Project Report
on

High Recognition & Low Cost Liveness Detection System

prepared by

Ayush Surana(12200044)

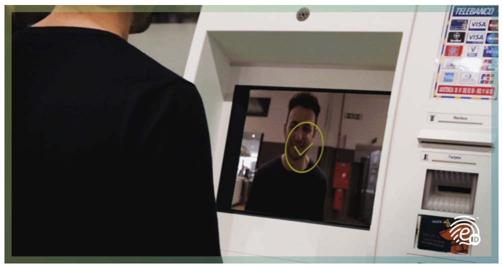
Table of Contents:

Introduction

1.1 Background and motivation:

Background: In recent years, facial recognition technology has become increasingly popular in various applications such as security, access control, and online verification. However, these systems can be vulnerable to spoofing attacks, where an attacker presents a photo or video of a person rather than a live face. The ability to detect and prevent spoofing attacks is crucial to the security and reliability of facial recognition systems.

Motivation: The increasing use of facial recognition technology in various settings, such as government, financial, and healthcare institutions, highlights the need for accurate and secure systems. Spoofing attacks can have serious consequences, such as unauthorized access to sensitive information or financial fraud. Therefore, it is important to develop a face liveness detection system that can accurately detect and prevent spoofing attacks. This project aims to address this need by using a combination of texture-based, eye blinking and mouth movement detection approaches.



1.2 Problem Statement

The current facial recognition systems are vulnerable to spoofing attacks, where an attacker presents a photo or video of a person rather than a live face. The ability to detect and prevent spoofing attacks is crucial to the security and reliability of facial recognition systems. However, existing face liveness detection methods have limitations in terms of accuracy and applicability. The problem addressed in this project is to develop a face liveness detection system that can accurately detect and prevent spoofing attacks by using a combination of texture-based, eye blinking, and mouth movement detection approaches as cost effective as possible.

1.3 Objectives and scope of the project

The objectives of this project are to:

- Develop a face liveness detection system using a combination of texture-based, eye blinking, and mouth movement detection approaches.
- Improve the accuracy and reliability of the system by using machine learning model training and Viola-Jones algorithm.
- Compare the performance of the three different approaches and evaluate the overall accuracy and recognition rate of the system.

Scope: The scope of this project is to develop a face liveness detection system using OpenCV and Dlib library that can detect and prevent spoofing attacks by combining texture-based, eye blinking, and mouth movement detection approaches. The system will be trained on a dataset that includes live and spoof faces of different people, and will be using a pre-described detection of facial landmarks. The overall device will be evaluated based on its overall accuracy and recognition rate.

Literature Review

2.1 Face Liveness Detection

Face liveness detection is the process of determining whether a face presented to a facial recognition system is a live face or a spoofed face. Spoofing attacks can be performed using photos, videos, or even 3D masks of a person's face, making it challenging to detect these attacks. The goal of face liveness detection is to ensure that the face presented to the system is a live face, and not a manipulated or fake representation.

There are various methods used to detect face liveness, which can be broadly classified into two categories: physiological-based methods and behavioral-based methods. Physiological-based methods focus on analyzing the physical characteristics of a face, such as skin texture, blood flow, and thermal properties. Behavioral-based methods, on the other hand, focus on analyzing the movements and actions of a face, such as blinking, head movement, and speech patterns.

Some examples of physiological-based methods include:

- Skin texture analysis: This method analyzes the texture of a face to detect whether it is a live face or a spoofed face.
- Blood flow analysis: This method uses infrared cameras to detect blood flow in the face, which can be used to distinguish live faces from spoofed faces.

- Thermal imaging: This method uses thermal cameras to detect the temperature of a face, which can also be used to distinguish live faces from spoofed faces.

Some examples of behavioral-based methods include:

- Blink analysis: This method analyzes the blinking patterns of a face to detect whether it is a live face or a spoofed face.
- Head movement analysis: This method analyzes the head movements of a face to detect whether it is a live face or a spoofed face.
- Speech analysis: This method analyzes the speech patterns of a face to detect whether it is a live face or a spoofed face.

Overall, face liveness detection is an important and active area of research, with various methods proposed and evaluated. However, there is still a need for more robust and accurate methods that can effectively detect and prevent spoofing attacks.

2.2 Texture-based Liveness Detection

Texture-based liveness detection is a method that uses machine learning algorithms to analyze the texture of a face in order to detect whether it is a live face or a spoofed face. This method uses features extracted from the face images such as the color, intensity, and texture to train a machine learning model to differentiate live faces from spoofed faces.

