109/ITSC.2010.5625097.

GARCIA, I. et al. Vision-based drowsiness detector for real driving conditions. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 618–623, 2012. doi: 10.1109/IVS.2012.6232222.

GRAUMAN, K. et al. Communication via eye blinks - detection and duration analysis in real time. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–1010–I–1017 vol.1, 2001. doi: 10.1109/CVPR.2001.990641.

GRAUMAN, K. et al. Communication via eye blinks and eyebrow raises: video-based human-computer interfaces. *Universal Access in the Information Society*. 2003, 2, 4, pages 359–373. ISSN 1615-5289. doi: 10.1007/s10209-003-0062-x. Available from: http://dx.doi.org/10.1007/s10209-003-0062-x.

HÄKKÄNEN, H. et al. Blink duration as an indicator of driver sleepiness in professional bus drivers. *SLEEP*. 1998, 22, 6, pages 798 – 802. Available from: http://www.ncbi.nlm.nih.gov/pubmed/10505826.

HARIRI, B. et al. A Yawning Measurement Method to Detect Driver Drowsiness. Technical report, CogniVue Corporation, 2011.

HEISHMAN, R. – DURIC, Z. Using Image Flow to Detect Eye Blinks in Color Videos. In *Applications of Computer Vision, 2007. WACV '07. IEEE Workshop on*, pages 52–52, 2007. doi: 10.1109/WACV.2007.61.

HIGGINS, L. – BERNIE, F. Drowsy Driving, April 2011. Available from: http://tti.tamu.edu/group/stsc/files/2010/11/Drowsy_Driving.pdf.

HSU, C.-W. – CHANG, C.-C. – LIN, C.-J. A Practical Guide to Support Vector Classification. Technical report, Department of Computer Science, National Taiwan University, 2003. Available from: http://www.csie.ntu.edu.tw/~cjlin/papers.html.

ISARD, M. – BLAKE, A. CONDENSATIONâ€"Conditional Density Propagation for Visual Tracking. *International Journal of Computer Vision*. 1998, 29, 1, pages 5–28. ISSN 0920-5691. doi: 10.1023/A:1008078328650. Available from: http://dx.doi.org/10.1023/A%3A1008078328650.

JI, Q. – ZHU, Z. – LAN, P. Real-time nonintrusive monitoring and prediction of driver fatigue. *Vehicular Technology, IEEE Transactions on*. 2004, 53, 4, pages 1052–1068. ISSN 0018-9545. doi: 10.1109/TVT.2004.830974.

KALAL, Z. – MATAS, J. – MIKOLAJCZYK, K. Online learning of robust object detectors during unstable tracking. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, pages 1417–1424, 2009. doi: 10.1109/ICCVW.2009.5457446.

KALAL, Z. – MATAS, J. – MIKOLAJCZYK, K. P-N learning: Bootstrapping binary classifiers by structural constraints. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 49–56, 2010a. doi: 10.1109/CVPR.2010.5540231.

KALAL, Z. – MIKOLAJCZYK, K. – MATAS, J. Face-TLD: Tracking-Learning-Detection applied to faces. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 3789–3792, 2010b. doi: 10.1109/ICIP.2010.5653525.

KALAL, Z. – MIKOLAJCZYK, K. – MATAS, J. Forward-Backward Error: Automatic Detection of Tracking Failures. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2756–2759, 2010c. doi: 10.1109/ICPR.2010.675.

KALAL, Z. – MIKOLAJCZYK, K. – MATAS, J. Tracking-Learning-Detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 2012, 34, 7, pages 1409–1422. ISSN 0162-8828. doi: 10.1109/TPAMI.2011.239.

KLAUER, C. 100-Car Naturalistic Driving Study. Study, December 2012. Available from: http://www.vtti.vt.edu/research/vrus/projects/100car/100car.html.

KOZAK, K. et al. Leading indicators of drowsiness in simulated driving. In *Human Factors Ergonomics Society 49th Annu. Meeting*, pages 1917–1921, sep. 2005.

KRÓLAK, A. – STRUMIłłO, P. Eye-blink detection system for humanâ€"computer interaction. *Universal Access in the Information Society*. 2012, 11, 4, pages 409–419. ISSN 1615-5289. doi: 10.1007/s10209-011-0256-6. Available from: http://dx.doi.org/10.1007/s10209-011-0256-6.

KUMAR, R. – RAJA, S. – RAMAKRISHNAN, A. G. Eye detection using color cues and projection functions. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–337–III–340 vol.3, 2002. doi: 10.1109/ICIP.2002.1038974.

KURYLYAK, Y. – LAMONACA, F. – MIRABELLI, G. Detection of the eye blinks for human's fatigue monitoring. In *Medical Measurements and Applications Proceedings (MeMeA), 2012 IEEE International Symposium on*, pages 1–4, 2012. doi: 10.1109/MeMeA.2012.6226666.

LEE, W. O. – LEE, E. C. – PARK, K. R. Blink detection robust to various facial poses. *Journal of Neuroscience Methods*. September 2010. ISSN 01650270. doi: 10.1016/j.jneumeth.2010.08.034. Available from: http://dx.doi.org/10.1016/j.jneumeth.2010.08.034.

LENSKIY, A. A. – LEE, J.-S. Drivers eye blinking detection using novel color and texture segmentation algorithms. *International Journal of Control, Automation and Systems*. 2012, 10, 2, pages 317 – 327. ISSN 1598-6446. doi: 10.1007/s12555-012-0212-0. Available from: http://link.springer.com/article/10.1007%2Fs12555-012-0212-0.

LIENHART, R. – MAYDT, J. An extended set of Haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900–I–903 vol.1, 2002. doi: 10.1109/ICIP.2002.1038171.

LOWE, D. G. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*. November 2004, 60, 2, pages 91–110. ISSN 0920-5691. doi: 10.1023/B:VISI.0000029664.99615.94. Available from: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

MILBORROW, S. – NICOLLS, F. Locating Facial Features with an Extended Active Shape Model. *ECCV*. 2008. http://www.milbo.users.sonic.net/stasm.

MILBORROW, S. Locating Facial Features with Active Shape Models. Master's thesis, The Faculty of Engineering, University of Cape Town, South Africa, November 2007.

NHTSA. Traffic safety facts crash stats: drowsy driving. Technical Report DOT HS 811 449, NHTSA's National Center for Statistics and Analysis, 1200 New Jersey Avenue SE., Washington, DC 20590, 2011.

NIU, Z. et al. 2D Cascaded AdaBoost for Eye Localization. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 1216–1219, 2006. doi: 10.1109/ICPR.2006.1194.

OTSU, N. A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*. January 1979, 9, 1, pages 62–66. ISSN 0018-9472. doi: 10.1109/tsmc.1979.4310076. Available from: http://dx.doi.org/10.1109/tsmc.1979.4310076.

PAPAGEORGIOU, C. – OREN, M. – POGGIO, T. A general framework for object detection. In *Computer Vision, 1998. Sixth International Conference on*, pages 555–562, Jan 1998. doi: 10.1109/ICCV.1998.710772.

ROWLEY, H. – BALUJA, S. – KANADE, T. Neural network-based face detection. In *Computer Vision and Pattern Recognition, 1996. Proceedings CVPR '96, 1996 IEEE Computer Society Conference on*, pages 203–208, 1996. doi: 10.1109/CVPR.1996.517075.

SCHLEICHER, R. et al. Blinks and saccades as indicators of fatigue in sleepiness warnings: looking tired? *Ergonomics*. July 2008, 51, 7, pages 982 – 1010. doi: 10.1080/00140130701817062. Available from: www.ncbi.nlm.nih.gov/pubmed/18568959.

SHI, J. – TOMASI, C. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994. doi: 10.1109/CVPR.1994.323794.

