UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Anti-spoofing system for facial recognition

Author:
Arsen SENKIVSKYY

Supervisor:
Oles DOBOSEVYCH

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2019

Declaration of Authorship

I, Arsen SENKIVSKYY, declare that this thesis titled, "Anti-spoofing system for facial recognition " and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Anti-spoofing system for facial recognition

by Arsen SENKIVSKYY

Abstract

The biometric recognition systems had massive success in recent years. Since webcameras are incorporated in many different devices(cell phones, tablets, laptops, entrance doors in some facilities, etc.), facial recognition systems become highly popular. Hence, the more people use these systems, the more people try to trick them to get unauthorized access.

There are three types of attack on the facial recognition system: picture-based attack, when an attacker is presenting a picture of another user's face; Video-based attack where an attacker is showing a prerecorded video of another user; Mask-based attack when attacker uses a mask of authorized user in order to spoof the facial recognition system.

In this work, I tackle picture-based and video-based attacks. For this reason, I develop a challenge-response system. The idea an approach is to detect where a user can do what system has challenged him to do. This way, we know that the face that is presented to the camera is alive. The user is required to watch a moving dot on the screen. The dot starts from the center of the screen and goes to the randomly chosen side of the screen, so this way user cannot present a prerecorded video. As the user follows the dot, the system estimates the direction where the user's eyes are moving. For these purposes, I implemented three different approaches. The custom neural network that takes as an input projections of three consecutive frames of an eye movement and classifies which the direction of the movement. In the third approach, I hypothesized then when the user is watching at collinear points on a vertical line, the x coordinates of the user's pupil will be approximately the same, having small variance. The same applies to y coordinates on a horizontal line. Thus by analyzing the variance of the coordinates, we can detect whether an attacker is not presenting some else's picture. ...

Acknowledgements

I want to say huge thank you to Oles Dobosevych for the supervision of this thesis and for those four years of dedication of his on and off work time to help with whatever I needed. Also, I would like to thank Lyubomyr Senyuk for his ideas about the approaches proposed here...

Contents

List of Figures

List of Tables

List of Abbreviations

CNN	**C**onvolutional neural network
NN	**N**eural network
KP	**K**eypoint

Chapter 1

Introduction

Traditional access control systems use passwords to remember or any kind of a key that user is required to carry with himself. Lately, this type of identification was displaced by biometric recognition. Biometric recognition systems that utilize user's fingerprint, iris, voice, face, palm veins, etc. had massive success in recent decades.

Moreover, global biometric authentication and identification market projected to Grow Over 51.98 Billion by 2023 source. The biometric recognition systems have become a convenient and reliable way for user identification. Instead of traditional methods of user identification, they are more favorable because they cannot be forgotten or lost.

Nowadays they are so widely spread(from logging in to laptops, phones to paying your receipt by your face). With such popularity number of people trying to trick the system increase as well. When one user tries to present himself with a false identity and have the intention to get unapproved access is called a spoofing attack.

Many different ways of protecting a facial recognition system from spoofing attacks were presented. We can categorize them by user involvement into **active** and **passive**.

In **active systems**, the user must do things that the system asks to, so it can recognize a user's liveness. They require a user to directly engage with a system;

The **passive** approach does not require a user to engage with a systems sensor; it doesn't even need a user to understand the way system works or what it does. It captures and analyzes involuntary facial movements, like blinking, facial muscles movement, iris movement, the way light reflects on face surface, etc.

In this work, I implemented an **active** system, where a user is being asked to watch the dot as it is moving from the center of the screen to randomly chosen a side of a screen. The first two approaches to anti-spoofing the system analyze the direction of the user's eye movement, and if the direction coincides it to the direction of the dot, then the user is considered to be alive. The third one requires a user to look not just to one episode of the dot moving to the side of the screen, but to a couple of at least. Then the system analyzes the variance of the x and y coordinates of the user's pupil center when he looks at a collinear set of point. If it is below a certain threshold that a user is considered to be alive.

Chapter 2

Related work

2.1 Spoofing attacks approaches

The usual types of spoofing attack is depicted in Figure 2.1. The unidentified user is trying to present somebody's face either as printed picture, video recording or a 3D mask.

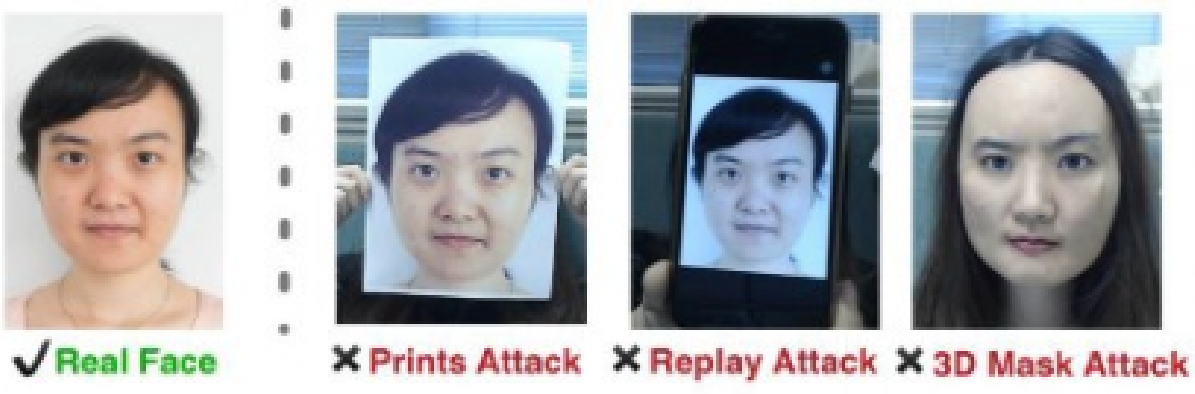


FIGURE 2.1: Types of spoofing attacks
Source article

- **Picture-based attacks.**
  - Present printed faces. Often attackers try to deform pictures so they can resemble the 3D shape of a real human face.
- **Video-based attacks**.
  - This type of attack has the same abilities as the previous type of attack. Besides, videos can give a piece of sequential information about changes in facial features and dynamic of environmental conditions to the sensor. Which gives an attacker more chances to gain unauthorized access.
- **Mask-based attacks.**
  - This type of attack helps to preserve 3D facial features and environmental conditions. Also, it gives an adversary an ability to move eyes, which can be helpful with bypassing challenge-response anti-spoofing systems.

