UNIVERSITÄT KOBLENZ · LANDAU
Fachbereich 4: Informatik

AGAS
Arbeitsgruppe Aktives Sehen

Evaluation von Eye-Tracking-Teilalgorithmen

Studienarbeit
im Studiengang Computervisualistik

vorgelegt von

Carola Schmidt

Betreuer: Dipl.-Inf. Detlev Droege, Institut für Computervisualistik, Fachbereich Informatik

Koblenz, im Februar 2008

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

Die Richtlinien der Arbeitsgruppe für Studien- und Diplomarbeiten habe ich gelesen und anerkannt, insbesondere die Regelung des Nutzungsrechts.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. ja ☐ nein ☐

Der Veröffentlichung dieser Arbeit im Internet stimme ich zu. ja ☐ nein ☐

Koblenz, den ..... Unterschrift

Inhaltsverzeichnis

Verzeichnis der Bilder

Kapitel 1

Einleitung

Die folgende Arbeit basiert auf der Diplomarbeit „Gaze-Tracking zur Interaktion unter Verwendung von Low-Cost-Equipment“ von Thorsten Geier, in der dieser ein echtzeitfähiges Blickrichtungserkennungssystem entwickelt hat.[GEI07] Mit dessen Hilfe kann man erfahren, auf welchen Bildschirmpunkt eine Person guckt. Benötigt wird dazu die von Thorsten Geier programmierte Software und die vorgegebene Hardware. Bedeutung hat solch ein System vor allem für Menschen, die aus physischen Gründen die Tastatur und Maus nicht bedienen können.Sie wären auf solch ein System angewiesen, wenn sie alleine am Computer arbeiten wollten. Daher ist es wichtig, dass die Augen korrekt gefunden werden und die Blickrichtungserkennung möglichst exakt ist, so dass bei der Handhabung des Systems keine Fehler und Abweichungen auftreten. Die Blickrichtungserkennung und deren Genauigkeit sind abhängig von den Ergebnissen der Berechnung des Pupillenmittelpunkts und des Hornhautreflektionsmittelpunkts. Aus diesen Daten kann man letztendlich den passenden Bildschirmpunkt berechnen. Momentan basiert diese Berechnung im Gazetracker auf der Bestimmung des Schwerpunkts von Pupille und Hornhautreflektion. Probleme können beispielsweise noch bei unpassender Umgebungshelligkeit auftreten, bei Tragen einer Brille seitens der Testperson oder Verdeckung der Pupille durch Wimpern beziehungsweise der Hornhautreflektion selbst. Zu beachten ist auch, dass die übergebenen Bilder zur Berechnung dieser beiden Positionen eine sehr niedrige Auflösung besitzen, es aber dennoch möglich sein soll, den Mittelpunkt auf Subpixelniveau zu berechnen.

Ziel dieser Arbeit ist es, neue Verfahren zu finden, mit denen man den Pupillenmittelpunkt und den Hornhautreflektionsmittelpunkt bestimmen kann. Mittels dieser Verfahren soll eine bessere Genauigkeit der Koordinaten auf Subpixelniveau erreichbar sein. Dabei werden durchaus einfache Verfahren, vergleichbar mit der Schwerpunktsberechnung, als auch komplexere Verfahren, die generell für hochauflösende Bilder entwickelt wurden, in Betracht gezogen. Die verschiedenen Verfahren zur Bestimmung der Mittelpunkte werden im nächsten Kapitel vorgestellt.
Die implementierten Algorithmen werden im Anschluss auf verschiedene Testserien angewandt. Diese wurden von dem studentischen Mitarbeiter Wladimir Krebs aufgenommen. Ebenso hat dieser im Gazetracker eine Funktion bereitgestellt, mit Hilfe derer man die Testserien einlesen kann und auf die man die Algorithmen anwenden kann. Im zweiten Kapitel werden die Testserien und die Bewertungsmethoden für die Algorithmen dann beschrieben. Es wird im Vergleich festgestellt, welcher Algorithmus die passendsten Ergebnisse liefert. Die Zusammenfassung liefert noch einmal einen Überblick über die Ergebnisse und gibt einen Ausblick, was für Verbesserungen für Mittelpunktsbestimmungen noch eingebracht werden könnten.

Kapitel 2

Algorithmen zur Berechnung der Mittelpunkte

Möchte man im Eye-Tracking den Mittelpunkt der Pupille finden, gibt es zwei von der Beleuchtung abhängige, verschiedene Methoden dies zu tun. Wird das Auge nur dem sichtbaren Licht ausgesetzt, ist es am Günstigsten, den Limbus, also die Kontur zwischen Iris und Sklera, zu finden und daraus den Irismittelpunkt zu berechnen. In diesem Fall entspricht der Irismittelpunkt dem Pupillenmittelpunkt. Es gibt in der Forschung einige Ansätze, wie ein Eye-Tracker bei sichtbarem Licht zu realisieren ist. Unabdingbar sind solche Systeme beispielsweise in der Biometrie im Gebiet der Iriserkennung. Hier müssen die Konturen der Iris klar erkennbar sein, um die Irisdaten korrekt auswerten zu können. Allerdings ist es schwierig, allein mit sichtbarem Licht zu arbeiten, da oft spiegelnde und diffuse Komponenten auftreten.
Diese Komponenten können mit Hilfe von Infrarotlicht, das für den Benutzer nicht wahrnehmbar ist, ausgeschlossen werden. Benutzt man Infrarotlicht, besitzt die Pupille die Kontur, die am Einfachsten zu tracken ist, da Iris und Sklera das Infrarotlicht stark reflektieren. Vorteil ist hier, dass sich die Pupillenkontur gegenüber der Limbuskontur genauer definieren lässt und kleiner ist. Zudem wird die Pupille nur selten von den Augenlidern überdeckt. Die Ergebnisse bei Benutzung des Infrarotlichts sind allerdings stark vom Umgebungslicht abhängig. Solch ein System kann dementsprechend nicht

ohne weiteres im Freien genutzt werden. Ebenso muss darauf geachetet werden, dass der Raum, in dem der Gazetracker mit Infrarotlicht genutzt wird, nicht zu hell ist, ansonsten kann es zu Problemen bei der Detektion der Augen kommen. Diese werden im Gazetracker mittels der Hough Transformation für Kreise gefunden.
Generell unterscheidet man zwischen zwei verschiedenen Arten von Verfahren, mit denen man den Mittelpunkt der Pupille beziehungsweise Iris nach der groben Detektion finden kann:
Die merkmalsbasierten Verfahren suchen Merkmale in der Augenregion, die der Iris oder Pupille zugeordnet werden können, beispielsweise Helligkeitswerte. Solch ein Verfahren ist im Gazetracker von Thorsten Geier implementiert worden.
Die modellbasierten Verfahren basieren auf geometrischen Modellen der Pupille/Iris. Hier wird ein Modell gesucht, das am Besten auf die Iris/Pupille passt. Dabei gibt es verschiedene Ansätze, welcher Form die Pupille letztendlich angepasst werden soll. So gehen einige Verfahren von einer kreisrunden Pupille aus, während andere Verfahren von einer ellipsenartigen Form ausgehen. Eine weitere Möglichkeit ist es, eine Form für das gesamte Auge zu definieren, an die die Pupillenform dann angepasst werden kann.
Der Gazetracker benutzt eine feste Kamera, die schwarz-weiß Bilder aufnimmt. Die Kameraposition befindet sich am Bildschirm, sodass der Benutzer seinen Kopf beim Betrachten des Bildschirms frei bewegen kann. Da ein Interlace-Effekt bei den Bilddaten besteht, sollte der Betrachter den Kopf möglichst ruhig halten. Der Interlace-Effekt entsteht durch das PAL Zeilensprungverfahren der Kamera. Bei der Bildbearbeitung wird nur jede zweite Zeile bei der Berechnung der Pupillenmitte miteinbezogen, damit keine Störung durch verschobene Zeileninhalte entsteht. Die Bildrate bei den Testsequenzen beträgt zwischen 15 und 20 FPS. Sie kann allerdings bei der Verwendung von aufwändigen Algorithmen noch sinken.
Das Augenbild, auf das die Algorithmen angewandt werden, hat insgesamt eine Größe von 44*44 Pixeln. Der Radius der Pupille selbst variiert zwischen 3 und 8 Pixeln.

