Eye Movement Pattern Based Authentication System [EPAS]

A PROJECT REPORT

Submitted by

Tushar Gupta (19BCY10019)
Blesson Biju Abraham (19BCY10079)
Subham Biswal (19BCY10136)
Chirag Ganguli (19BCY10158)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT BHOPAL UNIVERSITY

KOTHRI KALAN, SEHORE
MADHYA PRADESH - 466114

DECEMBER 2021

VIT BHOPAL UNIVERSITY, KOTHRI-KALAN, SEHORE MADHYA PRADESH – 466114

BONAFIDE CERTIFICATE

Certified that this project report titled **"Eye Movement Pattern Based Authentication System [EPAS]"** is the bonafide work of "**Tushar Gupta (19BCY10019), Blesson Biju Abraham (19BCY10079), Subham Biswal (19BCY10136), Chirag Ganguli (19BCY10158)"** who carried out the project work under my supervision. Certified further that to the best of my knowledge the work reported at this time does not form part of any other project/research work based on which a degree or award was conferred on an earlier occasion on this or any other candidate.

PROGRAM CHAIR
Dr. Rakesh R, Program Chair BCY
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

PROJECT GUIDE
Dr. Kannan Shanmugam
School of Computer Science and Engineering
VIT BHOPAL UNIVERSITY

The Project Exhibition II Examination is held on 09th December, 2021

ACKNOWLEDGEMENT

First and foremost I would like to thank the Lord Almighty for His presence and immense blessings throughout the project work.

I wish to express my heartfelt gratitude to Dr Shishir Kumar Shandilya, Head of the Department, School of Computer Science and Engineering (with spcl in Cyber Security & Digital Forensics) for much of his valuable support and encouragement in carrying out this work.

I would like to thank my internal guide Dr. Kannan Shanmugam, for continually guiding and actively participating in my project, giving valuable suggestions to complete the project work.

I would like to thank all the technical and teaching staff of the School of Aeronautical Science, who extended directly or indirectly all support.

Last, but not least, I am deeply indebted to my parents who have been the greatest support while I worked day and night for the project to make it a success.

ABSTRACT

This project is basically focused on a continuous authentication model where the model continually keeps recognizing Iris Movements and then records it in a temporary storage to authenticate a user using an advanced Biometric Authentication Technique.

After authenticating an user, the temporary storage blob is deleted and then the model is again ready to authenticate the next user.

TABLE OF CONTENTS

1. CHAPTER 1

1.1 INTRODUCTION

Authentication systems are the software's protective barrier. It ensures that the correct people enter the system and have access to the correct information.

Biometric authentication is a type of security that depends on an individual's unique biological traits to verify that they are who they say they are. Biometric authentication systems compare physical or behavioral characteristics to data in a database that has been verified and authenticated. Authentication is validated when both samples of biometric data match. Biometric authentication is commonly used to control access to physical and digital resources such as buildings, rooms, and computers.

1.2 MOTIVATION FOR THE WORK

Since privacy is a concern that must be addressed, particularly when it comes to digital data, it is critical to upgrade and improve existing authentication techniques, or to invent new methods. Considering this need a different strategy based on gaze movement was adopted, in which a gaze tracking authentication method would be employed to provide an additional layer of security to the conventional pin or password-based authentication model.

1.2 INTRODUCTION TO THE PROJECT

1.3

The geometric and motion characteristics of the eyes are unique which makes gaze estimation and tracking important for many applications such as human attention analysis, human emotional state analysis, interactive user interfaces, and human factors.

Iris Movement and gaze tracking have been an active research field in the past years as it adds convenience to a variety of applications. The goal of this project is to present a model on the existing literature on Iris Movement and Gaze Tracking and Develop an Efficient Technique that can revolutionize the field of

Computer Vision. A Motion analysis method is developed to track and detect Iris movement and gaze Tracking. They serve the purpose of being used as an authentication method in many login-based applications which require continuous authentication. The main purpose of the system is the motion analysis method and finding Horizontal Ratio to Find the Direction of Vision i.e., Left, Right or Centre.