2.2 Spoofing Detection Approaches

There are different ways of detecting spoofing attempts. Each of them utilizes the specific property of the person who is trying to get access. The are four main anti-spoofing approaches:

- **Texture analysis**
  - Extracts static features like blurriness, the difference in illumination, etc. Most systems of this type require just one picture.
- **Image quality analysis**
  - Assumes that when a user attempts spoofing attack(picture, video-based attacks), his face will be lower quality.
- **Face liveness detection**
  - An approach that uses sequential features. It analyzes how alive user is by analyzing involuntary facial movements, blinking, iris movements, etc. This approach also includes challenge-response techniques, where the system asks you to perform some action (following the dot with your eyes, turning head, smiling, etc.) and then analyzed whether user performed actions as was needed.
- **Hardware-based solutions**
  - Using Infrared camera, stereoscopic cameras or FaceID- like sensors. Cons of this are approach is that additional hardware can be expansive and system based on hardware cannot be used on users devices (laptops, phones, etc.) so it cannot be easily scaled and integrated.

We can also classify systems by **user involvement**. By these criteria, systems can be classified into **active** and **passive**.

The active approach requires a user to interact with the system's sensor(asking the user to perform an action like turning head, looking at stimulus, etc.).

The passive approaches to anti-spoofing are more convenient for the user since do not require users cooperation. They detect user's liveness(blinking, involuntary facial movements, the way the face reflects light, the texture of an image, the blurriness, etc.)

2.3 Existing solutions

As was described earlier, there are many different approaches to protect your facial recognition system from spoofing attacks. Multiple solutions were proposed in recent decades.

2.3.1 Static models

In the work of (Kim et al., 2012) the assumption that images of real faces and printed faces differ from one another by shape and quality. They used frequency and texture analyses by exploiting power spectrum and Local Binary Pattern(LBP). Then they apply fused frequency-based and texture-based classifiers to identify spoof pictures.

Unlike the previous method, this one utilizes a picture's chrominance component. (Dong, Tian, and Xu, 2017) introduced an approach that utilizes a color gradient of an image. This anti-spoofing system uses a Roberts cross operator that extracts a color gradient out of a live and spoof faces.

(Ali, Deravi, and Hoque, 2013) proposed a model that uses sharpness and blurriness. This method relies on the nature of digital focus where a 3D object will have regions that are closer to camera sharp and regions that are further from camera blurry. They used two face regions nose and cheek. Then they extracted blurriness level and the gradient magnitude with a threshold.

These approaches usually are easy to compute since they use only one image. Also, these types of systems do not require a user to engage with the sensor for a long period of time, which is really convenient. On the other hand, they don't work well when different illumination or surrounding conditions or image quality happen.

2.3.2 Dynamic models

The approach that (Jee, Jung, and Yoo, 2006), uses based on analyzing the nature of a live human behavior. In this approach, they extract coordinates of both eye pupils on users face. Then they calculate the variance of coordinates. If it surpasses a certain threshold, it is recognized as a real user, otherwise, a spoofing attack.

The other type of detecting a spoofing attack is challenge-response technique. This approach provides better robustness since the system is provided with dynamic features which are harder to forge in comparison to static ones that are extracted from photographs.

This type of systems was used by (Frischholz and Werner, 2003). The system asked users to look in randomly chosen directions, so that they needed to move their head. The model extracts two feature points(corners of the left eye) and knowing the 3D properties of humans face, the system estimates the user's head pose. If the user head position was right, the system verified the user as a live person.

The (Ali, Deravi, and Hoque, 2012) presented an approach where implemented a gaze tracking system where the users were presented with a stimulus and were required to look at it. Than system analyzed the frames where the stimulus was going through collinear points. The idea is that when the user looks at collinear

points, the coordinates on the x-axis of the eyes center should be close to each other. Then the variance of those coordinates is analyzed. Live user pupil's coordinates have small variance compared to fake ones.

Challenge-response techniques generally are more robust, they exploit the sequential nature of humans movements, so they are less sensitive to environmental changes and noise in comparison to static based models.

Chapter 3

Dataset collection

3.1 Process of collecting the dataset

The system setup for collecting the dataset was that one that is shown in Figure 3.1. The setup consists of a web camera, a PC, and a display monitor. I used the embedded camera from my laptop HP Probook 430(720p HD web camera). The camera is located on the top of the screen right in the middle. The screen is used the one that is in the laptop(13.3" LCD screen), a commonly used laptop monitor type, with a resolution of 19201080 pixels and response time is 5ms. The distance from the camera was 50 cm. The allowed head rotation angle was 10 degrees. The resolution of an image is 600 by 600 pixels.

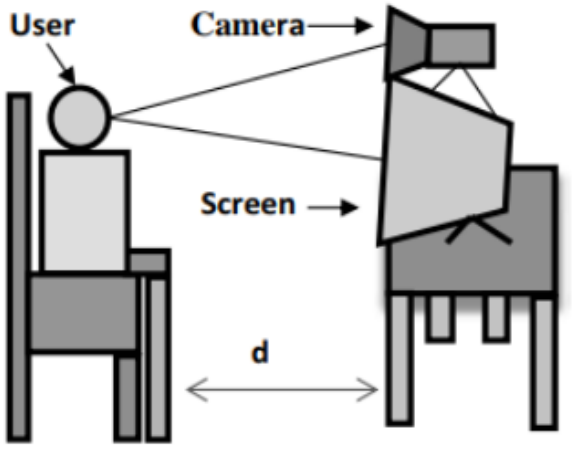
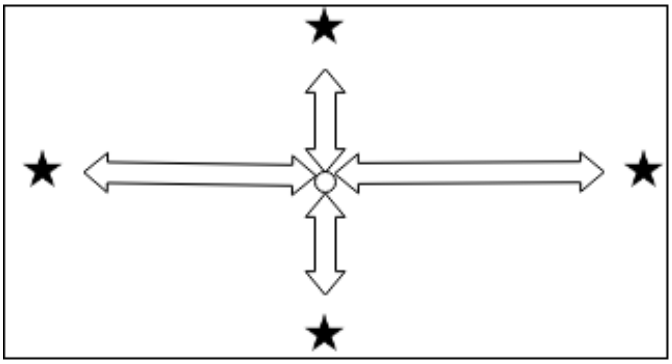