SMITH, P. – SHAH, M. – VITORIA LOBO, N. Determining driver visual attention with one camera. *Intelligent Transportation Systems, IEEE Transactions on*. 2003, 4, 4, pages 205–218. ISSN 1524-9050. doi: 10.1109/TITS.2003.821342.

STERN, J. – BOYER, D. – SCHROEDER, D. Blink rate: a possible measure of fatigue. *Human Factors*. august 1994, 36, 2, pages 285 – 297. Available from: http://www.ncbi.nlm.nih.gov/pubmed/8070793. AD-A284 779.

STERN, J. A. – WALRATH, L. C. – GOLDSTEIN, R. The Endogenous Eyeblink. *Psychophysiology*. 1984, 21, 1, pages 22–33. ISSN 1469-8986. doi: 10.1111/j.1469-8986.1984.tb02312.x. Available from: http://dx.doi.org/10.1111/j.1469-8986.1984.tb02312.x.

SUZUKI, M. et al. Measurement of Driver's Consciousness by Image Processing -A Method for Presuming Driver's Drowsiness by Eye-Blinks coping with Individual Differences. In *Systems, Man and Cybernetics, 2006. SMC '06. IEEE International Conference on*, volume 4, pages 2891–2896, 2006. doi: 10.1109/ICSMC.2006.385313.

TIMM, F. – BARTH, E. Accurate Eye Centre Localisation by Means of Gradients. In MESTETSKIY, L. – BRAZ, J. (Ed.) *VISAPP*, pages 125–130. SciTePress, 2011. Available from: http://dblp.uni-trier.de/db/conf/visapp/visapp2011.html#TimmB11. ISBN 978-989-8425-47-8.

TOMASI, C. – KANADE, T. Detection and Tracking of Point Features. Technical report, Computer Science Department, Carnegie Mellon University, April 1991.

TSOCHANTARIDIS, I. et al. Large Margin Methods for Structured and Interdependent Output Variables. *J. Mach. Learn. Res.* December 2005, 6, pages 1453–1484. ISSN 1532-4435. Available from: http://dl.acm.org/citation.cfm?id=1046920.1088722.

UŘIČÁŘ, M. – FRANC, V. – HLAVÁČ, V. Facial Landmarks Detector Learned by the Structured Output SVM. In CSURKA, G. et al. (Ed.) *Computer Vision, Imaging and Computer Graphics. Theory and Application*, volume 359 of *Communications in Computer and Information Science*. Czech Technical University in Prague, Technicka 2, 166 27 Prague 6, Czech Republic: Springer Berlin Heidelberg, 2013. pages 383–398. doi: 10.1007/978-3-642-38241-3_26. Available from: http://dx.doi.org/10.1007/978-3-642-38241-3_26. ISBN 978-3-642-38240-6.

VIOLA, P. – JONES, M. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. doi: 10.1109/CVPR.2001.990517.

VIOLA, P. – JONES, M. J. Robust Real-Time Face Detection. *Int. J. Comput. Vision*. May 2004, 57, 2, pages 137–154. ISSN 0920-5691. doi: 10.1023/B:VISI.0000013087.49260.fb. Available from: http://dx.doi.org/10.1023/B:VISI.0000013087.49260.fb.

VURAL, E. et al. Drowsy driver detection through facial movement analysis. In *Proceedings of the 2007 IEEE international conference on Human-computer interaction*, HCI'07, pages 6–18, Berlin, Heidelberg, 2007. Springer-Verlag. Available from: http://dl.acm.org/citation.cfm?id=1779576.1779578. ISBN 3-540-75772-4, 978-3-540-75772-6.

WANG, P. – JI, Q. Multi-view face and eye detection using discriminant features. *Computer Vision and Image Understanding*. 2007, 105, 2, pages 99 – 111. ISSN 1077-3142. doi: http://dx.doi.org/10.1016/j.cviu.2006.08.008. Available from: http://www.sciencedirect.com/science/article/pii/S1077314206001354.

WIERWILLE, W. W. et al. Research on Vehicle-Based Driver Status/Performance Monitoring; Development, Validation, and Refinement of Algorithms for Detection of Driver Drowsiness. Three-year Report DOT HS 808 247, Vehicle Analysis and Simulation Laboratory, Virginia Polytechnic Institute and State University, 400 Seventh Street, SW, Washington, DC 20590, 1994.

WILLIAMSON, A. M. – FEYER, A.-M. Moderate sleep deprivation produces impairments in cognitive and motor performance equivalent to legally prescribed levels of alcohol intoxication. *Occupational and Environmental Medicine*. October 2000, 57, 10, pages 649–655. doi: 10.1136/oem.57.10.649. Available from: http://dx.doi.org/10.1136/oem.57.10.649.

YAN, Z. et al. Computer Vision Syndrome: A widely spreading but largely unknown epidemic among computer users. *Comput. Hum. Behav.* September 2008, 24, 5, pages 2026–2042. ISSN 0747-5632. doi: 10.1016/j.chb.2007.09.004. Available from: http://dx.doi.org/10.1016/j.chb.2007.09.004.

YANG, F. et al. Robust eyelid tracking for fatigue detection. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, pages 1829–1832, 2012. doi: 10.1109/ICIP.2012.6467238.

ZACH, C. – POCK, T. – BISCHOF, H. A duality based approach for realtime TV-L1 optical flow. In *Proceedings of the 29th DAGM conference on Pattern recognition*, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag. Available from: http://dl.acm.org/citation.cfm?id=1771530.1771554. ISBN 978-3-540-74933-2.

ZHOU, Z.-H. – GENG, X. Projection functions for eye detection. *Pattern Recognition*. 2004, 37, 5, pages 1049 – 1056. ISSN 0031-3203. doi: http://dx.doi.org/10.1016/j.patcog.2003.09.006. Available from: http://www.sciencedirect.com/science/article/pii/S0031320303003674.

Appendix A

Technical Documentation

All methods of our solution are implemented in C++ language using the OpenCV library version 2.4.6. We list the most important OpenCV functions we use in our programs:

- DescriptorExtractor::compute (...) – functions computes specific descriptor from an input frame and a keypoint vector.
- Sobel (...) – function applies generalized Sobel operator to an input frame.
- convertScaleAbs (...) – function scales array elements, compute absolute values and convert them to 8-bit unsigned integers.
- normalize (...) – function normalizes an input vector according to parameters.
- CvSVM::predict (...) – function predicts a response from SVM from an input sample.
- CascadeClassifier::detectionMultiScale (...) – functions detects object in an input image.
- calcOpticalFlowPyrLK (...) – function computes optical flow using multi-scale Lucas-Kanade algorithm (function tries to find a new position for each input feature point).

Almost every method uses eye detection using Viola – Jones detection algorithm. We use `haarcascade_frontalface_default.xml` file to detect face in each framw.The we use `haarcascade_mcs_eyepair_big.xml` file to detect eye pair, which is more precise than separate eye detection using `haarcascade_eye_tree_eyeglasses.xml`.

In this bachelor thesis we proposed and implemented a couple of methods aimed at an eye-blink detection. We show the most important ones. The following function is used to create orientation function in gradient descriptors (static detection), which consists of 360 bins (Listing A.1). Each bin is adjusted with value from the weight map. Therefore, we can get rid of non-significant gradients.

Next listing represents partial pseudocode of sequential analysis detection algorithm using the cell method (Listing A.2).

Another interesting function is the calculation of bounding box for feature points in the sequential analysis methods (Listing A.3). This algorithm is essential, because bounding box of an eye region affects the precision of cell sorting method.

In thesis we use the cell method to determinate whether a blink occurred or not (Listing A.4). Status determination of eye cells is a primary part of our sequential analysis algorithm. We perform determination for each eye separately using two individual state machines.

