

Eyetracking Dokumentation

Idee hinter dem Projekt

Die Aufgabe war ein Projekt zu erstellen, das OpenCV verwendet.

Meine Idee war zuerst eine Eyetracking Software zu schreiben, die mit den Kontrasten im Auge arbeitet. Leider hat die Auflösung von Webcams meist nicht die Kapazität so nah heranzuzoomen oder man hätte die Kamera sehr nah ans Auge bringen müssen.

Jedoch ist eine Anzeige von den Richtungen allein relativ nutzlos, also habe ich das Spiel Pong selbst programmiert. Das Spiel nutzt eine Plattform, auch Paddle genannt und funktioniert wie Squash nur digital und 2 dimensional.

Ich habe nur 3 Blickrichtungen gewählt, da ein Spiel, das mehrere braucht zu Problemen führen kann, da es je nach Winkel/Perspektive zu Abweichungen führen kann. So ist die Blickrichtung immer klar definierbar für das Spiel.

Auch die Belichtung kann eine Rolle spielen, da die Iris so einen höheren oder niedrigeren Kontrast haben kann.

OpenCV vs MediaPipe Ansätze

1. OpenCV-basierte Tracking-Software



Code-Bestandteile:

1. Importieren der Bibliotheken:

```
import cv2  
import numpy as np
```

OpenCV wird für die Bildverarbeitung genutzt, NumPy unterstützt dabei bei der numerischen Berechnung.

2. Laden der Haar-Cascades:

```
face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_
frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_e
ye.xml')
```

Haar-Cascades sind vortrainierte Modelle, die Gesichter und Augen erkennen können.

3. Initialisierung der Referenzpunkte:

```
reference_points = {"left_eye": None, "right_eye": None}
```

Diese Punkte dienen als Referenz, um Augenbewegungen zu messen.

4. Funktion: `detect_pupils_and_direction()`

Diese Funktion erkennt Pupillen in einem Bild und bestimmt deren Bewegung relativ zu den Referenzpunkten.

- **Gesichtserkennung:**

Erkennt Gesichter im Graustufenbild.

```
faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minN
eighbors=4)
```

- **Augenerkennung:**

Erkennt Augen innerhalb des Gesichts.

```
eyes = eye_cascade.detectMultiScale(roi_gray)
```

- **Pupillenerkennung:**

Über Histogrammausgleich und Schwellenwertsetzung werden Pupillen hervorgehoben:

```
_, threshold_eye = cv2.threshold(eye_gray, 30, 255, cv2.THRESH_BINARY_I
NV)
```

- **Richtungsbestimmung:**

Die Position der Pupille wird mit den Referenzpunkten verglichen, um die Blickrichtung zu bestimmen:

```
if cx < ref_x - ew * 0.2:
    direction = "Links"
elif cx > ref_x + ew * 0.2:
    direction = "Rechts"
else:
    direction = "Mitte"
```

5. Hauptprogramm:

- Anzeige

```
webcam = cv2.VideoCapture(0)
while True:
    ret, frame = webcam.read()
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    result_frame = detect_pupils_and_direction(gray_frame, frame)
    cv2.imshow('Pupil Tracking', result_frame)
```

2. Mediapipe-basierte Eyetracking-Software



Code-Bestandteile:

1. Importieren der notwendigen Bibliotheken:

```
import cv2
import numpy as np
import mediapipe as mp
```

Mediapipe bietet eine präzise Gesichtsanalyse durch vordefinierte Modelle.

2. Initialisierung von Mediapipe:

```
mp_face_mesh = mp.solutions.face_mesh
face_mesh = mp_face_mesh.FaceMesh(refine_landmarks=True)
```

Mediapipe's FaceMesh-Modell wird initialisiert, um Gesichtslandmarken zu erkennen.