FIGURE 3.1: System setup

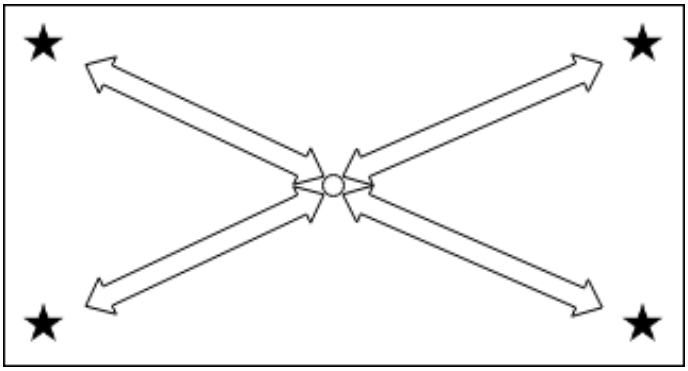
3.2 Dataset description

Initially, the idea was to track people's eye movements to key points depicted in Figure 3.2(A). The dot is moving from center of the screen 3.2(A)to the key point and backward. The eye movements from twenty-four participants were collected. Each of them was presented with a visual stimulus(a dot on the screen). They were asked to follow the dot with their eyes. The one episode(a process of moving dot from one key point to another) lasted one second. Twenty four frames are collected in one session resulting in two hundred frames per person(movement towards and away from the four KPs). Testing different approaches on this data, I did not manage to get satisfactory results. After that, I decided to try different KPs. I collected nine

people's eye movements to the KPs depicted in Figure 3.2(B). The hypothesis was that when a dot moves to the corner of the screen, the distance it goes is longer than to the side of the screen, so that the eye movement will be more distinct and easier to classify. The first dataset consist of twenty-four people that look at KPs 3.2(A), the second dataset consist of nine people



(A) Point's direction dataset 1



(B) Point's direction dataset 2

FIGURE 3.2: Moving direction comparison

3.3 Visual stimulus

A dot appears in the center of the screen Figure 3.5). The process of dot's moving to the right KP for the first dataset type 3.2(A) is depicted in Figure 3.5(B). When system is testes every new session it randomly chooses the direction the dot will go, so an attacker cannot present a prerecorded video. The user is required to watch the dot and not to get distracted. While dot is moving the camera is recording users face. The dot takes two seconds to go from the center of the screen to the any KP, this applies fro both dataset types 3.2(A), the same applies to the second type of key points setting 3.2(B). The example of eye dynamic is depicted in Figure 3.3. The numbers indicate the time stamps of the point position 3.5(B). That is eye movement when they look at the point which moves from the center of the screen to the right side. What I discovered that some people do not have some people don't have a distinct eye movement, though they we looking exactly at the point. That happens because of the distance to the camera. From the 50 cm to screen you don't have to move eyes as much as from even slightly less distance(40-45cm).The lack of movement is show in Figure 3.4.

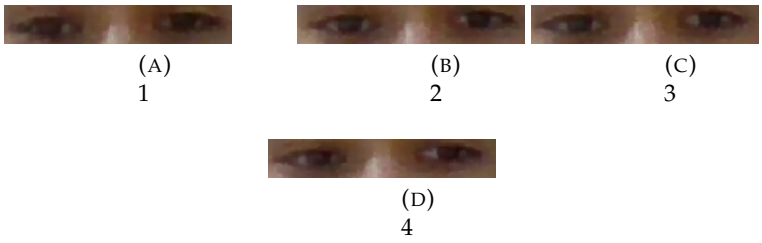


FIGURE 3.3: Eye movement dynamic

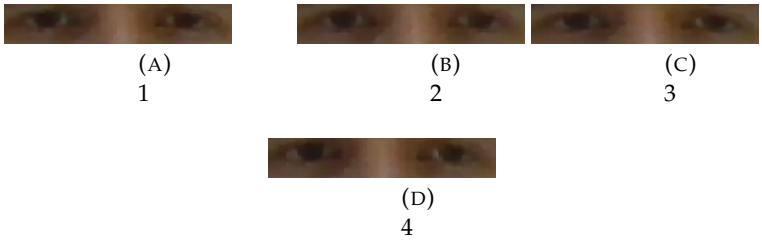
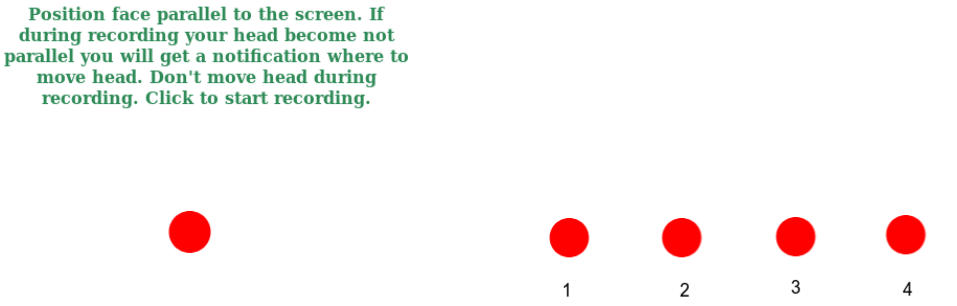


FIGURE 3.4: Small dynamic of eye movement



(A) Initial state of the stimulus

(B) Process of dot's moving

FIGURE 3.5: Stimulus

Chapter 4

Approaches

The proposed system design is depicted in Figure 4.1. The system does not require any additional hardware except the traditional web camera. A dot initially is located in the middle of the screen. The system randomly chooses the direction the dot will go, so an attacker cannot present a prerecorded video. When it starts moving web camera starts recording the process. The eye region is extracted from the picture. The direction of eye movement is classified.