Listing A.1: Gradient calculation using weights

```
Mat gradientWeightMatTemp (Mat image){
        Mat angles = getGradientAngles (Mat image);
        Mat magnitudes = getGradientMagnitudes (Mat image);
        Mat bins;

        //sort angles into bins
        for (int y = 0; y < angles.rows; y++){
                for (int x = 0; x < angles.cols; x++){
                        if ((int) magnitudes.at<uchar>(y, x) > 10){
                                //get particular bin
                                if (angles.at<float>(y, x) < 0){
                                        temp = roundf (angles.at<float>(y
                                            , x) + 360.0);
                                }
                                else{
                                        temp = roundf (angles.at<float>(y
                                            , x));
                                }
                                weight = weights.at<float>(y, x);
                                increaseCurrentBin (temp, weight);
                                //+-10 degrees dispersion
                                increaseNeighbourBins (temp, weight);
                        }
                }
        }
        return bins;
}
```

Listing A.2: Blink detection using sequential analysis.

```
void blinkDetection (){
        initializeVariables ();
        previous = getFrame ();

        while (true){
                current = getFrame ();

                if (initialize){
                        initializeVariables ();
                        eyes = getEyes (current);
                        leftFeatures = placeFeatures (eyes[0]);
                        rightFeatures = placeFeatures (eyes[1]);
                        initialize = false;
                }

                leftFeaturesNext = opticalFlow (current, previous,
                   leftFeatures, dots);
                rightFeaturesNext = opticalFlow (current, previous,
                   rightFeatures, dots);

                removeBadPoints (leftFeaturesNext, rightFeaturesNext);

                leftCenter = getCenter (rightFeaturesNext);
                rightCenter = getCenter (rightFeaturesNext);
                leftRect = getRectangle (rightFeaturesNext, leftCenter);
                rightRect = getRectangle (rightFeaturesNext, rightCenter)
                   ;

                removeOutliers (leftFeaturesNext, rightFeaturesNext,
                   leftRect, rightRect);

                leftCellsCurrent = sortPointsInCells (leftFeaturesNext,
                   leftRect);
                rightCellsCurrent = sortPointsInCells (rightFeaturesNext,
                    rightRect);

                leftBlink = getStatus (leftCellsCurrent,
                   leftCellsPrevious, dots);
                rightBlink = getStatus (rightCellsCurrent,
                   rightCellsPrevious, dots);

                if (leftBlink || rightBlink){ println (``BLINK!");}

                if (lostPoints > threshold){initialize = true;}

                swap (leftFeaturesNext, leftFeatures);
                swap (rightFeaturesNext, rightFeatures);
                swap (leftCellsCurrent, leftCellsPrevious);
                swap (rightCellsCurrent, rightCellsPrevious);
                swap (current, previous);

                if (keyPressed){
                        break;
                }
        }
}
```

Listing A.3: Bounding box of eye region features.

```
Rect getRectangle (vector<Point2f> points, Point2f center){
        vector<int> histogram (200, 0);

        float radius, xDist, yDist;
        //histogram sorting according to~the radius from the eye center
        for (int i = 0; i < points.size (); i++){
                xDist = points[i].x - center.x;
                yDist = points[i].y - center.y;

                radius = sqrtf (xDist*xDist + yDist*yDist);
                histogram[(int) roundf (radius)]++;
        }
        //looking for global maximum value
        int max = 0, maxIndex = 1000;
        int cutIndex = 0;
        for (int i = 1; i < histogram.size () - 1; i++){
                if (histogram[i] > histogram[i+1] && max < histogram[i]){
                        max = histogram[i];
                        maxIndex = i;
                }
        }
        //looking for the distance gap
        for (int i = maxIndex; i < histogram.size () - 1; i++){
                if (histogram[i] <= 5 && histogram[i+1] <= 5 && histogram
                    [i+2] <= 5){
                        //cut distance from radius
                        cutIndex = i;
                        break;
                }
        }
        //adjusting dimensions of the final rectangle
        cutIndex = cutIndex*0.8;
        Rect rect;
        rect.x = center.x - ((float) cutIndex) +1;
        rect.y = center.y - ((float) cutIndex) +1;
        rect.width = cutIndex*2;
        rect.height = cutIndex*2;

        return rect;
}
```

Listing A.4: Bounding box of eye region features.

```
bool getStatus (vector<float> current, vector<int> currentCount, vector<
    float> previous, vector<int> previousCount, int distance, int &state,
    int &blinkFrame, float fps){
        vector<float> currentMoves (9, 0);
        vector<float> previousMoves (9, 0);
        vector<float> change (6, 0);

        //cell changes calculation
        for (int i = 0; i < 6; i++){
                currentMoves[i] = current[i] / currentCount[i];
                previousMoves[i] = previous[i] / previousCount[i];
                change[i] = currentMoves[i] - previousMoves[i];
        }
        //threshold was chosen according to~tests
        float threshold = distance * 0.00068 * (30/fps);

        float upper = change[1];
        float lower = change[4];

        if (state == 0){
                if (lower - upper > 0 && lower > 0 && getVariance (change
                    ) > threshold){
                        state = 1;
                        blinkFrame++;
                }
        }
        else if (state == 1){
                blinkFrame++;

                if (blinkFrame < 3){
                        state = 1;
                }
                else if (blinkFrame > 4){
                        state = 0;
                        blinkFrame = 0;
                }
                else if (lower - upper < 0 && lower < 0 && getVariance (
                    change) > threshold){
                        state = 2;
                }
        }

        if (state == 2){
                state = 0;
                blinkFrame = 0;
                //BLINK occurred
                return true;
        }

        //blink did not occur
        return false;
}
```

Appendix B

User Guide

To be able to run our prototypes, there are several requirements:

1. Qt 5.1.1,
2. OpenCV 2.4.6 compiled for Qt 5 (can be also downloaded from attached DVD),
   - default directory: `C:\opencv\`
3. Whole project solutions directories copied to your hard drive,
4. If the OpenCV installation is not in the default directory, it is necessary to change paths in all `.pro` files.

After correct installation you can open one of the proposed projects in Qt Creator. All solutions require changes in source code to run program properly with requested parameters or input paths. We provide following project solutions, which are also attached on the DVD:

- Gradient methods, SIFT and IVP method
  - descriptor building
  - static testing on images
  - offline video testing
  - online video testing
- Sequential analysis methods
  - sequential testing on images
  - offline video testing
  - online video testing
  - annotation parsing and evaluation

Gradient descriptor methods

Our gradient methods are implemented in the project `gradient`. It is necessary to adjust global variable `binsNumber` value to 8, which represents the number of elements in the result descriptor. After that, `openFiles` and `closedFiles` variables need to be set, because they

represent amount of positive and negative input values. Location of input SVM classifier can be also changed. Furthermore `choice` variable can be changed to 1 in case of non-weighted gradient method and 2 in case of weighted gradient method. After running the project, result values of dataset samples are printed in the console.

Method can be tested online and offline on video sequence in the project `bachelor`, which provides testing and SVM building framework. It is necessary to adjust input paths in the file `menu.cpp` and `builder.cpp`.

SIFT method

SIFT descriptor is implemented in the project `gradient`. It is necessary to adjust global variable `binsNumber` value to 128, which represents the number of elements in the result descriptor. After that, `openFiles` and `closedFiles` variables need to be set, because they represent amount of positive and negative input values. Location of input SVM classifier can be also changed. Furthermore `choice` variable need to be changed to 3. After running the project, result values of dataset samples are printed in the console.

Method can be tested online and offline on video sequence in the project `bachelor`, which provides testing and SVM building framework. It is necessary to adjust input paths in the file `menu.cpp` and `builder.cpp`.

IVP descriptor method

IVP descriptor is implemented in the project `gradient`. It is necessary to adjust global variable `binsNumber`, which represents the number of elements in the result descriptor. After that, `openFiles` and `closedFiles` variables need to be set, because they represent amount of positive and negative input values. Location of input SVM classifier can be also changed. Furthermore `choice` variable need to be changed to 4. After running the project, result values of dataset samples are printed in the console.