3. Funktion: `detect_pupils_mediapipe()`

Diese Funktion verwendet Mediapipe zur Pupillenerkennung:

- **Verarbeitung des Frames:**

Mediapipe erfordert RGB-Bilder für die Verarbeitung.

```
rgb_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
results = face_mesh.process(rgb_frame)
```

- **Landmarks:**

Die Pupillenposition wird aus einer Gruppe von Landmarks berechnet:

```
left_eye_indices = [33, 160, 158, 133, 153, 144]
right_eye_indices = [362, 385, 387, 263, 373, 380]
```

- **Pupillenposition berechnen:**

Die Durchschnittsposition der Landmarks wird als Position verwendet:

```
cx, cy = int(sum(x_coords) / len(x_coords)), int(sum(y_coords) / len(y_
coords))
```

4. Hauptprogramm:

- Initialisiert die Webcam und zeigt die Ergebnisse in Echtzeit an:

```
webcam = cv2.VideoCapture(1)
while True:
    ret, frame = webcam.read()
    result_frame = detect_pupils_mediapipe(frame)
    cv2.imshow('Pupil Tracking (Mediapipe)', result_frame)
```

Vergleich der Ansätze

Merkmal	OpenCV	Mediapipe
Erkennungsgenauigkeit	Mittel	Hoch
Einführung der Pupillen	Basierend auf Schwellenwerten	Basierend auf Landmarken
Benutzerfreundlichkeit	Einfach	Etwas komplexer
Verarbeitungsgeschwindigkeit	Schnell	Kann langsamer sein

Spielmechanik

1. Ballbewegung und Kollisionen:

- Der Ball bewegt sich kontinuierlich über das Spielfeld und ändert seine Richtung, wenn er die Ränder berührt.
- Wenn der Ball auf das Paddle trifft, prallt er ab und die Richtung ändert sich ebenfalls. Zusätzlich erhöht sich der Punktestand um 1.
- Verfehlt der Ball das Paddle und berührt den unteren Spielfeldrand, ist das Spiel vorbei.

2. Steuerung des Paddles:

- Die Blickrichtung des Spielers wird mithilfe von Mediapipe und OpenCV erkannt.
- Je nach erkannter Blickrichtung (links, rechts oder Zentrum) bewegt sich das Paddle entsprechend, um den Ball abzufangen.

3. Spielstart und Neustart:

- Das Spiel beginnt, wenn die Leertaste gedrückt wird.
- Nach einem Game Over kann das Spiel durch Drücken der Taste „R“ neu gestartet werden.

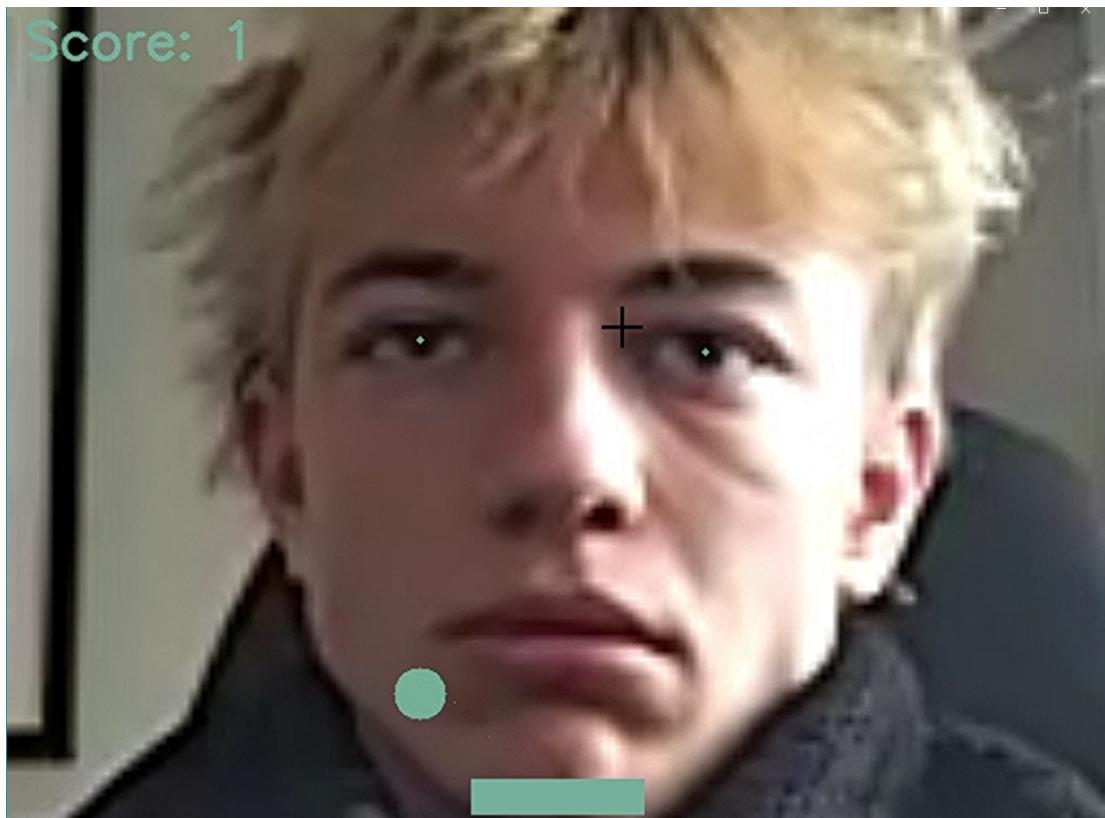
4. Soundeffekte:

