

Entwicklung eines Algorithmus zur videobasierten Blinzelerkennung

Bachelor-Arbeit

Tobias Ochs

KOM-B-0598



TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Elektrotechnik
und Informationstechnik

Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation
Prof. Dr.-Ing. Ralf Steinmetz

Entwicklung eines Algorithmus zur videobasierten Blinzelerkennung
Bachelor-Arbeit
KOM-B-0598

Eingereicht von Tobias Ochs
Tag der Einreichung: 01. Dezember 2017

Gutachter: Prof. Dr.-Ing. Ralf Steinmetz
Betreuer: Augusto Garcia-Agundez

Technische Universität Darmstadt
Fachbereich Elektrotechnik und Informationstechnik
Fachbereich Informatik (Zweitmitglied)

Fachgebiet Multimedia Kommunikation (KOM)
Prof. Dr.-Ing. Ralf Steinmetz

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelor-Arbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in dieser oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen. Die schriftliche Fassung stimmt mit der elektronischen Fassung überein.

Darmstadt, den 01. Dezember 2017

Tobias Ochs



Inhaltsverzeichnis

1 Einleitung	3
1.1 Motivation	3
1.2 Problemstellung	3
1.3 Struktur	4
2 Grundlagen zur Bildverarbeitung, Parkinson und Lidschlag	5
2.1 Bildverarbeitung	5
2.1.1 Graustufenbild	6
2.1.2 Histogrammausgleich	6
2.1.3 Tiefpass- und Hochpassfilter	7
2.1.4 Bildpyramide	8
2.2 Morbus Parkinson	9
2.3 Lidschlag	10
3 Stand der Technik zur Gesichtserkennung	13
3.1 Gesichtserkennung	13
3.1.1 Variationen in der Bildebene	13
3.1.2 Histogramm orientierter Gradienten mit einer Support-Vektor-Maschine	14
3.2 Erkennung von Gesichtsmerkmalen	17
3.3 Analyse zweier Gesichtserkennungsverfahren	18
4 Entwurf des Algorithmus zur Lidschlagerkennung	21
4.1 Augenseitenverhältnis	23
4.2 Lokalisierung der Augenmitte	25
4.3 Entscheidungsfindung eines Lidschlags	26
5 Implementierung des Entwurfs	29
5.1 Softwarebibliotheken	29
5.1.1 OpenCV	29
5.1.2 Dlib	29
5.2 Hardwarekomponenten und Entwicklungsumgebung	30
5.3 Klassen	30
5.4 Interaktionen der Komponenten	34
5.5 Bereitstellung	34
6 Evaluation	35
6.1 Probanden	35
6.2 Laufzeitanalyse	36
6.3 Prozessor- und Speicherauslastung	39
6.4 Genauigkeit	40
7 Fazit und Ausblick	43
Literaturverzeichnis	44



Abbildungsverzeichnis

2.1	Stufen der Bildverarbeitung und Bilderkennung nach [Wah89, S. 2]	5
2.2	Links: Die Graustufentabelle, bestehend aus 256 Grautönen von schwarz bis weiß, Rechts: Die Anwendung einer Graustufentabelle auf ein Foto [sDEDV17]	6
2.3	Links: Originalbild ohne Histogrammeinebnung, Rechts: Bild nach der Histogrammeinebnung [Wah89, S. 68]	7
2.4	Links: Originalbild in Graustufen, Rechts: Bild wurde mit einem Tiefpassfilter geglättet [wik17]	8
2.5	Bildpyramide mit Flächenreduktionsfaktor vier [dW17]	8
3.1	Beispiele von Gesichtern mit verschiedenen Posen und Individuen [Row99, S. 3]	14
3.2	Beispiele für die Veränderung von Gesichtern bei extremen Lichtverhältnissen [Row99, S. 3]	14
3.3	Ein Überblick über die Feature-Extraktion und Objekterkennung [DT05]	15
3.4	Überblick über die Fehlerrate bei Zellen- und Blockgrößen Veränderungen [DT05]	16
3.5	Nummerierung und Positionen der Landmarken [Gro17]	18
3.6	Links: Erkennungs- und Falsch-Positiv-Raten mit einem 80x80 px Suchfenster, Rechts: Erkennungs- und Falsch-Positiv-Raten mit einem 40x40 px Suchfenster [Chr16, S. 70-71]	19
4.1	Einteilung des Algorithmus in die einzelnen Komponenten	21
4.2	Daten- und Kontrollfluss des Algorithmus	22
4.3	Nummerierung und Positionen der Augen-Landmarken [Gro17]	22
4.4	Oben: Offenes und geschlossenes Auge mit Landmarken, Unten: Auszug der EAR-Werte über eine Sequenz von Bildern mit einem einzelnen Lidschlag nach [SC16]	23
4.5	Oben: Drei erkannte Lidschläge, Unten: Ergebnis des gefundenen Schwellenwertes ($SW = 0,2$) nach [SC16]	24
4.6	Auswertung der EAR-Werte bei einem Testlauf des Entwurfs zum Ermitteln des Schwellenwertes, Ergebnis: $SW=0,18$	24
4.7	Künstliches Beispiel mit einem dunklen Kreis auf einem hellen Hintergrund, ähnlich wie die Iris und die Sklera, Links: Der Verschiebungsvektor d_i und der Gradientenvektor g_i haben nicht die gleiche Orientierung, Rechts: Beide Orientierungen sind gleich [TB11] . .	25
4.8	Ergebnis nach der Berechnung der Augenmitte [TB11]	25
4.9	Lidschlag innerhalb von drei Bildern: (a) offen (50 ms), (b) am schließen (100 ms), (c) geschlossen (150 ms)	26
5.1	Links: Die Klasse <i>Detector</i> kümmert sich um das Finden von Gesichtern und dessen Verwaltung, Mitte: Die Klasse <i>Face</i> bildet ein Gesicht ab und dessen Funktionen, Rechts: Die Klasse <i>Helpers</i> bietet Methoden zur Berechnung an	31
5.2	Der <i>Detector</i> verwaltet mehrere Gesichter bzw. versucht diese zu finden, die <i>Face</i> Klasse bildet ein Gesicht ab und prüft auf Lidschläge, die <i>Helpers</i> Klasse steht jeder Klasse für wichtige Berechnungen bereit	34
6.1	Ausführung des Algorithmus in der Testumgebung	36
6.2	Grafischer Ausschnitt der Auswertung von der Laufzeit des Algorithmus	37
6.3	Grafischer Ausschnitt der Auswertung von der Laufzeit des Algorithmus mit zwei Gesichtern	38
6.4	Auslastung vom Prozessor und Speicher während der Ausführung des Algorithmus	39
6.5	Auslastung vom Prozessor und Speicher während der Ausführung des Algorithmus mit zwei Gesichtern	40

6.6	ROC-Kurve zwischen Sensitivität und Spezifität	41
6.7	EAR-Werte beim einem nach oben geneigten Kopf	42
6.8	EAR-Werte beim einem nach unten geneigten Kopf	42

Tabellenverzeichnis

6.1	Laufzeitverhalten des entwickelten Algorithmus	38
6.2	Aufstellung der Konfusion-Matrix zur Lidschlagerkennung	40
6.3	Testergebnisse aus den zehn erstellten Videos mit den Probanden	41



Zusammenfassung

In der vorliegende Bachelor-Arbeit wird die Entwicklung und Evaluation eines Algorithmus zur videobasierten Blinzelerkennung beschrieben. Anwendung soll der Algorithmus in Folgearbeiten bei der Selbstdiagnose von Parkinson-Patienten finden. Hierfür muss die Realisierung des Algorithmus auf handelsüblichen Hardwarekomponenten wie Smartphone oder Laptop stattfinden können. Parkinson-Patienten weisen eine verminderte Lidschlagfrequenz auf, anhand derer eine Diagnose des Gesundheitsstands abgeleitet werden soll. Grundlage hierfür ist die Entwicklung eines Algorithmus, der den Lidschlag effizient und effektiv detektieren kann. Die Begriffe Lidschlag und Blinzeln werden in dieser Arbeit als synonym verwendet.

Auf dem Weg zum leistungsstarken Algorithmus, der die gegebenen Rahmenbedingungen erfüllt, befasst sich diese Arbeit zu Beginn mit den allgemeinen Grundlagen der Bildverarbeitung sowie der Gesichtserkennung im Speziellen. Auf Basis der Informationen aus früheren Arbeiten wird ein entsprechender Programmablauf entworfen, der unter anderem Verfahren des Augenseitenverhältnisses und der Lokalisierung der Augenmitte beinhaltet. Neben der Lidschlagerkennung beschäftigt sich diese Arbeit aber auch mit der Aufbereitung des Videobilds mittels Filter, Bildpyramide und Histogrammausgleich. Somit umfasst der entworfene Algorithmus alle notwendigen Schritte von der Aufnahme des Videosignals bis zur Auswertung des Lidschlags. Die Implementierung des Algorithmus in C++ erfolgt schlussendlich unter anderem durch die Verwendung der Softwarebibliotheken OpenCV und Dlib sowie der Definition eigener Klassen.

Zur Ermittlung der Leistungsfähigkeit des entworfenen Algorithmus wird dieser anhand eines Probandentests evaluiert. Dabei wird der Lidschlag von 10 Probanden mittels handelsüblicher Hardware beim Schauen eines Youtube-Videos beobachtet und ausgewertet. Die Ergebnisse der Laufzeit- und Auslastungsanalyse zeigen, dass der entworfene Programmablauf eine sehr gute Effizienz aufweist und problemlos auf handelsüblicher Hardware verwendet werden kann. Dies gilt sowohl für die Erkennung von einem Gesicht sowie für die parallele Erkennung von zwei Gesichtern. Hinsichtlich der Effektivität ist anhand der aufgestellten Konfusion-Matrix festzustellen, dass der entworfene Ablauf bereits eine moderate Erkennungsrate von 75 % und eine sehr gute Fehlerrate von unter 1 % aufweist. Für die Anwendung zur Selbstdiagnose bei Parkinson muss die Erkennungsrate allerdings in fortführenden Arbeiten weiter gesteigert werden. Hierzu werden im Ausblick dieser Arbeit entsprechende Maßnahmen zur Steigerung der Erkennungsrate erläutert.



1 Einleitung

Der Jakobsweg kann auch der Gang zum Bäcker sein. Als Parkinson Patient das zu tun, was man noch kann und nicht über das zu jammern, was man nicht mehr kann, das ist wichtig.

Karl-Heinz Brass

Westdeutsche Zeitung, 08.04.2011

1.1 Motivation

Die vorliegende Bachelor-Arbeit befasst sich mit der Entwicklung eines Algorithmus zur videobasierten Erkennung des Lidschlags. Der vom Autor entworfene Algorithmus soll als Basis für weiterführende Anwendungen zur frühzeitigen Diagnose von Parkinson dienen und für Entwickler und Benutzer öffentlich zugänglich sein. Erkrankt eine Person an Parkinson, wird deren Bewegungsapparat mit der Zeit immer stärker beeinträchtigt. Dies hat sowohl eine Verlangsamung der Bewegungsabläufe zur Folge, wie auch der des Lidschlags [Gmb17]. Somit ist eine Diagnose der Krankheit durch die kontinuierliche Auswertung des Lidschlags möglich. Neben der reinen Erkennung von Parkinson soll auch das Voranschreiten der Krankheit erkennbar gemacht werden. Ziel ist es, den Algorithmus so zu entwerfen, dass den Patienten eine Selbstdiagnose bequem von zu Hause aus ermöglicht wird. Die Diagnoseergebnisse können dann digital an den betreuenden Arzt übermittelt oder vom Arzt selbst abgerufen und ausgewertet werden, wodurch eine kontinuierliche Behandlung ermöglicht wird.

Morbus Parkinson gehört zu den häufigsten Krankheiten des Nervensystems weltweit. In Deutschland geht man von einer Gesamtzahl von ungefähr 220.000 Parkinson-Patienten aus [Gmb17]. Der Einflussbereich des zu entwerfenden Algorithmus umfasst somit eine große Zahl an Personen. Die Häufigkeit der Parkinson-Krankheit nimmt mit dem Alter zu, aber auch junge Menschen sind von der Krankheit befallen. 10 % der Erkrankten zeigen Symptome schon vor dem 40. Lebensjahr [Gmb17]. Durch die demografische Entwicklung ist mit einer stetigen Zunahme der Patientenzahl zu rechnen, was das Vorhandensein eines modernen Diagnosetools unabdingbar macht.

1.2 Problemstellung

Die Herausforderung beim Entwurf des Algorithmus besteht darin, dass dieser auf Bilddaten in Echtzeit operieren muss. Vorhanden Ressourcen müssen demnach optimal ausgenutzt werden können. Für die bequeme Selbstdiagnose ist wichtig, dass handelsübliche Hardware, wie sie in jedem Haushalt aufzufinden ist, und nicht teure Spezialhardware zum Einsatz kommen kann. Durch Smartphones und Laptops sind heutzutage in nahezu jedem deutschen Haushalt hochauflösende Kameras vorhanden, welche für dieses Vorhaben nutzbar gemacht werden sollen. Dafür muss die Datenverarbeitung und Ergebnisberechnung ohne merkliche Verzögerung von statthen gehen.

Der Algorithmus muss zudem robust auf mögliche Beeinträchtigungen bei der Bilderfassung reagieren, die beispielsweise durch schlechte Lichtverhältnisse oder personenbezogene Hindernisse wie Brille, Bart oder Make-Up auftreten können. Erschwerend kommt bei der Erkennung des Lidschlags noch hinzu, dass dieser in wenigen Millisekunden vollzogen ist [SWG84] und daher jede Information im Datenstrom von Relevanz ist. Die Auswertung muss sowohl effizient als auch effektiv durchgeführt werden, um neben einer schnellen Datenverarbeitung auch eine hohe Genauigkeit der Erkennungsrate zu gewährleisten. Abgrenzend ist noch zu sagen, dass sich diese Arbeit auf die Entwicklung des Algorithmus zur Erkennung

des Lidschlags unter den gegebenen Bedingungen beschränkt und keine für den Endnutzer lauffähige Anwendung bereitstellt.

1.3 Struktur

Für die Entwicklung des Algorithmus werden zunächst Grundlagen der Bildverarbeitung benötigt, welche in Kapitel 2 kurz erläutert werden. Die Beschreibung der Grundlagen beschränkt sich dabei auf Gebiete, die für die Themen Parkinson und Lidschlag von Bedeutung sind. Im Anschluss an Kapitel 2 wird in Kapitel 3 auf die besonderen Herausforderungen bei der Gesichtserkennung eingegangen. Das Verfahren zur Gesichtserkennung, welches im entworfenen Algorithmus Verwendung findet, wird dabei detailliert beschrieben. Zusätzlich werden aktuelle Ansätze zur Extraktion der Gesichtsmerkmale kritisch hinterfragt. Anschließend werden in Kapitel 4 die in dieser Arbeit genutzten Ansätze für die Erkennung eines Lidschlags und zum Finden der Pupillen beschrieben. Diese Ansätze bilden die Basis für den Algorithmen-Entwurf. Die detaillierte Herangehensweise und die Entwicklung des Algorithmus werden in Kapitel 5 gezeigt und erläutert. Die Evaluation dieses Algorithmus sowie die dabei erhaltenen Resultate auf einem eigens erstellten Datensatz mit Probanden werden in Kapitel 6 präsentiert und bewertet. Zum Abschluss liefert Kapitel 7 ein Fazit der Ergebnisse und einen Ausblick auf weitere mögliche Optimierungen und Verbesserungen.

2 Grundlagen zur Bildverarbeitung, Parkinson und Lidschlag

Den Schwerpunkt dieser Arbeit bildet die Erstellung eines geeigneten Verfahrens zur Erkennung des Lidschlags einer Person. Dieser Algorithmus wird auf einem Digitalrechner ausgeführt, der die Bildinformation als einen zeitlich äquidistanten Datenstrom von Bilddaten von einer Kamera erhält. Zum besseren Verständnis wird an dieser Stelle kurz die allgemeine Bildverarbeitung erläutert. Danach wird näher auf die Parkinson-Krankheit und ihrer Besonderheiten eingegangen. Einzelne, für diese Arbeit wichtige Fakten zu einem menschlichen Lidschlag und Hinweise auf die Auswirkung von Parkinson auf den Lidschlag beenden Kapitel 2.

2.1 Bildverarbeitung

Nach Wahl [Wah89] handelt es sich bei der Bildverarbeitung um eine Signalverarbeitung mit dem Ziel, Informationen aus Bilddaten zu gewinnen. Zwei Arten sind dabei bestimmend, die Bildverbesserung und die Bilderkennung.

Die Bildverbesserung dient der Qualitätsverbesserung von Informationen in einem Bildmaterial mit Hilfe der Signalverarbeitung. Die Bilderkennung ist eine Methode zur Entscheidungsfindung beziehungsweise zur Klassifikation von Objekten aus vorliegenden Bilddaten.

Die Erkennung des Lidschlags kann der Bilderkennung zugeordnet werden, da eine Information aus dem Bilddatenstrom gewonnen werden soll. Dabei handelt es sich um die binäre Aussage, ob die Augenlider *offen* oder *geschlossen* sind. Bilderkennungssysteme nutzen jedoch auch die Methoden der Bildverbesserung, um beispielsweise eine Informationsreduktion vor der eigentlichen Erkennung vorzunehmen. Abbildung 2.1 zeigt den allgemeinen Aufbau eines Bilderkennungssystems nach [Wah89].

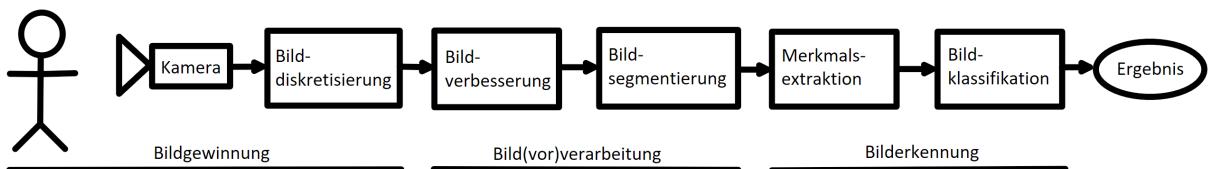


Abbildung 2.1: Stufen der Bildverarbeitung und Bilderkennung nach [Wah89, S. 2]

Die Szene mit den enthaltenen Objekten wird von der Kamera aufgenommen und durch die Bilddiskretisierung in einen Datenstrom umgewandelt. Danach werden die Bilddaten für die weiteren Prozessschritte mit Verbesserungsverfahren vorbereitet, um eine Reduktion der Information vorzunehmen, wie etwa eine Wandlung in Graustufen oder die Qualität zu verbessern wie z. B. Rauschfilterung oder Deinterlacing.

Darauffolgend findet eine Merkmalsextraktion statt, um aus den Daten Merkmale zu gewinnen. Die so extrahierten Merkmale werden dann mit Hilfe eines Klassifikators eingeteilt um schließlich die gewünschte Information zu generieren, die zur weiteren Verarbeitung einer Informationssenke bereitgestellt wird. Die Erkennungsmethoden zur Merkmalsextraktion und Klassifikation sind zumeist nichtlineare Verfahren, die nicht zwingend deterministisch arbeiten müssen.

Die gleichen Ansätze werden für Videodaten ebenso verwendet, wobei die dort zu verarbeitenden Daten zusätzlich eine zeitliche Komponente enthalten.