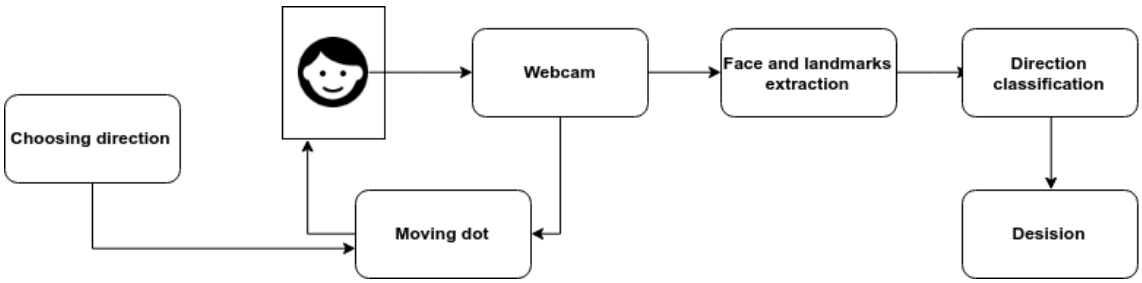


FIGURE 4.1: Model pipeline

4.1 Eyes projection neural network

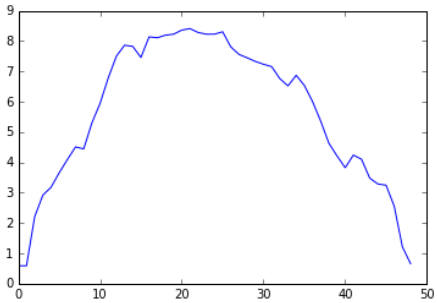
4.1.1 Overview

In this approach, I take three consecutive frames, with each of them, I do the following: I extracted face out of images by using the classic Histogram of Oriented Gradients (HOG) feature combined with a linear classifier, an image pyramid, and sliding window detection scheme from DLib library, after that 68 facial landmark coordinates are extracted by (Kazemi and Sullivan, 2014) algorithm. Using eyes landmarks, eye region is extracted out of the picture. Then eye's projection vectors onto both axes are calculated. By doing that i end up with 6 projections(two projections per frame). Finally, I use custom CNN for direction estimation.

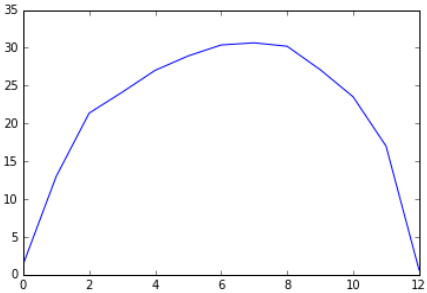
4.1.2 Projection calculation

The hypothesis is that grayscale image of an eye will consist of lighter shades of gray which is an eyeball and darker shades of gray which is an iris. So the location of an iris will be the location of maximum value of the projection function. The formula for projections calculating for each axis is the following: $f(x) = \sum(1 - x/255)$ where x is vector of points of one entry in axis. I calculated projections of a grayscale eye image 4.2(A) onto the axis x 4.2(B) and y 4.2(C). The eye size is (49x13) pixels.

(A) Raw eye



(B) Projection on x-axis



(C) Projection on y-axis

FIGURE 4.2: Eye processing

4.1.3 Network architecture

I wanted to utilize the sequential nature of human's eye movement, so as an input a give a projections on both axes of three consecutive eye frames. The architecture visualization is depicted in Figure 4.3. This architecture perfectly fits my needs, because I have two projections and I wish that network learn both projections separately. This way I want the network to learn how projection changes when the eye is moving in a certain direction. I give three consecutive pictures as an input it translates into six(2 projections per 3 frames) vectors.

4.2 Optic flow based approach

4.2.1 Overview

In previous approach I was presenting the network a set of pictures and training it to predict movement direction. In this approach, I want to calculate a direction vector of eyes on each frame by myself.

4.2.2 Architecture

Instead of the algorithm for facial landmarks extraction, I used in the previous approach, here I use STASM algorithm (Milborrow and Nicolls, 2014). I do this because, in addition to landmarks that the previous algorithm provided, this one also extracts centers of pupils location. An example is shown in Figure 4.5. Out of all landmarks I use only pupils location. The system pipeline is depicted in Figure 4.4.

4.2.3 Directional vector calculation

The optic flow of the moving eye is calculated as follows: for one episode(dot's movement away/towards the center of the screen to/from a keypoint) I have 24 frames. First of all, I extract the landmarks of right eye area 4.5 which include eyelid key points and center of the pupil. Then landmarks coordinates are normalized concerning minimum x and minimum y coordinated of eye area landmarks, this