1.4 PROBLEM STATEMENT

Pupil and Iris of Human Eyes are considered to be really strong biometrics that can be used for the unique authentication of an individual. In normal Biometric Authentication systems, there is a high probability of false matches which can be made resistant to in cases of Eye Movement Pattern Based Authentication systems. While validating a usual method of authentication, there is a form of validation needed which includes searching an entire database of known passwords which can be time-consuming and less effective. Normal authentication techniques can be altered by threat actors to gain unauthorized access to a secured system

1.5 OBJECTIVE OF THE WORK

Because the geometry and kinematic features of the eyes are unique, gaze estimation and tracking are critical for a variety of applications, including human attention monitoring, interactive user interfaces, and human factors. To track and detect Iris movement and gaze tracking, a motion analysis approach is created. They are used as an authentication technique in a variety of login-based applications that require continuous authentication.

1.6 SUMMARY

In this introduction part, we have talked about our problem, the objective of our work, and the techniques involved to detect pupils of a person in order to implement them in an authentication system.

2. CHAPTER 2 (Related work investigation)

2.1 INTRODUCTION

Optic movement and eye movements tracking have been a popular subject because they bring ease to a wide range of applications. Therefore, many previous works have been done in this particular field.

2.2 CORE AREA OF PROJECT

The core implementation of this project is in the field of authentication and security systems wherever continuous authentication is required.

2.3 EXISTING APPROACHES/ METHODS

As per our knowledge only on-paper and research implementation of these approaches are available. Any other similar type of approach found on the internet is just a part which has been already included in this project.

2.4 ISSUES/ OBSERVATIONS FROM INVESTIGATION

Continuous Authentication is still not implemented on a large scale and the Eye Movement Pattern Based Authentication is just a research model and has no implementation.

3. CHAPTER 3 (Requirement artifacts)

3.1 INTRODUCTION

Software and Hardware Requirements are the sole parts for the implementation of any project model. Here, for this project, we have tested it on several environments and have tried to make it as cross-platform as possible.

The bare minimum hardware and software requirements are specified below.

3.2 HARDWARE AND SOFTWARE REQUIREMENTS

a) Software Requirements:

- Operating system: Linux - Ubuntu 16.04 to 17.10, or Windows 7 to 10, with 2GB RAM (4GB preferable)
- OpenCV version 4.1.x (4.1.0 or 4.1.1 will both work just fine).
- Python version 3.6 (any Python version 3.x will be fine).

b) Hardware Requirements:

- 32- or a 64-bit computer.
- Minimum Processors: Intel Atom® processor or Intel® Core™ i3 processor
- Windows , macOS or Linux.

3.3 SUMMARY

This part explained all about the minimum hardware and software requirements required for running the project.

4. CHAPTER 4 (Design methodology and its novelty)

4.1 INTRODUCTION

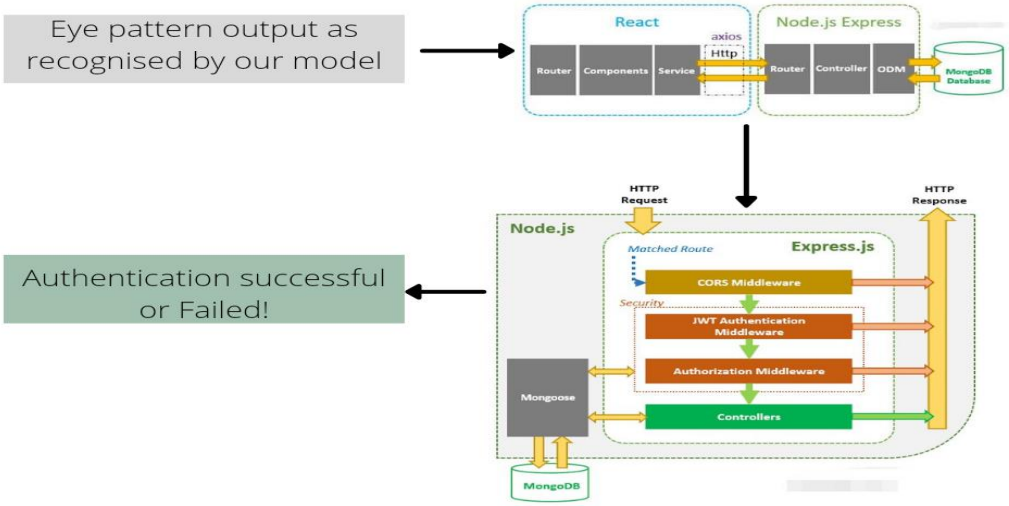
The design of this project is fairly simple. We are just working on the simple GUI provided by the opencv2 library. That is the user interface used in the project.

4.2 FUNCTIONAL MODULES DESIGN AND ANALYSIS

Since we rely on Machine Learning we have a number of modules communicating with each other to get the results. We have 2 main systems,

One is the ML model which detects the movement and generates the key and the other is the web application part of it which receives and checks whether an authentication check has been passed and logs the user into the application.