2.1.1 Graustufenbild

Nach Burger und Burge [BB09] spielen Graustufenbilder eine wichtige Rolle für die Gesichtserkennung, um die Verarbeitung in Echtzeit zu beschleunigen. Bilder von Gesichtern liegen oftmals in der Farbversion vor, weswegen diese erst in ein Graustufenbild umgewandelt werden müssen. Das Farbbild (RGB-Bild) hat mit Rot (R), Grün (G) und Blau (B) drei Kanäle zur Verfügung, um Farben zu generieren. Das Graustufenbild besitzt dagegen nur einen Kanal, dementsprechend kann ein Pixel des Graustufenbildes nur einen Grauwert annehmen.

Die Länge des Binärworts für den Normalgebrauch beträgt 8 Bits pro Pixel und kann daher $2^8 = 256$ Werte annehmen, wobei hier der Wert 0 der Farbe Schwarz und der Wert 255 der Farbe Weiß entspricht. Je höher der Wert des Grauwertes desto heller erscheint der Pixel. Somit können mehrere Grauwerte dargestellt werden. Ein möglichst einfacher Algorithmus für das Umwandeln eines RGB-Bildes in ein Graustufenbild ist der Durchschnitt über alle drei Kanäle. G (2.1) steht hierbei für den äquivalenten Grauwert nach [BB09]:

$$G = \frac{1}{3} \cdot R + \frac{1}{3} \cdot G + \frac{1}{3} \cdot B \quad (2.1)$$

Ein Beispiel über die Farbpalette von 0 bis 255 Grauwerten und einem Graustufenbild wird in Abbildung 2.2 dargestellt. Hier werden helle Farben, wie etwa das Blau des Himmels, auf einen hellen Grauwert abgebildet. Dunkle Farben, wie etwa die Blätter des Baumes, werden auf einen Bereich der dunklen Grauwerte abgebildet.

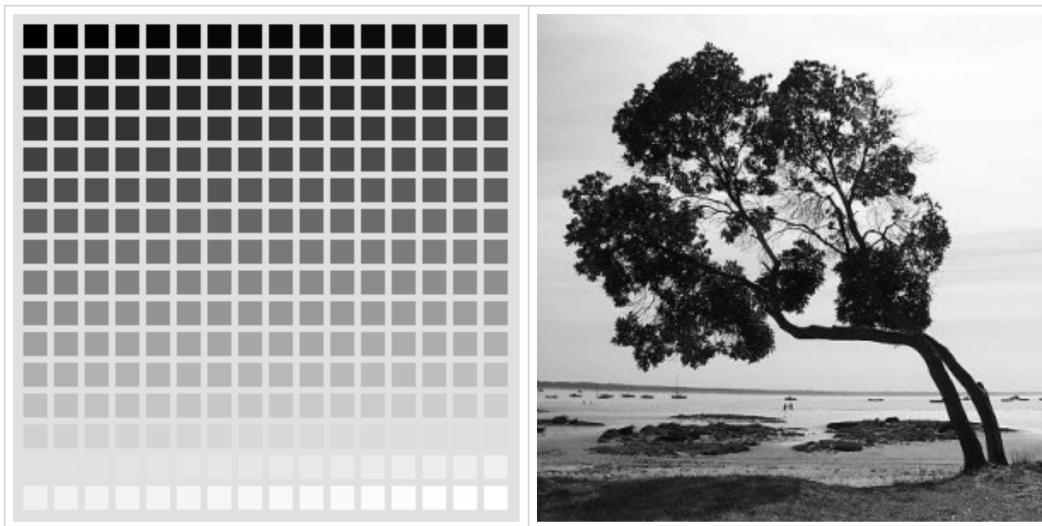


Abbildung 2.2: Links: Die Graustufentabelle, bestehend aus 256 Grautönen von schwarz bis weiß, Rechts: Die Anwendung einer Graustufentabelle auf ein Foto [sDEdV17]

2.1.2 Histogrammausgleich

Als Histogramm wird im Allgemeinen eine Häufigkeitsverteilung bezeichnet, die in der Bildverarbeitung die verschiedenen Häufigkeiten der Helligkeitswerte widerspiegelt. In einem Graustufenbild stellen diese Helligkeitswerte die unterschiedlichen Graustufen dar [BB09]:

$$h(i) = \text{card}\{(x, y) | I(x, y) = i\} \quad (2.2)$$

Somit gibt $h(i)$ (2.2) die Anzahl der Pixel in einem Bild mit der Intensität i wieder. Dabei steht $I(x, y)$ für den Pixelwert des Bildes an der Stelle (x, y) und card gibt die Mächtigkeit, also die Summe der

Vorkommnisse der einzelnen Pixelwerte an.

Der Histogrammausgleich wird genutzt, um den Kontrast von Bildern zu verbessern. Nach Rowley [Row99] führt ein geringer Kontrast oftmals dazu, dass die entscheidenden Gesichtsmerkmale bei der Gesichtserkennung nicht deutlich zu erkennen sind. Daher ist es notwendig Bilder soweit anzupassen, dass ihre Intensitätsverteilung ähnlich ist, um diese besser vergleichen und vereinheitlichen zu können [BB09]. Dadurch werden eine gute Belichtung und ein starker Kontrast erzeugt. Dies findet vor allem Verwendung bei über- oder unterbelichteten Bildern.

In Abbildung 2.3 wird ein Bild, welches im Originalzustand zu dunkel erscheint, durch einen linearen Histogrammausgleich verbessert. Dafür werden die Weißwerte im Bild angehoben und das Bild im Kontrast verbessert.



Abbildung 2.3: Links: Originalbild ohne Histogrammeinebnung, Rechts: Bild nach der Histogrammeinebnung [Wah89, S. 68]

2.1.3 Tiefpass- und Hochpassfilter

Durch die Anwendung spezifischer Filter wird die Objekterkennung in Bildern durch Verfahren der Reduktion oder Hervorhebung verbessert. Hierfür werden z. B. störende Anteile in einem Bild reduziert oder informative Anteile hervorgehoben. Filteroperationen beziehen zur Berechnung eines Grauwertes im Ereignisbild eine ganze Umgebung des ursprünglichen Bildpunktes mit ein. Dabei wird zwischen linearen und nichtlinearen Filter unterschieden.

Bei der Gesichtserkennung wird üblicherweise ein linearer Tiefpassfilter wie dem Gaußschen Rauschen angewendet, um z. B. Pupillen besser in einem Bild zu identifizieren. Dieser kann zum Glätten eines Bildes helfen, also zum Entfernen von hochfrequenten Störungen. Nachteil ist, dass Kanten verwaschen werden. Um Kanten und Spitzen in Bildern zu betonen werden Hochpassfilter verwendet [Wah89].

Abbildung 2.4 zeigt ein solches Bild das mit einem Tiefpassfilter geglättet wurde. Das geglättete Bild sieht danach etwas verwaschen bzw. verrauscht aus, wohingegen das Originalbild körniger dargestellt wird. Im Gegensatz zur menschlichen Objekterkennung ist das verwaschene Bild für die rechnerbasierte Erkennung besser, da in diesem die Pixel ineinander verlaufen und störende Anteile entfernt wurden.



Abbildung 2.4: Links: Originalbild in Graustufen, Rechts: Bild wurde mit einem Tiefpassfilter geglättet [wik17]

2.1.4 Bildpyramide

Bildpyramiden finden ihren Einsatz, da üblicherweise Gesichter und Objekte auf Bildern in unterschiedlichen Größen vorkommen können. Zur Korrektur wird das Bild, welches als Eingabe übergeben wurde, auf verschiedene Größen skaliert und jedes skalierte Bild für sich separat untersucht [DT05].

Abbildung 2.5 dient zur Veranschaulichung dieser Struktur, bei der ein Reduktionsfaktor von vier angewendet wurde. Dadurch wird das Ausgangsbild um $\frac{1}{4}$ seiner Fläche verkleinert. Das daraus resultierende Bild wird dann ebenso um $\frac{1}{4}$ seiner Fläche verkleinert. Dies wird bis zu einem bestimmten Verkleinerungsfaktor fortgeführt. Durch die Verkleinerung können nun auch Objekte erkannt werden, die aufgrund ihrer Darstellungsgröße im Ursprungsbild nicht erkannt werden konnten.

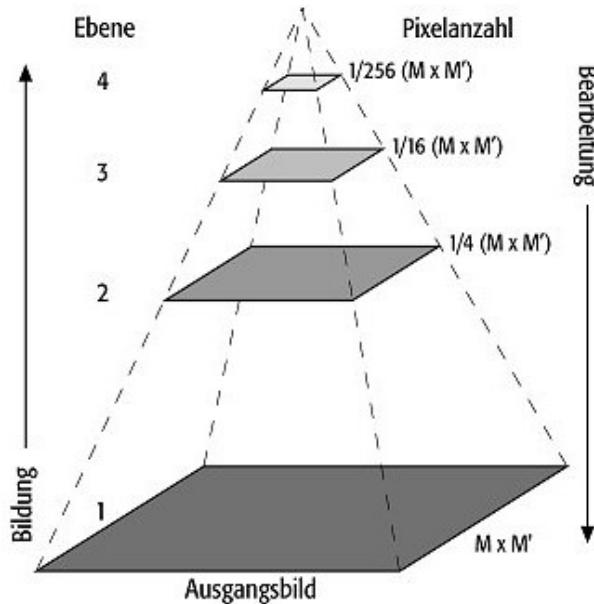


Abbildung 2.5: Bildpyramide mit Flächenreduktionsfaktor vier [dW17]

2.2 Morbus Parkinson

Die Parkinson-Krankheit (PD) ist eine neurodegenerative Erkrankung, die durch die fortschreitende Degeneration auf Dopamin reagierende Neuronen in der Substantia nigra pars compacta, durch das reduzierte Striatum-Dopamin und die Anwesenheit von Lewy-Körperchen verursacht wird. PD präsentiert sich jedoch auch als eine nicht-dopaminerge Degeneration in den cholinergen, Noradrenalin-, Serotonin-Neuronen, dem olfaktorischen System, dem Rückenmark, dem peripheren autonomen System und den Hemisphären. Derzeit betrifft es ca. 220.000 Person in Deutschland [Gmb17], aber basierend auf der prognostizierten Bevölkerungsalterung wird erwartet, dass seine Prävalenz in Zukunft drastisch zunehmen wird. Die Prävalenz von PD nimmt mit zunehmendem Alter zu, mit einem mittleren Erkrankungsalter von 60 Jahren, obwohl Fälle von PD bei bereits 20-jährigen Patienten berichtet wurden [LFK⁺14].

Gegenwärtig besteht die klinische Behandlung von PD hauptsächlich aus der dopaminergen Ersatztherapie (DRT) durch den metabolischen Dopamin Vorläufer L-DOPA. Diese Therapie mildert die dopaminergen Symptome von PD vorübergehend, verursacht aber sowohl akute Nebenwirkungen (Dyskinesie, orthostatische Hypotonie, Übelkeit) als auch Langzeitkomplikationen (Sturz, Gefrieren, vegetative Dysfunktion, Schlafstörungen, Demenz) sowie vermehrte motorische Komplikationen, da das Medikament die Effektivität verliert (EIN/AUS-Perioden, mit längeren AUS-Perioden und erhöhter Verzögerung beim Einsetzen von EIN-Perioden) [LFK⁺14].

Angesichts der Tatsache, dass das Hauptziel bei der PD-Behandlung die Milderung der dopaminergen Symptome ist, sind es die nicht-dopaminergen Symptome, sowie die Nebenwirkungen von Levodopa, die letztlich für die Verschlechterung der Lebensqualität und letztendlich für die Aufnahme im Pflegeheim verantwortlich sind. Zum Beispiel zeigte eine Studie an 143 PD-Patienten, dass motorische Komplikationen, insbesondere nächtliche Akinesie und biphasische Dyskinesie, die Lebensqualität von PD-Patienten am stärksten beeinflussen [COM⁺05]. Des Weiteren kann Demenz als ein potenzieller Indikator für ein hohes Risiko einer stationären Pflege angesehen werden, wobei die Denkstörung der stärkste Prädiktor ist [ALTL00].

Motorische Einschränkungen könnten einen negativen Effekt auf den entwickelten Algorithmus zur Lidschlagerkennung haben. Abhängig davon wie weit diese fortgeschritten sind und ob sich diese im Gesicht befinden. Kann ein Patient seine Augenlider nur noch eingeschränkt bewegen, wird es schwierig zu erkennen ob dieser geblinzelt hat oder nicht. Darüber hinaus deuten Studien darauf hin, dass diese nicht dopaminerge Degeneration vor der dopaminergen Degeneration beginnt und daher Symptome wie Herzderervation, Anosmie, Depression, Verstopfung oder REM-Schlafstörung den klassischen Symptomen der PD vorausgehen, was sie als Prodromal-PD definiert [PAB⁺12]. Dies eröffnet die Möglichkeit, vorherzusagen, wann Patienten in naher Zukunft ein Risiko haben, eine Parkinson-Krankheit zu entwickeln, oder genauer gesagt, welche Patienten wahrscheinlich innerhalb weniger Jahre die klassischen, dopaminergen Symptome der Parkinson-Krankheit zeigen werden [LFK⁺14].

Die positive Diagnose der idiopathischen Parkinson-Krankheit (IPD), die die häufigste Form von Parkinsonismus (das Auftreten Parkinson-artiger Symptomatik) ist (75 % der Fälle), folgt häufig dem Standard der Parkinson-Krankheit [HDKL92] und wird derzeit durch das Vorhandensein der Kardinal motorischen Merkmale: Ruhetremor, Asymmetrie und eine gute Reaktion auf DRT, was eine bestätigte Pathologie in 99 % der Fälle ermöglicht. Andere Ursachen wie Toxine, Stoffwechselerkrankungen oder die Behandlung mit bestimmten Medikamenten können jedoch auch Parkinsonismus verursachen. Diese anderen Formen des Parkinsonismus neigen im Gegensatz zur klassischen PD nicht dazu, mit der Zeit fortzuschreiten [LFK⁺14].

Die pharmakologische Behandlung von PD ist äußerst komplex und stark vom Zustand des Patienten

abhängig. Sobald die Diagnose bestätigt ist, ist die übliche Behandlung Carbidopa in Kombination mit Levodopa. Während die Krankheit fortschreitet, muss die Dosierung angepasst werden, um lange EIN-Perioden aufrechtzuerhalten und die Funktionalität des Patienten zu erhalten. Mit der Zeit ruft jedoch eine dopaminerge Ersatztherapie in fast allen Fällen schwere Nebenwirkungen, wie Dyskinesien und Halluzinationen, hervor [AGMANJJ13].

Zur Behandlung von nicht-dopaminergen Symptome kann Deep-Brain-Stimulation (DBS) auf den Nucleus subthalamicus oder Globus pallidus internus angewendet werden, um die AUS-Perioden und Dyskinesien zu reduzieren. Ein Problem ist, dass Demenz auch ein Ausschlusskriterium für DBS ist.

Eine kontinuierliche Kontrolle der Symptome von PD wird wesentlich. Dazu kann der Algorithmus zur Lidschlagerkennung eine wichtige Rolle spielen, um den Zustand und den Fortschritt von Patienten mit PD zu überwachen, mit besonderem Augenmerk auf die Kontrolle von Verschlechterungen von Bradykinesie.

2.3 Lidschlag

Nach Cramon [Cra80] ist ein Lidschlag eine phasische, bilaterale synchrone Lidkontraktion mit unterschiedlichem Bewegungseffekt unter Beteiligung beider Ober- und Unterlider.

Der Lidschlag besteht aus einem Lidschluss, dem unmittelbar die Lidöffnung folgt. Dabei werden die Augenlider nach oben und unten bewegt. Nach Galley [Gal01] wird der Hauptanteil der Bewegung beim Menschen vom Oberlid geleistet. Darüber hinaus begleitet zusätzlich das Oberlid die auf- und abwärts gerichteten Blickbewegungen, was zu einer fortlaufend wechselnden Lidstellung und Lidspalte führt.

Dem Lidschlag mit dem Lidschluss und der Lidöffnung steht der Lidschluss ohne Lidöffnung beim Einschlafen gegenüber. Beim Einschlafen werden fernerhin die Augäpfel nach oben in die Augenhöhle gedreht, was beim normalen Lidschlag nicht im gleichen Maße auftritt.

Um den Lidschlag von anderen Lid- oder Augenbewegungen und sonstiger Mimik zu unterscheiden, ist die Kenntnis der Wellenform des Lidschlages unerlässlich. Die Parameter Lidschlagamplitude und Lidschlagdauer bestimmen im Wesentlichen die Wellenform des Lidschlages. Die Lidschlagamplitude ist die Distanz, welche die Augenlider während des Lidschlusses überwinden. Sie ist vom Blick des Auges abhängig. Beim aufwärtigen Blick ist die Lidschlagamplitude größer als bei abwärts gerichtetem Blick, bei dem sie kleiner ist. Die Lidschlagdauer beträgt abhängig von der ihr zu Grunde liegenden Messvorschrift ungefähr 150 - 250 ms [SWG84].

Laut Moses [Mos81] blinzelt ein ruhiger Mensch etwa 10- bis 15-mal die Minute. Dies entspricht ungefähr alle 4 bis 6 Sekunden einmal.

Wie oft sich die Augenlider in der Minute über den Augapfel schieben, hängt aber von verschiedenen Faktoren ab. Beispielweise verändert sich die Lidschlag-Frequenz beim Fernsehen oder am Computer deutlich. Starrt ein Mensch, kann diese auf unter 5-mal pro Minute sinken [Ber99].

Nach Stern et al. [SWG84] beträgt die Zeit für einen Lidschluss während des Lidschlages ungefähr 50 ms für den schnellsten und 145 ms für den langsamsten Lidschluss. Außerdem ist sie abhängig von der Länge des Weges, den das Oberlid durchläuft.

Die Öffnungszeit des Augenlids dauert nach Stern et al. [SWG84] ungefähr 100 - 200 ms bei Müdigkeit und ist somit länger als die Zeit für das Schließen der Augenlider.

Wild [Wil83] stellte fest, "dass der Lidschluss vor allem dann erfolgt, wenn wichtige Informationen mit der geringsten Wahrscheinlichkeit zu erwarten sind, d. h., dass die Setzung der Lidschlussereignisse

nisse von der subjektiven Beurteilung der unmittelbar bevorstehenden Umweltereignisse abhängig ist“. Fernerhin fand Wild [Wil83], dass der Lidschluss nicht nur vom visuellen Informationssystem, sondern von der Erwartung bezüglich des unmittelbar bevorstehenden Gesamtverhaltens unter Einschluss auch anderer Sinneskanäle bestimmt wird: „Die Augen bleiben offen, wenn wichtiges Verhalten bevorsteht, und werden erst dann kurz geschlossen, wenn die entsprechende sensomotorische Verhaltensphase abgeschlossen ist“ [Wil83].

Bei Parkinson-Patienten wurde vermehrt Bradykinesie als vorzeitiges Symptom festgestellt. Ein möglicher Indikator für Bradykinesie ist eine verminderte Blinkfrequenz. Studien, die PD- und Kontrollpatienten vergleichen, zeigen, dass PD-Patienten im Durchschnitt eine Lidschlagreduzierung von durchschnittlich 7,11 Lidschlägen pro Minute haben. Der Schweregrad der Krankheit, der durch die HY-Skala definiert wird, scheint diesen Wert nicht weiter zu beeinflussen, aber die Differenz steigt mit dem Alter an. Fitzpatrick et al. [FHS⁺12] schlägt einen Wert von 20 Lidschlägen pro Minute oder niedriger als möglichen Indikator für PD vor.