Bild 2.1: Ein Augenbild des linken Auges im Gazetracker

Bild 2.2: Ein Augenbild des rechten Auges im Gazetracker

2.1 Verfahren zur Bestimmung des Pupillenmittelpunktes

Bei der Pupillendetektion im Gazetracker wird das *dark-pupil*-Verfahren angewandt, das darauf basiert, dass die Pupille der dunkelste Bereich im Bild ist. Im Gegensatz dazu existiert noch das *bright-pupil*-Verfahren, in dem die Pupille den hellsten Bereich im Bild darstellt. Mit der Dark-Pupil-Methode kann recht einfach ein Schwellwert gefunden werden, der alle zur Pupille zugehörigen Pixel findet. Im Gazetracker wird dieser Schwellwert aus der durchschnittlichen Pupillenhelligkeit berechnet, der ein variabler Wert aufaddiert wird.
Dem Algorithmus zur Bestimmung des Pupillenmittelpunktes liegt ein Augenbild zu Grunde, in welchem die Pupille, die zuvor mittels der Hough Transformation approximativ gefunden wurde, ungefähr im Mittelpunkt des Bildausschnitts plaziert wurde. Dies ist insbesondere günstig, da man so die Suche des Pupillenrandes auf einen kleinen Bildbereich um die Bildmitte beschränken kann.
Die Verfahren zur Bestimmung des Mittelpunktes enthalten alle eine Kantendetektion der Pupillenrandpunkte, aus denen der Pupillenmittelpunkt berechnet werden kann. Dabei wird teils auf klassische Kantendetektionsalgorithmen wie die Operatoren von Canny oder Sobel zurückgegriffen, teils sucht man die Kantenpunkte ausgehend vom Pupillenmittelpunkt über den Schwellwert oder die Differenz zwischen Iris-und Pupillenpunkten. Dabei gehen die meisten Verfahren davon aus, dass es sich bei der Pupille um einen Kreis oder eine Ellipse handelt. Die Kantenpunkte können dann an die geometrische

Form angepasst werden, sodass der Pupillenmittelpunkt der geometrische Mittelpunkt des berechneten Kreises beziehungsweise der Ellipse ist.

2.1.1 Der Starburst Algorithmus nach Li, Winfield und Parkhurst

Der Starburst Algorithmus [LWP05] ist ein Verfahren zum Finden der Pupillenmitte, das ursprünglich für ein Head-Mounted Eye Tracking System mit Infrarotbestrahlung entwickelt wurde. Es ist aber auch für Systeme mit frei platzierbarer Kamera geeignet. Dabei vereint der Algorithmus merkmalsbasierte und modellbasierte Verfahren, um einen genauen Mittelpunkt zu bestimmen. Der Algorithmus beginnt mit einer Bildvorverarbeitung, die eine Rauschentfernung und eine Entfernung der Hornhautreflektion vorsieht. Danach beginnt die Suche nach der Pupillenkontur. Für jedes Bild wird zunächst ein Pupillenmittelpunkt geraten. Im ersten Bild kann dies manuell eingestellt werden oder es wird eine als passend erscheinende Position im Bild als Ausgangspunkt genommen (bei Head-Mounted Systemen und auch bei dem Gaze-Tracker kann man zum Beispiel einfach die Mitte des Eingabebildes nehmen). Für weitere Bilder wird der Pupillenmittelpunkt des vorherigen Bildes verwendet.
Hat man diesen Punkt gefunden, werden Punkte auf N (hier 18) Strahlen, die sich vom Startpunkt aus strahlenförmig ausbreiten, jeweils pixelweise verglichen, bis ein Schwellwert (hier 20) erreicht ist. Ausgehend von der Dark- Pupil Methode nehmen die Helligkeitswerte der Strahlen zum Rand der Pupille hin zu und dementsprechend werden nur positive Differenzen beachtet. Ist der Schwellwert erreicht, wird an dieser Position ein Featurepunkt definiert, der potentiell den Rand der Pupille darstellt. Für jeden der gefundenen Featurepunkte wird dieses Verfahren wiederholt, allerdings ist der Grad der Strahlen diesmal beschränkt auf +- 50 Grad um den Strahl, der zum Featurepunkt geführt hat. So werden wiederum nur Feature- Punkte gefunden, die auf der Pupille liegen.
Problematisch wird dieses Verfahren allerdings, wenn der geratene Mittelpunkt nicht ohnehin in der Pupille liegt. Dies führt dazu, dass die Featurepunkte nach dem bisherigen Algorithmus schwerpunktmäßig nur auf einer Seite der Pupille zu finden sind. Um dieses Problem zu beseitigen, wird nach der ersten Iteration die durchschnittliche Position aller

Featurepunkte berechnet und als neuer Startpunkt für die nächste Iteration verwendet. Dieses Vorgehen wird solange wiederholt, bis der durchschnittliche Featurepunkt nicht mehr als d Pixel (hier 10 aufgrund der hohen Auflösung) vom Vorherigen abweicht. Dieser Prozess sollte nach circa zehn Iterationen abgeschlossen sein, ansonsten wird der Algorithmus für dieses Frame abgebrochen. In diesem Fall konnte kein Pupillenmittelpunkt gefunden werden.
Ist der Prozess allerdings abgeschlossen, folgt das Ellipse Fitting, um den genauen Mittelpunkt der Pupille zu errechnen. Dazu wird der RANSAC Algorithmus benutzt, um auszuschließen, dass Featurepunkte, die ausserhalb der Pupille liegen, Einfluss auf die Position der Ellipse haben. So werden zunächst fünf Featurepunkte zufällig ausgewählt. Mittels Singulärwertzerlegung werden dann die Parameter der Ellipse gefunden, die exakt auf diese Punkte passt. Falls die Werte nicht zusammenpassen, liegt der Mittelpunkt der Ellipse ausserhalb des Bildes oder die Länge der Achsen steht in keinem realistischen Verhältnis zueinander. Es werden solange fünf Punkte zufällig ausgewählt, bis die Werte für die Ellipse zueinander passen. Wenn fünf Punkte gefunden wurden, werden sämtliche Featurepunkte durchlaufen, um mittels eines Schwellwerts zu testen, wieviele Punkte zu dieser Ellipsendarstellung passen.
Diese Vorgehensweise wird einige Male wiederholt. Danach werden die gefundenen Ellipsen miteinander verglichen. Die Ellipse mit den meisten passenden Featurepunkten liefert dann den zu verwendenden Pupillenmittelpunkt als Parameter.
Der Starburst-Algorithmus mit seinem vergleichsweise aufwändigen Verfahren ist insbesondere für Augenbilder mit einer hohen Auflösung geeignet, die bei dem gegebenen Gazetracker nicht vorhanden sind. Die Pupille befindet sich im Gazetracker in einem Bereich von ungefähr neun mal neun Pixeln, und ist in dem Augenbild jeweils so platziert, dass der Pupillenmittelpunkt sich ungefähr in der Mitte des Bildes befindet. Aufgrund der geringen Auflösung dürfte das Starburstverfahren mit seinem Algorithmus eher Nachteile als Vorteile mit sich bringen. Bei dem ersten Aussenden der Strahlen würden so schon alle wichtigen Kantenpunkte detektiert, was die nachfolgende Iteration unnötig machen würde. Auch das RANSAC Verfahren würde vermutlich zu keiner Verbesserung bei der Mittelpunktssuche führen, da die Kantenpunkte so nah beieinander liegen, dass sich der Ellipsenmittelpunkt bei der Auswahl verschiedener Randpunkte nur minimal verändern dürfte. Dementsprechend darf man bei der späteren Auswertung der Ergebnisse keine

Verbesserung im Vergleich mit dem jetzigen Verfahren erwarten.

2.1.2 Der Algorithmus nach Daunys und Ramanauskas

Gintautas Daunys und Nerijus Ramanauskas stellen in ihrem Paper „The accuracy of eye tracking using image processing“ [DAR04] gleich zwei Algorithmen zur Bestimmung des Pupillenmittelpunkts vor.
Die Pupille wird zunächst mittels des *dark-pupil*-Verfahrens gefunden. Dabei geht man davon aus, dass die Helligkeit in der Region zwischen Pupille und Iris monoton zunimmt. In der Übergangsregion werden so die Helligkeitswerte einer polynomiellen Funktion zugeordnet, um die genauen Kanten mit Subpixelgenauigkeit zu finden.
Wenn man die Kantenpunkte gefunden hat, gibt es nun zwei Möglichkeiten, den Mittelpunkt der Pupille zu finden. Die erste vorgestellte Methode nennt sich Circle Approximation Method. Hier wird ein Kreis nach Chaudhuri and Kundu berechnet, der exakt zu den gefundenen Konturpunkten passen soll. Die Summe der Abstände der Kantenpunkte zum Mittelpunkt wird nach dieser Methode minimiert. Als Ergebnis werden Radius und Mittelpunkt des Kreises geliefert. Nach der ersten Iteration werden die Kantenpunkte neu gewichtet, je nach Abweichung vom errechneten Kreis. So wird der Einfluss von Artefakten minimiert. Insgesamt finden maximal acht Iterationen statt, ansonsten endet der Algorithmus, wenn die Ergebnisse sich nur noch minimal verändern.
Die zweite vorgestellte Methode ist die Coordinates Averaging Method. Hier werden die Randlinien der Pupille einmal horizontal und einmal vertikal gescannt.Bei diesen Scans ergeben sich jeweils zwei gegenüberliegende Punkte, von denen der Mittelpunkt berechnet wird. Aus den Mittelpunkten aller horizontalen und vertikalen Scans kann man dann insgesamt zwei lineare Gleichungen aufstellen. Um die Mitte der Pupille zu finden, wird ein Gleichungssystem bestehend aus diesen Gleichungen gelöst. Der Mittelpunkt ist somit der Schnittpunkt dieser beiden Gleichungen.
Besitzt die Pupille eine einfache, gleichmäßige Struktur kann die Coordinates Averaging Method durchaus gute Ergebnisse erzielen. Probleme könnte es allerdings geben, wenn sich die Hornhautreflektion am unteren Rand oder in der Pupille befindet. In diesem

Fall wird die berechnete horizontale Gerade nach oben verschoben und der y- Wert der Pupillenmitte wird verfälscht. Wenn zudem wenige Randpunkte gefunden werden und die Pupille verdreht oder asymmetrisch ist, ist es kaum möglich, mit dieser Methode einen korrekten Mittelpunkt zu finden.