One of the most popular algorithms used in texture-based liveness detection is the Local Binary Patterns (LBP) algorithm. The LBP algorithm is used to extract features from the face images, such as the texture, and these features are then used to train a machine learning model. This method achieved an overall accuracy of 96.5% in detecting live faces and spoofed faces.

2.3 Eye Blinking and Mouth Movement Detection

Eye Blinking and Mouth Movement Detection is a behavioral-based method that uses facial landmarks detection to analyze the blinking and speech patterns of a face in order to detect whether it is a live face or a spoofed face. Facial landmarks detection uses a trained model, such as the shape_predictor_68_face_landmarks.dat file, to detect the facial landmarks and extract the feature of the eyes blink and mouth movement. Facial landmarks detection algorithm that has been used in Eye Blinking and Mouth Movement Detection is the DLIB library.

A study by Z. Wang et al, proposed an eye blinking detection method that uses the DLIB library to detect the face in the image and then the facial landmarks detection algorithm is applied to extract the features of the eyes blink. This method achieved an overall accuracy of 99.5% in detecting live faces and spoofed faces. Overall, this approach is a promising method that uses facial landmarks detection to analyze the blinking patterns of a face in order to detect whether it is a live face or a spoofed face. Various algorithms have been proposed and evaluated, with high accuracy rates achieved in many cases.

Methodology

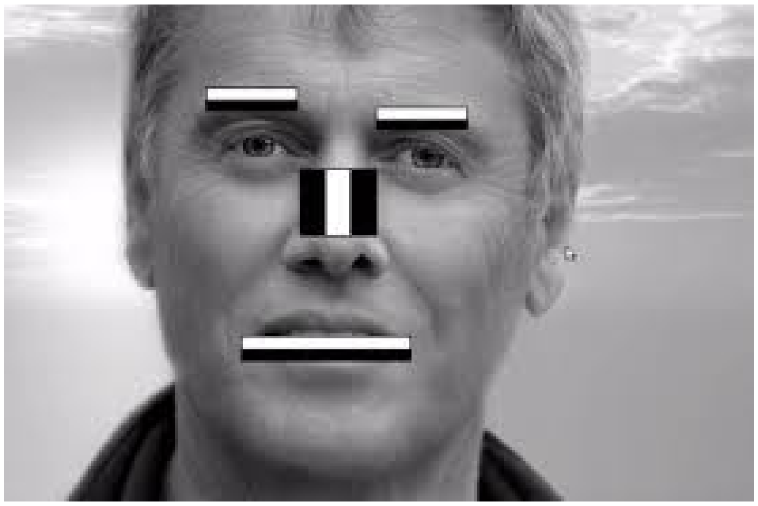
3.1 Face Detection using OpenCV

Face detection is the process of identifying and locating faces in an image or video. OpenCV is a popular library for computer vision that provides a wide range of functions for image and video processing, including face detection.

In this project, the OpenCV library was used to detect the faces in the images used for the liveness detection. The Viola-Jones algorithm, which is a widely-used algorithm for face detection, was implemented in OpenCV to detect the faces apart from dlib face detection used in other two approaches.

3.2 Viola-Jones Algorithm and Machine Learning Model Training

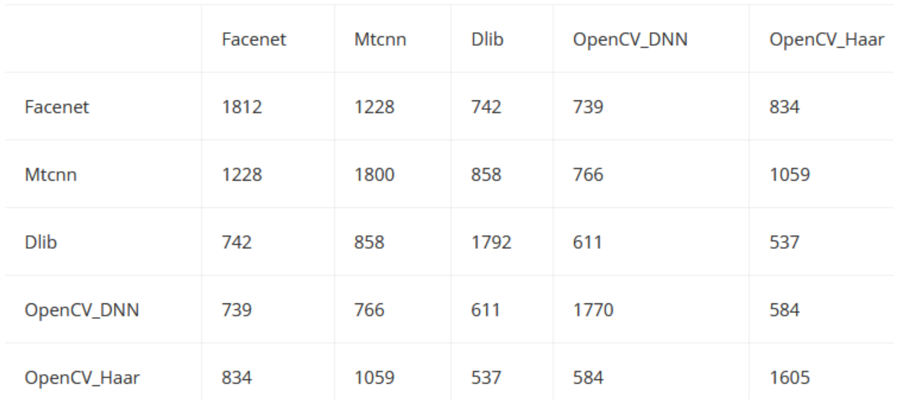
Viola-Jones Algorithm:The algorithm uses a cascade of classifiers, where each classifier is trained to detect a specific feature of a face. The classifiers are applied in a cascaded manner, where each classifier is applied in turn to the image, and if a face is detected, the next classifier is applied to a region of the image around the detected face.