Diese Informationen sind beim Entwurf des Algorithmus wichtige Randbedingungen. Da ein willkürlicher Lidschlag innerhalb von 150 – 200 ms vollzogen ist, ist es unabdingbar das die Informationsverarbeitung schnell durchgeführt wird, um genügend Bilder bei der Auswertung zur Verfügung zu haben. Des Weiteren ist der verminderte Lidschlag das Hauptsymptom von Parkinson, welches vom Erkennungsalgorithmus eindeutig erkannt werden muss. Hierzu muss nur die Anzahl der Lidschläge pro Minute gemessen werden und der daraus resultierende Wert kann der behandelte Arzt auswerten und in seine Diagnose mit einbeziehen.



3 Stand der Technik zur Gesichtserkennung

In diesem Kapitel wird der Stand der Technik zur Gesichts- und der Merkmalerkennung gegeben. Dies beinhaltet vorliegende Herausforderungen und eingesetzte Methoden. Zuerst werden Variationen von Gesichtern aufgezeigt und ein Ansatz zur Erkennung von Gesichtern in Bildern erläutert. Neben diesem Ansatz gibt es noch weitere, doch diese wurden bei der Analyse (siehe Abschnitt 3.3) der Ansätze ausgeschlossen.

3.1 Gesichtserkennung

Nach Rowley [Row99] kann das Erkennen von Gesichtern in Bilder durch mehrere Faktoren deutlich erschwert werden. Dies kann zum einen durch Probleme mit dem Bild oder zum anderen durch Probleme mit den Gesichtern auf dem Bild verursacht werden. Trotz dieser Faktoren wird versucht eine möglichst hohe Genauigkeit bei der Erkennung zu erreichen und diese in Echtzeit durchzuführen [VJ04]. Die häufigsten Störfaktoren werden im Folgenden aufgelistet und kurz erläutert.

3.1.1 Variationen in der Bildebene

Wenn das Gesicht durch gewisse Umstände nicht deutlich zu erkennen ist, führt dies zu Schwierigkeiten bei der Bestimmung. Das geschieht hauptsächlich durch eine Veränderung oder eine Verdeckung von Gesichtsmerkmalen.

Die am häufigsten auftretenden Störfaktoren werden im Folgenden kurz beschrieben.

Haltungsvariation

Haltungsvariationen können z. B. Rotation und Translation sein. Rotationen des Gesichts, die sich nicht in der Bildebene befinden, können einen größeren Einfluss auf das Erscheinungsbild haben. Eine andere Variation ist der Abstand der Verkrümmung von der Kamera, wobei Veränderungen in der Perspektive zu einer Verzerrung führen können [Row99]. Beispiele mit Gesichtern die eine Transformation aufweisen werden in Abbildung 3.1 gezeigt.

Licht und Textur Variation

Diese Variation wird verursacht durch das Objekt und seine Umgebung. Dabei spielt die Oberfläche-neigenschaften des Objekts und die umgebenen Lichtquellen eine große Rolle. Zum Beispiel können Veränderungen der Lichtquelle das Aussehen eines Gesichts radikal verändern [Row99] wie es in Abbildung 3.2 gezeigt wird.

Veränderung

Ein Punkt ist die Formveränderung des Gesichts, die u. a. durch Bärte oder Hüte herbeigeführt werden kann. Da manchmal eine ovale oder elliptische Form als Identifikationsmerkmal genutzt wird, kann dies zu Streckung oder Stauchung dieser Form führen. Enthaltene Orientierungspunkte des Gesichts, wie Augen, Nase oder Mund, werden dadurch verschoben und befinden sich dann außerhalb des Suchmusters. Auch unterschiedliche Gesichtsausdrücke oder Grimassen, wie ein offener Mund, können Veränderungen auf verschiedenste Art und Weise herbeiführen.



Abbildung 3.1: Beispiele von Gesichtern mit verschiedenen Posen und Individuen [Row99, S. 3]



Abbildung 3.2: Beispiele für die Veränderung von Gesichtern bei extremen Lichtverhältnissen [Row99, S. 3]

Verdeckung

Ursachen einer Verdeckung können beispielsweise Gegenstände sein, die Teile des Gesichts verdecken. Eine Verdeckung entsteht, wenn komplette Teile des Gesichts verdeckt und damit nicht mehr wahrnehmbar sind. Dies kann als Ursache eine Abdeckung durch Objekte, wie durch eine Brille oder einen Schal, oder auch eine seitliche Ansicht des Gesichts haben. Beispielhaft wäre eine Person, die Ihre Hand vor dem Mund hält. Auch ist es möglich, dass sich in Bildern mit mehreren Personen, Gesichter überlappen und damit verschiedenste Bestandteile, wie die Augen, nicht vollständig sichtbar sind. Die Ansicht des Gesichts spielt ebenfalls eine große Rolle für die Erkennung. Gesichter in der Frontalansicht werden leichter erkannt als Gesichter in der Profilansicht.

3.1.2 Histogramm orientierter Gradienten mit einer Support-Vektor-Maschine

Als Konzept zur Gesichtserkennung wird die Nutzung von Histogrammen orientierter Gradienten (HoG) mit einer Support-Vektor-Maschinen (SVM) von Dalal und Triggs [DT05] vom Jahr 2005 erläutert. Dies ist ein Objekterkennungsalgorithmus der zur Fußgängererkennung als Anwendungsfall entwickelt wurde. Diese Methode wird auch von der Softwarebibliothek Dlib zum Erkennen von Frontalgesichtern verwendet, welche zur Erstellung des Algorithmus Verwendung findet (siehe Abschnitt 5.1.2).

Neben dem Ansatz mit Support-Vektor-Maschinen gibt es auch Ansätze mit Kaskaden, beschrieben von Viola und Jones [VJ04], und Neuronalen Netzen, beschrieben von Rowley [Row99], welche in dieser

Arbeit aber nicht zum Einsatz kommen. Der Ansatz von Dalal und Triggs [DT05] kann sowohl mit Graustufenbildern als auch Farbbildern arbeiten.

Bei dieser Arbeit würde das Vorgehen mit Farbbildern den Algorithmus nur ausbremsen, da er nicht nur rein auf der Gesichtserkennung basiert, sondern das Bild noch in weiteren Schritten verarbeitet. Um dort auch den Vorteil der schnellen Bearbeitung ausnutzen zu können wird rein mit Graustufenbildern gearbeitet.

Wie in Abbildung 3.3 zu sehen ist, läuft während des Prozesses über das Eingabebild ein Suchfenster, das zuerst in ein Raster aus Zellen aufgeteilt wird, für die sogenannte Gradienten-Vektoren berechnet werden. Die Zellen werden dann wiederum in Blöcke zusammengefasst, für die, mithilfe der Gradienten-Vektoren, Histogramme erzeugt werden. Anschließend werden die Histogramme konkateniert und an eine trainierte Support-Vektor-Maschine weitergeleitet, die darüber entscheidet ob ein Gesicht enthalten ist oder nicht. Die Bildpyramide, die das Eingabebild in unterschiedlichen Auflösungen darstellt, wird genutzt, um Gesichter in verschiedenen Größen zu erkennen.



Abbildung 3.3: Ein Überblick über die Feature-Extraktion und Objekterkennung [DT05]

Die eigentliche Methode diente hauptsächlich zur Erkennung von Personen, beispielsweise für Fußgänger im Straßenverkehr. Jedoch lässt sich das Verfahren zur verallgemeinerten Objekterkennung nutzen [DT05], wodurch auch das Erkennen von Gesichtern ermöglicht wird.

In den folgenden Abschnitten werden die Grundprinzipien dieser Vorgehensweise kurz erläutert.

Gradienten-Vektor

Zur Erzeugung der benötigten Histogramme orientierter Gradienten werden als Grundlage Gradienten-Vektoren benötigt. Ein solcher Gradienten-Vektor wird für jeden Pixel innerhalb des Bildes berechnet und bildet sich anhand folgender Gleichungen [RK14]:

$$G_x(x, y) = H(x + 1, y) - H(x - 1, y) \quad (3.1)$$

$$G_y(x, y) = H(x, y + 1) - H(x, y - 1) \quad (3.2)$$

In diesem Fall definiert G_x (3.1) den horizontalen Richtungsgradienten und G_y (3.2) den vertikalen Richtungsgradienten des Vektors an der Stelle (x, y) des Bildes. $H(x, y)$ bezeichnet die Intensität der Pixelwerte. Mit $G(x, y)$ (3.3) wird die Gradientengröße und mit $\alpha(x, y)$ (3.4) die Gradientenrichtung des Vektors angegeben [RK14]:

$$G(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)} \quad (3.3)$$

$$\alpha(x, y) = \tan^{-1}\left(\frac{G_y(x, y)}{G_x(x, y)}\right) \quad (3.4)$$

Histogramm orientierter Gradienten

Mithilfe der vorliegenden Gradienten-Vektoren für jeden Pixel, wird ein Histogramm erzeugt. Bei Histogrammen orientierter Gradienten (HoG) wird dabei die Häufigkeit der Orientierung von Gradienten-Vektoren genutzt. Zur Verbesserung der Robustheit gegenüber kleineren Veränderungen von Lichtverhältnissen im Bild, werden mehrere Pixel in Zellen und mehrere Zellen wiederum in Blöcke unterteilt, für die Histogramme erzeugt werden. In der Arbeit von Dalal und Triggs [DT05] hat sich eine Blockgröße von 18×18 bzw. 16×16 Pixeln mit 3×3 bzw. 2×2 enthaltenen Zellen der Größe 6×6 bzw. 8×8 Pixeln als beste Wahl für die Personenerkennung erwiesen.

In Abbildung 3.4 ist zu erkennen, dass die Fehlerrate bei einer Zellengröße von 6×6 Pixeln und einer Blockgröße von 3×3 bei 10,4 % liegt. Das heißt, dass bei der Personenerkennung diese Block- und Zellengröße die besten Ergebnisse liefert.

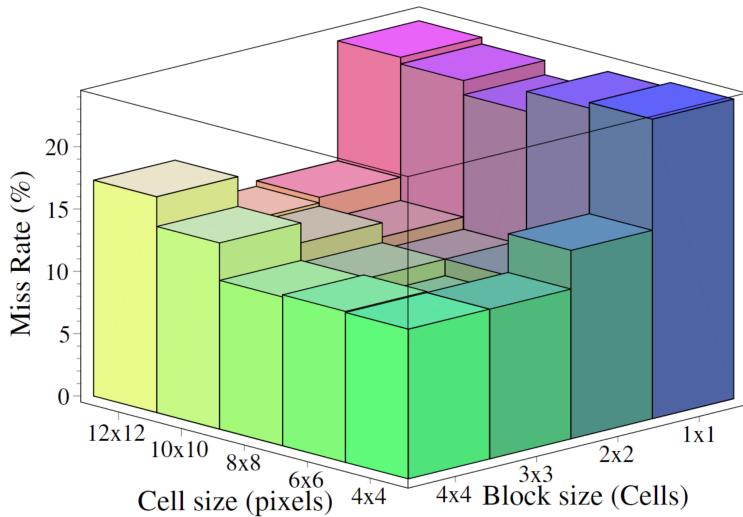


Abbildung 3.4: Überblick über die Fehlerrate bei Zellen- und Blockgrößen Veränderungen [DT05]

Für jede Zelle wird ein Histogramm mit den Gradientenrichtungen von allen enthaltenen Pixeln erstellt, wobei die Richtungen von 0° bis 360° auf eine feste Anzahl an Richtungskanälen abgebildet werden. Zum Beispiel umfasst Richtungskanal 1 den Bereich 0° bis 39° , Kanal 2 40° bis 79° und so weiter.

Support-Vektor-Maschine

Für das Training und die Klassifikation von Detektoren wird eine Support-Vektor-Maschine (SVM) genutzt. Die Idee der SVM geht auf die Arbeit von Vapnik und Chervonenkis [VC74] aus dem Jahr 1974 zurück. Dabei werden so genannte Stützvektoren trainiert. Es wird versucht zwischen verschiedenen Mustern bzw. dem Muster und allem anderen durch die Stützvektoren eine Hyperebene aufzuspannen, die ein Muster von den anderen trennt.

Für die Klassifikation werden die HoG-Features eines Eingangsbildes bzw. eines Ausschnitts als Suchfenster davon erstellt, die als Eingabevektor der SVM übergeben werden. Die SVM berechnet danach, auf Basis ihrer trainierten Parameter, auf welcher Seite von der Hyperebene ausgesehen, sich der Eingabevektor befindet und entscheidet demnach, ob das Bild zur Klasse des Musters gehört oder nicht.

Bei der Gesichtserkennung existieren lediglich zwei Klassen, nämlich eine für das *Enthalten* und eine für das *Fehlen* eines Gesichts im Bild. Dalal und Triggs [DT05] verwenden in ihrem Ansatz eine lineare Support-Vektor-Maschine zur Einteilung bzw. Klassifizierung. Im Falle der linearen Klassifikation mit der SVM liegen die Daten durch ihre Darstellung als Vektor so im Raum, dass sie voneinander linear separierbar sind.

3.2 Erkennung von Gesichtsmerkmalen

Im Unterschied zur Gesichtserkennung wird bei der Merkmalerkennung versucht, Positionen einzelner, bestimmter Punkte eines Gesichts zu bestimmen. Ziel der Merkmalerkennung ist es eine Form S zu finden [CWWS12] [KS14], die dem Gesicht von der Form her am nächsten aussieht. Dazu muss eine Regression erlernt werden, die extrahierte Gesichtsmerkmale auf eine Form abbildet. Dazu kommen Hilfsmittel zum Einsatz, wie die Histogramme orientierter Gradienten (siehe Abschnitt 3.1.2) oder Pixelunterschiede in dem Bereich des zu suchenden Merkmals. Die Form S wird dabei wie folgt aufgestellt (3.5):

$$S = [x_1, y_1, \dots, x_n, y_n] \quad (3.5)$$

Die Anzahl der unterschiedlichen Landmarken ist dabei n .

Das Erkennen läuft in mehreren Regressionsschritten [CWWS12] ab. Dabei werden sogenannte schwache Regressoren ($R^1, \dots, R^t, \dots, R^T$) mit einander additiv kombiniert, um eine möglichst genaue Form des Gesichts zu finden (3.6):

$$S^t = S^{t-1} + R^t(I, S^{t-1}), t = 1, \dots, T \quad (3.6)$$

Der Regressor R^t berechnet anhand eines Bildes I und der vorherigen Form S^{t-1} einen Inkrement, der zur vorherigen Form wiederum addiert wird, um eine aktualisierte Form zu erhalten. Dies geschieht so lange bis alle T Regressoren benutzt wurden. Die Regressoren selbst werden gelernt, um den Unterschied zwischen vorhergesagten und der tatsächlichen Form (Alignment-Fehler) zu minimieren. Für den Regressor R^t ergibt sich nachfolgende Gleichung (3.7):

$$R^t = \operatorname{argmin}_R \sum_{i=1}^N \|\widehat{S}_i - (S_i^{t-1} + R(I_i, S_i^{t-1}))\| \quad (3.7)$$

Dabei ist \widehat{S}_i die wirkliche Form des Gesichts an der Stelle I_i , wobei i der Index ist, und S_i^{t-1} die im Schritt davor geschätzte Form des Gesichts [CWWS12]. Viele heutige Implementierungen bauen auf diesen Ansatz auf, wobei diese jedoch um einen Regression-Baumes erweitert werden.

Die in Abbildung 3.5 dargestellte Silhouette zeigt ein Beispiel der Landmarken eines Detektors mit 68 Punkten. Die Silhouette enthält neben dem Kiefer noch Merkmale wie Mund, Augen, Augenbrauen und Nase. So kommt dieser auch in der Dlib Bibliothek (siehe Abschnitt 5.1.2) vor und wird im entwickelten Algorithmus verwendet, um die Regionen des linken und rechten Auges ausfindig zu machen.

Laut Kazemi und Sullivan [KS14] liegt die durchschnittliche Fehlerrate der Merkmalerkennung bei gerade mal 5 %. Die Fehlerrate gibt an, um wieviel Prozent das erkannte Landmark vom echten Landmark entfernt ist. Bei dem sehr häufig genutzten Datensatz LFW (Labeled Face Parts in the Wild) zur Merkmalerkennung erzielte es gerade mal eine Fehlerrate von 3,4 % und übertrifft damit die meisten anderen Merkmalerkennungsalgorithmen. Zu erwähnen ist, dass die Erkennung ebenfalls nur einen Bruchteil vorheriger Algorithmen dauert.

Dies kommt den Zielen des zu entwickelten Algorithmus für die Lidschlagerkennung zu gute. Dort spielt die Genauigkeit der Position der Augen eine große Rolle um die Augenseitenverhältnisse (siehe Abschnitt 4.1) korrekt berechnen zu können. Auch die Schnelligkeit des Verfahrens unterstützt, um in Echtzeit Daten zu verarbeiten.

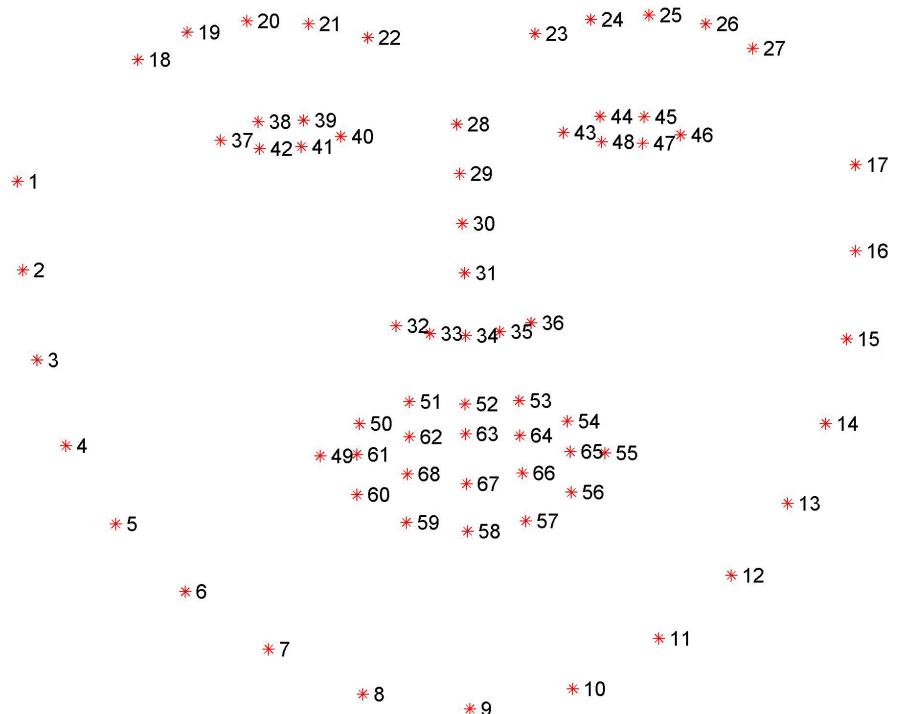


Abbildung 3.5: Nummerierung und Positionen der Landmarken [Gro17]

3.3 Analyse zweier Gesichtserkennungsverfahren

Der Ansatz von Dalal und Triggs, der in Abschnitt 3.1.2 kurz erläutert wird und auch bei der Gesichtserkennung in der Bibliothek Dlib zum Einsatz kommt, soll hier dem Ansatz von Viola und Jones [VJ04], wie er in der Bibliothek OpenCV (siehe Abschnitt 5.1.1) vorkommt, gegenübergestellt werden.

Christ hat in seiner Arbeit [Chr16] von 2016 eine Recherche, Anwendung und Evaluierung verschiedener Verfahren zur Detektion von Gesichtern erstellt. Dabei wurden auch die Ansätze von Dalal und Triggs sowie von Viola und Jones evaluiert.