2.1.3 Der Algorithmus nach Perez, Garcia, Mendez, Munoz, Pedraza und Sanchez

Dieser Algorithmus [PCG03] ähnelt in seinen Grundzügen dem Starburst-Verfahren. Zunächst wird ein Pupillenmittelpunkt durch den Schwerpunkt der Pupille bestimmt. Um die Pupillenkontur zu finden, wird nur der Bereich innerhalb der Iris zur Suche benutzt. Bevor die Kontur gesucht wird, wird das Bild gefiltert, um Rauschen zu entfernen und die Pixel, die über dem Schwellwert liegen, der die Pupille gefunden hat, werden invertiert. Danach wird der Laplaceoperator benutzt, um die Kanten zu finden.
Es wird in diesem Verfahren davon ausgegangen, dass die Pupille die Form eines Kreises hat. Der Algorithmus geht vom Pupillenmittelpunkt, der ja mittels Schwerpunkt vorher bestimmt wurde, aus und versucht, wie der Starburst-Algorithmus, mittels diagonalen Strahlen den Rand der Kontur zu finden. Wenn alle gefundenen Randpunkte äquidistant zum Mittelpunkt liegen, liegt ein Kreis vor und es kann der Pupillenradius ermittelt werden. Sind die Abstände nicht ähnlich, wird mittels der Mittelpunkte der Diagonalen ein neuer Mittelpunkt bestimmt. Dabei werden zu lange oder zu kurze Diagonalen nicht berücksichtigt, um fehlerhafte Featurepunkte nicht in die Rechnung miteinzubeziehen. Dieses Verfahren wird solange wiederholt, bis die Abstände der Featurepunkte äquidistant sind, alternativ kann man natürlich die Anzahl der Durchläufe beschränken. Diese Beschränkung ist vor allem zu empfehlen, da die Pupille ellipsenförmig sein kann und somit nicht unbedingt genau äquidistante Abstände gefunden werden können.

2.1.4 Der Algorithmus nach Ahmad Poursaberi und Babak N. Araabi

Dieses Verfahren [POA05] beschäftigt sich grundsätzlich mit der Detektion der Iris, beinhaltet jedoch auch das Finden der Pupille und die Berechnung ihres Mittelpunkts. Bei diesem Verfahren wird allerdings nicht mit Infrarotbestrahlung gearbeitet, so dass die Pupillendetektion bei der Implementation leicht abgewandelt werden muss.
Als Eingabe dient ein Grauwertbild des Auges, das nur den Irisbereich abdeckt und welches zunächst zur Artefaktbeseitigung invertiert wird. Im invertierten Zustand werden Löcher gefüllt, also helle oder dunkle Pixel, die von Pixeln mit stark unterschiedlicher Helligkeit umgeben sind. So können unter Umständen auch Artefakte der Hornhautreflektion entfernt werden. Nach dem Eliminieren der Artefakte wird wieder das Komplement des Bildes gebildet.
Nun werden die Kanten der Pupille mittels Schwellwerten gesucht. Dazu wird ein oberer und ein unterer Schwellwert bestimmt. In einer bestimmten Zahl von Durchläufen werden die Helligkeitswerte abhängig von den Schwellwerten auf 0 oder 255 gesetzt. Liegt ein Pixel zwischen den beiden Schwellwerten, wird die Intensität des Pixels herunterskaliert. Als Ergebnis dieses Verfahrens erhält man ein Binärbild.
Da immer noch Artefakte um die Pupille vorhanden sein können, werden nun morphologische Operationen eingesetzt, um diese zu beseitigen. Dies bedeutet, dass erneut dunkle Pixel, die von Hellen umgeben sind, entfernt werden. Dasselbe geschieht mit schwarzen Pixeln, die nur diagonal mit einem schwarzen Pixel benachbart sind. Weiterhin werden Lücken gefüllt, sodass weiße Pixel, die von Schwarzen umgeben sind, invertiert werden.
Zur Berechnung des Pupillenradius wird nun das Bild nach zusammenhängenden Komponenten abgesucht. So werden die Pixel, die zur Pupille gehören, gefunden. Von der gefundenen Komponente wird die Größe bestimmt und ein Quadrat berechnet, das um die Komponente gelegt werden kann (eine quadratische Bounding Box). Mittels des Quadrats wird ein Kreis berechnet, der genau in das Quadrat passt. Nun wird die zusammenhängende Komponente mit dem Kreis verglichen. Ist das Vergleichskriterium (welches im Paper nicht genannt wird) erfüllt, wird die Pupille als Kreis erkannt. Der Radius und der Mittelpunkt lassen sich durch den Mittelpunkt und die Höhe des Quadrates bestimmen.

Leider wird in dem Paper nicht beschrieben, nach welchen Kriterien die Pupille mit dem Kreis verglichen wird und was zu tun ist, falls die Pupille nicht als Kreis erkannt wird. Eine Möglichkeit, diesen Algorithmus zu implementieren, wäre zum Beispiel anzunehmen, dass die Pupille immer ein Kreis ist. Das Ergebnis würde so immer dem Mittelpunkt der Bounding Box entsprechen. Andernfalls könnte man das Verfahren nach Bestimmung der Bounding Box mit einem anderen Verfahren erweitern, so dass die Pupille an einen möglichen Kreis angepasst wird. Eignen würde sich hierzu beispielsweise die Circle Approximation Methode nach Daunys und Ramanauskas.

2.1.5 Der Algorithmus nach Ohno, Mukawa und Yoshikawa

Dieser Algorithmus zur Bestimmung des Pupillenmittelpunkts ist dem Gazetracking System FreeGaze[OMY02] entnommen.
Der Algorithmus beginnt mit dem Finden der Pupille in einem Augenbild. Dazu wird das Bild segmentiert, indem Regionen aus ähnlich hellen Pixeln gefunden werden. Für jede Region wird eine Boundary Shape berechnet, um Pupillenkandidaten zu detektieren. Dabei wird ausgenutzt, dass die Helligkeitswerte der Pupille dunkler sind als die Helligkeitswerte in anderen kreisähnlichen Regionen wie zum Beispiel Spiegelungen. Bei den Augenbildern des Gazetrackers allerdings ist die Suche nach der Pupille nicht mehr vorzunehmen. Bei der Implementation kann man die Vorverarbeitung für dieses Verfahren dementsprechend auslassen.
Das Verfahren zur Bestimmung des Mittelpunkts wird hier Double Ellipse Fitting genannt. Dabei wird das Ellipse Fitting Verfahren zweimal angewendet, um ein optimales Ergebnis zu erhalten. Im Detail sieht dies so aus:
Zuerst werden die Kanten der Pupille gesucht, indem Pixel radial vom Zentrum des Pupillenkandidats aus gescannt werden. Mittels eines Schwellwerts werden die Kantenpunkte der Pupille gefunden. Für diese Kantenpunkte wird nun ein Ellipse Fitting angewendet, um einen neuen Mittelpunkt der Pupille festzulegen. Von diesem Mittelpunkt aus sendet man diesmal feinere, also circa doppelt so viele, Strahlen aus und es wird erneut getestet, ob die Punkte, die der Kontur zugeordnet wurden, wirklich auf der Kante liegen. Pixel,

die weit von der Kante entfernt liegen, werden nicht mehr betrachtet. Abschließend wird ein zweites Ellipse Fitting durchgeführt, indem endgültig der Mittelpunkt der Pupille berechnet wird. Abbildung 3 stellt den Verlauf dieses Verfahrens grafisch dar.
Probleme könnte die Hornhautreflektion bei diesem Verfahren machen. Durch sie können bei Anwenden des Schwellwerts auf das Bild Artefakte außerhalb der Pupille mit gleichem Helligkeitswert entstehen. Diese können einerseits durch morphologische Operatoren beseitigt werden, andererseits kann man den Suchraum um den Mittelpunkt des Bildes mittels eines bestimmten Radius beschränken. So werden außerhalb der Pupille liegende Kantenkandidaten gar nicht detektiert.

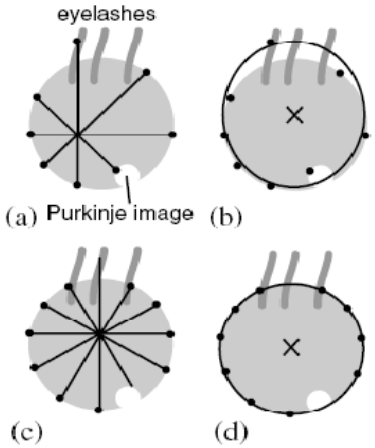

Figure 5: Double ellipse fitting, which consists of four steps: (a) rough edge detection, (b) temporal fitting of ellipse, (c) detailed edge detection, and (d) final fitting of ellipse.