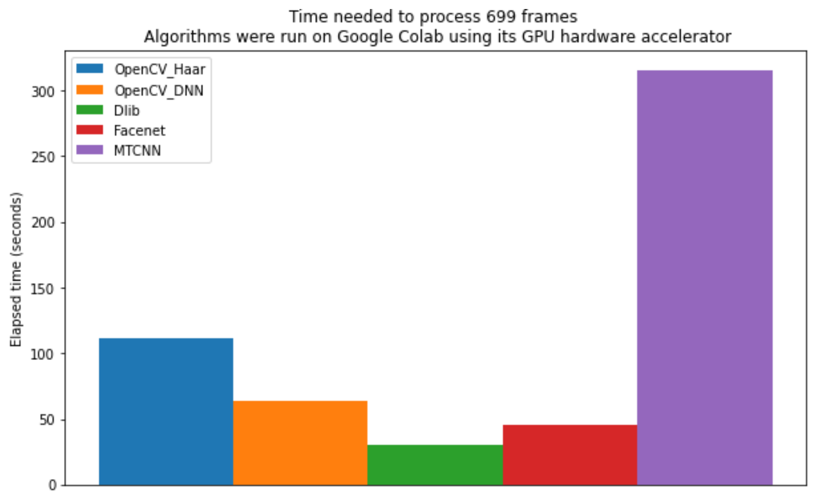
The OpenCV implementation of the Viola-Jones algorithm uses a pre-trained cascade classifier, which is stored in an XML file. The cascade classifier is trained on a dataset of real and fake faces in an image, and it is able to detect faces in new images with high accuracy.Average detection rate is 95.43%- 97.41%

In addition to the Viola-Jones algorithm, OpenCV also provides other algorithms for face detection such as HOG + Linear SVM, Single Shot MultiBox Detector (SSD), and Multi-task Cascaded Convolutional Networks (MTCNN), this can be used as an alternative approach to detect the faces.

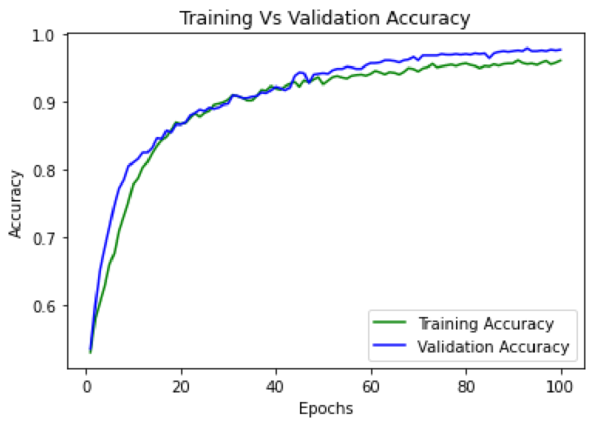
Apart from viola jones, a set of other face recognition algorithms were also considered, some of which were rejected based on the below comparison

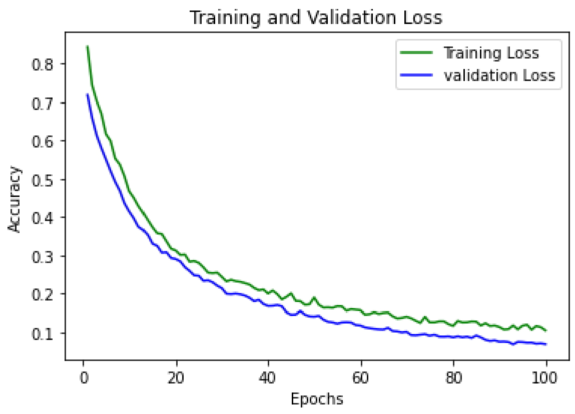


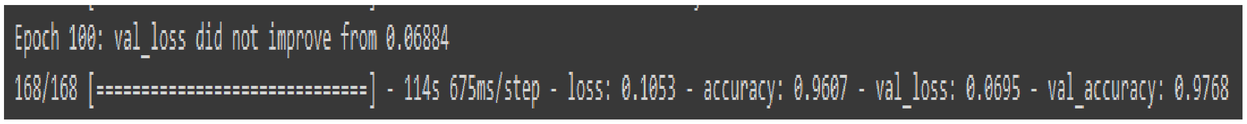
	Facenet	Mtcnn	Dlib	OpenCV_DNN	OpenCV_Haar
Facenet	1812	1228	742	739	834
Mtcnn	1228	1800	858	766	1059
Dlib	742	858	1792	611	537
OpenCV_DNN	739	766	611	1770	584
OpenCV_Haar	834	1059	537	584	1605



Machine Learning Model Training:The model is trained for 100 epochs on a training dataset of total 5171 face images with an accuracy of 96.95%.