Seine Ergebnisse sind in Abbildung 3.6 dargestellt. Es wurde ein Datensatz von Fußballmannschaften mit 1678 Gesichtern in verschiedenen Einstellungen erstellt und mit zwei verschiedenen Suchfenstergrößen, einmal in 80x80 Pixeln und einmal in 40x40 Pixeln untersucht. Die Graphen zeigen auf der x-Achse die prozentuale Falsch-Positiv-Rate und auf der y-Achse die prozentuale Erkennungsrate. Dabei wurde die Bibliothek OpenCV mit verschiedenen Skalierungsfaktoren getestet, da dies als eine zusätzliche Konfiguration bereitgestellt wurde. Ein kleinerer Skalierungsfaktor bedeutet eine erhöhte Anzahl an zu untersuchenden Bildausschnitten was zu einer geringeren Erkennungsrate führt. Dabei ist Christ zu den nachfolgenden Ergebnissen gekommen.

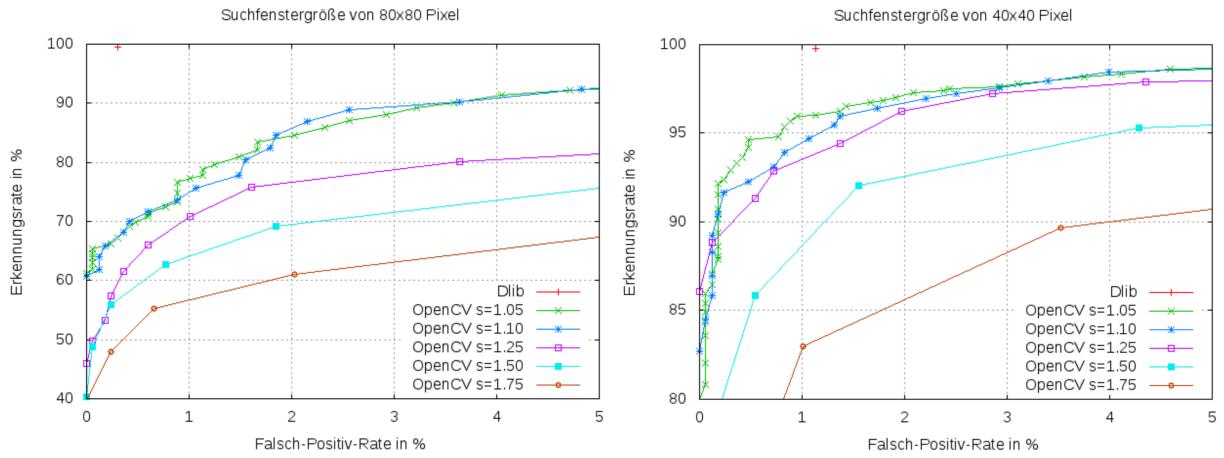


Abbildung 3.6: Links: Erkennungs- und Falsch-Positiv-Raten mit einem 80x80 px Suchfenster, Rechts: Erkennungs- und Falsch-Positiv-Raten mit einem 40x40 px Suchfenster [Chr16, S. 70-71]

Erkennungsrate

Bei beiden Bibliotheken sinkt die Erkennungsrate mit größerem Suchfenster, jedoch ist der Unterschied bei Dlib weniger deutlich als bei OpenCV. Hier hat Dlib eine Erkennungsrate von 99,46 % (1669 von 1678 erkannten Gesichtern) und bei einem Suchfenster von 40x40 px 99,74 % (1674 von 1678 erkannten Gesichtern). Werden die Erkennungsraten von OpenCV bei einer Mindesterkennung von 1 betrachtet, erreicht OpenCV eine Erkennungsrate von 98,63 % bei 80x80 px und 99,76 % bei 40x40 px mit einem Skalierungsfaktor von 1,05. Jedoch muss hier eine Falsch-Positiv-Rate von 43,98 % in Kauf genommen werden. Dlib kommt dabei nur auf 1,13 %.

Falsch-Positiv-Rate

Wird bei einer gleichen Falsch-Positiv-Rate beide Verfahren getestet, erlangt Dlib eine bessere Erkennungsrate. OpenCV erreicht hier eine Erkennungsrate von 96,01 % und somit 3,75 % weniger als Dlib. Den Vorteil erlangt Dlib bzw. der Ansatz von Dalal und Triggs wegen der Aktualität und die damit verbundenen Optimierungen des Verfahrens.

Beurteilung

Angesichts dieses Ergebnisses ist die Bibliothek Dlib mit ihrem Ansatz zur Erkennung von Gesichtern für den Algorithmus zur videobasierten Blinzelerkennung besser geeignet. Einmal ist dort die Erkennungsrate höher, was bedeutet das mehr Gesichter gefunden werden und die Falsch-Positiv-Rate ist sehr viel geringer und dadurch werden viel weniger falsch erkannte Gesichter erkannt.



4 Entwurf des Algorithmus zur Lidschlagerkennung

Das Ziel des Entwurfs ist es, dass der Algorithmus in Echtzeit operiert und die vorhandenen Ressourcen optimal nutzt. Zudem muss er robust zur Lidschlagerkennung sein und trotzdem die Genauigkeit besitzen einen Lidschlag korrekt zu erkennen. Der entworfene Algorithmus soll dabei als Bibliothek zur Verfügung gestellt werden, sodass er wiederum vielfach eingesetzt werden kann, z. B. bei der Erstellung eines Endanwender-Programms in einer an diese Arbeit anschließende Arbeit.

Der Entwurf lässt sich in drei große Komponenten einteilen. In Abbildung 4.1 wird gezeigt, in welche Bereiche diese unterteilt werden. Die erste Komponente bildet die Aufnahme der Bilddaten und die Methoden zum Vorverarbeiten der Daten. Diese Daten werden der zweiten Komponente zur Verfügung gestellt, die versucht in diesen Daten Gesichter zu erkennen, damit die dritte und letzte Komponente die zur Entscheidungsfindung erforderlichen Informationen aus den Daten extrahieren und auswerten kann. Somit bilden alle drei Komponenten einen Kreislauf und sind voneinander abhängig.

Dabei sollen die Schnittstellen so ausgelegt werden, dass jede Komponente problemlos ausgetauscht werden kann, ohne dass eine andere Komponente davon Einfluss nimmt.

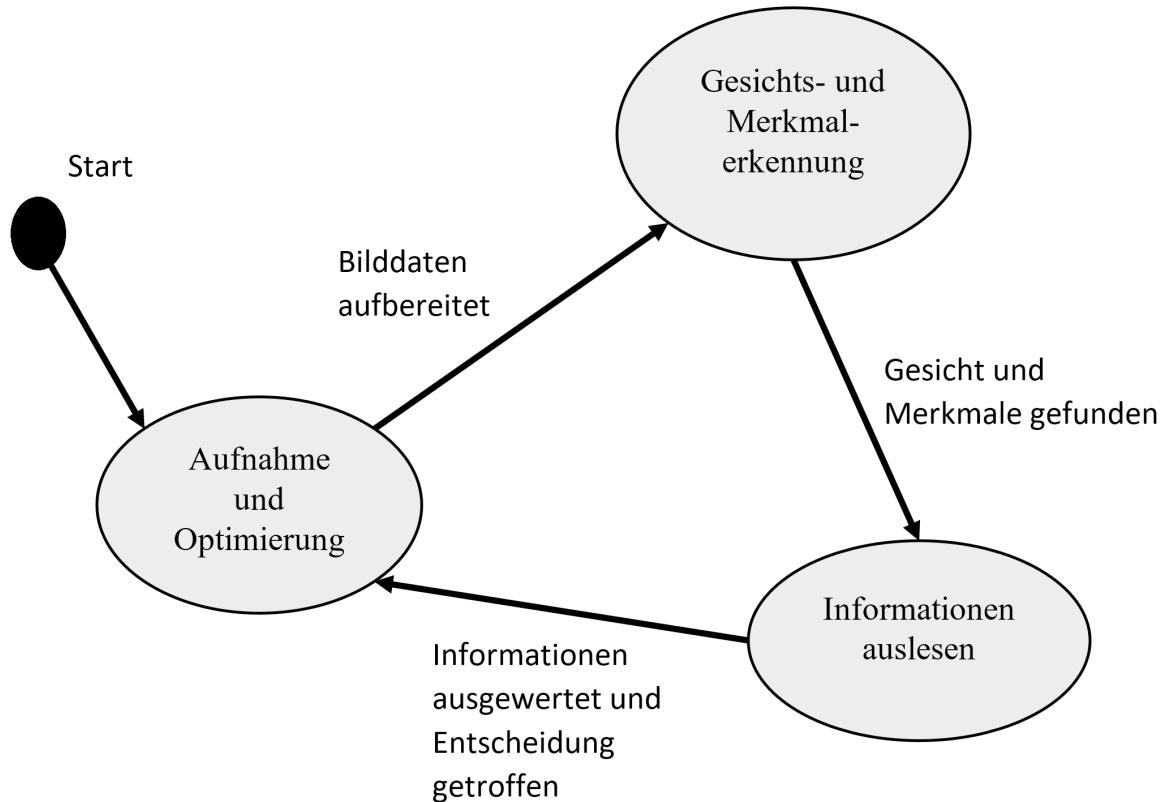


Abbildung 4.1: Einteilung des Algorithmus in die einzelnen Komponenten

Der Daten- und Kontrollfluss für diesen Entwurf ist in Abbildung 4.2 dargestellt. Der Datenstrom beginnt mit der Übergabe des Datenstroms. Dabei ist der Datenstrom eine Menge M (4.1):

$$M := \{P_0(t_0), P_1(t_0 + T), \dots, P_{n-1}(t_0 + (n-1) \cdot T), P_n(t_0 + n \cdot T)\} \quad (4.1)$$

von Bildern, die in äquidistanten Zeitabständen T vorliegen. Die Anzahl n der Bilder in dem Datenstrom ist bei einem aufgezeichneten Video begrenzt, bei der Echtzeitverarbeitung jedoch undefiniert.

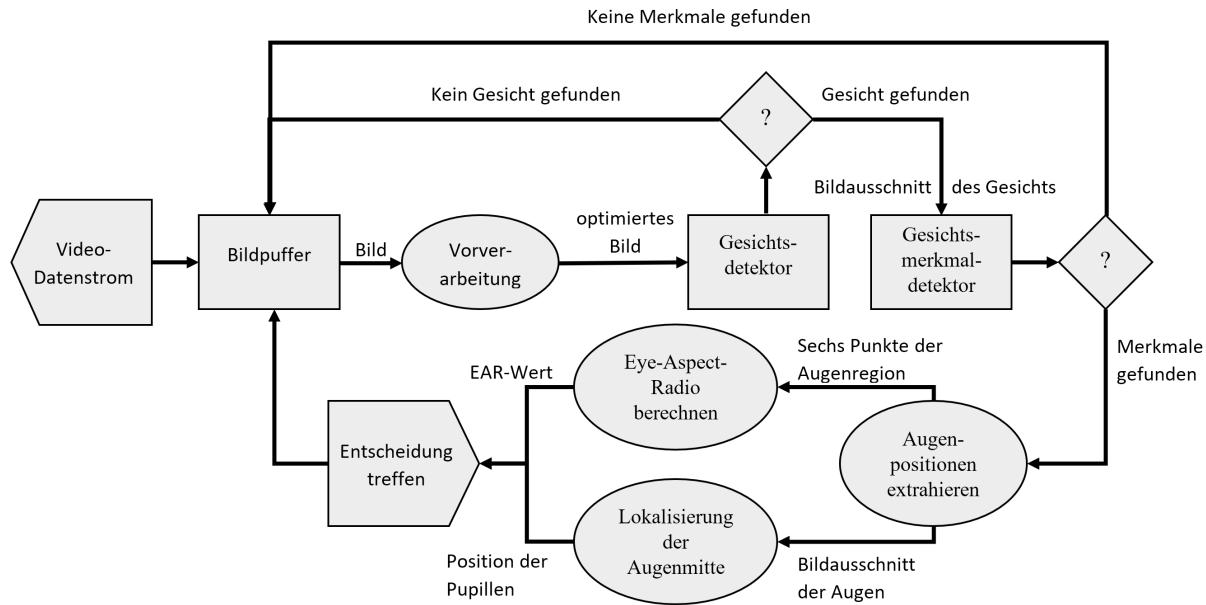


Abbildung 4.2: Daten- und Kontrollfluss des Algorithmus

Die einzelnen Videobilder P repräsentieren ein digitales Farbbild und sind definiert als ein 3-Tupel (4.2):

$$P(t) := (R, G, B) \quad (4.2)$$

Jedes Element des Tupels aus (4.2) stellt eine Matrix der Größe $M \times N$ dar, die die digitalen Bilddaten für die einzelnen Grundfarben Rot, Grün und Blau enthalten.

Der Bildpuffer dient zum Zwischenspeichern der einzelnen Videobilder aus dem Videostrom. Die Videobilder, die als Farbbilder im RGB-Farbraum vorliegen, werden zunächst vorverarbeitet. Dabei werden diese zuerst auf eine reduzierte Größe verkleinert. Dies hat zum Vorteil, dass dadurch Bearbeitungszeit eingespart werden kann. Das an Größe reduzierte Bild wird dann in ein Graustufenbild umgewandelt. Ist dies geschehen, wird ein Histogrammausgleich ausgeführt, um den Kontrast des Graustufenbildes (siehe Abschnitt 2.1.1) zu verbessern.

An dieser Stelle endet die erste Komponente aus Abbildung 4.1 und die optimierten Bilddaten werden dem Detektor zur Erkennung von Gesichtern (siehe Abschnitt 3.1) übergeben. Wird ein Gesicht ermittelt, wird der Ausschnitt mit den Daten an den Gesichtsmerkmalendetktor (siehe Abschnitt 3.2) gegeben. Dieser versucht in dem Teilausschnitt die x - und y -Koordinaten von 68 Landmarken zu finden. Die Punkte beschränken sich dabei auf Stellen im Gesicht, wie Mund, Nase, Kiefer, Augenbrauen und Augen (vgl. Abbildung 3.5).



Abbildung 4.3: Nummerierung und Positionen der Augen-Landmarken [Gro17]

Mit Hilfe der Positionen der Punkte 37 bis 48 (Abbildung 4.3) kann so die Region der Augen aus dem Bild extrahiert werden. Mit diesem Wissen kann der Eye-Aspect-Ratio-Wert, welcher im Abschnitt 4.1

genauer beschrieben wird, berechnet werden.

Des Weiteren wird in der Region des linken und des rechten Auges mit Hilfe der Gradientenberechnung in Abschnitt 4.2 die genaue Position der Pupillen des jeweiligen Auges ermittelt. Diese Informationen tragen zur Entscheidung bei, ob ein Lidschlag vollzogen wurde oder nicht.

An dieser Stelle endet die zweite Komponente aus Abbildung 4.1 und die gewonnene Information über den Lidschlag wird an Folgeprogramme übergeben. Nach der Informationsverarbeitung wird aus dem Puffer ein neues Bild entnommen und der sequentiellen Verarbeitung erneut übergeben.

4.1 Augenseitenverhältnis

Anhand der Arbeit von Soukupová et al. [SC16] kann das Augenseitenverhältnis (Eye-Aspect-Ratio) *EAR* (4.3) zur Bestimmung eines Lidschlages verwendet werden, welches zur Schätzung des Augenöffnungszustandes dient.

Der *EAR* wird als ein Verhältnis zwischen Höhe und Weite eines Auges, wobei p_1 bis p_6 die Landmarken sind, berechnet:

$$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|} \quad (4.3)$$

Die Landmarken p_1 bis p_6 entsprechen beim linken Auge den Erkennungspunkten 37 bis 42 und beim rechten Auge den Erkennungspunkten 43 bis 48 (vgl. Abbildung 4.3). Laut Soukupová et al. [SC16] ist der *EAR* die meiste Zeit über konstant über eine Wert von 0,25 bei offenem Auge und geht gegen den Wert 0 wenn sich dieses schließt. Abbildung 4.4 zeigt einen Ausschnitt von *EAR*-Werten über eine kurze Zeitspanne, in der ein einzelner Lidschlag zu erkennen ist. Die Methode ist teilweise Person und Kopfhaltung unempfindlich.

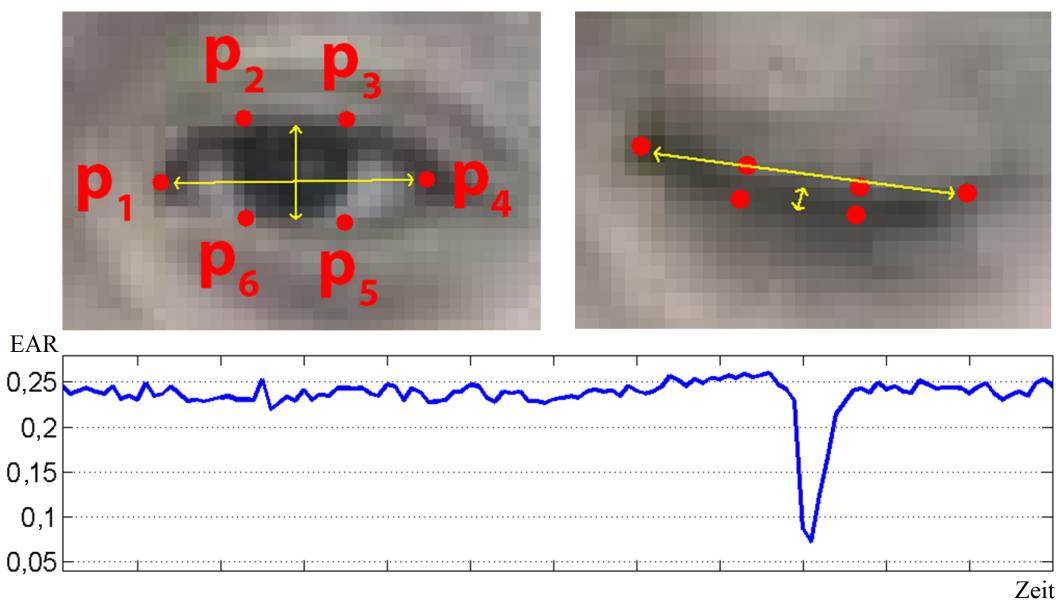


Abbildung 4.4: Oben: Offenes und geschlossenes Auge mit Landmarken, Unten: Auszug der *EAR*-Werte über eine Sequenz von Bildern mit einem einzelnen Lidschlag nach [SC16]

Dabei ist festzustellen, dass der Wert bei geöffnetem Zustand relativ konstant bei ungefähr 0,25 liegt und bei einem geschlossenen Zustand gegen 0 geht. Mit Hilfe eines Schwellenwerts *SW* kann nun über eine gewisse Sequenz von Bildern bestimmt werden, ob ein Blinzeln vorliegt oder nicht. Soukupová et al. [SC16] kam bei eigenen Tests auf einen Schwellenwert von *SW* = 0,2 was in der unteren Abbildung

von 4.5 zu erkennen ist.