Bild 2.3: Verlauf des Algorithmus nach Ohno, Mukawa und Yoshikawa [OMY02]

2.2 Verfahren zur Bestimmung des Irismittelpunktes

Verfahren zum Finden des Irismittelpunktes werden oft als Alternative zur Bestimmung des Pupillenmittelpunkts genutzt, da sie auch bei sichtbarem Licht eingesetzt werden können. Eine besondere Bedeutung bekommen solche Verfahren in der Biometrie, in der man aufgrund des fehlenden Infrarotlichts nicht den Dark-Pupil-Effect ausnutzen kann. Oft wird aber auch hier der Rand der Pupille und der Iris bestimmt, um die zur Iris gehörenden Merkmale von dem Rest des Auges zu unterscheiden. So kann man im Anschluss mittels verschiedener Verfahren eine Iriserkennung durchführen.
Die nachfolgenden Verfahren sind für die Studienarbeit demnach ebenso für Pupillendetektion einsetzbar, da sie sich durch minimale Änderungen leicht für die Zwecke des Gazetrackers anpassen lassen. Ebenso werden in diesem Abschnitt Algorithmen mit Templates beschrieben, da diese auch oft zur Bestimmung des Irismittelpunktes verwendet werden, jedoch für die Augenbilder des Gazetrackers eher ungeeignet sind.

2.2.1 Der Algorithmus nach Li und Parkhurst

Bei diesem Algorithmus [LWP06] handelt es sich grundsätzlich um den Starburst-Algorithmus von den gleichen Autoren, der für das Finden der Iriskontur leicht abgeändert wurde.
Auch hier werden von einem potentiellen Mittelpunkt Strahlen ausgesandt, die mittels eines Schwellwerts Featurepunkte finden sollen. Allerdings werden hier pro Strahl direkt zwei Featurepunkte für die Iris- und die Pupillenkontur gesucht. Auch ist der Grad, in dem die Strahlen ausgesendet werden begrenzt. Die Strahlen werden so nur in dem Bereich von -45 bis 45 und 135 bis 225 Grad ausgesendet. Diese Beschränkung soll dafür sorgen, dass die Augenlider, die die Iris überlappen können, nicht als Featurepunkte erkannt werden.
Um Featurepunkte, die ausserhalb der Iris liegen, zu entfernen, wird ein Distance-Filter benutzt, in dem Punkte, die 1.5 Mal weiter vom Startpunkt wegliegen als der Durchschnitt, eliminiert werden.

Der Vorgang des Ellipse Fitting entspricht wiederum dem des Starburst-Algorithmus. Hier werden nur zwei Einschränkungen gemacht: der Ellipsenradius darf nur um 1.5 größer oder kleiner als die Durchschnittsdistanz sein und der Quotient aus dem größeren und dem kleineren Radius muss größer als 0.75 sein.
Da dieser Algorithmus sich nur in wenigen Zügen von dem originalen Starburst-Algorithmus unterscheidet, wird er nicht speziell für diese Studienarbeit implementiert.

2.2.2 Der Algorithmus nach Zhu und Yang

In der Methode von Zhu und Yang, die in ihrem Paper „Subpixel Eye Gaze Tracking“ [ZHY02] beschrieben wird, wird die Kontur der Pupille mit einem Kantendetektionsalgorithmus auf Subpixelniveau berechnet. Anschließend wird der Mittelpunkt der Pupille durch Ellipse Fitting bestimmt.
Bei der Kantendetektion wird davon ausgegangen, dass der Irisrand schon bestimmt wurde. Der Algorithmus zur Kantendetektion auf Subpixelniveau verläuft theoretisch folgendermaßen:
Auf jedes Kantenpixel zwischen Iris und Sklera wird der vertikale und horizontale Sobelkantenoperator angewandt, der zwei Werte liefert. Mithilfe dieser Werte berechnet man die Gradientenrichtung der Punkte. Entlang der Gradientenrichtung werden zu jedem Kantenpunkt jeweils weitere Werte mittels des Sobeloperators berechnet. Die gewonnenen Werte werden nun mit der eindimensionalen kubischen Interpolation interpoliert. Der gesuchte Kantenpunkt mit Subpixelwerten befindet sich nun an der Subpixel Position, an der der interpolierte Sobelkantenwert maximal wird.
Dieser Algorithmus kann für die Praxis modifiziert werden, so dass die Gradientenrichtung als horizontal angenommen wird, da ja nur die Punkte zwischen Iris und Sklera beachtet werden sollen. Der gesuchte y-Wert entspricht dann dem Mittelpunkt des Kantenpixels und der x-Wert wird, wie oben beschrieben, mittels kubischer Interpolation berechnet.
Hat man nun die Kantenpunkte mit Subpixelwerten erhalten, wird der Irismittelpunkt anhand von Ellipse Fitting mit der Least-Square Methode ermittelt. Damit keine Punkte

des Augenlids in das Ellipse Fitting miteinbezogen werden, werden von dem höchsten und dem niedrigsten Punkt des Augenlids Kanten zu den Augenecken gezogen. Nur Punkte, die innerhalb des entstehenden Rechtecks liegen, werden im Algorithmus berücksichtigt. Da die Augenecken in den Augenbildern des Gazetrackers nicht enthalten sind, muss diese letzte Optimierung bei der Implementation des Verfahrens ausgelassen werden. Da allerdings der Mittelpunkt des Augenbildes innerhalb der Pupille liegt und der Rand um die Pupille herum geschlossen ist, dürfte es kein Outlierproblem bei dem vorhandenen Bildmaterial geben.

2.2.3 Algorithmen mit Templates

Um den Irismittelpunkt in einem Augenbild zu finden, kann man verschiedene Templates benutzen, die man von Hand erstellt oder aus einer Datenbank entnimmt und dann mit dem Eingabebild abgleicht. Dabei gibt es unterschiedliche Möglichkeiten die Größe der Templates zu bestimmen. So kann man beispielsweise verschiedene Größen ausprobieren oder die Templategröße aus den geometrischen Informationen über das Gesicht festlegen. Im Algorithmus nach Peng, Chen, Ruan und Kukharev wird ein Template genutzt, das einem realen Augenbild entnommen wurde. Die Größe der Augen ermittelt man durch die Breite des Gesichts und dem Wissen über dessen geometrische Struktur. So kann man die Größe des Templates skalieren und damit beginnen, das Augenbild mit dem Template abzugleichen.
Zum Vergleich von Template und Augenbild werden die Helligkeitswerte der beiden Bilder miteinander verglichen. Da diese wahrscheinlich nicht exakt übereinstimmen, wird ein Unähnlichkeitswert berechnet, beispielsweise mittels Sum of Squared Errors. Man lässt das Template so über das Bild laufen und berechnet die Kreuzkorrelation zwischen Template und Bild. Das Ziel ist es, lokale Maxima oder Punkte über einem bestimmten Schwellwert zu finden. Da die Helligkeitswerte im Bild je nach Region variieren, muss dazu eine normalisierte Kreuzkorrelation genutzt werden.
Laut den Entwicklern funktioniert dieser Algorithmus auch bei unterschiedlichen Kopfausrichtungen gut. Allerdings gibt es aufgrund von Spiegelungen starke Probleme bei

dem Suchen der Iris bei Brillenträgern.
Der Algorithmus bietet in erster Linie eine Augendetektion und kann nur grob den Irismittelpunkt bestimmen. Grundsätzlich ist für die Bestimmung des Mittelpunktes ein weiterer, genauerer Algorithmus zu empfehlen.[PCR05]
Der Algorithmus nach Kawaguchi und Rizon verwendet einfache kreisförmige Templates in Verbindung mit Templates aus einer Datenbank, um die Iris zu finden. [KAR02]
Funabiki, Isogai, Higashino und Oda stellen in ihrem Paper eine Lösung vor, die Deformable Templates benutzt. Dabei modelliert das Template das gesamte Auge mit Hilfe von drei quadratischen Funktionen und einem Kreis. Zwei quadratische Funktionen bilden das obere Augenlid, die Dritte das untere Lid. Der Kreis beschreibt folglich die Iriskontur. Zur Auswertung der Funktionen werden zehn Parameter benötigt, darunter der Irismittelpunkt und der Radius. Vor der Anwendung des Templates wird der Canny Operator benutzt, um ein Kantenbild aus dem Eingabebild zu erzeugen. Dabei läuft der Template-Matching Algorithmus nur über das Augenbild. Es werden verschiedene Positionen des Kantenbilds gesucht. Aus diesen Daten werden die korrespondierenden Punkte des Templates berechnet. So kann das Template auf das Auge gelegt werden und der Radius, und damit der Mittelpunkt, kann errechnet werden.[FIH06]

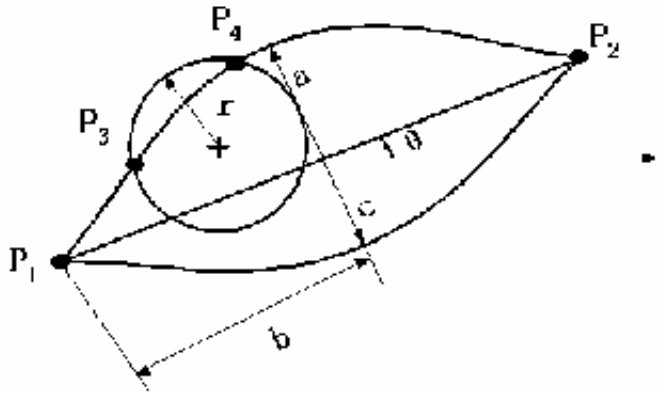

Bild 2.4: Darstellung eines verformbaren Templates [FIH06]

Der Algorithmus von Funabiki, Isogai, Higashino und Oda würde allerdings nur funktionieren, wenn ein Augenbild vorliegt, auf dem das gesamte Auge mit den Augenecken

vorliegt. Die ist beim Gazetracker nicht der Fall. Dementsprechend wäre auch dieser Algorithmus auf der Grundlage des Pupillenbildes für unsere Zwecke ungeeignet.