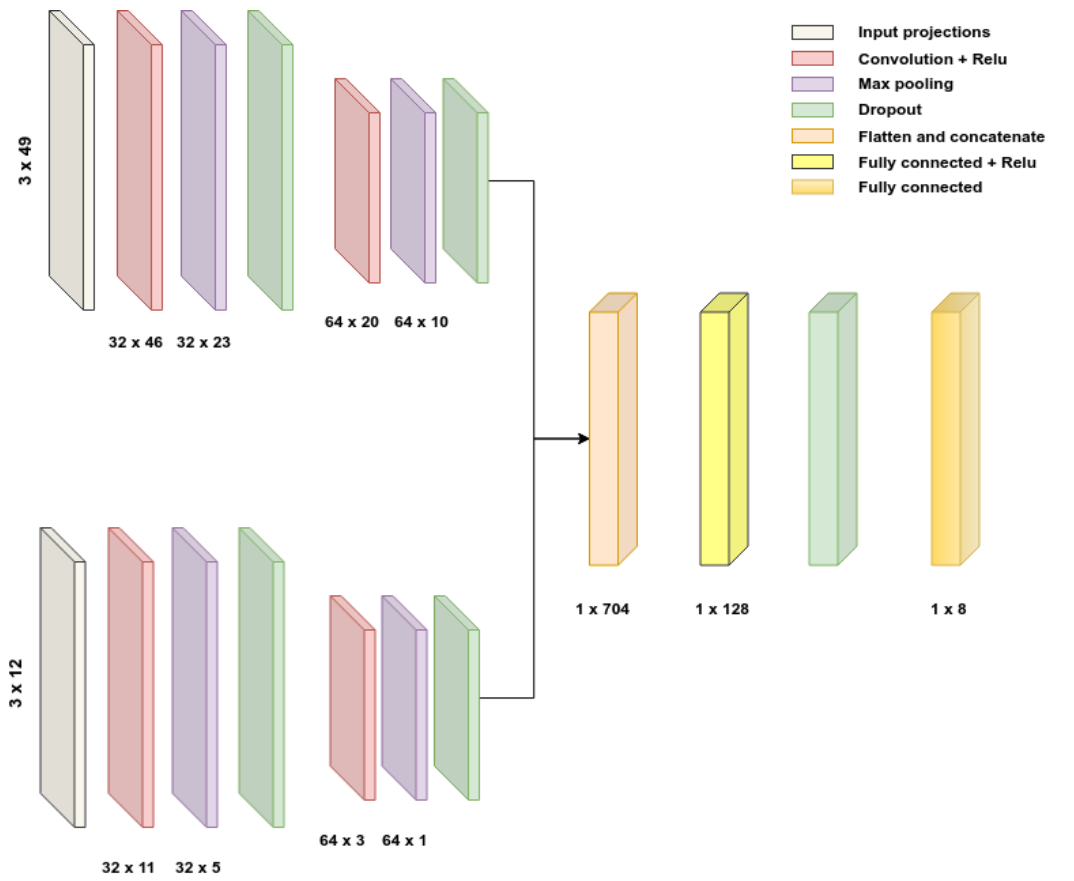


FIGURE 4.3: Neural network architecture for eight class classification

way, the changing location of the face in the picture will not affect the direction vector calculations. Having normalized landmarks, I extract one coordinate - the pupil center. I take the mean of X and Y coordinates from the first **five frames**, this way I minimize the STASM algorithm (Milborrow and Nicolls, 2014) error for pupil center localization, on top of that I empirically concluded that first five frames do not contain much pupil movement, so this is a good estimation of initial pupil's coordinate. Having the base coordinate of the pupil, I take pupil's coordinates from **last three** frames of the session and calculate the directional vector of them. Then I take mean of those three directional vectors to have one that represents the movement direction of the episode. After that, I calculate the angle I compute whether the angle is corresponding to the desired direction.

4.2.4 Limitations

In the previous approach, I used the neural network to classify eight directions of eye movement(towards and away to/from four key points 3.2). In this approach, I can only classify four(away from the center of the scree), because this approach is not accurate by itself. Since the eye area is small and the pupil is taking most of it, the coordinate of the center of an eye barely passes one-two pixels in the best case scenario. On top of that the STASM algorithm (Milborrow and Nicolls, 2014) produces a small error localizing the pupil's center on each frame, so even when the eye is still the system detects a movement and wrongly calculates the vector. That's why I will not be able to correctly distinguish the eye movement from the

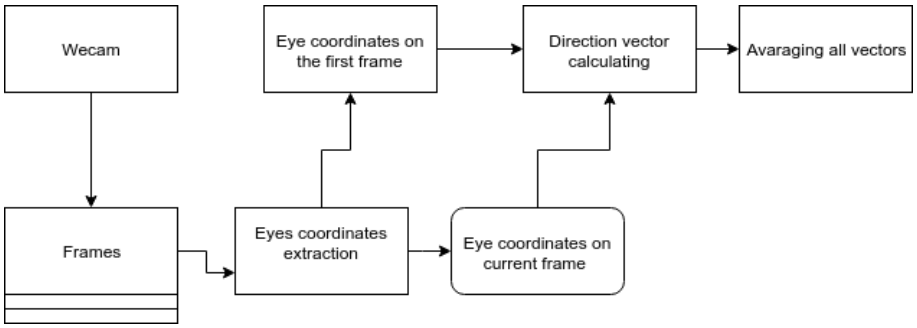


FIGURE 4.4: Optic flow model pipeline

right keypoint to the center from the eye movement from the center to the left side keypoint since the direction of movement is the same(this applies to all collinear key points).

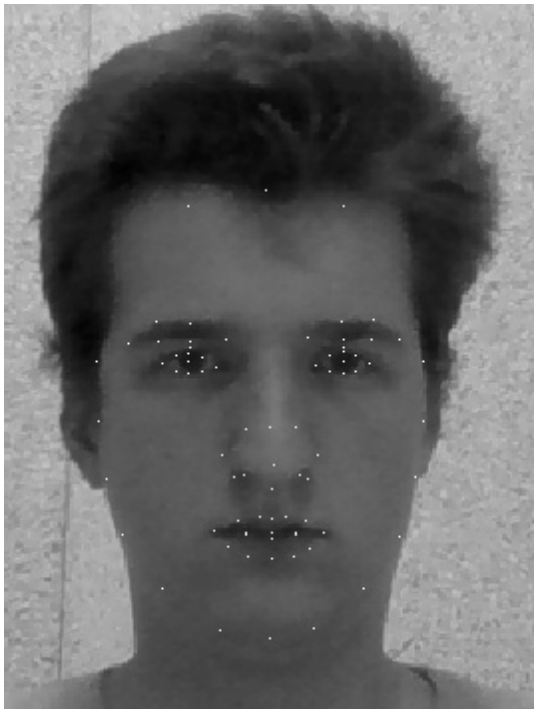


FIGURE 4.5: STASM algorithm landmark extraction

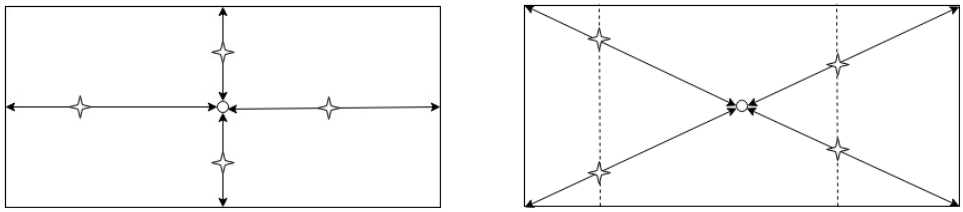
4.3 Variance-based algorithm

4.3.1 Overview

I also implement the approach proposed by (Ali, Deravi, and Hoque, 2012). The hypothesis utilized in this approach is that when the user is watching at collinear points on a vertical line, the **X coordinates** of the user's pupil will be approximately the same, having small variance. The same applies to **Y coordinates** on a horizontal line.

4.3.2 Description

Having twenty-four frames per one episode and eight moving directions per one user I decided to to use all four directions for vertical collinear coordinates variance extraction and all four directions for horizontal coordinates extraction. The key points used in first dataset are shown in Figure 4.6(A), the second dataset example is shown in Figure 4.6(B). The start represents the pair of frames that are being compared to one another on vertical and horizontal lines. In my approach I compare all twenty-four frames of each movement with one another. The landmarks of an eye center I extract by STASM algorithm (Milborrow and Nicolls, 2014). The way I normalize landmarks is exactly the same as in previous approach.