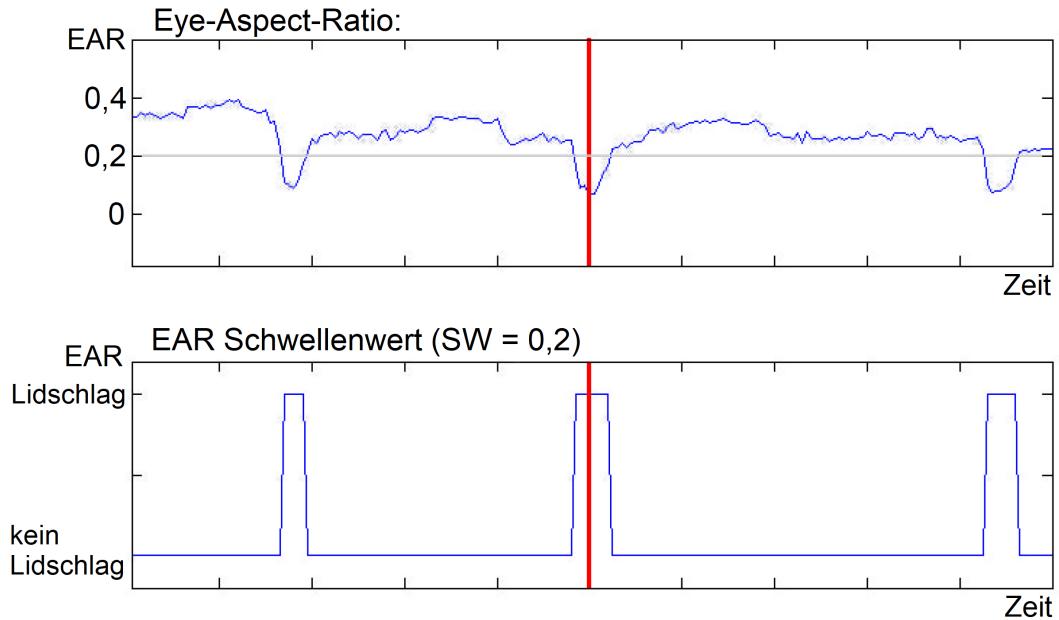


Abbildung 4.5: Oben: Drei erkannte Lidschläge, Unten: Ergebnis des gefundenen Schwellenwertes ($SW = 0,2$) nach [SC16]

Die Abbildung 4.6 zeigt einen Ausschnitt über eine Zeitspanne von 50 Sekunden eines vom Autor dieser Arbeit aufgebauten Testprogramms (siehe Kapitel 6) für den Entwurf des Algorithmus. In der Abbildung liegt der EAR-Wert relativ konstant über 0,2 bis 0,4 und bei einem Lidschlag sinkt dieser meist deutlich unter 0,15. In dieser Arbeit wurde daher ein Schwellenwert von $SW = 0,18$ gewählt. Somit wird gewährleistet, dass es nicht zu einer Fehlererkennung kommen kann.

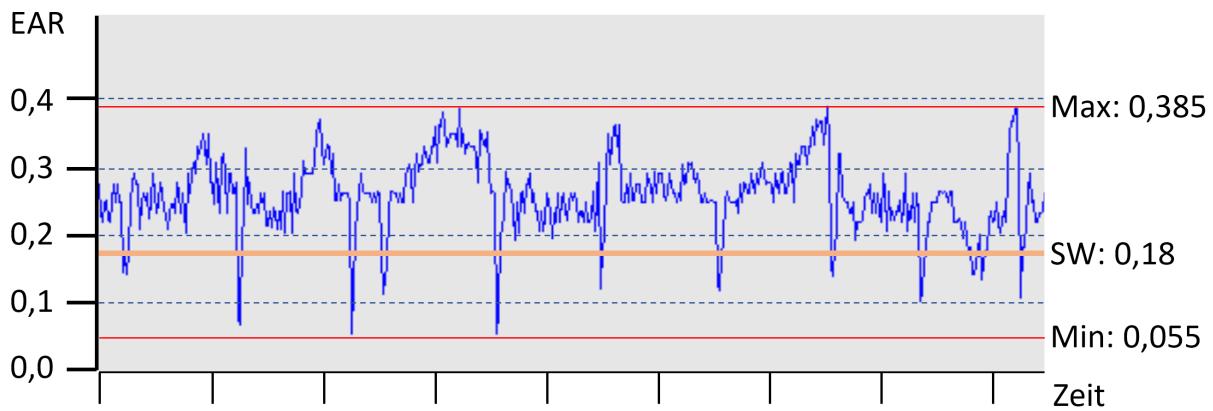


Abbildung 4.6: Auswertung der EAR-Werte bei einem Testlauf des Entwurfs zum Ermitteln des Schwellenwertes, Ergebnis: $SW=0,18$

Das Augenseitenverhältnis wurde in den Entwurf des Algorithmus integriert, da anhand der Gleichung des EAR-Werts (4.3) eine schnelle Berechnung garantiert ist, was in der Evaluation des Algorithmus in Kapitel 6 auch bestätigt werden konnte. Die Lidschlagerkennung kann somit anhand eines einzelnen Werts, dem Schwellenwert, berechnet werden, was eine effiziente Informationsverarbeitung ermöglicht und wenig Rechenleistung benötigt.

4.2 Lokalisierung der Augenmitte

Timm und Barth stellen in Ihrer Arbeit [TB11] von 2011 einen neuartigen Ansatz zur Erkennung von Pupillen bei Verwendung von Gradienten vor. Dazu analysieren Sie das Vektorfeld von Bildgradienten und leiten darüber eine mathematische Formulierung der Vektorfeldcharakteristik ab. Die Formulierung beschreibt dabei die mathematische Beziehung zwischen einem möglichen Zentrum und den Orientierungen aller Bildgradienten.

Abbildung 4.7 zeigt ein mögliches Zentrum c und den Gradientenvektor g_i an der Position x_i . Hier sollte der Verschiebungsvektor d_i die gleiche Orientierung wie der Gradientenvektor g_i haben.

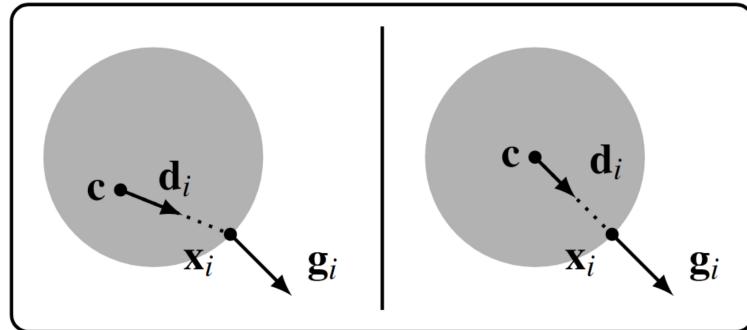


Abbildung 4.7: Künstliches Beispiel mit einem dunklen Kreis auf einem hellen Hintergrund, ähnlich wie die Iris und die Sklera, Links: Der Verschiebungsvektor d_i und der Gradientenvektor g_i haben nicht die gleiche Orientierung, Rechts: Beide Orientierungen sind gleich [TB11]

Der optimale Mittelpunkt c^* (4.4) eines kreisförmigen Objekts in einem Bild wird wie folgt definiert:

$$c^* = \operatorname{argmax}_c \left\{ \frac{1}{N} \sum_{i=1}^N (d_i^T g_i)^2 \right\} \quad (4.4)$$

$$d_i = \frac{x_i - c}{\|x_i - c\|_2}, \forall i : \|g_i\|_2 = 1 \quad (4.5)$$

Hierzu wird das Vektorfeld der Gradienten verwendet, indem die Produktprodukte zwischen den normalisierten Verschiebungsvektoren und dem Gradientenvektor berechnet werden. Die Verschiebungsvektoren d_i (4.5) werden für ein gleiches Gewicht über alle Pixelpositionen auf Einheitslänge skaliert. Gleichermaßen gilt für die Gradientenvektoren g_i , um dort robuster gegenüber Änderungen in Beleuchtung und Kontrast zu sein.

Das Ergebnis solch einer Berechnung kann aus der Abbildung 4.8 entnommen werden.

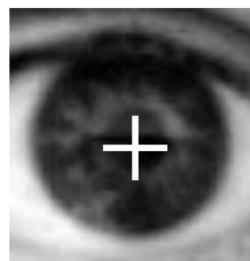


Abbildung 4.8: Ergebnis nach der Berechnung der Augenmitte [TB11]

Während des Algorithmen-Entwurfs hat sich herausgestellt, dass die Region, über die die Gradienten berechnet werden, eine Breite von 50 Pixeln haben muss. Liegt der Wert darunter sind es zu wenige Informationen, um eine genaue Bestimmung zu machen. Werden mehr Werte hinzugenommen, langt die Zeit zum Berechnen nicht aus um diese in Echtzeit durchzuführen.

Die Pupillenextraktion wurde zum Entwurf hinzugenommen um eine Möglichkeit haben zu können, eine Verfolgung der Pupillen zu ermöglichen. Einerseits ist die Berechnung sehr schnell und anderseits liefert diese eine sehr gute Trefferrate. Damit könnten

Die Pupillenextraktion stellt eine Erweiterung des Algorithmus zur Lidschlagerkennung dar, welcher für die Selbstdiagnose von Parkinson-Patienten primär nicht benötigt wird. Allerdings kann die Pupillenextraktion modular zum Programmablauf hinzugefügt werden und ermöglicht so die Anwendung des Algorithmus auch zur Pupillenverfolgung. Damit könnten Programme nur mit den Augen bedient werden. Der Nutzer kann so nur über die Blickrichtung den Mauszeiger auf dem Bildschirm bewegen und Felder oder Funktionen auswählen. Die Auswahl von Feldern, bzw. der Mausklick erfolgt dann über ein gezieltes Blinzen. Die Blinzelzeit könnte dabei individuell in Millisekunden eingestellt werden.

4.3 Entscheidungsfindung eines Lidschlags

Bei der Entscheidungsfindung geht es darum, zu entscheiden ob es sich um einen Lidschlag handelt oder nicht. Dafür werden Informationen über den EAR-Wert in einer vorher gewählten Anzahl von Bildern benötigt, um eine sichere Aussage treffen zu können.

In Abbildung 4.9 ist zu erkennen, dass innerhalb einer Sequenz von drei Bildern das Auge den Zustand von *offen* zu *geschlossen* wechselt. Die Aufnahme der Kamera erfolgt mit 20 Bildern pro Sekunde, was zu einer Bildzykluszeit T von 50 ms (4.6) führt:

$$T = \frac{1}{20 \frac{\text{Bilder}}{\text{Sekunde}}} = 50 \text{ ms} \quad (4.6)$$

Das bedeutet das der Zustandswechsel der Augenlider im vorliegenden Fall in ungefähr 150 ms passiert ist und durch die Messung von drei Bildern à 50 ms bestätigt wurde. Dafür muss der Algorithmus immer mindestens zusätzlich zwei Bilder vor dem aktuellen mit in die Betrachtung einbeziehen.

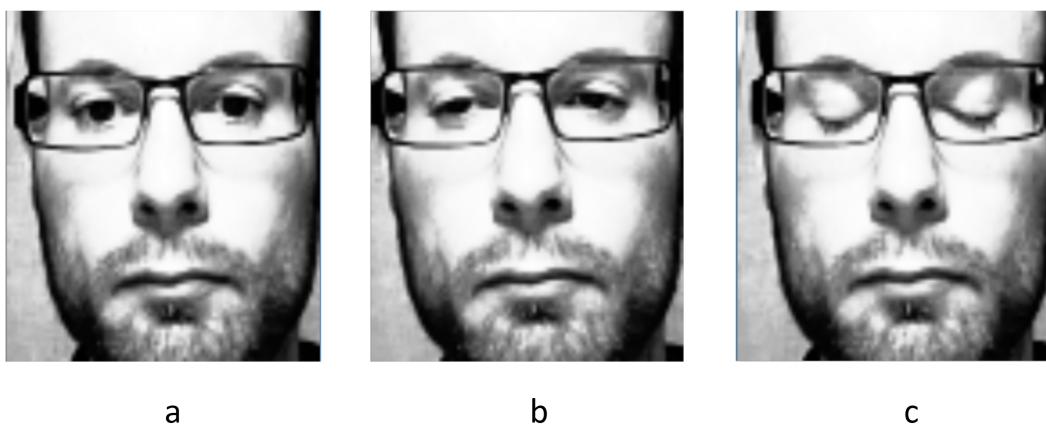


Abbildung 4.9: Lidschlag innerhalb von drei Bildern: (a) offen (50 ms), (b) am schließen (100 ms), (c) geschlossen (150 ms)

Somit wird deutlich, dass zum Treffen einer Entscheidung mindestens 3 Bilder in Betracht gezogen werden müssen. Das erste Bild wäre, wenn die Augen halb geschlossen, das zweite, wenn die Augen geschlossen und das dritte, wenn die Augen wieder halb offen sind.

Das bedeutet, dass der EAR-Wert innerhalb dieser Sequenz unter den definierten Schwellenwert liegen muss. Ist dies der Fall, fällt die Entscheidung dahingegen aus, dass es sich um einen Lidschlag gehandelt hat.

Um vorzubeugen das ein geschlossen sein der Augen über einen längeren Zeitraum, also wenn der Betrachter die Augen zum Schlafen schließt, als ein Dauerblinzeln erkannt wird, wurde ein Maximalwert von drei definiert der diesen Fall abfängt. Sind die Augen länger als drei Bilder in Folge geschlossen wird kein Lidschlag mitgezählt.



5 Implementierung des Entwurfs

In diesem Kapitel werden die Details des Entwurfs beschrieben, welcher in Kapitel 4 vorgestellt wurden. Des Weiteren werden die Softwarebibliotheken erläutert, die zur Erstellung und Evaluation des in der Arbeit erstellten Algorithmus genutzt wurden.

5.1 Softwarebibliotheken

Für die Gesichts- sowie Gesichtsmerkmalerkennung wurden die Open-Source-Softwarebibliotheken OpenCV [Ope17a] und Dlib [Dli17a] verwendet, die dazu verschiedene Herangehensweisen als Grundlage nutzen. Der Einsatz von Open-Source-Bibliotheken ist damit begründet, dass der entworfene Algorithmus ebenfalls frei zugänglich sein soll.

5.1.1 OpenCV

OpenCV ist eine im Jahre 2000 entstandene frei zugängliche Bibliothek für den Bereich der Computer Vision und Bildbearbeitung. OpenCV stellt generelle Funktionalitäten zur Bildbearbeitung, zum Maschinenlernen und speziell auch zum maschinellen Sehen bereit, was die Erkennung von Gesichtern beinhaltet. Unterstützt werden die Betriebssysteme Microsoft Windows, Linux, MacOS und Android, wobei Schnittstellen für die Programmiersprachen C++, C, Python, Java und MATLAB existieren [Ope17b]. Die Implementation der Gesichtserkennung basiert auf dem Ansatz von Viola und Jones aus dem Jahre 2001 [VJ01], da der Fokus auf Algorithmen liegt, die Erkennung in Echtzeit leisten können [Ope17c]. Die Bibliothek weist dabei über 2500 optimierte Algorithmen auf, die sowohl klassische als auch State-of-the-Art-Algorithmen umfassen [Ope17b].

Durch die modular gehaltene Struktur von OpenCV müssen nur die Teile der Bibliothek geladen werden, die auch gebraucht werden. Wichtige Module neben dem Kernmodul, sind unter anderem diese Module:

- **imgproc** – für Aktionen rund um die Bildbearbeitung
- **imgcodecs** – zum Laden und Abspeichern von Bildern
- **objdetect** – zur Detektion von Objekten mithilfe von Kaskaden

Des Weiteren ist zu erwähnen, dass OpenCV die BSD (Berkeley Software Distribution) Lizenzierung nutzt, wodurch sich diese Bibliothek sowohl für die gewerbliche, als auch nicht gewerbliche Nutzung eignet [Ope17d]. Im Verfahren wird die aktuelle Version 3.3.0 benutzt.

5.1.2 Dlib

Dlib ist eine Universal-Bibliothek, die für verschiedenste Zwecke eingesetzt werden kann. Dabei zeichnet sich diese vor allem durch die Implementierung von relativ neuen Algorithmen zur Computer Vision aus. Darunter zählen u. a. Data Mining, Netzwerktechnik, Maschinenlernen und Bildverarbeitung, die auch eine Methodik zur Gesichtserkennung enthält. Die Softwarebibliothek ist in C++ geschrieben und wurde auf den Betriebssystemen Mac OSX, Microsoft Windows, Linux, Solaris, Berkeley Software Distributions (BSD), sowie Hewlett Packard Unix (HP-UX) bereits erfolgreich genutzt [Dli17d].

Zur Erkennung von Gesichtern wird das Histogramm orientierter Gradienten mit einer Support-Vektor-Maschine von Dalal und Triggs (siehe Abschnitt 3.1.2) aus 2005 [DT05] verwendet und wurde durch deformierbare Teil-Modelle von Felzenszwalb et al. aus 2010 [FGMR10] weiter verbessert [Dli17b] [Dli17c].

Die Implementierung der Gesichtsmerkmalerkennung von Dlib baut auf den Ansatz von Kazemi und Sullivan (siehe Abschnitt 3.2) aus dem Jahr 2014 [KS14] auf und ermittelt anhand eines Regression-Baumes die Landmarken [Dli17f].

Ein wichtiges Modul neben dem Kernmodul, ist unter anderem dieses Modul:

- **imageprocessing** – für Aktionen rund um die Bildbearbeitung und zur Detektion von Objekten

Durch die Open-Source-Lizenzierung eignet sich Dlib sowohl für den privaten als auch gewerblichen Gebrauch [Dli17e]. Im Verfahren wird die aktuelle Version 19.7 benutzt.

5.2 Hardwarekomponenten und Entwicklungsumgebung

Für die Entwicklung steht ein Microsoft Surface Pro 4 von der Firma Microsoft Corporation zur Verfügung. Dieses ist mit einem Intel(R) Core(TM) i5-6300U @ 2,4 GHz Prozessor, 8 GB - DDR3 Speicher und einer Intel(R) HD Graphics 520 von der Intel Corporation ausgestattet. Entwickelt wird mit Microsoft Visual Studio Enterprise 2015 Update 3, welches unter Windows 10 Pro in der Version 1709 läuft.

Die Programmiersprache ist die objektorientierte Sprache C++ in der Version 14, C++ ermöglicht sowohl die effiziente und maschinennahe Programmierung als auch eine Programmierung auf hohem Abstraktionsniveau.

Die Aufnahme geschieht über die eingebaute Frontkamera des Microsoft Surface und nimmt mit 20 Bildern die Sekunde auf. Die Auflösung beträgt 5-Megapixel, das Seitenverhältnis ist 16:9 und sie zeichnet Videos in HD-Qualität auf.

5.3 Klassen

Der Entwurf besteht größtenteils aus denen in der Abbildung 5.1 angezeigten Klassen. Die Klassen *Detector*, *Face* und *Helpers* werden in den nachfolgenden drei Abschnitten genauer beschrieben. Außerdem gibt es noch die Klasse *Constants*. Diese implementiert die Konstanten, die von den anderen Klassen abgerufen werden. Dort können zum Beispiel der Schwellenwert und die Anzahl der Bilder konfiguriert werden, die für die Entscheidungsfindung (siehe Abschnitt 4.3) eine große Rolle spielen. Zusätzlich können Werte geändert werden, die eine schnellere Verarbeitung gewährleisten, wie z. B. die Größe pro Bild um die es reduziert oder ob eine gewisse Anzahl an Bildern übersprungen werden soll. Ein weiterer wichtiger Parameter ist die Matrixgröße für die Gradientenberechnung bei der Suche der Pupillen (siehe Abschnitt 4.2), welcher mit 50 Pixeln vorbelegt ist.

Die in der Abbildung 5.1 angezeigten Methoden und Felder sind nur ein Ausschnitt der wichtigsten Methoden und Felder. Der komplette Programmcode steht in einem Repository, dessen Link im Abschnitt 5.5 steht, bereit.

Detector

Der Detektor ist der Haupteinstiegspunkt und benötigt für die Merkmalerkennung (*den ShapePredictor*) eine Daten-Datei. Dafür wird die von der Dlib bereitgestellte Datei *shape_predictor_68_face_landmarks.dat* übergeben. Wurde eine Instanz vom Detektor erzeugt, muss lediglich die Methode *Detect* aufgerufen werden. Diese benötigt zum Verarbeiten ein einzelnes Bild, muss also ständig mit neuen Bildern versorgt werden. Die Klasse hält zudem Möglichkeiten zum Zugriff auf gefundene Gesichter bereit und gibt die Anzahl aller erkannten Lidschläge zurück.