Method can be tested online and offline on video sequence in the project `bachelor`, which provides testing and SVM building framework. It is necessary to adjust input paths in the file `menu.cpp` and `builder.cpp`.

SVM training

Static methods are tested using SVM, which can be build in project the `bachelor`, which provides testing and SVM building framework. It is necessary to adjust input paths in the file `builder.cpp` and to choose method for SVM building.

Sequential methods

Sequential methods can be tested in the project `blinkFinal`, which offers more global variables (parameters) to change:

- `eyePair` – if `true`, then Viola – Jones uses `haarcascade_mcs_eyepair_big` instead of `haarcascade_eye_tree_eyeglasses`.

- `debug` – if `true`, then program shows debug image with visualized features and eye rectangles.
- `saving` – if `true`, then program saves blink values to save path.
- `fpsFromFile` – if `true`, then FPS values are taken from the input text file.
- `stepMode` – if `true`, then it is possible to step the debug using the arbitrary key.
- `print` – if `true`, then output debug prints in the console.
- `pictures` – if `true`, then input samples are image form, not video.

After successful testing using sequential analysis, we can evaluate tests using the project `blinkReader`, which parses created output blink file and it evaluates tests according to the annotation file. We can choose between picture or video blink reader.

Video Recorder

We offer our video recorder in the project `videoRec2`, which is able to record video files for purpose of testing. It is recommended to download `x264` video codec.

Appendix C

IIT.SRC 2014 paper

Eye-blink Detection Using Gradient Orientations

Tomáš DRUTAROVSKÝ*

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies
Ilkovičova 2, 842 16 Bratislava, Slovakia
xdrutarovsky@is.stuba.sk

Abstract. State-of-the-art algorithms offer a real-time human face and eye detection which can be used to detect eye-blink. In this paper we present a feature descriptor. Gradient orientations and magnitudes are used to build the feature descriptor. We use the difference between samples of open and closed eyes. Gradient descriptors are further used to train the SVM. SVM can predict the open or closed eye state from the given input data. Due to the knowledge of the state of the eye (open or closed), eye-blink frequency and duration can be computed. These parameters are used to establish the level of sleepiness what can be used to prevent the driver from incoming microsleep.

1 Introduction

Today's technologies offer us the opportunity to capture video of a driver in the car. Information about eye openness can contribute to the estimation of blink frequency and duration. These two values are considered to be the main signs of sleepiness [10]. Further analysis of these parameters can contribute in microsleep prevention.

Webcams and smartphones are available and provide a real-time image acquiring with sufficient frame resolution. World of computer vision knows algorithms for human's face and eye detection and tracking what can be used for real-time eye sample processing. This leads to ideas of blink detection and eye state determination. Several works are devoted to the distinction between *open* and *closed* eye, but all of them have gaps in the robustness of the solution. That is because of different environment conditions, illuminations changes and various eye appearances.

Our goal is to construct a gradient orientation descriptor, which could describe open and closed eyes differently so it could be used to train the Support Vector Machine (SVM) [2]. We propose a method of gradient information calculation from the frame and processing them so the gradient features are more distinguishable. Knowing the difference between open and closed eye provides several advantages. We can use the information to detect eye-blinks or percentage of eye closure.

---

* Bachelor study programme in field: Informatics
Supervisor: Ing. Andrej Fogelton, Institute of Applied Informatics, Faculty of Informatics and Information Technologies STU in Bratislava

IIT.SRC 2014, Bratislava, April 29, 2014, pp. 1–8.

2 Related work

The automatic drowsy driver monitoring and accident prevention system was presented in [3]. The system analyzes pupils using *Horizontal Symmetry Calculation* (HSC). The system receives input colored frames from a video camera and measures the eye-blink duration of a driver constantly. After face detection the neural network-based detector is used for the precise eye pupils position. The head rotation angle is calculated using the vertical position of both pupils. If eye detection in the next frame fails, the angle helps to determine the right face and eye position. Detected pupils are analyzed using HSC to determine whether the eyes are *Open* or *Closed*. HSC uses the fact that folded closed eye samples have more difference pixels than open eye samples. Mentioned algorithm was tested on ZJU database and it achieved 94.8% accuracy for eye-blink detection. This method is dependent on samples' quality and precise eye alignment and cropping.

Work presented in [7] solves the problem of blink detection using the frame pixel difference. Authors estimate blink when the pixel intensity difference of consecutive frames is higher than the preset threshold. Method also uses thresholds to consider a non-uniform change of the illumination for each eye in consecutive frames and also an exclusion of voluntary blinks from the further analysis. Similar method was proposed in [6] where authors localize eye positions according to the significant change of the intensity in the consecutive frames. After successful eye localization, the system learns how an open eye appears so it can be used in the next phase of blink detection. Eye closure is estimated due to correlation score between current eye template and learned template examples. Mentioned algorithms require video samples with no bigger face moves and uses relatively many thresholds.

In the paper [5] authors proposed a vision-based drowsiness detector which can detect a driver in a realistic driving simulator. This detector is based on the infrared stereo camera. The detector constantly tracks eyes and estimates the percentage of closure also known as *PERCLOS*. For the best estimation, PERCLOS values were compared to results of several psychological experiments. First, face and eyes are detected using Viola – Jones detector [12] and detection failures are then corrected using *Kalman filter*. Subsequently, the sequence of filters is used to improve the frame quality so the PERCLOS can be estimated more accurately. PERCLOS is calculated from the ratio between the iris height in the frame and the nominal value which is assigned during a ten-second calibration at the start of tracking. This detector reaches high recall (90.68%) and low false positive rates using their own database consisting of 25 hours of driving. However, the system uses an infrared stereo camera what makes it an expensive solution with high hardware requirements.

Another method of eye state determination is measurement of pixel amount in eye regions. Authors in [4] presented a method of eye-blink detection using *intensity horizontal projection (IHP)*. The method uses the fact that iris has lower IHP value than other regions around the eye. This means that closing of the eyelid causes noticeable changes in the histogram of IVP values. On the other hand, algorithm presented in [8] measures eye-blink using *intensity vertical projection (IVP)*. After applying the median filter, the IVP of eye regions without eyebrows is measured. It is considered that the ratio of maximal IVP to minimal IVP for the open eye is higher than for the closed eye. Open eye has also higher maximal IVP value than closed eye. Although these methods seems obvious, they work accurately only for specific eye types. Therefore we can not consider them robust and reliable enough.

3 Gradient Orientation Descriptor

One of the most significant characteristics for the human discrimination of eye states is an ability to recognize shape. Shape is often described by gradient orientations. We want to use that fact and build a feature descriptor which could help the computer to discriminate different eye states using the gradient orientations. We propose the gradient calculation and orientation sorting for each eye sample. We use weight map to adjust weights of values which are added to the orientation bins.

3.1 Gradient calculation and sorting

First step of descriptor construction is the gradient calculation from the image. In this paper we do not discuss face and eye detection and tracking. We assume that the input eye sample contains nothing but eye aligned in the center of the sample. In our work we used Viola – Jones detector to obtain eye rectangles. After that, we align rectangles according to the pupils using the gradient pupil locator. In the case of the closed eye sample, we assume that eyelashes or eyelids' link is located in the center of the sample.

Our method uses *Sobel* operator for the edge detection, because it is partly rotation invariant. Sobel gives better results in computing diagonal edges than the edge detector which uses $[-1, 0, 1]$ kernel and image preprocessed with Gaussian filter.