(A) First dataset collinear points

(B) Second dataset collinear points

FIGURE 4.6: Collinear key points for both datasets example

Since there are many sets of collinear points I take mean of these variances to describe one user.

4.3.3 Classification

The classification rule for live user classification for **first dataset** is $var(x) < 100\ AND\ var(y) < 60$. For the **second one** the classification threshold is $var(x) < 140\ AND\ var(y) < 40$
Also i did not perform any outlier filter criteria because it did not give any performance boost.

4.3.4 Limitations

Fake users may still have low variance of eye coordinates, that is because some attackers when doing this challenge-response technique may decide not to move picture in needed direction, as much as other people who perform the spoofing attack.

4.4 Challenges of gaze tracking

It is not an easy task to do the model I intended to because of several of reasons.

4.4.1 Unconscious eyes movement

Since by design of the system is supposed to track changes pupil direction changes in the response to the stimulus. And when user unconsciously distracts, the direction vector changes. Live users that looking the dot may seem to be classified as a fake one. A lot of the time the user doesn't even notice the eye movements, it may even seem to the user that he is doing everything right. This research (Galdi et al., 2016) shows that a user's gaze can unconsciously exhibit rapid changes, that results in unexpected eye movements.

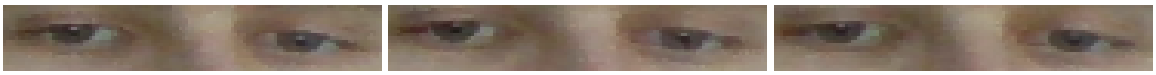
4.4.2 Blinks

On top of that, **blinking eyes** would also introduce noises in the observed iris trajectory. Every time user blinks, one's eye slowly closes, the system recognizes it as if a user if looking down and then back up.

4.4.3 Distance from the screen

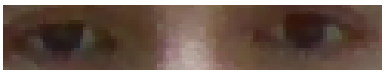
The motivation of this work was to implement the anti-spoofing system that can track eye movements of the user. This system could further be applied in combination with facial recognition system for logging into to your computers, be installed on the facility doors to let only authorized people in, etc. I used the minimum the preferred viewing distance from a monitor - 50cm (source). As show in Figure 3.4 from the distance of 50cm eye movement is minimal. Moreover, it's even worse when the users look at the horizontal key points (top or bottom key point)3.2(A). The eye movement dynamic to bottom key point is shown in Figure 4.7

(A) 1



(B) 2 (C) 3 (D) 4

FIGURE 4.7: Eye movement to the bottom key point



(A) 1

(B) 2 (C) 3 (D) 4

FIGURE 4.8: Eye movement to the left key point

4.4.4 System setup

The system **system setup** for a video recording can be a problem as well. Some cameras produce horizontally flipped images while others don't. Thus, the results of calculating the directional vector of eye movement are going to be different depending on the fact whether pictures are flipped or not. You need to make sure the hardware and software on which the system is trained will behave exactly the same as the one you will be exploiting the system on.

4.5 Other approaches

In the course of working on this project I tried many different iris tracking approaches.

4.5.1 Summing opposite vectors

The initial hypothesis when I started doing this project was that when a dot is moving from the center of the screen to a key point and back it goes through the same path, so eyes movement should follow the same trajectory. Hence calculated directional vector of eye movement from user's eye that were following the dot that is moving from the center to a key point should be opposite to the vector calculated from user's eye that were following the dot that is moving from a key point to the center. So when we add them together they are supposed to compensate each other so the magnitude of a resulting vector should be approximately zero. But on practice this rule does not hold. The error of algorithm that extracts coordinates of iris's center combined with blinks and involuntary eye movements do not let this approach to provide satisfactory results.

4.5.2 Raw optic flow calculation

Also I tried to calculate optical flow without coordinates of eye centers of provided by SMASM. I used Lucas–Kanade method (Lucas and Kanade, 1981) to do an optical flow estimation in eye region. But I didn't manage to extract good enough features to track.

4.5.3 Different image processing techniques

I also tried to do different image processing techniques. I was trying to extract an iris by applying threshold to grey scale image. While on some pictures it did a great job, on most of them it worked unreliably. Also I used Canny edge detector (Canny, 1987) but with my dataset I didn't manage to get good results either. Finally, I applied Hough transformation (Duda and Hart, 1972) to eyes region in order to extract iris. But with image quality that I had and the fact that most people that participated in collecting the dataset did not have their eyes open enough for the algorithm to detect a circle consistently.

Chapter 5

Results

5.1 Dataset used

Having **twenty-four** users in my **first dataset**, I trained the projection based neural network 4.3 on **fifteen** users, validated on **four** and tested on **five** users. Having **nine** users in **second dataset**, I trained the model on **five** users, validated on **two** users and and tested on **two** users.

All results that are presented are being shown on test part of each dataset.

5.2 Projection based neural network

5.2.1 Overview

This network take as an input two projections from each of three consecutive frames of eye movement. The numbers of frames per episode is twenty-four, so I for each episode I take all three consecutive frames sets with stride one, after that I pick the class that occurs the most frequently out of all predictions in episode.

Initially, I assumed that four class classification will be more accurate so I trained neural networks to classify only direction towards the key points. After that I trained the model for eight class classification. The comparison of the performance of all four networks in one episode direction classifying is depicted in Table 5.1. The four class classification predicts direction of an eye only towards 4 key points, the eight class classification - towards and backwards. As we can see in Table 5.2 the model provides a good protection against spoofing attacks. The low score means that user was supposed to look in certain direction but looked in other(the lower, the better).

You may notice that the performance on train and test set are different, on the test set models tend to perform better. That is because with botch dataset I shuffled all users and assigned them to train-validation-test. So users who don't tend to move their eyes as much as others were not in test dataset because they we randomly chosen to be in either train or validation. And because one user is giving either four of eight examples(either 4 classification model or 8 classification model) the overall accuracy differs, because one user has indistinct eye movement patters and he produces 4 or 8 examples of an eye movement to dataset.