Der Entwurf der in Kapitel 4 aufgestellt wurde, wird durch die Methode *Detect* implementiert. Die Implementierung wird in Algorithmus 1 dargestellt.

Zuerst wird das Eingabebild in seiner Größe reduziert, vorbelegt ist eine Größe von 320 px und sorgt

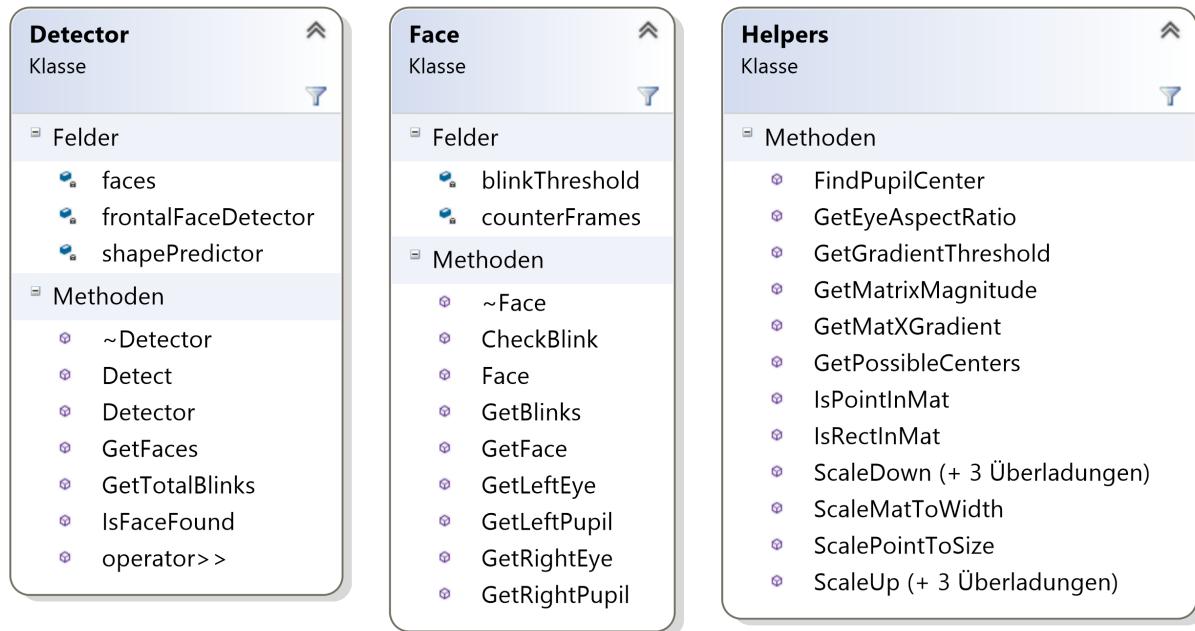


Abbildung 5.1: Links: Die Klasse *Detector* kümmert sich um das Finden von Gesichtern und dessen Verwaltung, Mitte: Die Klasse *Face* bildet ein Gesicht ab und dessen Funktionen, Rechts: Die Klasse *Helpers* bietet Methoden zur Berechnung an

dafür, dass das Bild schneller bearbeitet werden kann. Dann wird das Bild in ein Graustufenbild umgewandelt. Ab jetzt wird nur noch mit einem in der Größe geänderten Graustufenbild fortgefahrene. Zusätzlich wird der Kontrast des Graubildes ausgeglichen, um eine bessere Erkennung zu gewährleisten. Nun wird mit Hilfe des Gesichtsdetektors versucht die enthaltenen Gesichter zu extrahieren. Wurde eins gefunden, wird dies in die Liste der gefundenen Gesichter aufgenommen, falls dort noch keins existiert und die Abmessungen am Gesicht gesetzt. Jetzt kann der Merkmalendetektor mit Hilfe der Abmessungen und dem Graustufenbild nach Merkmalen, die zu diesem gehören, suchen. Sobald diese Aktion abgeschlossen ist, sind die Augenregionen verfügbar, die zum Finden der Pupillen (siehe Abschnitt 4.2) und zum Berechnen der Augenseitenverhältnisse (siehe Abschnitt 4.1) nötig sind.

Face

Die Klasse *Face* bildet ein Gesicht mit seinen Merkmalen und Eigenschaften ab. Als Merkmale werden neben den Augenregionen, die Region für Mund, Nase, Kiefer und Augenbrauen bereitgestellt sowie die Abmessungen für den ganzen Kopf. Die Eigenschaften werden einmal für Dlib und für OpenCV umgesetzt, da beide verschiedene Konstrukte zum Abbilden von Punkten und Rechtecke besitzen.

Im Algorithmus 2 wird die Implementation aus Abschnitt 4.3 exemplarisch dargestellt. Die Methode *CheckBlink* wird für jedes Gesicht pro Bild einmal aufgerufen. Es werden die beiden EAR-Werte (siehe Abschnitt 4.1) mit denen aus der Merkmalerkennung ermittelten Positionen (P_{37} bis P_{48}) anhand der Gleichung berechnet. Liegen die errechneten Werte unter einem der in der Klasse *Constants* festgelegten Schwellenwert, wird die Anzahl der erkannten Bilder um eins erhöht. Ist die Anzahl der Bilder mit einem Wert gleich, der ebenfalls in der Klasse *Constants* vorher definiert wurde, wird die Anzahl der Lidschläge um eins erhöht. Das heißt, dass ein Lidschlag erkannt wurde. Liegen die errechneten EAR-Werte über den Schwellenwert, wird die Anzahl der Bilder wieder auf den Wert 0 zurückgesetzt.

Algorithm 1 Detect Methode

```
1: function DETECT(frame)                                ▷ Methode zum Erkennen eines Lidschlags
2:   Resize(frame, size)                                 ▷ Bild auf eine Breite von 320 Pixeln verkleinern
3:   Gray(frame)                                         ▷ Bild in ein Graustufenbild transformieren
4:   Equalize(frame)                                    ▷ Helligkeit und Kontrast des Bildes anpassen
5:   foundFaces ← FaceDetector(frame)                  ▷ Gefundene Gesichter als Rechtecke
6:   for i ← 0; i < foundFaces.size(); i++ do
7:     foundFace ← foundFaces.at(i)
8:     if faces.size <= i then
9:       faces.pushback(newFace)
10:    end if
11:    face ← faces.at(i)
12:    face.SetFace(foundface)                         ▷ Rechteck vom Gesicht setzen
13:   end for
14:   for i ← foundFaces.size(); i < faces.size(); i++ do ▷ Laufe nicht gefundene Gesichter durch
15:     face = faces.at(i)
16:     face.Reset()                                  ▷ Werte am Gesicht zurücksetzen
17:   end for
18:   for all faces do                                ▷ Laufe alle Gesichter durch
19:     if face.HasFace() then
20:       shape ← ShapePredictor(frame, face.GetFace()) ▷ Merkmale fürs Gesicht setzen
21:       face.SetShape(shape)
22:       face.SetLeftPupil(Helpers.FindPupilCenter(frame, face.GetLeftEye())) ▷ Pupillen setzen
23:       face.SetRightPupil(Helpers.FindPupilCenter(frame, face.GetRightEye()))
24:       face.CheckBlink()                            ▷ Auf Lidschlag prüfen
25:     end if
26:   end for
27: end function
```

Algorithm 2 CheckBlink Methode

```
1: function CHECKBLINK(frame)                           ▷ Methode zum Entscheiden über einen Lidschlag
2:   leftEar ← Helpers::GetEyeAspectRatio(GetLeftEye()) ▷ EAR-Werte berechnen
3:   rightEar ← Helpers::GetEyeAspectRatio(GetRightEye())
4:   if leftEar <= EAR_BLINK_THRESHOLD and rightEar <= EAR_BLINK_THRESHOLD then
5:     counterFrames++                                ▷ Anzahl der Bilder unter dem Schwellenwert mit zählen
6:   else
7:     counterFrames ← 0
8:   end if
9:   if counterFrames == MAX_EAR_FRAMES then
10:    blinks++                                     ▷ Blinzeln hoch zählen
11:   end if
12: end function
```

Helpers

Die Klasse *Helpers* bietet statische Methoden, Methoden die ohne eine Instanz der Klasse aufgerufen werden können, zur Berechnung oder zum Prüfen, wie z. B. ob ein Rechteck in einem Bildausschnitt liegt, an. Eine interessante Methode ist die zum Finden der Pupillen, was die Umsetzung des im Abschnitt 4.2 angesprochenen Verfahrens ist. Der Algorithmus 3 liefert dazu einen Einblick.

Zuerst wird aus den Bilddaten die Region, welche von Interesse ist, extrahiert. Danach wird der Ausschnitt auf 50 px skaliert. Dieser Wert hat sich als äußerst robust herausgestellt. Alles kleiner als 50 px ist zu klein und es werden keine guten Werte berechnet und alles größer als diesen Wert verlangsamt die Berechnung drastisch. Jetzt können die Gradienten-Matrizen x und y berechnet und aus denen im darauf folgenden Schritt die Magnituden ermittelt werden. Mit Hilfe einer mittleren Standardabweichung die sich aus den Magnituden ergibt, können die Matrizen x und y normalisiert werden. Anschließend wird nach möglichen Mitten für jeden Gradienten gesucht. Abschließend wird aus den gefundenen Mitten diejenige ermittelt, welche den maximalsten Schwarzwert besitzt. Von diesem Punkt wird angenommen, dass dieser die Mitte der Pupille am ehesten trifft. Da die Pupille ein kleiner schwarzer Kreis innerhalb der Iris ist, befindet sich der Punkt innerhalb dieses Kreises.

Algorithm 3 FindPupilCenter Methode

```
1: function FINDPUPILCENTER(frameInGray, eyeRect)                                ▷ Methode zum Finden einer Pupille
2:   eyeRoi ← frameInGray(eyeRect)                                              ▷ Bildbereich des Auges extrahieren
3:   Helpers::ScaleMatToWidth(eyeRoi, MAT_WIDTH)                                 ▷ Bildbereich auf passende Größe ändern
4:   gradientX ← Helpers::GetMatXGradient(eyeRoi)                               ▷ Den Gradienten für X berechnen
5:   gradientY ← Helpers::GetMatXGradient(eyeRoi.t()).t()                      ▷ Den Gradienten für Y berechnen
6:   mags ← Helpers::GetMatrixMagnitude(gradientX, gradientY)                ▷ Alle Magnituden berechnen
7:   gradientThresh ← Helpers::GetGradientThreshold(mags, THRESHOLD)          ▷ Standardabweichung berechnen
8:   Normalize(mags, gradientThresh, gradientX, gradientY)                     ▷ Gradienten normalisieren
9:   weight ← BlurredAndInvertedImage(eyeRoi)                                    ▷ Gewichtung berechnen
10:  outSum ← Mat.zero()
11:  for i ← 0; i < weight.rows; ++i do
12:    xR ← gradientX.at(i)
13:    yR ← gradientY.at(i)
14:    for j ← 0; j < weight.cols; ++j do
15:      gX ← xR.at(j)
16:      gY ← yR.at(j)
17:      if gX == 0 and gY == 0 then
18:        continue
19:      end if
20:      outSum += Helpers::GetPossibleCenters(i, j, weight, gX, gY)           ▷ Mitten finden
21:    end for
22:  end for
23:  maxPoint ← MinMax(outSum)                                                 ▷ Mitte mit dunkelstem Wert ermitteln
24:  return maxPoint
25: end function
```

5.4 Interaktionen der Komponenten

In Abbildung 5.2 wird das Klassendiagramm dargestellt, welches die Interaktionen der einzelnen Klassen untereinander wiederspiegelt. Die Klasse *Detector*, welche den Ablauf koordiniert und verwaltet, kann mehrere Objekte der Klasse *Face* besitzen. Beide Klassen greifen auf die Klasse *Helpers* und die darin enthaltenen Methoden zurück. Das Konzept ist so umgesetzt das jede Klasse für sich selbst unabhängig agiert. Änderungen an den Klassen können somit ohne großen Einfluss auf die anderen Klassen durchgeführt werden.

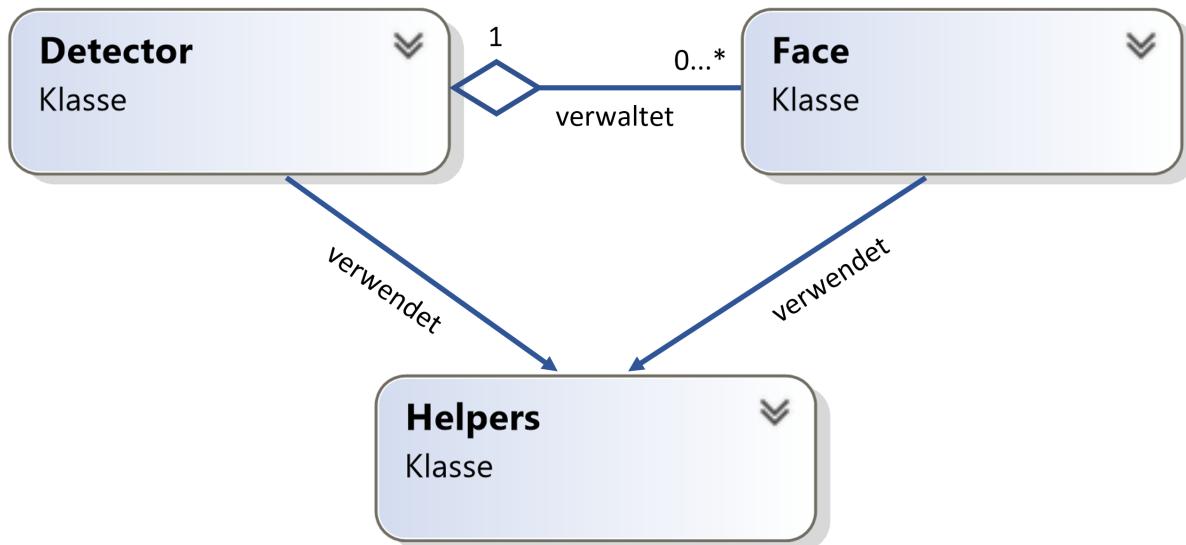


Abbildung 5.2: Der *Detector* verwaltet mehrere Gesichter bzw. versucht diese zu finden, die *Face* Klasse bildet ein Gesicht ab und prüft auf Lidschläge, die *Helpers* Klasse steht jeder Klasse für wichtige Berechnungen bereit

5.5 Bereitstellung

Der Algorithmus bzw. die dazugehörigen Klassen liegen auf dem webbasierten Online-Dienst GitHub unter folgendem Link <https://github.com/Sniger87/RealTimeEyeBlink> bereit. Für die Inbetriebnahme ist die Installation von OpenCV und Dlib notwendig. OpenCV kann unter <https://opencv.org/releases.html> und Dlib unter <http://dlib.net/> geladen werden. Sobald diese eingerichtet sind, kann z.B. über das Modul *VideoCapture* von OpenCV der Zugriff auf ein Video am übergebenen Pfad oder auf die Kamera, sofern eine vorhanden ist, erfolgen. Nach dem Erstellen einer Instanz der Klasse *Detector* kann mit dem Aufruf der Methode *Detect* am *Detector* die Erkennung gestartet werden.

6 Evaluation

In diesem Kapitel wird der Entwurf und die Implementation aus den beiden vorherigen Kapiteln 4 und 5 getestet und bewertet. In der Abbildung 6.1 wird der Algorithmus in der eigens dafür erstellten Testumgebung ausgeführt. Die Probanden schauen sich während des Tests ein YouTube-Video vom YouTuber DailyBroccoli [Dai17] an, dass einen Farbentest für die Augen enthält.

Das Video zeigt mehrere Sequenzen von jeweils 10 Farben an, wo sich eine Farbe von der anderen abhebt. Der Betrachter muss vier Schwierigkeitsgrade absolvieren und möglichst viele Farbunterschiede erkennen. Desto mehr er erkannt hat desto besser sind seine Augen.

Dieses Video wurde gewählt, da die Probanden durch die zu erfüllende Aufgabe abgelenkt sind und so mit nicht bewusst auf den Lidschlag achten. Hierdurch werden möglichst viele willkürliche Lidschläge getätigt, welche für eine aussagekräftige Evaluation des Algorithmus benötigt werden. Die Bestimmung der Erkennungsrate bei willkürlich auftretenden Lidschlägen ist präziser, da willkürliche Lidschläge aufgrund ihres zufälligen Auftretens schwieriger zu erkennen sind.

In Abbildung 6.1 ist außerdem zu sehen, dass die gefundenen Gesichtsmerkmale sowie die Pupillen farblich gekennzeichnet sind. Wird ein Lidschlag erkannt, wird der Zähler über dem Gesicht des Probanden automatisch hochgezählt. Während der Ausführung wird das Aufnahmefeld der Kamera in einer Datei aufgezeichnet und sofern der Proband damit einverstanden ist, für späteres testen wiederverwendet.

6.1 Probanden

Zum Evaluieren wurde ein eigener Datensatz aus 10 Personen erstellt. Die Altersspanne liegt von 18 bis 60 Jahre. Die Testpersonen teilen sich in fünf männliche und fünf weibliche Personen auf. Drei männliche Teilnehmer haben eine Gesichtsbehaarung und vier von den zehn tragen eine Brille.

Die Evaluation fand immer bei Tageslicht, also mit ausreichend Helligkeit statt. Die Testpersonen schauten sich das Video auf dem in Abschnitt 5.2 zur Entwicklung genutzten Microsoft Surface Pro 4 an.

Um eine gleichbleibende Situation zu haben, wurde jeweils immer nur eine Person zur Evaluation herangezogen. Dabei saßen diese mit einem Abstand von ca. 50 cm vom Bildschirm entfernt und mit einem maximalen Neigungswinkel von 35° vor dem Laptop.

Für das Erstellen des Datensatzes wurde immer nur die Frontalansicht mit kaum merklicher Rotation des Kopfes verwendet, das heißt Testdaten mit einer Profilansicht wurden nicht erstellt.