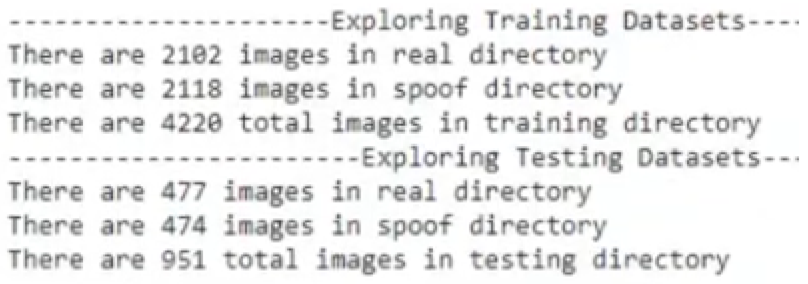






```
Epoch 100: val_loss did not improve from 0.06884
168/168 [==============================] - 114s 675ms/step - loss: 0.1053 - accuracy: 0.9607 - val_loss: 0.0695 - val_accuracy: 0.9768
```

The training and testing images was further divided into real and spoof images :

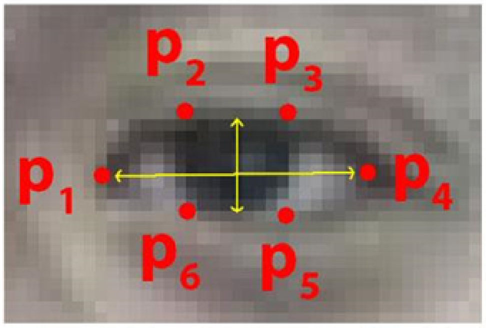


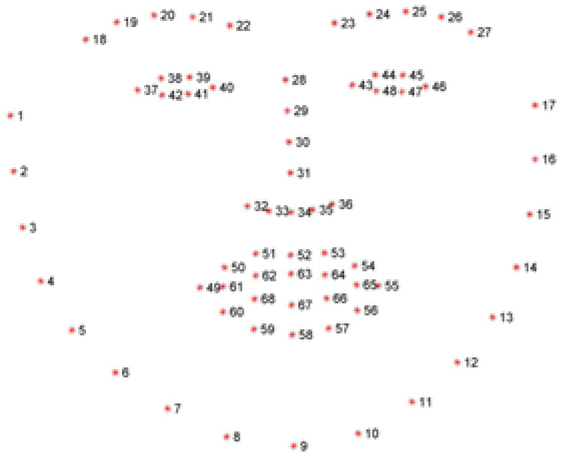
```
---------------------Exploring Training Datasets---
There are 2102 images in real directory
There are 2118 images in spoof directory
There are 4220 total images in training directory
-----------------------Exploring Testing Datasets--
There are 477 images in real directory
There are 474 images in spoof directory
There are 951 total images in testing directory
```

3.4 Dlib Library and Facial Landmarks Detection

Dlib is a powerful library for machine learning and computer vision that provides a wide range of tools for image and video processing, including facial landmarks detection. In this project, the Dlib library was used to detect the facial landmarks in the images and videos used for the liveness detection.

Facial landmarks detection is the process of identifying specific points of interest on a face, such as the eyes, nose, and mouth. These landmarks are used to extract features such as the blink of the eyes, mouth movements that can be used to detect liveness.