2.2.4 Der Algorithmus nach Daugman und ähnliche Algorithmen

Der Algorithmus [DAU02], der hier vorgestellt wird, ist ein Teil von John Daugmans Algorithmen, auf denen nahezu alle kommerziellen Iris-Erkennungs-Systeme basieren.
Daugmans Algorithmus intendiert die innere und äußere Kontur der Iris zu finden, um damit eine genaue Iriserkennung zu realisieren. Zum Finden der Iriskontur und des Radius benutzt er einen integrodifferentialen Operator unter der Annahme, dass die Iris in Kreisform ist. Der Operator sucht die Iriskontur und die Pupillenkontur gleichermaßen, so dass die Mittelpunktswerte und der Radius leicht gefunden werden können.
Der integrodifferentiale Operator wird in Formel 2.1 beschrieben.

$$\max_{r,x_0,y_0} \left| G_\sigma(r) \cdot \frac{\partial}{\partial r} \oint_{r,x_0,y_0} \frac{I(x,y)}{2\pi r} \mathrm{d}s \right| \tag{2.1}$$

Der Grundgedanke hier ist, dass am Rand von Pupille und Iris ein starker Helligkeitswechsel stattfindet, der die Kontur auszeichnet. Um zunächst die Grenze zwischen Iris und Pupille zu finden, werden ausgehend vom angenäherten Pupillenmittelpunkt Kreisintegrale mit ansteigendem Radius berechnet. Wird der Radius größer, wird folglich auch das Kreisintegral stetig größer. Wenn der Übergang zur helleren Iris stattfindet, steigt der Betrag des Kreisintegrals sprunghaft an. Das Ergebnis des Operators ist ein Radius um den Mittelpunkt (x0, y0). Mithilfe des Radius kann mittels Gradientenabstieg ein neuer, genauerer Mittelpunkt gefunden werden. Der Operator wird dann nochmal auf den neuen Mittelpunkt angewendet. Dieses Verfahren wird so oft angewendet, bis sich der neu errechnete Mittelpunkt nicht mehr stark vom alten Mittelpunkt unterscheidet.
Ist der Mittelpunkt gefunden, kann zusätzlich die Grenze zwischen Iris und Sklera gesucht werden.

Probleme wird es bei diesem Algorithmus geben, wenn die Hornhautreflektion nah an der Pupille liegt. Der Übergang von der Pupille zur Hornhautreflektion wird einen sehr starken Helligkeitsunterschied hervorrufen, wodurch der Daugmanalgorithmus annehmen wird, dass der Pupillenmittelpunkt dort ist, wo der Kreisrand des Punktes durch die Grenze zwischen Pupille und Hornhautreflektion wandert. Somit kann sich in solchen Fällen der berechnete Mittelpunkt sehr stark vom realen Mittelpunkt unterscheiden.
Es gibt einige Algorithmen, die dem von Daugman sehr ähneln, da sie ebenfalls mit einem Operator arbeiten, der maximiert werden muss und so den Irismittelpunkt finden soll. Solche Formeln findet man unter Anderem bei Camus und Wilde und Martin-Roche et al.. [PRA06]
All diese Operatoren basieren wie Daugman auf der Annahme, dass Iris und Pupille kreisförmig sind und es eine große Helligkeitsveränderung beim Übergang von Pupille zu Iris und Iris zu Sklera gibt.

2.3 Verfahren zur Bestimmung des Mittelpunkts der Hornhautreflektion

Bei den Verfahren zur Glint Detection und der Bestimmung des Hornhautreflektionsmittelpunkts wird oft der Schwerpunkt der Pixel als Mittelpunkt genommen. So ist es im Gazetracker implementiert. Zusätzlich kann man beim Aufsummieren der Pixel Gewichte abhängig von der Helligkeit verwenden, wodurch ein genauerer Mittelpunktswert gefunden werden kann. Speziell die Gewichtung der Helligkeitswerte wurde im Rahmen dieser Studienarbeit implementiert. Dennoch seien hier zwei Algorithmen genannt, die die Bestimmung des Mittelpunkts der Hornhautreflektion beinhalten.

2.3.1 Der Algorithmus nach Li, Winfield und Parkhurst

In dem Starburst Algorithmus [LWP05] wird ein Verfahren zur Entfernung der Hornhautreflektion vorgestellt. Die Reflektion wird dabei nur in der Region der Iris gesucht und zunächst durch das Setzen eines maximalen Schwellwerts lokalisiert. Dieser berücksichtigt, dass die Hornhautreflektion eine der hellsten Regionen im Bild ist. Der Schwellwert wird dann langsam heruntergesetzt und der Quotient zwischen dem größten Reflektionskandidaten und den Restkandidaten wird berechnet. Die Hornhautreflektion wird zum Rand hin in ihrer Intensität schwächer. Von daher wächst sie beim Heruntersetzen des Schwellwerts stärker an als andere Punkte. Dieses Vorgehen endet, wenn der Quotient zwischen den gefundenen Bereichen am Größten ist. Als Mittelpunkt der Hornhautreflektion wird der geometrische Mittelpunkt des größten Bereichs unter diesem Schwellwert genommen. Der Radius der Hornhautreflektion kann zusätzlich mit einer Gradientensuche berechnet werden.

2.3.2 Der Algorithmus nach Mulligan

Bei diesem Algorithmus [MUL95] wird zunächst ein Schwellwert auf das weichgezeichnete Bild angewendet, sodass nur sehr dunkle Pixel selektiert werden. Die Hornhautreflektion stellt ein Loch in dem Ergebnisbild dar. Das Bild wird noch einmal geblurrt und zu einem Wertebereich von 0 bis 1 renormalisiert. Pixel über einem gewissen Schwellwert werden nun gesucht. So wird eine Maske erstellt, die auf das Originalbild angewendet wird, um zur Pille gehörige Pixel auszusortieren.
Als Nächstes wird die Hornhautreflektion gefunden und danach eliminiert. Dazu wird ein Filter, der einen hellen Kreis mit dunkler Umgebung sucht, zur groben Lokalisation benutzt. Dieser Filter wird auf das Bild angewandt, das danach renormalisiert wird. Daraufhin werden die Pixel mit einem hohen Helligkeitswert selektiert. Eine Kreismaske wird dann über diese Pixel gelegt. Der Mittelpunkt wird folgendermaßen bestimmt: Ein Bild wird erstellt, indem der Wert jedes Pixels seiner x-Koordinate entspricht. Dieses Bild wird mit dem Bild der Hornhautreflektion pixelweise multipliziert. Die Summe dieser Berechnung wird gebildet. Diese Summe wird normalisiert durch die Summe des Hornhautreflektionsbildws und so wird die x-Koordinate des Mittelpunktes errechnet. Dieses Verfahren ist nur bei einem Grauwertbild sinnvoll, da sonst nur die durchschnittliche x-Koordinate berechnet wird. Zudem geht der Autor hier davon aus, dass der Glint kreisförmig ist, was nicht notwendigerweise der Fall sein muss.

Kapitel 3

Implementation und Bewertung der Algorithmen

3.1 Implementation der Algorithmen

Von den beschriebenen Algorithmen wurden die acht folgenden Algorithmen implementiert:

- der Starburst-Algorithmus für die Bestimmung des Pupillenmittelpunktes nach Li, Winfield und Parkhurst,
- der Coordinates Average und der Circle Approximation Algorithmus nach Daunys und Ramanauskas,
- der Algorithmus nach Perez, Garcia. Mendez, Munoz, Pedraza und Sanches,
- der Bounding-Box-Algorithmus nach Poursaberi und Araabi,
- der Freegaze-Algorithmus nach Ohno, Mukawa und Toshikawa,
- der Subpixel Eye Gaze Tracking Algorithmus von Zhu und Yang

- und der Daugman-Algorithmus.