- Hintergrundmusik läuft während des Spiels, um eine angenehme Atmosphäre zu schaffen.
- Bei bestimmten Aktionen, wie einem Treffer des Balls auf das Paddle oder einem Game Over, werden Soundeffekte abgespielt, um das Spielerlebnis zu intensivieren.

Startscreen



Spiel



Game Over Screen



Wie starten man das Spiel nun?

[PongGazeTracking.zip](#)

Die Code-Beschreibung mit allen Bibliotheken steht im README-File. Zuerst muss man alles aus dem PongGazeTracking.zip extrahieren und in einem Editor wie VS-Code öffnen. Danach installiert man folgende Libraries:

- OpenCV
- Mediapipe
- Pygame

Der Command dafür steht noch im README.txt.

Hardware-Voraussetzungen

Man benötigt eine Webcam, man kann dafür auch das eigene Smartphone mit Apps wie IvCam nutzen. Es könnten eventuell Probleme auftreten, wenn die Auflösung nicht auf 640×480 ist, bis jetzt hat es jedoch immer gut funktioniert.

Die Kamera muss nun bestenfalls gerade auf Augenhöhe platziert werden.

Quellen und Unterstützungen

GazeTracking

Real Time iris Detection from Scratch with Python and OpenCV – Step-by-Step Guide!
Learn Iris Detection with Python, OpenCV, and Mediapipe – From Scratch! 

Ready to dive into the world of iris detection using the power of Python? In this detailed tutorial,
 https://www.youtube.com/watch?v=k_p2RGqMITY

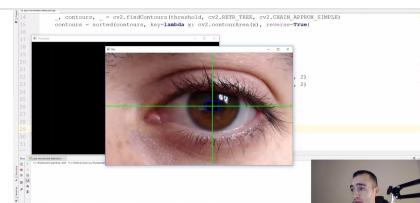


Eye motion tracking - Opencv with Python

Build your own AI vision solutions: <https://pysource.com/community>

We're going to learn in this tutorial how to track the movement of the eye using

 <https://www.youtube.com/watch?v=kbdbZFT9NQI>



Pong

AI for Everyone: Developing a Gesture Based Arcade Game Based on Mediapipe and Python

In this video lesson we introduce the concept of using OpenCV and Mediapipe to create gesture based arcade games in Python. In this lesson we suggest creating a gesture based version of the classic Pong arcade game. We show enough to get started, and your homework is to complete the game.

 https://toptechboy.com/ai-for-everyone-developing-a-gesture-based-arcade-game-based-on-mediapipe-and-python/?utm_source=chatgpt.com

pygame.mixer — pygame v2.6.0 documentation

Most useful stuff:

[Color](#) |
[display](#) |

 <https://www.pygame.org/docs/ref/mixer.html>