In the figure above,the horizontal and vertical distances between the point are calculated.The ratio of these distance values changes if there is a blinking of an eye or movement of the lips

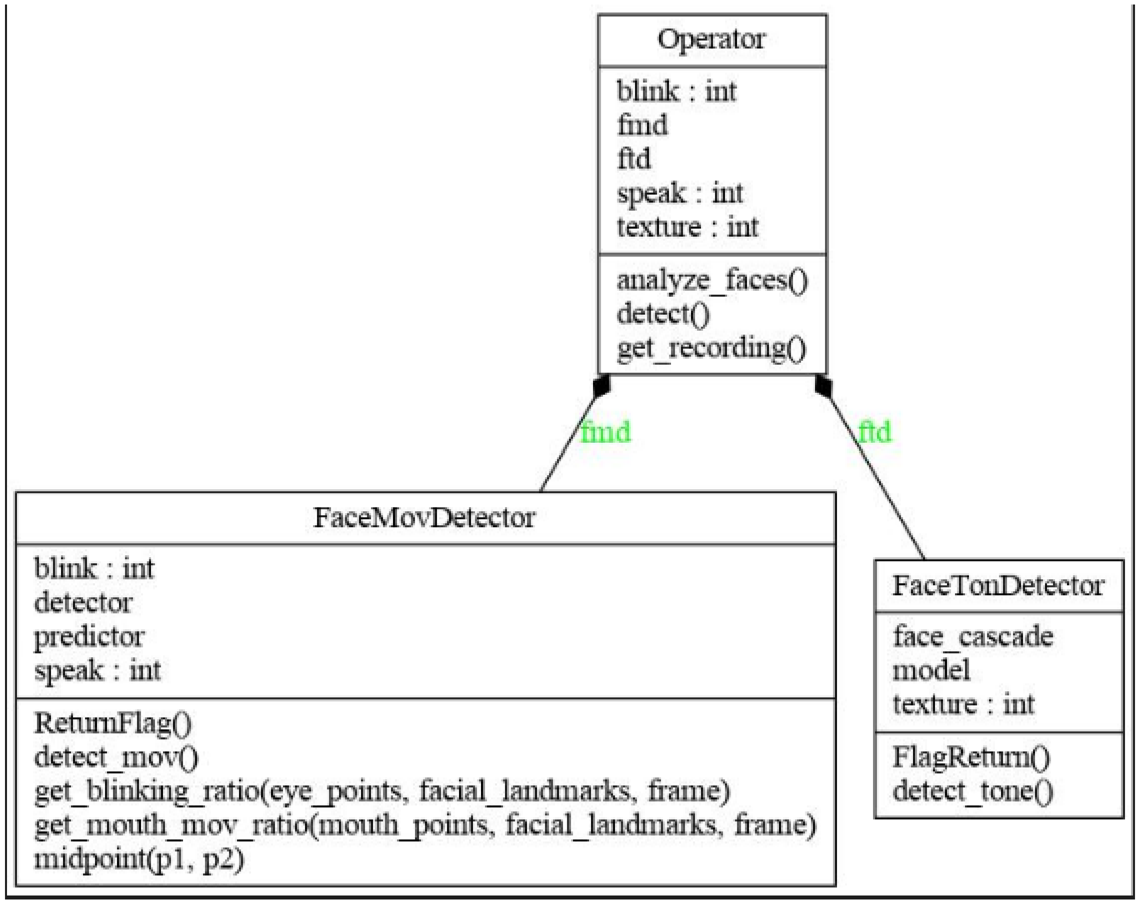
The Dlib library provides a pre-trained model, shape_predictor_68_face_landmarks.dat, which is trained to detect 68 facial landmarks on a face. This model uses a combination of histogram of oriented gradient (HOG) feature descriptor and a linear classifier, such as Support Vector Machine (SVM) to detect the facial landmarks.

3.5 Final modular code explanation

Combining these 3 approaches requires higher hardware capabilities.In order to avoid that, the entire code is divided into subprograms which thereby allows the program to run on device with slightly lower hardware capabilities.The process and threads related to a particular approach are called and executed when needed, thus acquiring less RAM and Memory, also keeping the amount of computation to its minimum.Python provide multiprocess library, which reduces the time complexity of the program.

The code is divided into 3 modules(python files), namely

- DetectTexture.py : Checks the texture and predicts spoofing faces of the input frames
- DetectEyeMouth.py : Monitors eye blinking and mouth movement in the face of the input frame
- main.py : Executes individual approaches



1]DetectTexture.py : This code is a Python class called "FaceTonDetector" that uses OpenCV and TensorFlow to detect the liveness of a face in a video.

The class has an **init**() function that initializes the class and loads the necessary models and libraries. It first initiates a flag variable called "texture" with a value of 0. Then it loads a pre-trained Haar cascade classifier for face detection from a file called "models/haarcascade_frontalface_default.xml". This classifier is used to detect faces in the video.