	Four class classification		Eight class classification	
	Train	Test	Train	Test
Dataset type 1	0.76	0.94	0.59	0.7
Dataset type 2	0.3	0.34	0.24	0.27

TABLE 5.1: F1 score comparison real people dataset

	Four class classification	**Eight class classification**
Dataset type 1 spoofing	0.14	0.05
Dataset type 2 spoofing	0.18	0.13

TABLE 5.2: F1 score comparison spoofing dataset

	F1 score
Dataset type 1	1
Dataset type 2	1
Dataset type 1 spoofing	0.89
Dataset type 2 spoofing	1
Combined type 1 and spoofing	0.9
Combined type 2 and spoofing	1

TABLE 5.3: F1 score comparison variance-based algorithm

In Figure 5.1 you can see the confusion matrix of the network classifying the direction of the movement. As you can see in figure the four-class model classifies moving direction pretty accurately, though not perfectly, but as you can see in Table 5.1 it doesn't affect the overall accuracy of movement direction classification of the episode because most moving examples(three consecutive frames) out of one episode(user watching the dot moving towards/away a keypoint) are classified correctly.

5.3 Variance-based algorithm

5.3.1 Overview

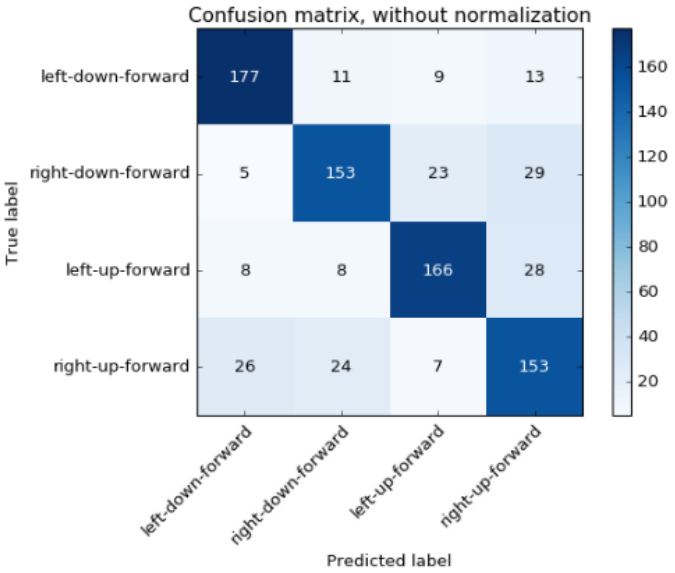
This approach also showed really good results. The main downside is that this approach requires all users episodes, so this system cannot use just one episode. This can be inconvenient, since it takes more user's time to go through all the eight moving directions. The F1 score comparison is shown in Table 5.3. In spoofing first dataset there was one spoofing attack where variance was really low. That's the main downside of this approach, sometimes attacker can choose not to follow the challenge(try to mimic watching the dot by deforming the picture). The feature distribution of type 1 dataset is depicted in Figure 5.2, he feature distribution of type 2 dataset is depicted in Figure 5.3

5.4 Optic flow based approach

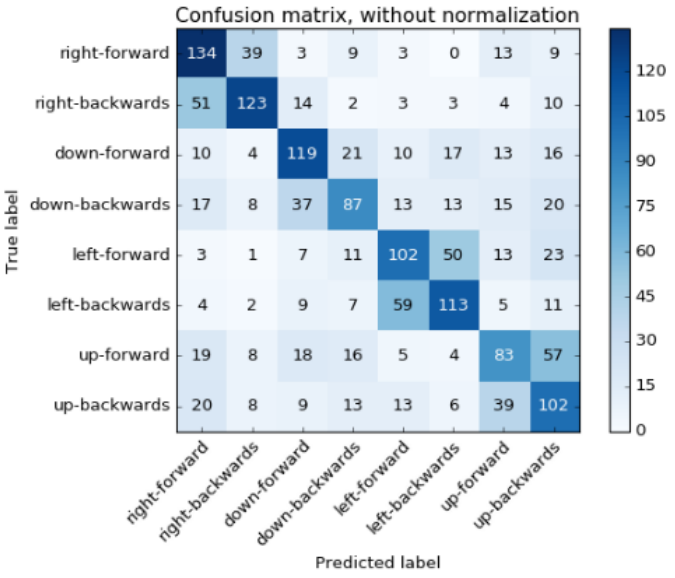
This approach showed the worst results. The main struggles with this approach is that when user is looking either up or down the coordinate of pupil's center doesn't change, that's why the results are pretty poor. I also tried to use eyelid landmarks to better estimate the direction of the vector but it didn't show any boost in performance. The F1 score comparison of the optic flow algorithm is depicted in Table 5.4. As you can see in Figure 5.4(A) the right-forward move and left-forward were the most accurately predicted, that's because the eye movement is the most distinct on horizontal trajectory. I think more landmarks utilization is needed in order to predict up and bottom movement. I tried to use eyelid landmarks but it did not give the desired accuracy.

	F1 score
Dataset type 1	0.3
Dataset type 2	0.3
Dataset type 1 spoofing	0.11
Dataset type 2 spoofing	0.22

TABLE 5.4: F1 score comparison optic flow algorithm

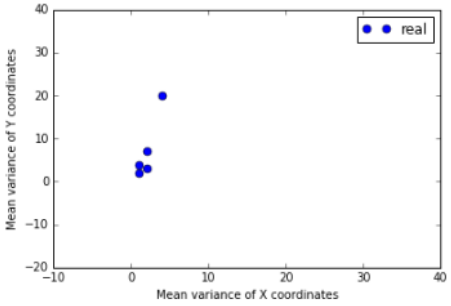


(A) Four class classification

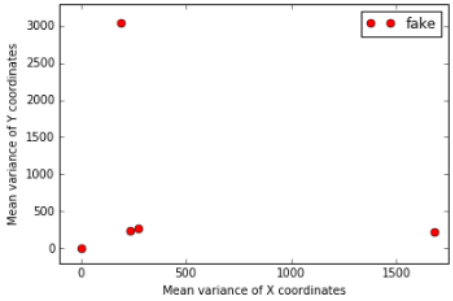


(B) Eight class classification

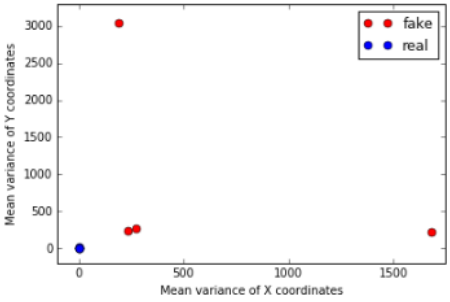
FIGURE 5.1: Confusion matrix for movement direction classification on dataset type 1