4.3 SOFTWARE ARCHITECTURAL DESIGN



4.4 SUMMARY

The uniqueness lies in the fact that this forms a new form of authentication method hands free which is different than normal biometric and face identification and very efficient as it relies only on a webcam access, and is perfect for small IOT based and similar devices.

5. CHAPTER 5 (Technical implementation and analysis)

5.1 OUTLINE

We have worked on the simple GUI provided by the opencv2 library.

5.2 TECHNICAL CODING AND CODE SOLUTIONS

Demo.py :- To run the recognition application.

```
import cv2
from gaze_tracking import GazeTracking

gaze = GazeTracking()
webcam = cv2.VideoCapture(0)

while True:
    # We get a new frame from the webcam
    _, frame = webcam.read()

    # We send this frame to GazeTracking to analyze it
    gaze.refresh(frame)

    frame = gaze.annotated_frame()
    text = ""

    if gaze.is_blinking():
        text = "0"  # Blinking
```

```
        exit()

    elif gaze.is_right():

        text = "1"  # Right

    elif gaze.is_left():

        text = "2"  # Left

    elif gaze.is_center():

        text = "3"  # Center



    cv2.putText(frame, text, (90, 60), cv2.FONT_HERSHEY_DUPLEX, 1.6,
(0, 0, 255), 2)



    left_pupil = gaze.pupil_left_coords()

    right_pupil = gaze.pupil_right_coords()

    cv2.putText(frame, "Left pupil:  " + str(left_pupil), (90, 130),
cv2.FONT_HERSHEY_DUPLEX, 0.9, (0,255,0), 1)

    cv2.putText(frame, "Right pupil: " + str(right_pupil), (90, 165),
cv2.FONT_HERSHEY_DUPLEX, 0.9, (0,255,0), 1)



    """Store the coordinates in a variable and write it to a file"""



    left_co=str(text)

    file1=open("pswdfile.txt", "a")

    file1.write(left_co)

    file1.close()



    cv2.imshow("GAZE", frame)
```

```
    if cv2.waitKey(1) == 27:
        break

webcam.release()
cv2.destroyAllWindows()
```

To perform authentication

```
import cv2
from gaze_tracking import GazeTracking

gaze = GazeTracking()
webcam = cv2.VideoCapture(0)

while True:
    # We get a new frame from the webcam
    _, frame = webcam.read()

    # We send this frame to GazeTracking to analyze it
    gaze.refresh(frame)

    frame = gaze.annotated_frame()
    text = ""
```

```
    if gaze.is_blinking():
        text = "0"  # Blinking
        exit()
    elif gaze.is_right():
        text = "1"  # Right
    elif gaze.is_left():
        text = "2"  # Left
    elif gaze.is_center():
        text = "3"  # Center

    cv2.putText(frame, text, (90, 60), cv2.FONT_HERSHEY_DUPLEX, 1.6,
(0, 0, 255), 2)

    left_pupil = gaze.pupil_left_coords()
    right_pupil = gaze.pupil_right_coords()
    cv2.putText(frame, "Left pupil:  " + str(left_pupil), (90, 130),
cv2.FONT_HERSHEY_DUPLEX, 0.9, (0,255,0), 1)
    cv2.putText(frame, "Right pupil: " + str(right_pupil), (90, 165),
cv2.FONT_HERSHEY_DUPLEX, 0.9, (0,255,0), 1)

    """Store the coordinates in a variable and write it to a file"""

    left_co=str(text)
    file1=open("pswdfile.txt", "a")
    file1.write(left_co)
```

```
    file1.close()


    cv2.imshow("GAZE", frame)


    if cv2.waitKey(1) == 27:
        break


webcam.release()
cv2.destroyAllWindows()
```