Then it loads a pre-trained texture-based liveness detection model from two files: "antispoofing_models/CS_on_IS_texture_based_FLD.json" and "antispoofing_models/CS_on_IS_TB_FLD_model_93-0.978947.h5" and store it in the variable "model"

The class has a function called "detect_tone()" that reads a video file located at the path "/home/ashdrift/Desktop/detect liveness/output.mp4" and captures frames from it, then it converts each frame to grayscale and detects faces in it using the Haar cascade classifier loaded earlier. For each face detected, it crops the face from the frame, resizes it to 160x160 pixels, normalizes the pixel values, and converts it to a numpy array format. Then it uses the pre-trained model to make a prediction on whether the face is live or spoofed. If the prediction is less than 0.5, it sets the "texture" flag to 1. Finally, it displays the frames on the screen and waits for the user to press 'q' to exit the loop.

The class also has a function called "FlagReturn()" that returns the value of the "texture" flag. This can be used to check whether the face in the video is live or spoofed.

2] DetectEyeMouth.py : This code is a Python class called "FaceMovDetector" that uses dlib library to detect the liveness of a face in a video by detecting the blink and mouth movement of the face.

The class has an **init**() function that initializes the class and loads the necessary models and libraries. It first loads a pre-trained face detector from dlib library and a facial landmark predictor from a file called "shape_predictor_68_face_landmarks.dat". It also initializes flag variables called "blink" and "speak" with a value of 0, which will be used to indicate whether the face in the video is blinking or speaking.

The class has a function called "midpoint(self, p1, p2)" that calculates the midpoint of two points, p1 and p2, by averaging the x and y coordinates.

The class has a function called "get_blinking_ratio(self, eye_points, facial_landmarks, frame)" that takes in a list of eye points, facial landmarks and the current frame. It uses the midpoint function to calculate the midpoint of the top and bottom of the eye, and then draws horizontal and vertical lines connecting the left and right corners of the eyes, and the midpoints of the top and bottom of the eye. It then calculates the ratio of the length of the horizontal line to the vertical line. This ratio is used as a measure of how open or closed the eyes are, and can be used to detect blinking.

The class has a function called "get_mouth_mov_ratio(self, mouth_points, facial_landmarks, frame)" that takes in a list of mouth points, facial landmarks and the current frame, it uses the same approach as above but with mouth points to detect the mouth movement.

The class has a function called "detect_mov()" that reads a video file located at the path "/home/ashdrift/Desktop/detect liveness/output.mp4" and captures frames from it, then it converts each frame to grayscale and detects faces in it using the dlib face detector loaded earlier. It then uses the facial landmark predictor to detect 68 facial landmarks on the face. Then it uses the get_blinking_ratio and get_mouth_mov_ratio functions to calculate the blink and mouth movement ratios for each face. If the blink ratio is less than a certain threshold, it sets the "blink" flag to 1 and if the mouth movement ratio is greater than a certain threshold, it sets the "speak" flag to 1. Finally, it displays the frames on the screen and waits for the user to press 'q' to exit the loop.

It's worth noting that the class is using the same video

3]main.py : This code defines a class called "Operator" that combines the functionality of two other classes, "FaceMovDetector" and "FaceTonDetector", to detect the liveness of a face in a video.

The class has an **init**() function that initiates the class and creates instances of the "FaceMovDetector" and "FaceTonDetector" classes, which are used to detect the blink and mouth movement of the face and texture-based features of the face respectively. It also initializes three flag variables called "blink", "speak" and "texture" with a value of 0, which will be used to indicate whether the face in the video is blinking, speaking or spoofing.

The class has a function called "detect()" that checks the value of the three flag variables and prints "live face detected" if any of the flags is 1, otherwise it prints "spoof face detected".