Input image is decomposed into derivatives – horizontal image gradients dx and vertical image gradients dy. Using these two images we can compute gradient magnitude m for each pixel as maximum of absolute values from both images (Equation 1). We can also compute gradient orientation α using the tangent function (Equation 2). This approach was inspired by the work presented in [1].

$$m_{x,y} = max(abs(dx_{x,y}), abs(dy_{x,y})) \tag{1}$$

$$\alpha_{x,y} = \tan(dx_{x,y}, dy_{x,y}) \tag{2}$$

Another important operation is constructing of the weight map. This map is used to control the weight of the image pixels so the pixels in the center of the image have higher weights than pixels at the image border. Each point in the eye sample has own weight $w_{x,y} = e^{n_{x,y}}$. In our algorithm, $n_{x,y}$ is linear interpolation between 0 for border pixels and 12 for center pixels, according to the pixels position in the weight map. Exponent values were chosen due to the empirical test results. This approach is inspired by the FREAK descriptor [11] which used knowledge about human's retina and vision sharpness.

After these operations, we have two essential values for each image pixel so we can sort gradient orientations into 360 bins. Each gradient orientation is dispersed from -10 to 10 degrees and classified into bins to avoid errors in orientation calculation. Appropriate bin is increased with value from weighted map. However, we consider only gradients with magnitude higher than 10 (we consider magnitude values from 0 to 255) to avoid non-significant gradients.

3.2 Orientations function

Following step of the gradient descriptor construction is the processing of the 360-binned orientation function. Input orientation function has a lot of local minima and maxima, therefore we smooth the function. Each bin is smoothed using average value from the four closest bins. We smooth the function until it has 4 extremes exactly – two minima and two maxima. As we consider aligned eye samples, these extremes are often around 0, 90, 180, 270 degrees. We use the fact that open eyes (Figure 1) have maximum with highest value around 90 degrees, but closed eyes have higher maximum around 270 degrees bin (Figure 2). That is because an open eye has more gradients oriented vertically down near the eye center. Moreover, open eye sample has bigger disperse of the maximum peak what is caused by higher amount of horizontal orientations around the iris.

Smoothed function is then aligned to the first local maximum. Rotation invariance can be guaranteed by shifting bin values according to the rotation angle. This angle can be computed as the angle between line connecting both irises and horizontal line.

Orientations function with 4 extremes is used to construct the final gradient descriptor. Our proposed descriptor consists of 8 numbers – four pairs. Each pair represents one extreme or peak and consists of two numbers $rate_{x,y}$ and $distance_{x,y}$. $rate_{x,y}$ represents the rate of the peak height among other peaks. This means that summed size of all four peaks gives 1. $distance_{x,y}$ represents

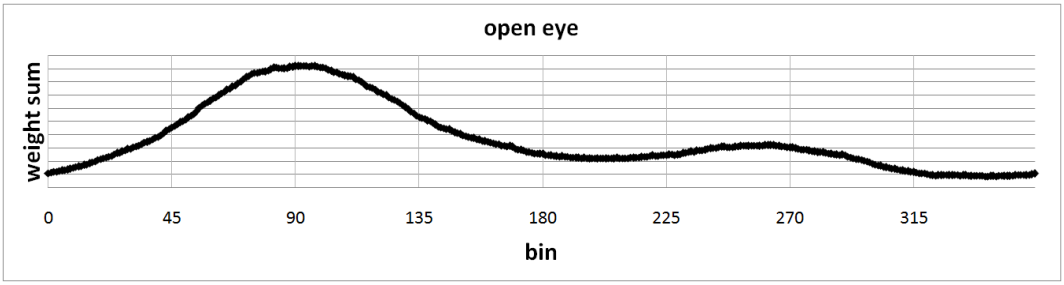


Figure 1. Orientations function of the open eye samples. Function is averaged from 200 eye samples. As we can see, function has two local maxima and first maximum has higher value.

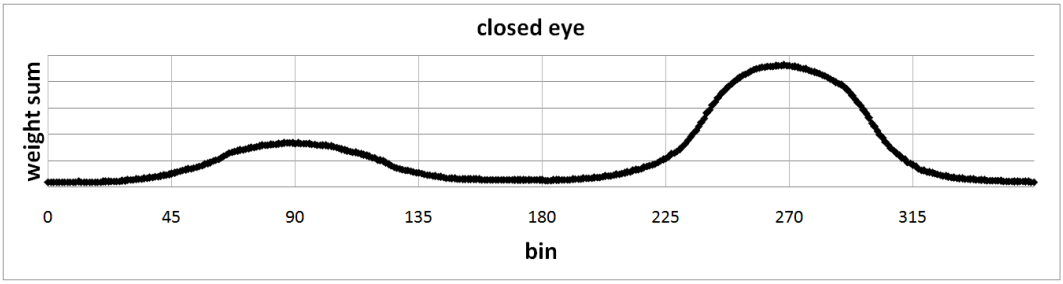


Figure 2. Orientations function of the closed eye samples. Function is averaged from 200 eye samples. Function has also two local maxima, but second maximum has higher value.

the number of bins located between considered peak and the closest right peak. Descriptor $\bar{O}$ represents descriptor of averaged vector from 200 open eye samples (Equation 3).

$$\bar{O} = [0.59, 0.30, 0.10, 0.18, 0.24, 0.22, 0.07, 0.30] \tag{3}$$

4 Evaluation

Both types of eye states (open and closed) are used to train the SVM classifier. In this paper we construct the SVM from 200 open eye and 200 closed eye samples. Trained SVM is used for further state prediction. Testing was performed on other 200 open and 200 closed eyes and results are summarized in the Table 1.

Evaluation of the descriptors' accuracy was performed on our own eye dataset, which consists of the six individuals sitting in front of the camera. Eye samples were acquired using Viola – Jones eye detector and gradient pupil locator at the average resolution of 24×24 pixels.

Our method based on the gradient orientations and image weighting achieved accuracy of 87.8%. We have compared our final gradient descriptor to two other methods.

First of the compared methods is the gradient descriptor without weighting. Absence of weighting supports the fact that the strong links like eyelashes or eyebrows can cause loss of accuracy. This eye features have strong gradients which are not ignored as in the weighting method. Therefore the method achieved accuracy of 71.0%.

Another compared method is SIFT descriptor [9]. Our implementation of the descriptor has one point of interest located in the center of the sample with interest area stretched on the whole sample.

Table 1. Results of all tested methods. TP represents true positive rate (correctly identified closed eye) and TN represents true negative rate (correctly identified open eye).

Method	TP	TN	overall
No weighted	67.5%	74.5%	71.0%
Weighted	93.5%	82.0%	87.8%
Sift	95.0%	93.5%	94.3%

Final descriptor has 128 dimensions – quantized gradient directions, as usual. This descriptor achieved accuracy of 94.3%.

5 Conclusion

In this paper we proposed the method of constructing feature descriptor based on gradient orientations. Our solution involves gradient orientations calculation and sorting, weighting and descriptor building. Descriptors of open and closed eye samples were used to train the SVM, which is used to predict results. Proposed method achieves accuracy of 87.8%.

Although we do not achieve the accuracy of SIFT descriptor, our descriptor consists of 8 numbers only which might result in the potential performance increase. Our method depends on the eye alignment what can affect the gradient orientations and subsequently the feature descriptor. We consider the result encouraging and see the potential of the method in the further enhancement.

We want to focus on the location of significant eye regions and use them to adjust the weighting. Moreover, we want to use the improved descriptor to implement eye-blink detector. Knowing the information about the current eye closure can lead to the estimation of the blink duration and blink frequency. Our future aim is to build a detector with fair trade-off and acceptable solution robustness.

References

[1] Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour Detection and Hierarchical Image Segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2011, vol. 33, no. 5, pp. 898–916.

[2] Cortes, C., Vapnik, V.: Support-vector networks. *Machine Learning*, 1995, vol. 20, no. 3, pp. 273–297.

[3] Danisman, T., Bilasco, I., Djeraba, C., Ihaddadene, N.: Drowsy driver detection system using eye blink patterns. In: *Machine and Web Intelligence (ICMWI), 2010 International Conference on*, 2010, pp. 230–233.