Remaining code as well as the training models can be found at:

https://github.com/chirag-ganguli/EPAS

5.3 PROTOTYPE SUBMISSION

```
import cv2

import dlib

import numpy as np



def shape_to_np(shape, dtype="int"):

    coords = np.zeros((68, 2), dtype=dtype)

    for i in range(0, 68):

        coords[i] = (shape.part(i).x, shape.part(i).y)

    return coords



def eye_on_mask(mask, side):

    points = [shape[i] for i in side]

    points = np.array(points, dtype=np.int32)

    mask = cv2.fillConvexPoly(mask, points, 255)

    return mask



def contouring(thresh, mid, img, right=False):

    cnts, _ = cv2.findContours(thresh,
cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_NONE)

    try:

        cnt = max(cnts, key = cv2.contourArea)

        M = cv2.moments(cnt)
```

```
        cx = int(M['m10']/M['m00'])
        cy = int(M['m01']/M['m00'])
        if right:
            cx += mid
        cv2.circle(img, (cx, cy), 4, (0, 0, 255), 2)
    except:
        pass


detector = dlib.get_frontal_face_detector()
predictor =
dlib.shape_predictor('shape_predictor_68_face_landmarks.dat')


left = [36, 37, 38, 39, 40, 41]
right = [42, 43, 44, 45, 46, 47]


cap = cv2.VideoCapture(0)
ret, img = cap.read()
thresh = img.copy()


cv2.namedWindow('image')
kernel = np.ones((9, 9), np.uint8)


def nothing(x):
    pass
cv2.createTrackbar('threshold', 'image', 0, 255, nothing)
```

```
while(True):

    ret, img = cap.read()

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    rects = detector(gray, 1)

    for rect in rects:



        shape = predictor(gray, rect)

        shape = shape_to_np(shape)

        mask = np.zeros(img.shape[:2], dtype=np.uint8)

        mask = eye_on_mask(mask, left)

        mask = eye_on_mask(mask, right)

        mask = cv2.dilate(mask, kernel, 5)

        eyes = cv2.bitwise_and(img, img, mask=mask)

        mask = (eyes == [0, 0, 0]).all(axis=2)

        eyes[mask] = [255, 255, 255]

        mid = (shape[42][0] + shape[39][0]) // 2

        eyes_gray = cv2.cvtColor(eyes, cv2.COLOR_BGR2GRAY)

        threshold = cv2.getTrackbarPos('threshold', 'image')

        _, thresh = cv2.threshold(eyes_gray, threshold, 255,
cv2.THRESH_BINARY)

        thresh = cv2.erode(thresh, None, iterations=2) #1

        thresh = cv2.dilate(thresh, None, iterations=4) #2

        thresh = cv2.medianBlur(thresh, 3) #3

        thresh = cv2.bitwise_not(thresh)
```

```
        contouring(thresh[:, 0:mid], mid, img)
        contouring(thresh[:, mid:], mid, img, True)


    cv2.imshow('eyes', img)
    cv2.imshow("image", thresh)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


cap.release()
cv2.destroyAllWindows()
```

5.4 TEST AND VALIDATION

Training models:-

```
from __future__ import division
import cv2
from .pupil import Pupil



class Calibration(object):



    def __init__(self):
        self.nb_frames = 20
```

```
            self.thresholds_left = []
            self.thresholds_right = []


        def is_complete(self):
            """Returns true if the calibration is completed"""
            return len(self.thresholds_left) >= self.nb_frames and
len(self.thresholds_right) >= self.nb_frames


        def threshold(self, side):


            if side == 0:
                return int(sum(self.thresholds_left) /
len(self.thresholds_left))
            elif side == 1:
                return int(sum(self.thresholds_right) /
len(self.thresholds_right))


        @staticmethod
        def iris_size(frame):


            frame = frame[5:-5, 5:-5]
            height, width = frame.shape[:2]
            nb_pixels = height * width
            nb_blacks = nb_pixels - cv2.countNonZero(frame)
            return nb_blacks / nb_pixels
```

```
        @staticmethod
        def find_best_threshold(eye_frame):

            average_iris_size = 0.48
            trials = {}

            for threshold in range(5, 100, 5):
                iris_frame = Pupil.image_processing(eye_frame, threshold)
                trials[threshold] = Calibration.iris_size(iris_frame)

            best_threshold, iris_size = min(trials.items(), key=(lambda p:
abs(p[1] - average_iris_size)))
            return best_threshold

        def evaluate(self, eye_frame, side):

            threshold = self.find_best_threshold(eye_frame)

            if side == 0:
                self.thresholds_left.append(threshold)
            elif side == 1:
                self.thresholds_right.append(threshold)
```