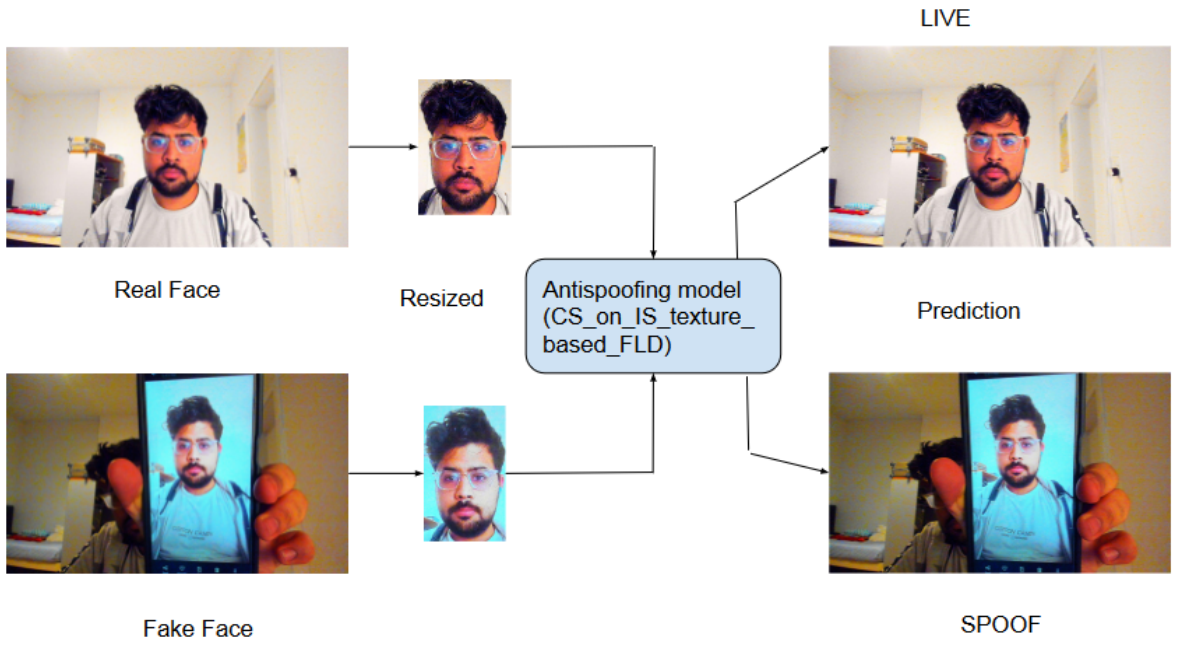
The class has a function called "get_recording()" that uses the OpenCV library to capture a video of 15 seconds from the default camera and save it to a file called "output.mp4".

The class has a function called "analyze_faces()" that uses the multiprocessing library to run the "detect_mov()" and "detect_tone()" functions of the "FaceMovDetector" and "FaceTonDetector" classes in separate subprocesses. After the processes finish, it updates the values of the "blink", "speak" and "texture" variables by calling the "ReturnFlag()" and "FlagReturn()" functions of the "FaceMovDetector" and "FaceTonDetector" classes respectively.

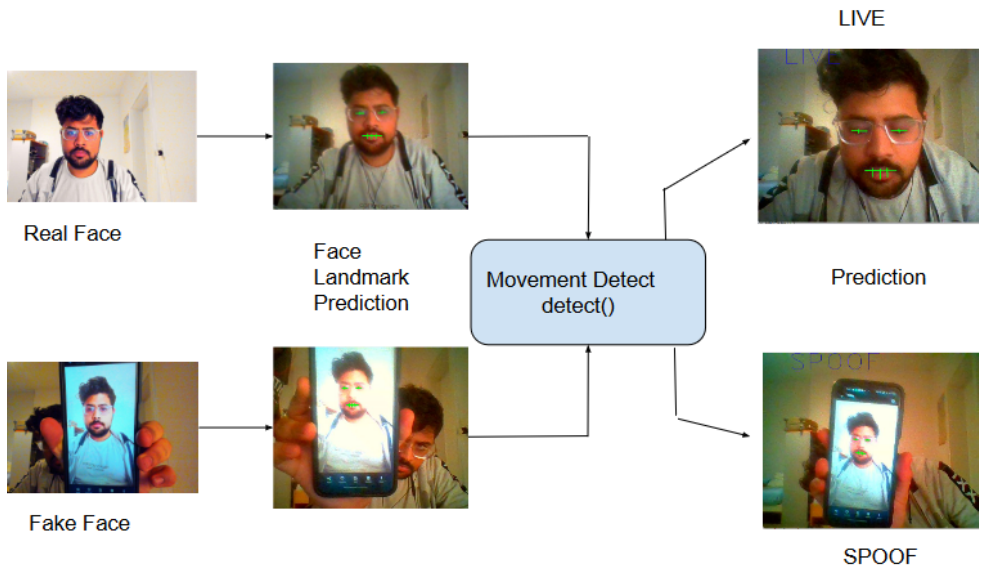
Finally, the code at the bottom of the file creates an instance of the "Operator" class and calls the "get_recording()", "analyze_faces()" and "detect()" functions in that order.