Desweiteren wurde auch der aktuelle Algorithmus, der den Schwerpunkt der Pupille als Mittelpunkt setzt, ausgewertet und in Relation zu den anderen Algorithmen gesetzt. Für die Implementation wurde jeder Algorithmus in eine eigene Klasse geschrieben. Die verschiedenen Klassen sind im Ordner „Algorithmen“ im SVN im Gazetrackerverzeichnis zu finden. Ebenso befinden sich verschiedene Ordner in diesem Verzeichnis, die die Ergebnisse zu den verschiedenen Testserien enthalten sowie eine kurze Beschreibung der Sequenzen in einem Tabellendokument.
Aufgerufen werden die verschiedenen Algorithmen in der Klasse „Algorithms“, die ihrerseits in der Klasse Eyedata parallel zu dem ursprünglich implementierten Algorithmus aufgerufen wird. Zum Aufruf der Klassen muss man dabei das Augenbild mit Schrittweite und Höhe, den Helligkeitsdurchschnitt der Pupille und den Pupillenradius übergeben. Der Helligkeitsdurchschnitt und der Pupillenradius wurden dabei zuvor schon im Gazetracker berechnet und sind auch die Übergabeparameter des Originalalgorithmus. Dabei ist vor allem der Pupillenradius ein hilfreicher Wert zur Begrenzung des Suchraums. Sucht man nur innerhalb dieses Radius nach Kantenpunkten der Pupille, verhindert man, dass die Kantenpunkte der Iris als Bestandteile des gefundenen Kantenpunktsets erkannt werden, wodurch man die Algorithmen in ihrer Vorverarbeitung vereinfachen kann, da keine Pupille mehr gesucht werden muss und das Problem von Outliern nicht mehr auftreten kann.
Für die Implementation wurden einige Algorithmen leicht abgeändert beziehungsweise um positive Eigenschaften anderer Algorithmen erweitert. So wird die Subpixelkantendetektion, wie sie von Zhu und Yang beschrieben wird, auch bei den Algorithmen Daunys und Ramanauskas und dem Algorithmus nach Poursaberi und Araabi eingesetzt, um bei diesen vergleichsweise einfachen Algorithmen durch die Kantendetektion ein besseres Ergebnis zu erlangen. Dazu wurde eine Funktion implementiert, die aus den vier horizontal und vertikal umliegenden Punkten eines Pixels eine kubische Interpolation des Helligkeitswertes berechnet und mittels Auflösung der kubischen Gleichung den reellen Wert der Pixelposition auf Subpixelniveau zurückgibt.
Ebenso wurde der Algorithmus zum Füllen der Lücken nach Poursaberi und Araabi bei

der Coordinates Average Methode nach Daunys und Ramanauskas angewandt, da hier eine starke Abweichung des Mittelpunkts bei Artefakten festgestellt wurde.
Generell boten sich für die einfachen Algorithmen verschiedene Methoden zur Bestimmung der Kantenpunkte an. Einmal kann man den Kantendetektionsalgorithmus von Canny einsetzen, bei einer geringen Anzahl von Pixeln, wie sie im Gazetracker vorkommen, kann man allerdings auch Randpunkte über den Schwellwert suchen. Dabei ist das Starburst- Verfahren, in dem man vom Mittelpunkt aus Strahlen zu den Randpunkten aussendet, um diese durch Helligkeitsdifferenzen zwischen den verschiedenen Pixeln zu finden, bei den einfachen Algorithmen ein relativ uneffizientes und zu komplexes Verfahren. Gerade bei den einfachen Verfahren ergab der Algorithmus von Canny die besten Ergebnisse, da hier keine Abhängigkeit zu einem Schwellwert besteht.
Für das Ellipse Fitting, das in drei Algorithmen implementiert werden musste, wurde die Funktion „cvFitEllipse2(...)“ aus OpenCV verwendet, die sechs Randpunkte als Eingabeparameter braucht. Dementsprechend wurde im Starburst- Algorithmus die Anzahl der Punkte, die mittels RANSAC bestimmt wurden, von fünf auf sechs erhöht.
Bei schlechter Beleuchtung kann es dennoch passieren, dass die Kantendetektion nicht ausreichend viele Kantenpunkte für die Algorithmen findet. Wenn die Testperson zu weit vom Bildschirm entfernt ist, so dass die Pupille sehr klein aufgenommen wird, werden ebenso ungenügend viele Kantenpunkte gefunden. Die Algorithmen, die Ellipse Fitting benutzen, brauchen, wie bereits beschrieben, eine Basis von sechs Punkten zur Bestimmung einer Ellipse. Wenn keine sechs Kantenpunkte gefunden werden, schlägt die Mittelpunktsbestimmung fehl. In solch einem Fall könnte man einen Defaultwert als Mittelpunkt ausgeben (zum Beispiel der Bildmittelpunkt) oder einen Fehlerwert. Von einem Defaultwert ist allerdings eher abzuraten, da so die Ergebnisse der Mittelpunktssuche verfälscht werden und die Blickrichtung nicht richtig bestimmt werden kann.

3.2 Bewertung der Algorithmen

3.2.1 Bewertungsmethoden

Zur Bewertung der Algorithmen wurden von Wladimir Krebs verschiedene Aufnahmen von Personen in Interaktion mit dem Gazetracker gemacht. Dabei wurde darauf geachtet, dass die Aufnahmen die Fälle abdecken, die bei der Benutzung des Gazetrackers auftreten können. So wurden Personen mit und ohne Brille gefilmt, desweiteren gab es drei unterschiedliche Szenarien, die die Testpersonen anschauen sollten. Diese Szenarien beinhalten das Fixieren auf einen Punkt, das Verfolgen eines Punktes und das Anschauen einer Website, was das Anschauen vieler weit verteilter Punkte bedeutet.
Diese Aufnahmen wurden in den Gazetracker geladen, um die verschiedenen Algorithmen darauf anzuwenden. Die Ergebnisdaten wurden daraufhin ausgelesen und gespeichert. Zu den Daten gehören die berechneten Pupillenmittelpunkte und die Hornhautreflektionsmittelpunkte des jeweils linken und rechten Auges. Die Grundeinstellung für die Augensuche und die Suchmaske waren für alle Sequenzen gleich.
Möchte man die Ergebnisse bewerten, kommen verschiedene Methoden in Frage. Eine davon ist, für die Auswertung der Ergebnisse den Vektor zwischen Pupillenmittelpunkt und Hornhautreflektionsmittelpunkt zu bilden und die Ergebniswerte grafisch darzustellen. Praktisch gesehen wird mit dieser Methode angenommen, dass der Hornhautreflektionsmittelpunkt sich in der Mitte des unteren Bildschirms bei der Kamera befindet und demnach sind die Punkte des Differenzvektors die Punkte, die auf dem Bildschirm betrachtet werden. Dies ist natürlich kein genaues Ergebnis, da für die Berechnung der Bildschirmpunkte noch weitere Methoden benötigt werden, aber es ist eine vereinfachte Darstellung des Sachverhaltes, so dass man die Algorithmen miteinander vergleichen kann und schnell sieht, wie sinnvoll die Ergebnispunkte sind. Auf den Aufnahmen der Testpersonen ändert sich die Position des Hornhautreflektionsmittelpunktes nur minimal, während die Pupillenposition sich ändert. So lässt sich mittels des Vektors herausfinden, ob die Pupille beim Ansehen des Bildschirms gewandert ist und in welche Richtung sie sich bewegt hat.
Im folgenden Abschnitt werden die Ergebnisse, die mit dieser Bewertungsmethode erlangt wurden, vorgestellt.

3.2.2 Auswertung der Ergebnisse

Grundsätzlich sollten sich die Algorithmen in ihren Ergebnissen nur um Subpixel unterscheiden. Plottet man dementsprechend den vorhin beschriebenen Vektor, so müssen auch die grafischen Ergebnisse übereinstimmen. Als Vergleichsgrundlage dient hier der Originalalgorithmus aus Thorsten Geiers Gazetracker, von dem sich die restlichen Algorithmen nur minimal unterscheiden dürften.

Bei dem optischen Vergleich der Tests haben die Algorithmen von Daugman, der Coordinates Average Algorithmus von Daunys und Ramanauskas und der Starburst - Algorithmus schlecht abgeschnitten und haben selten Ergebnisse geliefert, die mit den anderen Algorithmen übereinstimmten.
Bei dem Fixieren eines Punktes beispielsweise liefert der Originalalgorithmus das Ergebnis, welches in Abbildung 3.1 gezeigt wird.

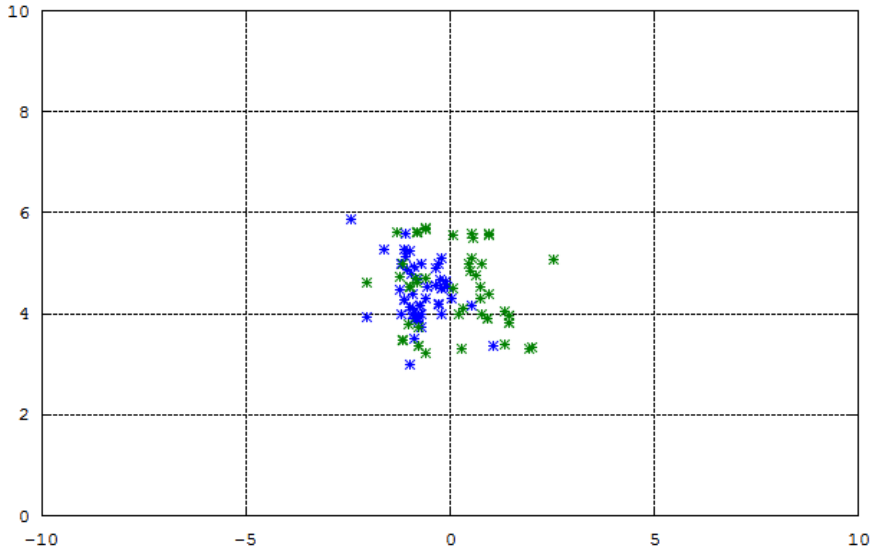

Bild 3.1: Ergebniswerte des Originalalgorithmus in Gnuplot

Die Grafik stellt die Punkte des Vektors zwischen Pupillenmittelpunkt und Hornhautre-

flektionsmittelpunkt für beide Augen (links: blau, rechts: grün) dar. Man erkennt, dass sich der Betrachter einen Punkt in der Bildschirmmitte angesehen hat, die Abweichungen entsprechen den Sakkaden des Auges.
Das Ergebnis des Starburst-Algorithmus sieht man dagegen in Abbildung 3.2.