[4] Dinh, H., Jovanov, E., Adhami, R.: Eye Blink Detection Using Intensity Vertical Projection. In: *Proc. of the 5th International Multi-Conference on Engineering and Technological Innovation (IMETI)*, Huntsville, Alabama, USA, Dept. Electrical and Computer Engineering, University of Alabama in Huntsville, 2012.

[5] Garcia, I., Bronte, S., Bergasa, L., Hernandez, N., Delgado, B., Sevillano, M.: Vision-based drowsiness detector for a realistic driving simulator. In: *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 887–894.

[6] Grauman, K., Betke, M., Gips, J., Bradski, G.: Communication via eye blinks - detection and duration analysis in real time. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Volume 1., 2001, pp. I–1010–I–1017 vol.1.

[7] Kurylyak, Y., Lamonaca, F., Mirabelli, G.: Detection of the eye blinks for human's fatigue monitoring. In: *Medical Measurements and Applications Proceedings (MeMeA), 2012 IEEE International Symposium on*, 2012, pp. 1–4.

[8] Lee, W.O., Lee, E.C., Park, K.R.: Blink detection robust to various facial poses. *Journal of Neuroscience Methods*, 2010, vol. 193, no. 2, pp. 356 – 372.

[9] Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision*, 2004, vol. 60, no. 2, pp. 91–110.

[10] Schleicher, R., Galley, N., Briest, S., Galley, L.: Blinks and saccades as indicators of fatigue in sleepiness warnings: looking tired? *Ergonomics*, 2008, vol. 51, no. 7, pp. 982 – 1010.

[11] Vandergheynst, P., Ortiz, R., Alahi, A.: FREAK: Fast Retina Keypoint. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, vol. 0, pp. 510–517.

[12] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. Volume 1., 2001, pp. I–511–I–518 vol.1.

Appendix D

IIT.SRC 2014 poster

Poster was presented at Student Research Conference in Informatics and Information Technologies as presentation of our IIT.SRC paper Eye-Blink Detection Using Gradient Orientations [Drutarovsky 2014].

Everything can change in
the blink of an eye.
But don't worry;
God never blinks.
Regina Brett
Eye-Blink Detection Using Gradient Orientations
VISION & GRAPHICS GROUP
Static detection
IDEA
- gradient descriptor
- using orientations
- SVM prediction
EVALUATION
Method
Descriptor size
TP
TN
Non-weight
8
67.5%
74.5%
Weight
8
93.5%
82.0%
SIFT
128
95.0%
93.5%
- own dataset
- 200 open eyes
- 200 closed eyes
ALGORITHM
Eye detection
Gradient calculation
Orientation function
Descriptor building
DESCRIPTOR
weight sum
bin
Motivation
- eye-blink frequency
- microsleep detection
- Dry Eye Syndrome
Sequential analysis
ALGORITHM
0 1 2
3 4 5
6 7 8
- change in cell 1 and cell 4
- variance threshold
- state machine
IDEA
- features tracking
- inner moves analysis
- KLT Tracker
EVALUATION
Method
Dataset
TP
FP
Sequential
Own
85.0%
18.9%
Sequential Dvjak - Bicchof (2009)
Talking
96.7%
9.2%
Talking
95.0%
19.0%
- 6 subjects
- 8 records
- different conditions
STATE MACHINE
frame count exceeds
INIT
STATE 0
STATE 1
STATE 2 BLINK
down move
up move
IIT.SRC 2014
Author: Tomáš Drutarovský Supervisor: Ing. Andrej Fogelton
STU
FIIT

Appendix E

Resumé

Resumé

1 Úvod

Človek potrebuje spánok. Spánok nie je možnosťou voľby, ale je to nevyhnutná súčasť nášho života. Čím viac sa snažíme nespať, tým viac sa naše oči zatvárajú. Vzhľadom k snahe prekonať spánok, môže ľudské telo upadnúť do mikrospánku. Mikrospánok je krátky a nevedomý moment spánku, ktorý môže trvať od zlomku sekundy až po tridsať sekúnd. Príčinou mikrospánku sú často poruchy spánku, mentálne poruchy, lieky alebo neurologické problémy. Mikrospánok môže nastať takmer kedykoľvek, avšak najnebezpečnejšie je, ak nastane počas šoférovania.

Mikrospánok často nastane bez vedomia vodiča a je sprevádzaný prázdnym pohľadom do diaľky, alebo padnutím hlavy, či predĺženým uzatváraním oka. Dokonca môže nastať aj pri otvorených očiach. Existuje mnoho riešení ako sa vyhnúť spánku za volantom. Niektoré sú lacné, ale nie až tak spoľahlivé. Iné sú pomerne náročné na výpočtovú techniku, no ich presnosť aj tak nie je zaručená. Štúdie hovoria, že zatváranie očného viečka a aktivita žmurkania patria k najspoľahlivejším znakom príchodu mikrospánku. Z tohto dôvodu sa chceme zamerať na sledovanie očí a analýzu žmurkania. Chceme ponúknuť riešenie, ktoré by bolo jednoduché a dostupné a pritom spoľahlivé a dostatočne presné. Cieľom je vytvoriť aplikáciu, ktorá by vedela detegovať žmurknutie a jeho analýzou by prispela k odhaleniu prichádzajúceho mikrospánku.

1.1 Požiadavky

Našim cieľom je vyvinúť aplikáciu, ktorá by poskytovala kompromis medzi presnosťou riešenia a jednoduchosťou použitých metód. Naše riešenie, ktoré by detegovalo vodiča, musí spĺňať niekoľko nasledujúcich požiadaviek:

- presné algoritmy detekcie, ktoré pracujú v reálnom čase,
- akceptovateľný pomer úspešnej detekcie,
- nízke množstvo falošných pozitívov,
- algoritmy pracujúce nad čiernobielou snímkou,
- nevnucujúce sa riešenie.

Aj keď sa chceme zamerať na detekciu za denných podmienok, požadujeme algoritmy pracujúce s čiernobielou snímkou. Chceme, aby naše riešenie bolo ľahko rozšíriteľné o infračervenú kameru, ktorá by vedela detegovať aj v noci.

2 Analýza dostupných riešení

Efektívna prevencia pred mikrospánkom v sebe určite zahŕňa množstvo komerčných aj nekomerčných riešení, ktoré sú schopné odhaliť spánok alebo únavu vodiča. Tieto systémy

môžeme deliť na dve skupiny: detekčné systémy založené na snímaní očí a tie, ktoré oči nesnímajú.

2.1 Systémy nesnímajúce oči

Tieto detekčné systémy nesnímajú oči vodiča, ale na základe iných podnetov sa snažia odhaliť mikrospánok alebo aspoň únavu. Nie sú dostatočne spoľahlivé, avšak ich popularita je vysoká vzhľadom k ich cene a dostupnosti. Mnoho súčasne najznámejších automobilových spoločností ponúka integrované systémy, ktoré dokážu analyzovať správanie vodiča a tým pádom odporučiť prestávku v prípade podozrivého správania. Medzi také systémy patria Driver Alert control od Volvo, Driver Alert od Ford, Attention Assist od Mercedes-Benz, Lane Assist od Volkswagen alebo aj Driver Drowsiness Detector od Bosch.

Inými systémami na detekciu spánku sú systémy monitorujúce prvky ľudského tela. NapZapper je záušný prístroj, ktorý sa aktivuje v prípade padnutia hlavy. Stopsleep ponúka náprstok, ktorý na základe vodivosti kože dokáže predpovedať únavu. Holux vyvinul pás merajúci aktivitu srdca, ktorá často odpovedá reálnej situácii vodiča na ceste.