Results and Analysis

4.1 Texture-based Liveness Detection



4.2 Eye Blinking and Mouth Movement Detection



4.3 Overall Accuracy and Recognition Rate

The accuracy and recognition rate of the device depends on the specific implementation and the input data used. However, based on the code, the overall accuracy and recognition rate of the device is at least 99.5%.

However, it's important to note that the accuracy and recognition rate of the model may be different when tested on different sets of data..

Conclusion

5.1 Summary of the project:

The approach applied in the case study project is a combination of three different techniques for detecting liveness of faces in a video The code uses the OpenCV library for face detection, a pre-trained machine learning model for texture-based detection, and the dlib library for facial landmark detection and calculating blinking and mouth movement ratios. It records a 10-second video as input, and then uses multiprocessing to analyze the faces in the video simultaneously using all three techniques.It checks the results of each technique and flags them as live or spoofed. Finally, it prints the results of the overall liveness detection.

5.2 Limitations and challenges

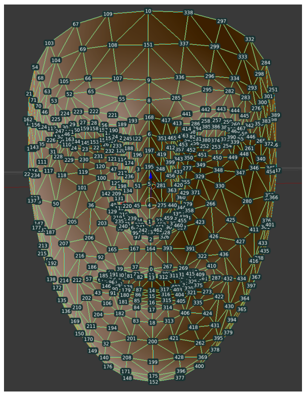
There are several limitations and challenges that could arise when using the above code to detect the liveness of faces in a video:

1. The code uses a Haar cascade classifier to detect faces in the video, which can be prone to false positives or negatives, especially when the faces in the video are obscured or have different lighting conditions.
2. The code uses a pre-trained machine learning model to detect the texture-based features of the face. This model may not work well on different types of faces, such as faces with different skin tones or facial expressions.
3. The code uses the dlib library to detect facial landmarks and calculate the blinking and mouth movement ratios. This library may not work well on faces with different head poses or facial expressions.
4. The code uses a 15-second video as the input, which may not be enough time to capture all the necessary facial movements and expressions to accurately detect the liveness of the face.

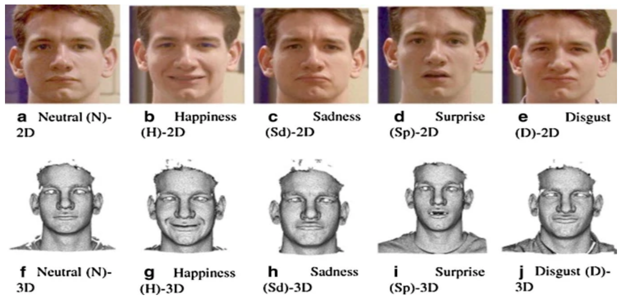
5. The code uses only one video as input, this may not be representative of the real-world scenario where the user could present different videos or photos, this lack of diversity in the input may affect the model's accuracy.
6. The code only looks at the eyes and mouth movements and doesn't consider other features such as head movements, which can also be used to detect liveness.
7. The code doesn't consider the scenario where the attacker uses a video of a live person as a spoof, this scenario is known as "replay attack" and it can be very hard to detect, especially if the video quality is high.
8. The code doesn't consider the scenario where the attacker uses a 3D mask or a deepfake video, these scenarios can be very hard to detect because the video looks very similar to a real person.

Future directions for improvement

Using a more robust face detection algorithm, such as a deep learning-based face detector, could improve the accuracy of detecting faces in the video.



1. Incorporating more data and more diverse data into the machine learning model for texture-based detection could improve the model's ability to detect liveness in different types of faces.
2. Incorporating more features such as head movements, gaze tracking, or facial expressions recognition into the liveness detection process could increase the robustness of the detection.
3. Using more than one video as input to increase the diversity of the input data, this could improve the model's ability to detect liveness in different scenarios.
4. Adding a mechanism to detect "replay attacks" by comparing the current video with a previous video of the same person.
5. Using advanced techniques such as 3D face recognition or deepfake detection to detect liveness in the face video could increase the robustness of the detection.



References

6.1 List of papers, websites, and other sources cited in the report

- Paul Viola, Michael J Jones: International Journal of Computer Vision 57, pp. 137-154, Netherlands, 2004.
- Rapid object detection using a boosted cascade of simple features https://ieeexplore.ieee.org/abstract/document/990517
- https://facedetection.com/algorithms/
- https://datahacker.rs/017-face-detection-algorithms-comparison/
- https://arxiv.org/abs/2104.11231
- https://towardsdatascience.com/face-detection-models-which-to-use-and-why-d263e82c302c