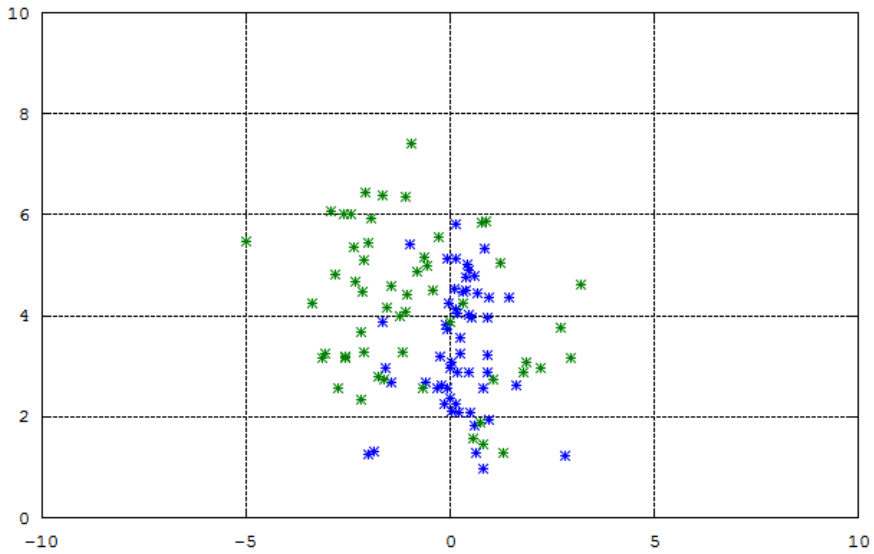

Bild 3.2: Ergebniswerte des Starburst-Algorithmus in Gnuplot

Hier erkennt man, dass sich die berechneten Punkte zwar grob an einer Stelle des Bildschirms befinden, aber die Streuung der Punkte sehr viel größer ist als im Originalplot. Die Coordinates Averaging Methode von Daunys und Ramanauskas liefert ebenso ein stark abweichendes Ergebnis, wie in Abbildung 3.3 zu sehen ist.

Die Ergebniswerte des Algorithmus von Daugman sind in Abbildung 3.4 dargestellt.

Ebenso kann man ein abweichendes Verhalten der Algorithmen erkennen, wenn man die Grafiken zu den Sequenzen betrachtet, in der Punkte verfolgt werden (Abbildung 3.5).

An den Bildern ist gut zu erkennen, dass der Starburst-Algorithmus und der Average

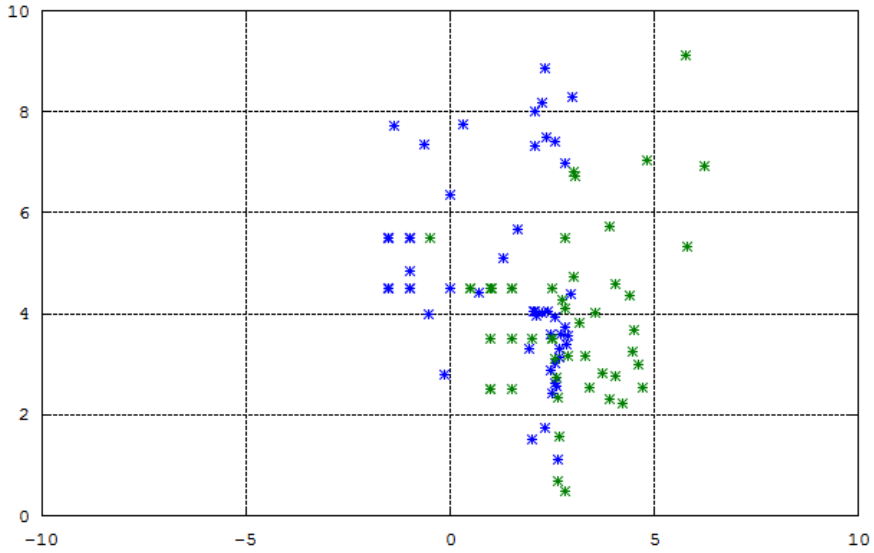

Bild 3.3: Ergebniswerte des Algorithmus von Daunys und Ramanauskas in Gnuplot

Coordinates Algorithmus von der Struktur her dem Originalbild noch nahe kommen, während Daugmans Algorithmus nicht annähernd ähnliche Ergebnisse liefert.
Daugman ist von allen Algorithmen wohl der aufwändigste Algorithmus, der sehr schnell falsche Ergebnisse liefert, wenn die Hornhautreflektion nah an der Pupille liegt. Ursprünglich wurde Daugmans Algorithmus für die Irisdetektion entwickelt, also für Bilder mit starken Helligkeitsunterschieden zu den Bildern, die der Gazetracker übergibt. Wenn der Algorithmus bei den Gazetrackerbildern nun nach den Pupillenmittelpunkten sucht, wird er immer die Stelle finden, wo die Hornhautreflektion auf dem Umkreis des Pupillenpunktes liegt. Dies ist meist nicht der Mittelpunkt der Pupille, sondern ein Randpunkt, der nahezu genau über dem Reflektionspunkt liegen müsste. So ist auch die relativ symmetrische Anordnung auf den geplotteten Ergebnisbildern zu erklären.
Die schlechten Ergebnisse des Starburst-Algorithmus sind durch die niedrige Auflösung der Bilder in Kombination mit dem RANSAC Algorithmus zu erklären. Ähnliche Algorithmen, die entweder die gleiche Kantensuche oder Ellipse Fitting gebrauchten, haben ohne RANSAC sehr viel bessere Ergebnisse gebracht.
Der Algorithmus von Daunys und Ramanauskas könnte ebenso unter der Hornhautreflektion leiden, die, wenn sie nah an der Pupille liegt, die gefundenen Werte für die

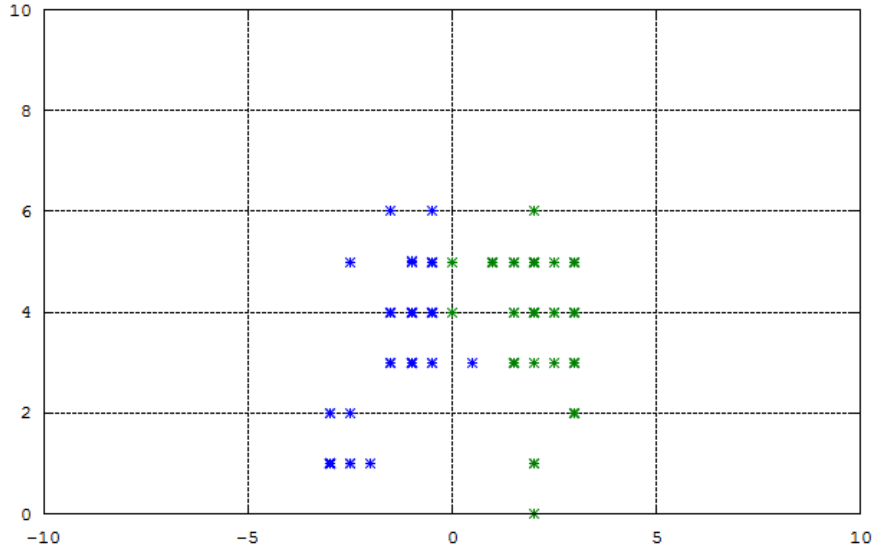

Bild 3.4: Ergebniswerte des Algorithmus von Daugman in Gnuplot

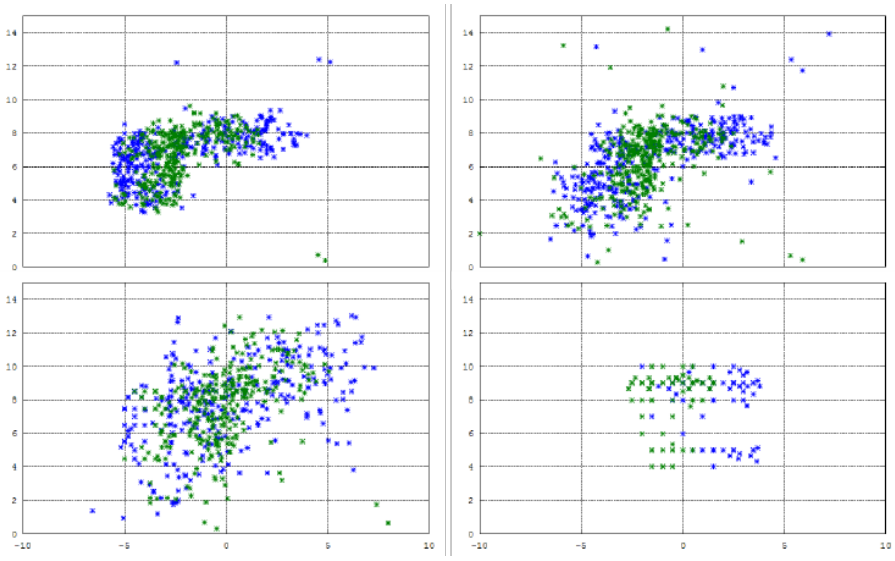

Bild 3.5: Ergebniswerte des Originalalgorithmus, des Starburst Algorithmus, des Algorithmus von Daunys und Ramanauskas und des Algorithmus von Daugman für eine Sequenz mit Punktverfolgung in Gnuplot (von links oben nach rechts unten gelistet)

Berechnung verfälschen könnte. Wenn bei dem Coordinates Average Algorithmus nur ein gefundener Kantenpunkt ein stark abweichendes Verhalten zeigt, ändert dies die berechneten Gleichungen für den Pupillenmittelpunkt sehr stark.