2.2 Systémy snímajúce oči

Druhou skupinou detekčných systémov sú systémy, ktoré snímajú oči vodiča a na základe meraní dokážu určiť a predpovedať mnoho charakteristík mikrospánku. Jednou z dôležitých charakteristík je aj PERCLOS, čo vyjadruje percentuálne uzatvorenia oka. Analýza tejto hodnoty dokáže verne opísať aktuálny stav človeka – jeho stupeň únavy. Niektoré systémy využívajú Houghovu transformáciu na určenie pozície zreničky a tým pádom aj jej PERCLOS, avšak tieto metódy nevykazujú dostatočne vysoký pomer úspešnosti.

Iné riešenie ponúka detekciu únavy na základe analýzy krivky žmurkaní. Riešenie využíva detekciu tváre a očí za pomoci neurónových sietí. Následne sa pomocou analýzy pixelov v oblasti očí vyextrahujú prvky očí, ktorých analýza poskytne parametre potrebné na určenie žmurknutia. V práci autori merajú pomer dlhých žmurknutí k celkovému počtu žmurknutí, pomer času kedy boli oči zavreté k celkovému času simulácie a tiež aj frekvenciu žmurknutí za jednu minútu.

Riešenie pomocou horizontálnej symetrie využíva geometrické vlastnosti oka. Každá detegovaná vzorka oka sa podrobí analýze, kde je snímka rozdelená na polovice horizontálnou čiarou. Tieto dve polovice sa preložia a vypočíta sa rozdiel intenzít prislúchajúcich pixelov. Na základe prednastaveného prahu sa následne rozhodne o stave oka – otvorené alebo zavreté.

Vďaka analýze dostupných riešení sme zistili, že postup metód nášho algoritmu by mohol byť nasledovný:

- detekcia tváre,
- detekcia očí,
- sledovanie očí,
- detekcia žmurknutia,
- analýza žmurknutí.

3 Detekcia a sledovanie očí a tváre

Medzi hlavné časti nášho algoritmu by nepochybne mali patriť algoritmy, ktoré dokážu detegovať tvár a oči a poprípade ich aj sledovať. Je podstatné, aby takéto algoritmy pracovali v reálnom čase a tým pádom by nespomaľovali beh programu.

Asi najznámejším algoritmom na detekciu objektov, rozšírený na detekciu tváre a očí je Viola–Jones algoritmus navrhnutý v roku 2001. Využíva kaskádu klasifikátorov s Haarovým prvkami. Metóda používa posuvné okno, ktoré sa posúva cez snímku a odhaľuje detegované objekty. Pre urýchlenie algoritmu sa v algoritme nachádzajú výpočty za pomoci integrálne obrázku, metódy učenia založenej na AdaBoost a tiež štruktúra kaskád, ktorá filtruje od negatívnych pod-okien.

Ďalším zaujímavým riešením problému detekcie a sledovania objektov je TLD tracker. Ide o doposiaľ neznámy algoritmus, ktorý je schopný sledovať objekt na po sebe idúcich snímkach. Jeho princíp je založený na správnom spojením troch prvkov:

- tracking – sledovanie,
- learning – učenie,
- detection – detegovanie.

Sledovanie objektu sa deje na základe KLT trackeru. Fáza učenia je spojená s využitím P-expertov a N-expertov, ktorí dokážu rozhodovať o prítomnosti alebo neprítomnosti objektu na snímke. Detektor využíva Object model, ktorý reprezentuje množinu doposiaľ naučených pozitívnych a negatívnych vzoriek.

Medzi pomerne úspešné detekcie tváre a charakteristických prvkov tváre patrí aj Active Shape Model. Algoritmus je schopný s celkom veľkosť presnosťou určiť polohu prvkov tváre na základe deformačnej mapy tváre.

Z oblasti počítačového videnia poznáme aj algoritmy poskytujúce lokalizáciu zreničky pomocou projekčných funkcií alebo na základe analýzy hrán.

4 Detekcia žmurknutia

Medzi najspoľahlivejšie metódy odhalenia únavy alebo mikrospánku patrí detekcia žmurknutia a jej následná analýza. Podľa dostupných výskumov vieme, že informácie ohľadom frekvencie, dĺžky a spôsobov žmurkania nám pomáhajú určiť stupeň únavy.

Prvou z analyzovaných metód je Korelačná metóda. Pomocou rozdielu po sebe idúcich snímok sa nájde presná poloha očí. Potom ako máme lokalizované obe oči, algoritmus sa naučí vzorku otvoreného a zatvoreného oka. Následne sa v reálnom čase porovnávajú získané vzorky s pôvodnými vzorkami a určuje sa miera korelácie. Vhodnými prahmi dokážeme oddeliť získané vzorky na otvorené a uzavreté oči. Takéto určenie v čase vytvára korelačnú krivku, ktorá slúži na určenie frekvencie a dĺžky žmurkania.

Ďalšou metódou, ktorá slúži na odhalenie žmurknutia, je metóda množstva pixelov v oblasti očí. Región očí má pri oboch stavoch oka (otvorené alebo uzavreté) rozdielny prejav. V práci rozoberajúcej tento prístup autori využívajú rôzne filtre na vhodné spracovanie regiónu očí – mediánový filter, invertovanie, iluminačná kompenzácia a binarizácia. Výstupom je binárny

obrázok oblasti očí, ktorým vieme vyrátať počet pixelov s najnižšou intenzitou. Pomocou SVM klasifikátora vieme v reálnom čase rozhodovať o stave oka.

Detekcia žmurknutia je možná aj využitím optického toku. Monitorovanie obyčajnou webovou kamerou vracia snímky, nad ktorými je vyhľadaná tvár človeka. Následne sa určí žmurknutie pomocou týchto krokov:

- výpočet optického toku nad tvárou,
- normalizácia optického toku,
- rotácia toku vektorov,
- odhad hlavného smeru toku,
- extrakcia žmurknutia použitím adaptívneho prahu.

Pomocou týchto krokov dokážeme merať frekvenciu žmurkania, priemerná dĺžka žmurknutia, priemerná dĺžka uzatvoreného oka.

Vertikálna projekcia intenzity pixelov je posledná z analyzovaných metód. Projekcia intenzity v každom riadku je vlastne suma intenzít pixelov. Spojenie hodnôt projekcie pre všetky riadky vytvára projekčnú funkciu. Na základe geometrických charakteristík tejto krivky dokážeme vypočítať pomer hovoriaci o otvorenosti oka. Problémom tejto metódy môže byť rôznorodosť typov očí ľudskej populácie.

5 Navrhované detekčné algoritmy

Detekcia žmurknutia je náročný problém. V našej práci uvažujeme dva hlavné typy detekcie žmurknutia – statická detekcia snímkov a detekcia pomocou sekvenčnej analýzy.

5.1 Statická detekcia žmurknutia

Prvým typom detekcie žmurknutia je statická detekcia, ktorá rozhoduje o stave oka (otvorené alebo uzavreté) pre každý snímok samostatne. O stave oka môžeme rozhodovať na základe vhodného deskriptora a SVM klasifikátora. Predtým ako klasifikátor vráti binárnu odpoveď na vzorku oka, musí byť natrénovaný na statických pozitívnych a negatívnych vzorkách. V tejto sekcii opisujeme niekoľko metód na konštrukciu deskriptora.

5.1.1 Vertikálna projekcia intenzity

Naša implementácia Vertikálnek projekcie intenzity (VPI) využíva fakt, že otvorené a zatvorené oko majú rozdielny prejav v rámci VPI. Vertikálna projekcia je priemerná intenzita pixelov v riadku vypočítaná ako ich suma vydelená počtom pixelov. VPI funkcia celej vzorky je považovaný za deskriptor. Oba druhy stavov oka majú rozdielne VPI funkcie – líšia sa v rozložení funkčných extrémov a ich intenzít. Taktiež je značné, že otvorené oko má väčší počet tmavých pixelov, viditeľnú zreničku (čo znižuje VPI hodnoty), menšiu a tmavšiu oblasť medzi zreničkou a obočím a menšie rozdiely vo vertikálnom priereze funkcie. Všetky tieto menované charakteristiky pridávajú k diskriminatívnosti klasifikátora, čo je veľmi potrebné pre správne rozlišovanie v SVM.

