Mattis Dietrich, Felix Tischler & Jan-Nicolas Weider

Bericht zum dritten Meilenstein

Arbeitsfortschritt

In der Entwicklung wurden erfolgreich Kompatibilitätsprobleme zwischen Kivy und Mediapipe durch eine geschickte Auswahl der Versionen behoben. Zudem wurde die zuvor ausgelagerte Funktionen wieder in die Klasse "detection_screen" integriert, um die Kommunikation zu optimieren und Abhängigkeiten zu gewährleisten.

Ein weiterer Fortschritt besteht darin, dass ein zu langes Blinzeln über mehrere Frames nun direkt erkannt wird. Sobald ein solches langes Blinzeln erkannt wird, wird ein Warnsignal abgespielt, um den Anwender zu alarmieren.

Des Weiteren wurde eine Funktion zur Berechnung des PERCLOS-Wertes erfolgreich implementiert. Dabei wurde auch eine Nachforschung angestellt, um die Interpretation dieses Wertes besser zu verstehen und zu analysieren.

Klassifikation	**PERCLOS-Wert**
Wach	< 0.075
Zwischenstadium	> 0.075 und < 0.15
Müde	> 0.15

Um eine begrenzte historische Betrachtung zu ermöglichen, wurden dynamische Listen als Fenster eingeführt. Die Länge dieses Fensters kann im Code variabel festgelegt werden, wobei sie momentan auf 1000 Frames eingestellt ist.

Um die anfänglichen EAR- und PERCLOS-Werte zu berechnen, wurde eine Kalibrierungsfunktion entwickelt. Diese Werte sind für die Trendanalyse von Bedeutung und helfen, individuelle Unterschiede zu eliminieren und Schwellenwerte festzulegen.

Darüber hinaus wurden erste Untersuchungen durchgeführt, um die Relevanz bestimmter Features abzuwägen und diese mit Hilfe eines Nearest-Neighbour Klassifikators zu bewerten. Dies ist ein Schritt zur weiteren Verbesserung der Erkennungsfunktionen.

Für die Überlegung, anhand welcher Features klassifiziert werden kann, sind wir auf folgendes Ergebnis gekommen:

- PERCLOS
- EAR
- NN-Klassifikator
- Blink duration
- Trendanalyse (EAR / PERCLOS / Blink duration)
- Zukünftig: gähnen, Kopfbewegung

Für das Deployment wurde versucht, ein Recipie zu erstellen, um Mediapipe und Kivy als apk-Datei bereitzustellen. Es stellte sich jedoch heraus, dass es zu viele Abhängigkeiten gibt und der Build-Prozess der apk-Datei nicht erfolgreich durchgeführt werden kann. Trotz der Schwierigkeiten bei der Bereitstellung als apk-Datei wurden Anpassungen an der GUI vorgenommen, um das Erscheinungsbild der App zu verbessern. Die GUI hat nun einen neuen Stil, der ansprechender gestaltet ist, jedoch stehen hierbei weitere Änderungen noch offen.

Es wurden Versionen der App entwickelt, in denen sowohl Dlib als auch Mediapipe lokal auf einem PC laufen. Diese Versionen sind noch nicht als apk-Dateien verfügbar. Eine weitere Idee ist, mit Hilfe von generativen Sprachmodellen wie GPT-4 soll eine Übersetzung des Python-Codes in Kotlin bzw. Java-Code durchgeführt werden. Dadurch soll es möglich sein, den Code mit Android Studio in eine App zu integrieren. Diese Idee basiert auf der Tatsache, dass Java bzw. Kotlin im Bereich der App-Entwicklung für Android state-of-the-art sind und das Builden reibungsloser ermöglichen.

Erkenntnisse

Wir haben getestet:

Blink Threshold	**Benutzte Bibliothek**	**Detektierte/reale Blinzelschläge (Mattis)**	**Detektierte/reale Blinzelschläge (Jannic)**
0,16	MediaPipe	45/45	54/54
0,18	Dlib	38/48 57/52 33/52	49/47
	MediaPipe	60/60 60/53	61/50 42/41
0,20	Dlib	50/52 47/53	65/44
	MediaPipe	64/64 57/57	60/37 70/45
Awake EAR	MediaPipe	0.295	0.252
	Dlib	0.32	0.259
Awake PERCLOS	MediaPipe	0.063	0.107
	Dlib	0.08	0.109

Die Kalibrierung von Mediapipe dauerte 33 Sekunden, während die Kalibrierung von Dlib 45 Sekunden beanspruchte. Für beide Kalibrierungen wurden jeweils 1000 Frames verwendet.

Wir konnten feststellen, dass es individuelle Unterschiede bei den EAR-Werten gibt, weshalb verschiedene Thresholds fürs Blinzeln verwendet werden sollten. Bei Mattis und Jannic hat sich ergeben, dass der der Threshold maximal 63% des awake EAR betragen sollte, tendenziell eher weniger.

Die visuelle Anzeige der Blinzeldetektion funktioniert aufgrund der ständigen Aktualisierung der Frames nicht zuverlässig und der Text wird nicht immer angezeigt, was jedoch kein Problem darstellt, da die Berechnungen fehlerfrei und davon nicht betroffen sind.

Anhand unserer Tests konnten wir für uns feststellen, dass MediaPipe für uns die Bibliothek der Wahl ist, da hier eine höhere Framerate erreicht wird, die Werte scheinen zuverlässiger und während der Ausführung ist uns aufgefallen, dass der Dlib Face Detector einzelne Aussetzer hat, welche bei MediaPipe nicht zu verzeichnen sind.

Die Schätzung des Pulses und somit das Einbeziehen physiologischer Daten empfinden wir als äußerst interessant, jedoch zu zeitaufwändig, um es im Rahmen dieses Projektes zu realisieren. Anhand der von uns ausgewählten Features sollten bereits gute Genaugkeitswerte erzielt werden können.