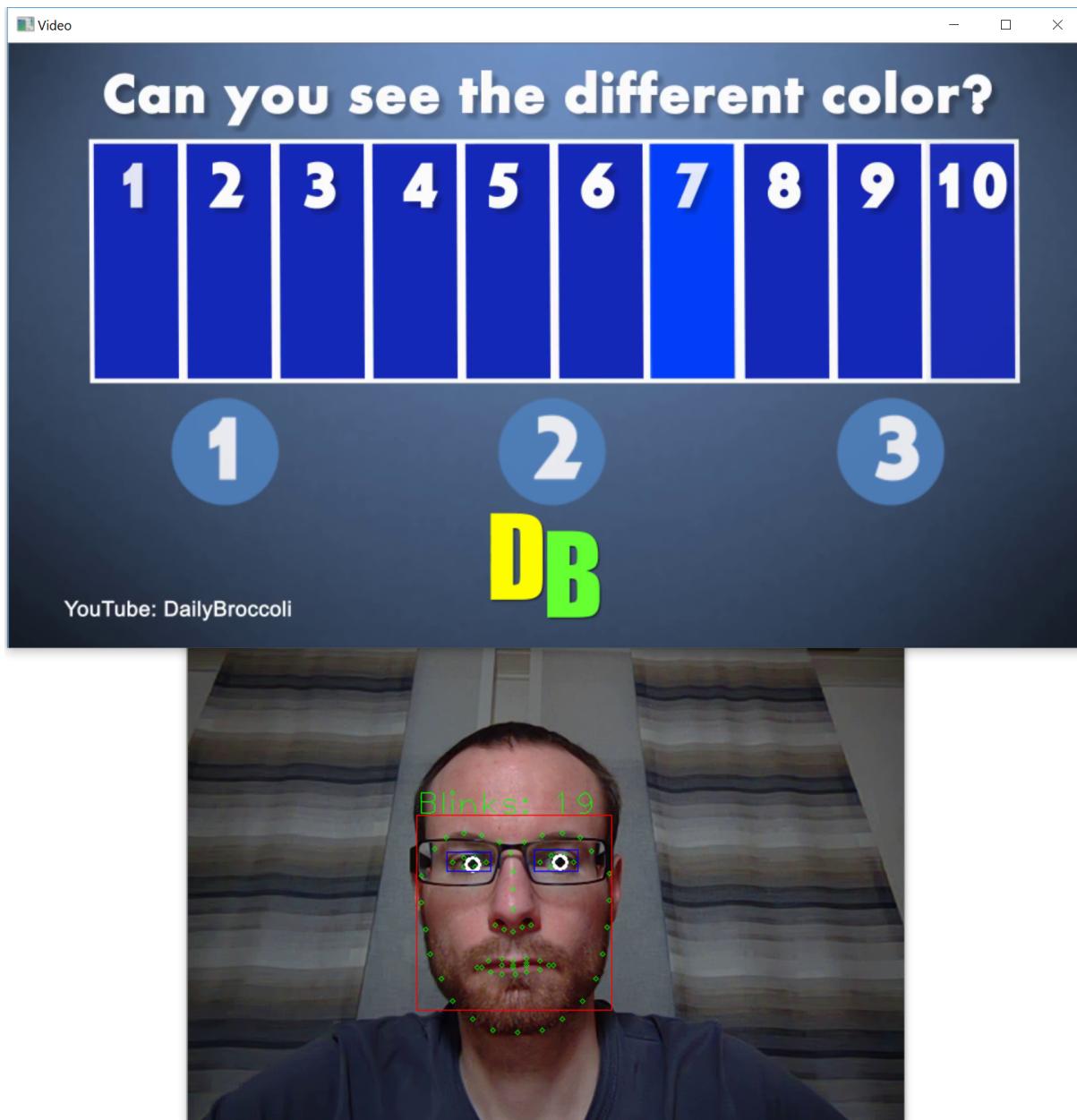


Abbildung 6.1: Ausführung des Algorithmus in der Testumgebung

6.2 Laufzeitanalyse

Die Laufzeit des Algorithmus wurde mit der Diagnosesitzung von Visual Studio untersucht. Als Eingangsdaten für den Algorithmus wurden die Testvideosequenzen aus dem Datensatz eingelesen und verarbeitet. Während des Programmlaufs wurden dann statistische Daten erzeugt, die durch das Werkzeug ausgewertet wurden.

Erstellt wurde die Analyse mit den gleichen Hardwarekomponenten wie unter Abschnitt 5.2 angegeben:

- Intel(R) Core(TM) i5-6300U @ 2,4 GHz Prozessor
- 8 GB - DDR3 Speicher
- Intel(R) HD Graphics 520
- Visual Studio Enterprise 2015 Update 3 mit der C++ Version 14
- Windows 10 Pro in der Version 1709
- Frontkamera des Surface Pro 4 mit 5-Megapixel

Die Abbildung 6.2 zeigt eine grafische Auswertung eines Testlaufs. Die gemessene Zeit wird in Millisekunden angegeben. Die y-Achse geht von 0 bis 50 ms. Auf der x-Achse werden die Anzahl der Bilder dargestellt. Die dunkelblaue Linie zeigt eine kumulative Auswertung des gesamten Algorithmus pro Bild an. Die hellblaue Linie ist die zeitliche Angabe des Gesichtsdetektors, der sich im Durchschnitt bei 21 ms befindet. Die Detektion der Gesichtsmerkmale liegt im Durchschnitt bei 5 ms und wird mit der violetten Linie dargestellt. Die grüne Linie zeigt die zeitliche Bearbeitung pro Bild zum Finden der Pupillen und liegt im Durchschnitt bei 3 ms.

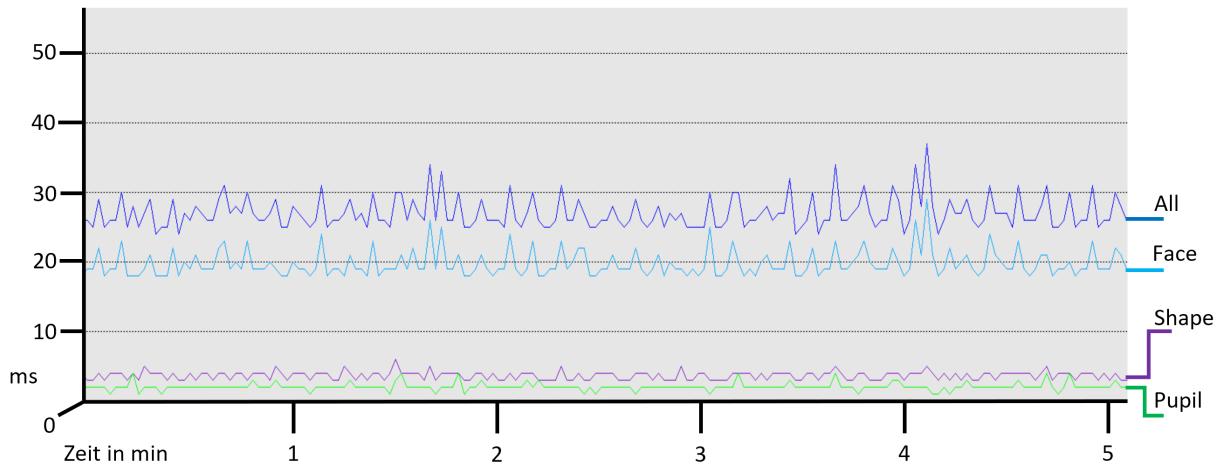


Abbildung 6.2: Grafischer Ausschnitt der Auswertung von der Laufzeit des Algorithmus

Die Vorverarbeitung der Bilder und das Prüfen ob ein Lidschlag vorliegt oder nicht, wurde ebenfalls gemessen. Diese Werte fallen jedoch aus dem Diagramm 6.2 heraus, da ihre Dauer im Nanosekundenbereich liegt und so keinen größeren Einfluss haben. Diese Werte wurden trotzdem in die kumulative Auswertung mit einbezogen. Die größte Auswirkung auf die Laufzeit hat der Aufruf im Detektor zum Suchen und Finden von Gesichtern in den Bilddaten.

Ausgewertet wurde die Laufzeit von 10 Testvideos die jeweils eine Länge von ca. 5 Minuten haben. Aus den Messwerten wurde jeweils die komplette Zeit des Algorithmus für ein einzelnes Bild genommen und über alle Werte den Durchschnitt berechnet. Wie Tabelle 6.1 zu entnehmen ist, ergibt sich eine durchschnittliche Laufzeit von 28,3 ms.

Ein entscheidendes Ziel beim Entwurf des Algorithmus war es, dass dieser schnell genug sein muss, um in der Zeit in der ein Lidschlag durchgeführt wird genug Bilddaten auswerten zu können, damit eine korrekte Aussage getroffen werden kann. Mit der durchschnittlichen Laufzeit des Algorithmus von 28,3 ms, führt dies zu einer Bildrate B_R (6.1) von:

$$B_R = \frac{1}{28,3 \text{ ms}} = 35,34 \frac{\text{Bilder}}{\text{Sekunde}} \quad (6.1)$$

Da Lidschläge mit einer Bildrate von $20 \frac{\text{Bilder}}{\text{Sekunde}}$ (siehe Abschnitt 4.3) aufgelöst werden, reicht diese Bildrate von $35,34 \frac{\text{Bilder}}{\text{Sekunde}}$ für die vorliegende Problemstellung aus.

Darüber hinaus lag die Bildrate in allen einzelnen Durchläufen ebenfalls über der Bildrate von $20 \frac{\text{Bilder}}{\text{Sekunde}}$ (vgl. Tabelle 6.1).

Testvideo	Laufzeit (ms)	Bildrate ($\frac{\text{Bilder}}{\text{Sekunde}}$)
1	28	35,71
2	30	33,30
3	27	37,04
4	29	34,48
5	31	32,26
6	27	37,04
7	30	33,30
8	27	37,04
9	28	35,71
10	26	38,46
\emptyset	28,30	35,34

Tabelle 6.1: Laufzeitverhalten des entwickelten Algorithmus

In Abbildung 6.3 wurde die Laufzeit mit zwei Gesichtern gemessen. Im Vergleich zur Abbildung 6.2, ist klar erkenntlich, dass die Bearbeitungszeit pro Bild merklich um ca. 10 ms im Durchschnitt zugenommen hat. Das Erkennen von Gesichtsmerkmalen und der Pupillen haben kaum zugenommen, dafür aber das Suchen und Finden von Gesichtern in den Bilddaten. Die Werte von der Vorbereitung der Bilddaten und das Prüfen auf einen Lidschlag nehmen auch bei zwei Gesichter keinen Einfluss auf die gesamte Ausführungszeit pro Bild. Somit wird diese hauptsächlich von der Gesichtserkennung bestimmt. Das gelingt dadurch, dass jedes gefundene Gesicht sequentiell abgearbeitet wird, dadurch kann im Endeffekt nur die Gesichtserkennung die Gesamtaufzeit erhöhen. Dies bedeutet im Umkehrschluss aber auch, dass für jedes zusätzliche Gesicht die Laufzeit durch jede zusätzliche Merkmalerkennung und Pupillensuche verlängert wird.

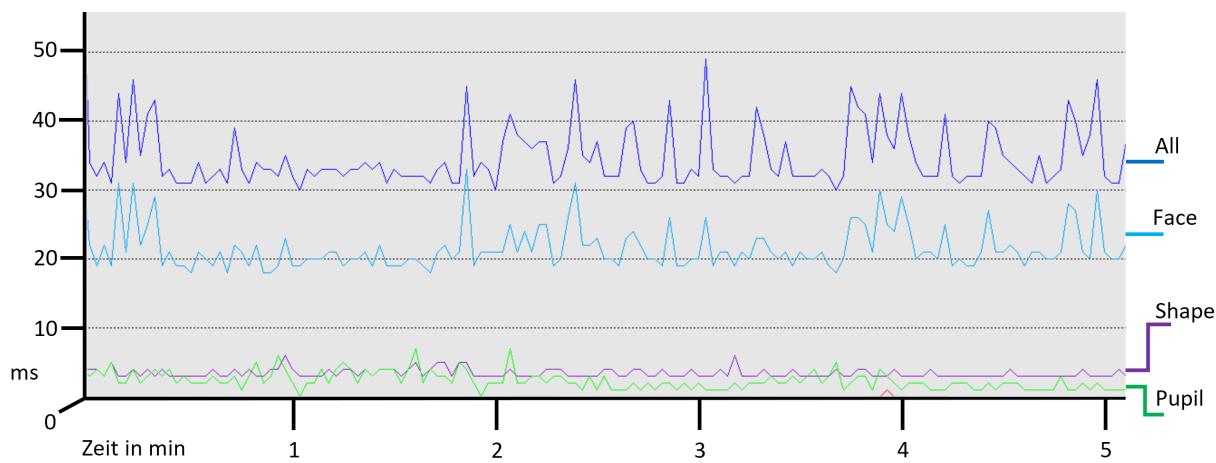


Abbildung 6.3: Grafischer Ausschnitt der Auswertung von der Laufzeit des Algorithmus mit zwei Gesichtern

Werden auf die Werte in Tabelle 6.1 jeweils 10 ms, die mehr für die Gesichtserkennung gebraucht werden, draufgeschlagen und zusätzlich für das zweite Gesicht die 8 ms für die Merkmalerkennung

und die Pupillensuche, ergibt sich eine durchschnittliche Laufzeit von 46,3 ms. Daraus ergibt sich eine Bildrate B_R (6.2) von:

$$R = \frac{1}{46,3 \text{ ms}} = 21,6 \frac{\text{Bilder}}{\text{Sekunde}} \quad (6.2)$$

Selbst mit zwei Gesichtern langt die Bildrate von $21,6 \frac{\text{Bilder}}{\text{Sekunde}}$ noch aus, um die Bilddaten in Echtzeit auszuwerten. Ab mehr als zwei Gesichter könnte die Verarbeitung allerdings ins Stocken geraten.

Hauptanwendung des hier entworfenen Algorithmus ist die Verwendung in der Selbstdiagnose von Parkinson-Kranken. Es ist davon auszugehen, dass für diesen Anwendungsfall lediglich ein Gesicht gleichzeitig erkannt werden muss und der entworfene Algorithmus somit die gestellten Anforderungen komplett erfüllt.

Die Pupillensuche, welche für die Selbstdiagnose von Parkinson-Kranken keine Rolle spielt, könnte ebenfalls abgeschaltet werden, womit Bearbeitungszeit des Algorithmus pro Bild eingespart wird.

6.3 Prozessor- und Speicherauslastung

Weitere Ziele bei der Entwicklung des Algorithmus waren, dass dieser die vorhandenen Ressourcen optimal nutzt und nicht nur effektiv, sondern auch effizient ist. Da die Aufnahme der Bilddaten z. B. auch über ein Handy aufgenommen und vom Algorithmus auf dem Gerät verarbeitet werden könnten, darf dieser die Ressourcen des Geräts nicht vollständig aufbrauchen.

Zur Messung der Prozessor- und Speicherauslastung wurde wieder das Diagnosetool von Visual Studio herangezogen. Dafür wurde der Algorithmus unter Last, also mit einer ständigen Präsenz eines Gesichtes, 4 Minuten lang ausgeführt.

Das Ergebnis wird in Abbildung 6.4 präsentiert. Zu erkennen ist, dass die Speicherauslastung bei 222 MB liegt und diese konstant beibehalten wird. Geräte mit heutiger Standardausstattung haben meist schon als minimalen Speicherbaustein 1024 MB verbaut. Die Speicherauslastung stellt somit kein Problem dar. Die Prozessorlast liegt weitestgehend konstant unter 40 %. Wird berücksichtigt das mit den Bilddaten einige Prozesse absolviert werden und die Bearbeitung rein auf der CPU abläuft, sind 40 % ein sehr guter Wert.

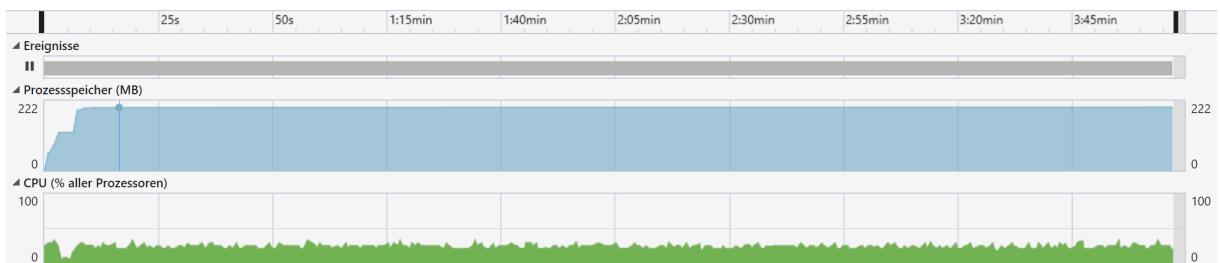


Abbildung 6.4: Auslastung vom Prozessor und Speicher während der Ausführung des Algorithmus

In Abbildung 6.5 wurde der Algorithmus wieder unter Last, aber diesmal mit zwei Gesichtern gleichzeitig getestet und die Werte von Prozessor und Speicher festgehalten. Diesmal liegt die Speicherauslastung bei konstanten 223 MB, also nur eins höher als davor. Selbst der Prozessor bleibt konstant unter der 40 % Grenze.

Mit den Ergebnissen der Auslastungstests ist gezeigt, dass der Algorithmus die Ressourcen optimal benutzt und somit Effizient arbeitet. Selbst wenn mehr als ein Gesicht erkannt wurde, ändert sich die Nutzung der Ressourcen kaum.

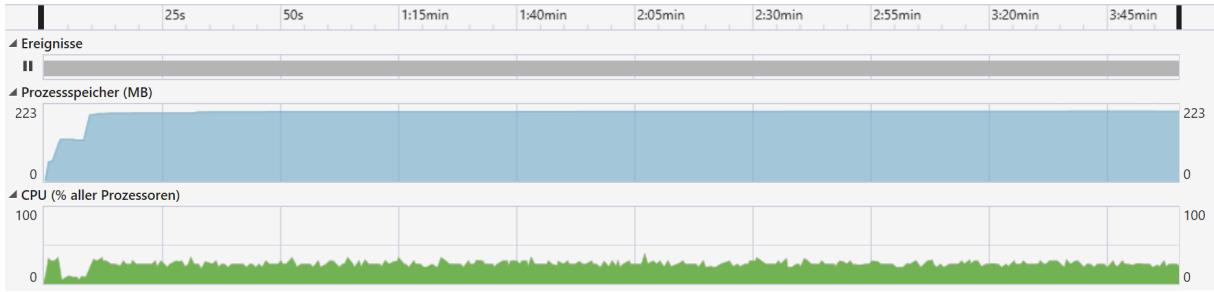


Abbildung 6.5: Auslastung vom Prozessor und Speicher während der Ausführung des Algorithmus mit zwei Gesichtern

6.4 Genauigkeit

Die Genauigkeit wird über die Erkennungsrate (*Sensitivität*) und die Fehlerrate ($1 - \text{Spezifität}$) bestimmt. Dazu werden mit einer Konfusion-Matrix (siehe Tabelle 6.2) die folgenden Werte ermittelt:

- **True-Positive** – Lidschlag gemacht und erkannt
- **False-Positive** – kein Lidschlag gemacht aber erkannt
- **False-Negative** – Lidschlag gemacht aber keinen erkannt
- **True-Negative** – kein Lidschlag gemacht und keinen erkannt

	Lidschlag gemacht	kein Lidschlag gemacht
Lidschlag erkannt	True-Positive	False-Positive
kein Lidschlag erkannt	False-Negative	True-Negative

Tabelle 6.2: Aufstellung der Konfusion-Matrix zur Lidschlagerkennung

Das Testvideo hat eine Länge von $G_F = 4684$ Bildern. Wird beachtet das ein Lidschlag 3 Bilder dauert, nimmt dieser also jeweils 3 Bilder für sich ein. Die insgesamt gemachten Lidschläge G_L (6.3) ergeben sich aus den Werten True-Positive tp und False-Negative fn .

$$G_L = tp + fn \quad (6.3)$$

Dieser Wert muss mit drei multipliziert werden um auf die Anzahl von Bildern zu kommen die alle Lidschläge eingenommen haben. Wird diese Anzahl von G_F abgezogen ergibt dies die Anzahl an Bildern aller nicht gemachter Lidschläge. Hier von muss der False-Positive-Wert fp abgezogen werden. Das ergibt den Wert für True-Negative tn (6.4):

$$tn = (G_F - (G_L \cdot 3)) - fp \quad (6.4)$$

Die Sensitivität E_R lässt sich über die Gleichung (6.5) und die Spezifität F_R über die Gleichung (6.6) berechnen:

$$E_R = \frac{tp}{(tp + fn)} \quad (6.5)$$

$$F_R = \frac{tn}{(tn + fp)} \quad (6.6)$$

Die Tabelle 6.3 zeigt die jeweiligen ermittelten Testergebnisse zu den zehn erstellten Videos mit den Probanden. Die Probanden wurden mit einem Schwellenwert von $SW = 0,18$ und einer maximalen Anzahl an Bildern von 3, wie in Kapitel 4 erklärt wurde, getestet. Die Tabelle enthält die Werte aus der Konfusion-Matrix und die dazu berechneten Werte zur Sensitivität und Spezifität. Über alle erhobenen Daten wird jeweils der Durchschnitt errechnet.