Calibration:-

```
from __future__ import division

import cv2

from .pupil import Pupil




class Calibration(object):




    def __init__(self):

        self.nb_frames = 20

        self.thresholds_left = []

        self.thresholds_right = []


    def is_complete(self):

        """Returns true if the calibration is completed"""

        return len(self.thresholds_left) >= self.nb_frames and
len(self.thresholds_right) >= self.nb_frames


    def threshold(self, side):


        if side == 0:

            return int(sum(self.thresholds_left) /
len(self.thresholds_left))

        elif side == 1:
```

```
                return int(sum(self.thresholds_right) /
len(self.thresholds_right))


        @staticmethod

        def iris_size(frame):


            frame = frame[5:-5, 5:-5]

            height, width = frame.shape[:2]

            nb_pixels = height * width

            nb_blacks = nb_pixels - cv2.countNonZero(frame)

            return nb_blacks / nb_pixels


        @staticmethod

        def find_best_threshold(eye_frame):


            average_iris_size = 0.48

            trials = {}


            for threshold in range(5, 100, 5):

                iris_frame = Pupil.image_processing(eye_frame, threshold)

                trials[threshold] = Calibration.iris_size(iris_frame)


            best_threshold, iris_size = min(trials.items(), key=(lambda p:
abs(p[1] - average_iris_size)))

            return best_threshold
```

```
    def evaluate(self, eye_frame, side):

        threshold = self.find_best_threshold(eye_frame)

        if side == 0:
            self.thresholds_left.append(threshold)
        elif side == 1:
            self.thresholds_right.append(threshold)
```

5.5 SUMMARY

The model is very efficient and the recognition and movement calculation is almost instantaneous as seen in the demonstration. It can effectively determine left, right and blinking type of movements. The main focus was to reduce the time gap between recognizing the movement, detecting if the pattern matches and authenticating the individual we managed to have very short intervals of about 5 seconds for the whole process.

6. CHAPTER 6 (PROJECT OUTCOME AND APPLICABILITY)

6.1 OUTLINE

Our main objective with this project was to create a novel authentication method which would be gesture based and free from the shortcomings of the traditional pin or password-based authentication since gestures become a habit while pins and passwords often tend to be alphanumeric and harder to remember.

6.2 SIGNIFICANT PROJECT OUTCOMES

Our system was successfully able to detect the movement, generate the pin and perform authentication. As a result of this project we got hands-on experience working with libraries like OpenCV and Dlib and learnt a lot about application development as well

6.3 PROJECT APPLICABILITY ON REAL-WORLD APPLICATIONS

- Banking Applications
- ATM's
- Employee checkpoints
- EV's authentication
- Smart Appliances
- Any IOT based devices needing authentication

6.4 INFERENCE

Most software products need a modern authentication method. Pins can get stolen so can passwords but its very hard to look at someone's eye movement and gauge what's happening internally in the system, that's the main advantage of implementing this.

7. CHAPTER 7 (CONCLUSIONS AND RECOMMENDATION)

7.1 OUTLINE

A robust authentication system should have certain key features like:
It should be fast (nearly instantaneous), error resistant and resistant to attacks, consistent and accurate so as to give less headache to the user while being sufficient to provide access to only the right users.

7.2 LIMITATION/CONSTRAINTS OF THE SYSTEM

Some limitations and constraints we had to work with were the learning curve could be a bit steep with this system since its a completely new way of authenticating apps different from the usual ones like pin/password etc.

7.3 FUTURE ENHANCEMENTS

We can enhance this further and use it with very secure timed OTP based systems etc. to create really secure applications resistant to even the most persistent of attacks. Such systems could be used by organizations dealing with very sensitive data like Citizen database, credit card companies etc.

7.4 INFERENCE

Information security is a significant factor when it comes to digital gadgets. We are continually concerned about the misuse of our personal information. That is why, in order to verify that a person is who he claims to be, the Iris Movement and Gaze tracking approach was adopted here, as it is also considered an effective and dependable human-computer interface. This extra layer of security will make sure that an unauthorized person will not be able to access the data even if he has access to the pin or password.

REFERENCES

1.Eye tracking in human interaction:-
https://link.springer.com/article/10.3758/s13428-020-01517-x

2.Observing response processes with eye tracking in international large-scale assessments: evidence from the OECD PIAAC assessment:-
https://link.springer.com/article/10.1007/s10212-018-0380-2

3.RemoteEye: An open-source high-speed remote eye tracker:-
https://link.springer.com/article/10.3758/s13428-019-01305-2

4.https://towardsdatascience.com/opencv-complete-beginners-guide-to-master-the-basics-of-computer-vision-with-code-4a1cd0c687f9

5.https://towardsdatascience.com/getting-started-with-opencv-249e86bd4293

6. https://towardsdatascience.com/facial-mapping-landmarks-with-dlib-python-160abcf7d672