(A) Real users feature distribution

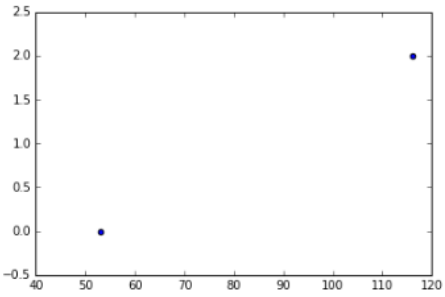


(B) Fake users feature distribution

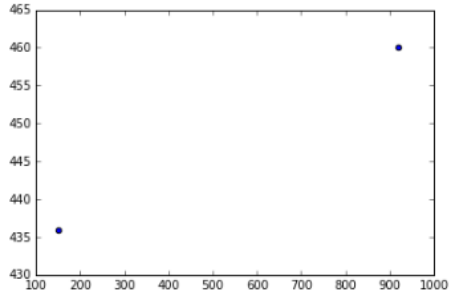


(C) Both real and fake users feature distribution

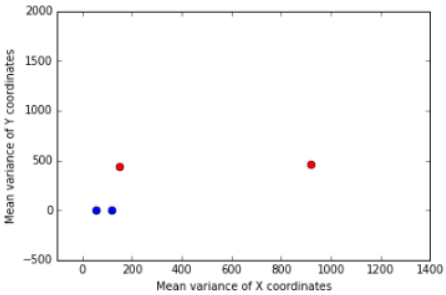
FIGURE 5.2: Feature distribution dataset type 1



(A) Real users feature distribution

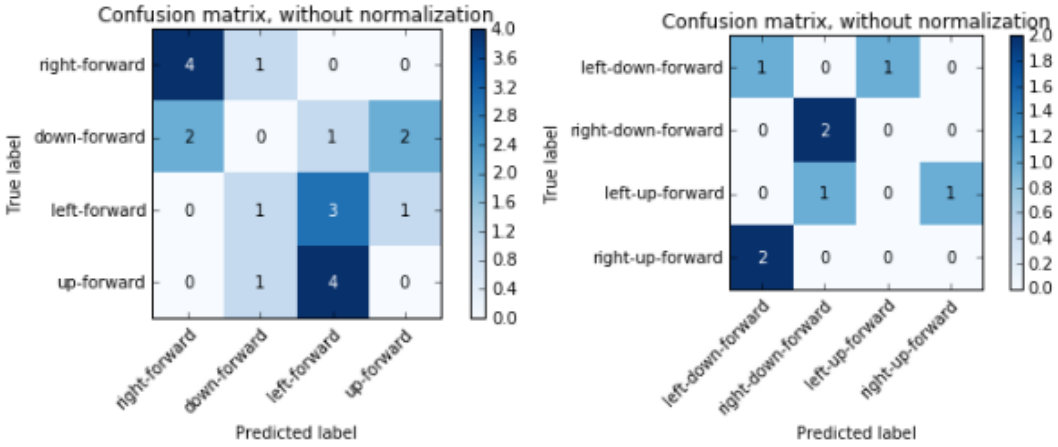


(B) Fake users feature distribution



(C) Both real and fake users feature distribution

FIGURE 5.3: Feature distribution dataset type 2



(A) Confusion matrix dataset type 1

(B) Confusion matrix dataset type 2

FIGURE 5.4: Confusion matrices comparison optic flow algorithm

Chapter 6

Summary

The neural network approach is the most promising. It requires only twenty-five frames to estimate the direction of user's eye movement, it is accurate and with more data collected I think the eight class classification will be more accurate which will give even more robustness to the system. The second dataset-based models did not show much performance. I think more data and research is needed to utilize the full potential of type 2 key points. They have the longer distance so eye movement should be more distinct.

Bibliography

Ali, Asad, Farzin Deravi, and Sanaul Hoque (2012). "Liveness detection using gaze collinearity". In: *2012 Third International Conference on Emerging Security Technologies*. IEEE, pp. 62–65.

— (2013). "Spoofing Attempt Detection using Gaze Colocation". In: *2013 BIOSIG - Proceedings of the 12th International Conference of Biometrics Special Interest Group, Darmstadt, Germany, September 4-6, 2013*, pp. 135–146. URL: https://dl.gi.de/20.500.12116/17663.

Canny, John (1987). "A computational approach to edge detection". In: *Readings in computer vision*. Elsevier, pp. 184–203.

Dong, Jixiang, Chunwei Tian, and Yong Xu (2017). "Face liveness detection using color gradient features". In: *International Conference on Security, Pattern Analysis, and Cybernetics, SPAC 2017, Shenzhen, China, December 15-17, 2017*, pp. 377–382. DOI: 10.1109/SPAC.2017.8304308. URL: https://doi.org/10.1109/SPAC.2017.8304308.

Duda, Richard O. and Peter E. Hart (1972). "Use of the Hough Transformation to Detect Lines and Curves in Pictures". In: *Commun. ACM* 15.1, pp. 11–15. ISSN: 0001-0782. DOI: 10.1145/361237.361242. URL: http://doi.acm.org/10.1145/361237.361242.

Frischholz, Robert W and Alexander Werner (2003). "Avoiding replay-attacks in a face recognition system using head-pose estimation". In: *Analysis and Modeling of Faces and Gestures, 2003. AMFG 2003. IEEE International Workshop on*. IEEE, pp. 234–235.

Galdi, Chiara et al. (2016). "Eye movement analysis for human authentication: a critical survey". In: *Pattern Recognition Letters* 84, pp. 272–283.

Jee, Hyung-Keun, Sung-Uk Jung, and Jang-Hee Yoo (2006). "Liveness detection for embedded face recognition system". In: *International Journal of Biological and Medical Sciences* 1.4, pp. 235–238.

Kazemi, Vahid and Josephine Sullivan (2014). "One millisecond face alignment with an ensemble of regression trees". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1867–1874.

Kim, Gahyun et al. (2012). *Face liveness detection based on texture and frequency analyses*. DOI: 10.1109/ICB.2012.6199760.

Lucas, Bruce D, Takeo Kanade, et al. (1981). "An iterative image registration technique with an application to stereo vision." In: *IJCAI*. Vol. 81, pp. 674–679.

Milborrow, S. and F. Nicolls (2014). "Active Shape Models with SIFT Descriptors and MARS". In: *VISAPP*.