Test	True-Positive	False-Negative	False-Positive	True-Negativ	Sensitivität	Spezifität	Beschreibung
1	43	16	3	4507	0,72	0,99	Männlich, Brille, Bart
2	58	11	2	4477	0,84	0,99	Männlich, Bart
3	40	14	3	4519	0,74	0,99	Weiblich
4	45	15	1	4503	0,75	0,99	Weiblich
5	18	8	0	4606	0,69	1	Männlich, Brille
6	27	5	2	4586	0,84	0,99	Männlich, Bart
7	25	2	3	4600	0,93	0,99	Weiblich
8	23	20	3	4552	0,53	0,99	Weiblich, Brille
9	47	13	7	4497	0,78	0,99	Männlich
10	35	17	5	4523	0,67	0,99	Weiblich, Brille
\emptyset	36,1	12,1	2,9	4536,5	0,75	0,99	

Tabelle 6.3: Testergebnisse aus den zehn erstellten Videos mit den Probanden

Aus den Mittelwerten der Sensitivität und der Spezifität wird die in Abbildung 6.6 angezeigte ROC-Kurve erzeugt. Anhand dieser sieht man das Verhältnis der Erkennungsrate von 0,75 und der Fehlerrate ($1 - \text{Spezifität}$) von 0,01.

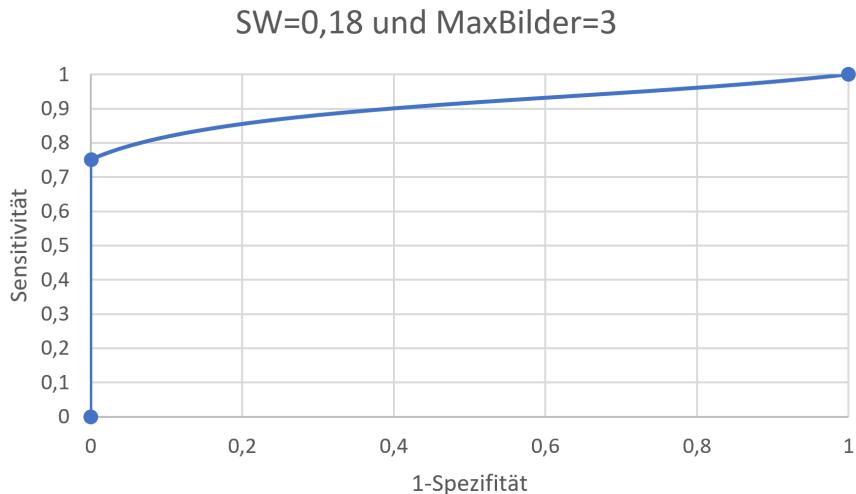


Abbildung 6.6: ROC-Kurve zwischen Sensitivität und Spezifität

Mit einer Erkennungsrate von 0,75 kann das Ziel des Algorithmus, dass dieser eine hohe Genauigkeit erzielt, leider nicht erfüllt werden. Die Fehlerrate ($1 - \text{Spezifität}$) von 0,01 ist ein gutes Ergebnis, da

dadurch nur wenige nicht gemachte Lidschläge als gemacht erkannt werden. Mögliche Ursachen für diese Erkennungsrate werden nachfolgend aufgezählt.

Nachteile beim Augenseitenverhältnis

Der Vergleich der berechneten EAR-Werte mit einem Schwellenwert hat einige Nachteile was die Erkennungsrate senken lässt.

In Abbildung 6.7 ist der Graph zu EAR-Werten bei einem nach oben geneigten Kopf zu erkennen. So gut wie alle Werte liegen unterhalb des gewählten Schwellenwerts von 0,18. Es werden bei diesem Fall sehr viele False-Positive-Werte erzeugt, also Lidschläge erkannt obwohl keine gemacht wurden.

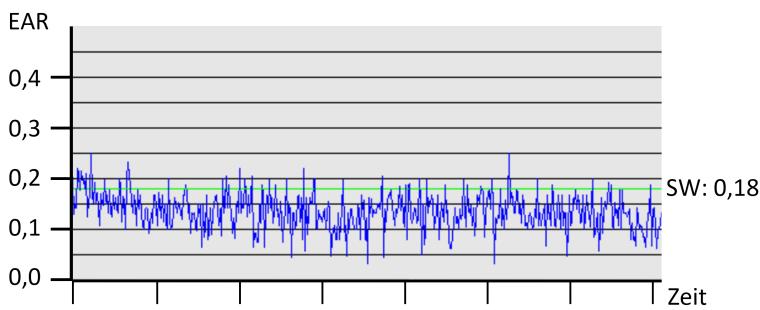


Abbildung 6.7: EAR-Werte beim einem nach oben geneigten Kopf

In Abbildung 6.8 ist der Graph zu EAR-Werten bei einem nach unten geneigten Kopf zu erkennen. So gut wie alle Werte liegen oberhalb des gewählten Schwellenwerts von 0,18. Es werden bei diesem Fall sehr viele False-Negative-Werte erzeugt, also keine Lidschläge erkannt obwohl welche gemacht wurden.

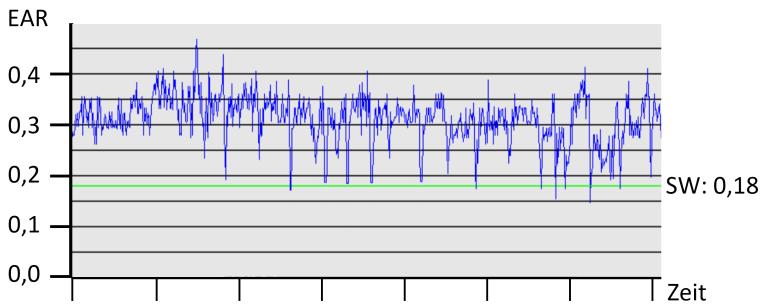


Abbildung 6.8: EAR-Werte beim einem nach unten geneigten Kopf

Ein weiterer Nachteil kommt auf, wenn die Person eine Brille trägt und der Rahmen der Brille sehr breit ausfällt. Dann liefert die Merkmalerkennung zu meist falsche Punkte von der die Berechnung des EAR-Wertes abhängt.

Aufgefallen ist auch wenn Personen schmale Augen besitzen. Dort ist der Abstand zwischen den Punkten sehr gering und die Werte des Augenseitenverhältnisses fallen dementsprechend schlecht aus.

Falsche Lidschläge werden meist auch erkannt, wenn eine Person grinst und dadurch die Grubenfalten unter den Augen nach oben gezogen werden. Auffällig war auch, dass die Entfernung des Gesichts zur Kamera eine Rolle spielt. Ist das Gesicht näher an der Kamera sind die Werte breiter verteilt. Desto weiter weg ein Gesicht ist, desto mehr häufen sich die Werte in einem bestimmten Bereich.

7 Fazit und Ausblick

Die Umsetzung der Arbeit hat gezeigt, dass mit dem heutigen Stand der Technik und Methoden eine videobasierte Blinzelerkennung in Echtzeit mit handelsüblichen Hardwarekomponenten wie Smartphone oder Laptop durchgeführt werden kann.

Die Herausforderungen des Algorithmus wurden dabei wie folgt gelöst. Aufgenommene Videodaten, entweder über eine Datei oder eine Kamera, werden mit den Methoden zur Bildreduktion und Bildverbesserung aus der OpenCV-Bibliothek vorverarbeitet. Hieraus ergeben sich in der Größe reduzierte Graustufenbilder. Da die Bibliothek Dlib nur eine Unterstützung mit einzelnen Bildern anbietet, wird der Datenstrom aus einer Abfolge von Bildern mit OpenCV in das erforderliche Format für DLib umgewandelt. Mit den Detektoren aus der Dlib-Bibliothek werden als erstes Gesichter in den Bilddaten gesucht. Wurde das mit Erfolg abgeschlossen, wird für jedes gefundene Gesicht die Positionen der Merkmale wie Mund, Nase, Kiefer und für diesen Algorithmus wichtigen Augenregion extrahiert. Dabei besteht die Augenregion jeweils aus sechs Punkten einmal für das linke und rechte Auge. Mit diesen Punkten kann der Eye-Aspect-Ratio nach Soukupová et al. [SC16] berechnet werden. Dieser beschreibt das Verhältnis zwischen Höhe und Weite eines Auges. Dieser Wert wird Bild für Bild ausgewertet. Liegt dieser unter einem Schwellenwert entscheidet der Algorithmus, ob ein Lidschlag vorgelegen hat oder nicht.

Abseits der Anwendung bei Parkinson-Kranken kann der entworfene Algorithmus auch zum Steuern von Programmen oder zum Spielen eines Serious Games eingesetzt werden. Dafür wurde der Ansatz nach Timm et al. [TB11] zum Ermitteln von Pupillen umgesetzt. Durch das verfolgen der Pupille und das Auslösen eines Lidschlages könnte zum Beispiel eine Aktion auf einem Button ausgelöst werden der gerade angeschaut wird. Dadurch das die Pupillenmitte vom Algorithmus schon gefunden wird und die Lidschlagerkennung vorhanden ist könnte der Algorithmus mit wenig Aufwand erweitert werden, um so eine Umsetzung zu realisieren.

Der Algorithmus erfüllt weitestgehend alle Anforderungen. Das Laufzeitverhalten ist gut gelöst, sodass sogar mehr als eine Person gleichzeitig vom System erfasst und ausgewertet werden kann. Die Ressourcen des Systems werden optimal genutzt und nicht übermäßig ausgelastet. Somit können nebenher noch weiter Prozesse Problemlos laufen.

Die Erkennungsrate von 0,75 fällt relativ schlecht aus. Hier muss noch nachgearbeitet und verbessert werden, um dort eine Rate von über 0,9 zu erzielen. Die Fehlerrate fällt gegenüber der Erkennungsrate sehr positiv aus und erreicht einen Wert von 0,01.

Mögliche Verbesserungen und Optimierungen werden nachfolgend aufgezählt.

Optimierungen

Damit die Verarbeitung noch weiter beschleunigt wird, sollten Teile des Algorithmus, wie zum Beispiel die Gesichtserkennung auf die GPU ausgelagert werden. In der Laufzeitanalyse (siehe Abschnitt 6.2) stellte sich heraus, dass bisher die Gesichtserkennung den größten Einfluss auf die Gesamlaufzeit hat. Soll der Algorithmus für mehr als zwei Personen zur Lidschlagerkennung eingesetzt werden, muss im Endeffekt eine Optimierung erbracht werden.

Eine weitere Optimierung der Laufzeit kann ein Umbau auf parallele Verarbeitung sein. Jedes Gesicht könnte dabei separat berechnet werden. Da auf die Bilddaten nur ein lesender Zugriff notwendig ist, gäbe es für die Parallelität kein Hindernis. Der Detektor müsste beim Erzeugen eines neuen Gesichts dieses nur in einen separaten Prozess auslagern. Mit dieser Optimierung könnte Laufzeit eingespart und Ressourcen noch besser genutzt werden.

Um den Entscheidungsprozess weiter zu optimieren, kann eine dynamische Anpassung des Schwellenwertes weitere Vorteile bringen. Bei der Evaluation ist aufgefallen das Gesichter die weiter weg sind einen EAR-Wert haben der höher liegt und selten unter einen Schwellenwert von 0,2 fällt.

Was eher weniger eine Optimierung ist aber auch Vorteile bringen kann, ist die Wiedererkennung einer Person. So könnten zu jeder Person Konfigurationen angelegt und diese geladen werden sobald die entsprechende Person wiedererkannt wurde. Dies macht Sinn, wenn der Algorithmus zur Selbstdiagnose eines Parkinson-Patienten verwendet wird, zu dem schon eine Krankenakte und vorherige Messergebnisse vorliegen.

Ausblick

In der vorliegenden Arbeit wurde ein lauffähiger Algorithmus zur Lidschlagerkennung entworfen, implementiert und an gesunden Menschen evaluiert. Als nächsten Schritt müssen nun Tests mit Patienten erfolgen die an Parkinson erkrankt sind. Dafür kann der Algorithmus für fortführende Arbeiten zur Selbstdiagnose von Parkinson-Patienten Anwendung finden. Dort muss die Erkennungsrate mit den vorgeschlagenen Optimierungen allerdings noch gesteigert werden.

Literaturverzeichnis

- [AGMANJJ13] J. A. Agúndez, E. García-Martín, H. Alonso-Navarro, and F. J. Jiménez-Jiménez. Anti-parkinson's disease drugs and pharmacogenetic considerations. *Expert opinion on drug metabolism & toxicology*, 9:859–874, 2013.
- [ALTL00] D. Aarsland, J. P. Larsen, E. Tandberg, and K. Laake. Predictors of nursing home placement in parkinson's disease: a population-based, prospective study. *Journal of the American Geriatrics Society*, 48:938–942, 2000.
- [BB09] W. Burger and M. J. Burge. *Principles of Digital Image Processing: Fundamental Techniques*. Springer London, London, 2009.
- [Ber99] A. Berke. *Biologie des Auges: eine Einführung in die Anatomie und Physiologie des Auges*. WVAO, 1999.
- [Chr16] D. Christ. Recherche, anwendung und evaluierung verschiedener verfahren zur detektion von gesichtern, Oktober 2016.
- [COM⁺05] S. Chapuis, L. Ouchchane, O. Metz, L. Gerbaud, and F. Durif. Impact of the motor complications of parkinson's disease on the quality of life. *Movement disorders*, 20:224–230, 2005.
- [Cra80] D. von Cramon. *Die spontanen Lidbewegungen: Ein Spiegel psychosomatischer Aktivität?*, volume 6. 1980.
- [CWWS12] X. Cao, Y. Wei, F. Wen, and J. Sun. Face alignment by explicit shape regression. In *CVPR 2012*, January 2012.
- [Dai17] DailyBroccoli. Do you have good eyes? (test with answers). Oktober 2017. Zugriff: 20.10.2017, <https://www.youtube.com/watch?v=GIQGzCcIrE8>.
- [Dli17a] Dlib. Dlib c++ library. November 2017. Zugriff: 22.11.2017, <http://dlib.net/>.
- [Dli17b] Dlib. Dlib c++ library - frontal face detector. November 2017. Zugriff: 22.11.2017, http://dlib.net/imaging.html#get_frontal_face_detector.
- [Dli17c] Dlib. Dlib c++ library - hog. November 2017. Zugriff: 22.11.2017, http://dlib.net/imaging.html#scan_fhog_pyramid.
- [Dli17d] Dlib. Dlib c++ library - introduction. November 2017. Zugriff: 22.11.2017, <http://dlib.net/intro.html>.
- [Dli17e] Dlib. Dlib c++ library - license. November 2017. Zugriff: 22.11.2017, <http://dlib.net/license.html>.
- [Dli17f] Dlib. Dlib c++ library - shape predictor. November 2017. Zugriff: 22.11.2017, http://dlib.net/imaging.html#shape_predictor.
- [DT05] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, pages 886–893, 2005.

- [dW17] Spektrum der Wissenschaft. Bildpyramide. November 2017. Zugriff: 22.11.2017, <http://www.spektrum.de/lexikon/geowissenschaften/bildpyramide/1742>.
- [FGMR10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. 32(9), September 2010.
- [FHS⁺12] E. Fitzpatrick, N. Hohl, P. Silburn, C. O'Gorman, and S. A. Broadley. Case-control study of blink rate in parkinson's disease under different conditions. *Journal of neurology*, 259:739–744, 2012.
- [Gal01] N. Galley. *Physiologische Grundlagen, Messmethoden und Indikatorfunktion der okulomotorischen Aktivität*. In Frank Rösler: Enzyklopädie der Psychologie, Göttingen: Hogrefe, 2001.
- [Gmb17] UCB Pharma GmbH. Parkinson aktuell. November 2017. Zugriff: 22.11.2017, <http://www.parkinson-aktuell.de>.
- [Gro17] Intelligent Behaviour Understanding Group. Facial point annotations. November 2017. Zugriff: 22.11.2017, <https://ibug.doc.ic.ac.uk/resources/facial-point-annotations/>.
- [HDKL92] A. J. Hughes, S. E. Daniel, L. Kilford, and A. J. Lees. Accuracy of clinical diagnosis of idiopathic parkinson's disease: a clinico-pathological study of 100 cases. *Journal of Neurology, Neurosurgery & Psychiatry*, 55:181–184, 1992.
- [KS14] V. Kazemi and J. Sullivan. One millisecond face alignment with an ensemble of regression trees. In *Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [LFK⁺14] D. L. Longo, A. S. Fauci, D. L. Kasper, S. L. Hauser, J. L. Jameson, and J. Loscalzo. Harrison's principles of internal medicine 19th. 2014.
- [Mos81] R. A. Moses. Adler's physiology of the eye clinical application. pages 1–15, 1981.
- [Ope17a] OpenCV. Open source computer vision library. November 2017. Zugriff: 22.11.2017, <https://opencv.org/>.
- [Ope17b] OpenCV. Open source computer vision library - about. November 2017. Zugriff: 22.11.2017, <https://opencv.org/about.html>.
- [Ope17c] OpenCV. Open source computer vision library - face detection. November 2017. Zugriff: 22.11.2017, https://docs.opencv.org/master/d7/d8b/tutorial_py_face_detection.html.
- [Ope17d] OpenCV. Open source computer vision library - license. November 2017. Zugriff: 22.11.2017, <https://opencv.org/license.html>.
- [PAB⁺12] R. B. Postuma, D. Aarsland, P. Barone, D. J. Burn, C. H. Hawkes, W. Oertel, and et al. Identifying prodromal parkinson's disease: Pre-motor disorders in parkinson's disease. *Movement Disorders*, 27:617–626, 2012.
- [RK14] N. Rekha and Dr. M. Z. Kurian. Face detection in real time based on hog. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, 3:1345–1352, April 2014.
- [Row99] H. A. Rowley. *Neural Network-based Face Detection*. PhD thesis, Pittsburgh, PA, USA, 1999.

-
- [SC16] T. Soukupová and J. Cech. Real-time eye blink detection using facial landmarks, Februar 2016.
- [sDEdV17] selfhtml Die Energie des Verstehens. Grafik/grundbegriffe/graustufen. November 2017. Zugriff: 22.11.2017, <https://wiki.selfhtml.org/wiki/Grafik/Grundbegriffe/Graustufen>.
- [SWG84] J. A. Stern, L. C. Walrath, and R. Goldstein. The endogenous eyeblink. *Psychophysiology*, 21(1):22–33, 1984.
- [TB11] F. Timm and E. Barth. Accurate eye centre localisation by means of gradients. volume 1, pages 125–130, Algarve, Portugal, 2011. In Proceedings of the Int. Conference on Computer Theory and Applications (VISAPP).
- [VC74] V. Vapnik and A. Chervonenkis. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.
- [VJ01] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Accepted Conference on Computer Vision and Pattern Recognition*, 2001.
- [VJ04] P. Viola and M. J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.
- [Wah89] F. Wahl. *Digitale Bildsignalverarbeitung : Grundlagen, Verfahren, Beispiele*. Informatik-Institute, Braunschweig, 1989.
- [wik17] wikipedia.org. Gauß-filter. November 2017. Zugriff: 22.11.2017, <https://de.wikipedia.org/wiki/Gau\T1\ss-Filter>.
- [Wil83] H. Wild. *Experimentelle Untersuchungen über die zeitliche Verteilung der spontanen Lidschlussereignisse (Dissertation am Psychologischen Institut der Universität Innsbruck)*. Universität Innsbruck, Innsbruck, 1983.