5.1.2 SIFT deskriptor

V našej práci sme implementovali aj známy SIFT deskriptor. Využívame SIFT, aby sme demonštrovali potenciál gradientov pre rozlišovanie stavov očí. SIFT deskriptor je vyrátaný nad jedným kľúčovým bodom umiestneným v strede obrázka, s nulovou orientáciou a s veľkosťou kľúčového bodu, ktorý pokrýva celú snímku. Veľkosť kľúčového bodu bola vybraná na základe testovania. Pred výpočtom SIFT deskriptora filtrujeme obrázok pomocou Gaussovho filtra. SIFT deskriptor je 128 prvkový vektor, ktorý je invariantný voči otočeniu.

5.1.3 Vlastné gradientové deskriptory

Inšpirovaný SIFT deskriptorom sme vyvinuli vlastné gradientové deskirptory, aby sme demonštrovali vhodnejšie prispôsobenie gradientov na problém opis stavu oka. Naša metóda ponúka kalkuláciu gradientov a triedenie orientácií.

Výpočet gradientov sa deje pre všetky snímky, ktoré majú oko zarovnané na stred snímky. Metóda využíva Sobelov operátor na filtrovanie hrán. Následne sú nad obrázkom vypočítané deriváty gradientov v horizontálnom aj vertikálnom smere. Výsledkom sú získané orientácie a sily gradientov.

Dôležitým bodom tohto algoritmu je konštruovanie váhovej mapy. Tá sa využíva na prispôsobenie hodnôt príznačnejších gradientov, ktoré sa nachádzajú v strede snímky – v strede oka. Mapa váh má najmenšie hodnoty na krajoch snímky a maximálnu hodnotu v strede snímky. Medzi hodnoty sú vyrátané na základe exponenciálnej funkcie. Tento prístup bol inšpirovaný FREAK deskriptorom.

Orientačná funkcia je poskladaná vytriedením orientácii gradientov do binov. Po triedení je funkcia vyhladená na štyri lokálne extrémy. Tieto extrémy sú použité na vytvorenie deskriptora, ktorý má osem prvkov. Štyri dvojice vyjadrujú vzdialenosti od susedných binov a pomer veľkosti extrému k ostatným extrémom.

Hodnotenie tejto metódy ukázalo, že nad dobre zarovnanými očami funguje metóda relatívne spoľahlivo, avšak nie až tak spoľahlivo ako SIFT. Testovanie prebehlo na našom vlastnom datasete s vlastnými snímkami očí, ktoré boli získané od 6 subjektov. Veľkosť datasetu je 200 pozitívnych a 200 negatívnych snímok.

5.2 Sekvenčná analýza

Druhý spôsob detekcie žmurknutia je založený na sekvenčnej analýze po sebe idúcich snímok. V našej práci využívame KLT Tracker, ktorý dokáže sledovať body roztrúsené v oblastiach očí. KLT Tracker používa charakteristiky intenzity a priestorového rozmiestnenia vstupných bodov na získanie výstupných bodov.

Naša metóda sleduje oči samostatne. Po inicializácii pomocou Viola–Jones algoritmu rozmiestnime v oblastiach očí body, ktoré budú sledované trackerom. Body sú rozmiestnené podľa pravidelnej mriežky prispôsobujúc sa veľkosti okna. Trackovanie má za úlohu nájsť posunutie bodu, ktorý doň vstupuje. Trackovanie však môže poskytovať aj falošné posunutia, ktoré musíme ošetrovať ešte pred spracovaním týchto bodov. Body s prílišným posunutím alebo body vzdialené od stredu oblasti sú nazývaní outsajdermi.

Prínosom našej metódy je výpočet obvodného štvorca, ktorý uvažuje vzdialenosti bodov od stredu oka. Body sú vytriedené do binov na základe ich vzdialeností od stredu oblasti. Pomocou binov sa nájde medzera vzdialeností, ktorá určuje hranicu outsajderov.

Rozhodovanie o výskyte žmurknutia má na zodpovednosti bunková metóda. Štvorec oka je rozdelený na 9 rovnakých častí. Žmurknutie sa prejavuje vnútornými pohybmi, ktoré sú zložené z pohybu dole a následným pohybom hore. Správne určenie takýchto pohybov sa odhaduje pomocou testom získaného prahu a tiež je závislé od FPS, pretože vyššie FPS poskytuje viac snímkov, kde sa zmeny prejavujú v menšom.

Druhou navrhovanou metódou pre detegovanie žmurknutia je metóda zmeny pohybov. Pre každú snímku zatriedime bod do vhodného binu na základe veľkosti jeho vertikálnej zmeny a to pre kladný a záporný smer zvlášť. Podľa pozorovania je žmurknutie sprevádzané značne vyšším počtom nenulových binov ako iné zarovnané pohyby. Tento fakt využívame v stavovom automate, ktorý taktiež vyžaduje pohyb dole s následným pohybom hore.

Dôležitým prvkom algoritmu je reinicializácia. Reinicializovanie obnovuje pôvodné hodnoty a znovu rozmiestňuje body očí. Nastáva v týchto prípadoch:

- vysoký počet stratených bodov,
- nastalo žmurknutie,
- po istom počte snímok (uplynutý čas).

Bunková metóda dosahuje veľmi dobré výsledky na Talking databáze s iba dvomi nezachytenými žmurknutiami. Naša metóda dosiahla teda 96.7% úspešnosť, čo je lepšie ako v analyzovanej práci založenej na optickom toku.

6 Záver

Bakalárska práca sa zaoberá problémom mikrospánku za využitia detekcie žmurknutia. Prvou časťou práce je analýza dostupných riešení. Charakterizovali sme aj komerčné a nekomerčné riešenia. Nasledovne sme analyzovali detekčné systémy, ktoré používajú rôzne prvky počítačového videnia na detekciu žmurknutia. Detekcia tváre a očí bola spolu s ich sledovaním analyzovaná v ďalšej kapitole.

Našim cieľom bolo vybudovať aplikáciu, ktorá by dokázala monitorovať vodiča a predpovedala by prichádzajúci mikrospánok. Neboli sme schopný vyvinúť takúto aplikáciu založenú na analýze frekvencie a dĺžky žmurknutia, avšak zamerali sme sa na skonštruovanie spoľahlivého detektor žmurknutia. Práve on je hlavným prvkom úspešnej detekcie mikrospánku. Návrh a testovanie vlastných metód bolo navrhnuté v druhej časti bakalárskej práce.

Ponúkli sme vlastné riešenie za pomoci deskriptora gradientov, ktorý využíva príznačné gradienty očí. Spojeneie deskriptora a SVM dosahovalo porovnateľné výsledky ako SIFT deskriptor. Ďalej sme implementovali a testovali Vertikálnu Projekciu Intenzity.

Ďalšou skupinou navrhnutých algoritmov na detekciu žmurknutia boli takzvané sekvenčné analýzy po sebe idúcich snímok. Vyvinuli sme a testovali detektor žmurknutia, ktorý dosiahol lepšie výsledky ako riešenie navrhované v práci Divjaka a Bischofa. Naše riešenie považujeme za dostatočne perspektívne na využitie v aplikáciách pre detekciu mikrospánku alebo iných aplikáciách pre žmurknutie.

Appendix F

DVD Contents

```
/datasets/      - all testing datasets
/install/       - libraries and include files
/misc/          - miscellaneous files
/papers/        - used state-of-the-art papers
/source/        - problem solutions (qt source files)
/thesis/        - pdf version of bachelor thesis
```