Die Ergebniswerte für die restlichen Algorithmen sieht man in Abbildung 3.6.

Diese Bilderserien unterscheiden sich in Einzelheiten. Dabei ist zu beachten, dass auch außerhalb des dargestellten Koordinatenbereichs noch Punkte zu finden sind, die dadurch entstehen, dass von den jeweiligen Algorithmen zwar ein Auge aber kein Pupillenmittelpunkt gefunden wird.
Ebenso erkennt man in dieser Serie zwar eine Richtung, in die sich das Auge bewegt, aber es zeichnet sich keine klare, einfache Linie ab, wie sie der Punkt, der verfolgt wurde, auf dem Bildschirm abgelaufen ist. Das liegt daran, dass der Punkt recht langsam über den Bildschirm lief und dadurch viele Sakkaden entstanden sind.
Für die Auswertung der Ergebnisse der besseren Algorithmen wurde nun beispielgebend eine Sequenz genommen, in der der Betrachter den Punkt etwas schneller verfolgt, sodass auf den Plotergebnissen eine Linie erkennbar ist, die allerdings immer noch durch Sakkaden gestört wird, was man bei den Aufnahmen mit dem Gazetracker aber kaum verhindern kann. Aus dem Anfangs- und Endpunkt der Linie kann man dann ungefähr eine Ideallinie errechnen und daraus erkennen, wie abweichend die umliegenden Punkte von dieser Linie sind. Um bessere Ergebnisse zu bekommen, wird dieses Verfahren für das linke und rechte Auge getrennt durchgeführt. Grafisch dargestellt sind die Ergebnisse des Plots in Abbildung 3.7.

Berechnet wurde die durchschnittliche Abweichung der Punkte in die vertikale Richtung. Die Ergebnisse werden im Folgenden tabellarisch angezeigt.

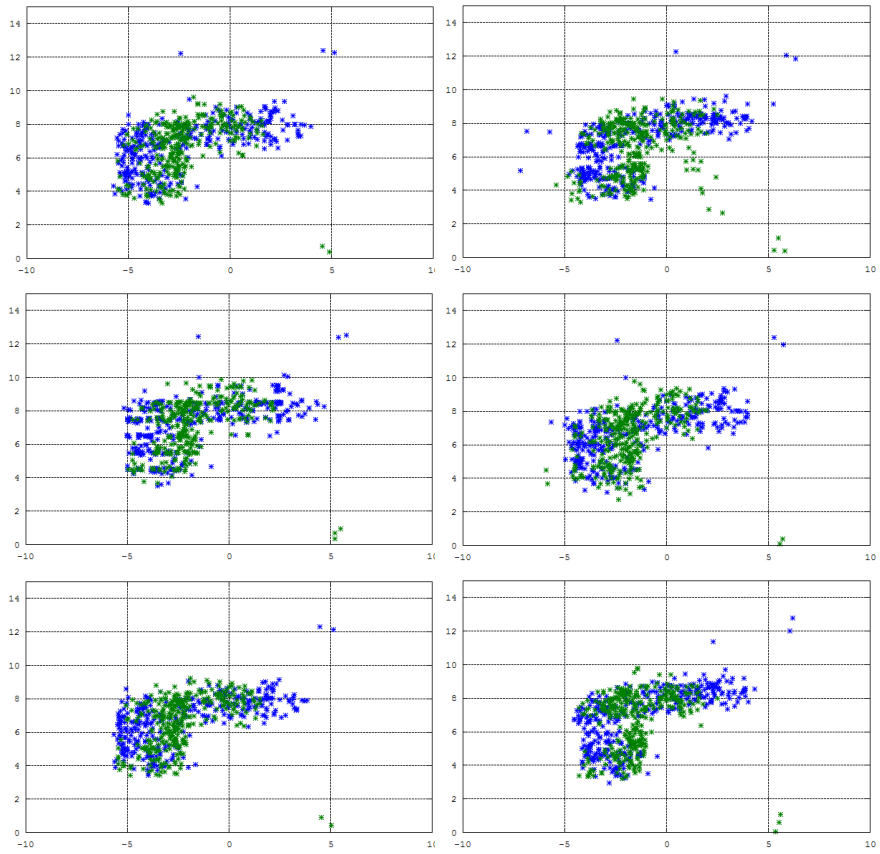

Bild 3.6: Ergebniswerte des Originalalgorithmus, des Algorithmus von Zhu und Yang, des Algorithmus von Poursaberi und Araabi, des Algorithmus von Perez et al., des Algorithmus von Ohno, Mukawa und Toshikawa und des Circle Approximation Algorithmus für eine Sequenz mit Punktverfolgung in Gnuplot (von links oben nach rechts unten gelistet)

Algorithmus	Abweichung linkes Auge	Abweichung rechtes Auge
Originalalgorithmus	1.4508	1.1996
Algorithmus nach Zhu und Yang	1.8148	1.7803
Algorithmus nach Poursaberi und Araabi	1.3013	1.0860
Algorithmus nach Perez, Garcia. Mendez, Munoz, Pedraza und Sanches	1.3842	1.2090
Algorithmus nach Ohno, Mukawa und Yoshikawa	1.2832	1.2341
Circle Approximation Algorithmus nach Daunys und Ramanauskas	1.2898	1.3303

Schnell erkennbar ist, dass der Algorithmus von Zhu und Yang eine höhere durchschnittliche Abweichung aufweist als die anderen Algorithmen. Zudem enthalten die von Zhu berechneten Werte einige Daten, die weit außerhalb des Bereichs der Pupille liegen, was auf eine unkorrekte Berechnung durch das Ellipse Fitting zurückzuführen ist. Dagegen schneidet der Algorithmus von Ohno, Mukawa und Yoshikawa, der ein doppeltes Ellipse Fitting nutzt, besser ab und weist auch bei den generell berechneten Werten der Pupillenmitte keine großen Abweichungen oder Besonderheiten auf. Beim Testen des linken Auges der obigen Serie erhält der Freegaze Algorithmus von Ohno, Mukawa und Yoshikawa sogar im Vergleich die geringste Abweichung.
Bei den Ergebnissen für das rechte Auge dagegen schneidet der einfache Bounding-Box Algorithmus von Poursaberi und Araabi vergleichsweise am besten ab. Wie man auf den Bildern auch erkennen kann, liegen die Punkte hier auch relativ nah aneinander und der Abstand jedes Punktes vom Durchschnittsmittelpunkt ist geringer als bei den anderen Algorithmen. Somit wäre diese vergleichsweise triviale Implementation des Algorithmus

eine gute Alternative zur Ermittlung des Schwerpunkts. Das Verfahren von Ohno dagegen erscheint etwas komplexer, und es ist anzunehmen, dass die Punkte, die hier berechnet werden, eher dem korrekten Pupillenmittelpunkt entsprechen würden, als der Bounding Box Algorithmus. Nichtsdestotrotz weisen die anderen Algorithmen außer Zhus und Yangs Algorithmus in diesem Testbeispiel ebenso gute Ergebnisse auf. Dabei tritt im Originalalgorithmus öfter der Fall auf, dass ein Mittelpunkt nicht korrekt gefunden wird und somit ein falscher Wert ausgegeben wird, was bei den anderen Algorithmen nicht der Fall ist. Dies dürfte der Grund für das relativ hohe Ergebnis der Abweichung des linken Auges sein.

Betrachtet man noch weitere Testsequenzen, fallen bei einigen Algorithmen allerdings noch bestimmte Probleme im Zusammenhang mit einzelnen Sequenzen auf. Der Freegaze Algorithmus, der im obigen Fall sehr gut abschneidet, weist so im grafischen Vergleich mit den anderen Algorithmen oft eine Verschiebung der Punkte des rechten und linken Auges auf. So überlappen sich diese Punkte auf dem geplotteten Bild oft nicht, wie es der Fall sein sollte und auch bei den anderen Algorithmen gegeben ist. Deswegen wäre der Algorithmus von Ohno trotz der guten Ergebnisse in dem obigen Test keine gute Alternative zur jetzt implementierten Berechnung des Schwerpunkts, da er in Einzelfällen doch ein abweichendes Verhalten zeigt. Der Average Coordinates Algorithmus von Daunys und Ramanauskas zeigt in den geplotteten Bildern stets wenig abweichendes Verhalten, enthält aber stellenweise fehlerhafte Werte ähnlich wie der Originalalgorithmus.

Der Algorithmus nach Perez et al. und der Algorithmus nach Poursaberi und Araabi zeigen auch bei den grafischen Ergebnissen ein gutes Verhalten und enthalten oft auch weniger abseits liegende Punkte als der Originalalgorithmus. In beiden Algorithmen liegen die Punkte, die gefunden werden, nah beieinander und weisen eine dichte Verteilung auf, was für die Algorithmen spricht. In dem obigen Test schneidet der Bounding Box Algorithmus etwas besser ab, Perez entspricht dagegen den durchschnittlichen Abweichungswerten. Somit wären beide Algorithmen als Alternative zur Bestimmung des Schwerpunkts denkbar.

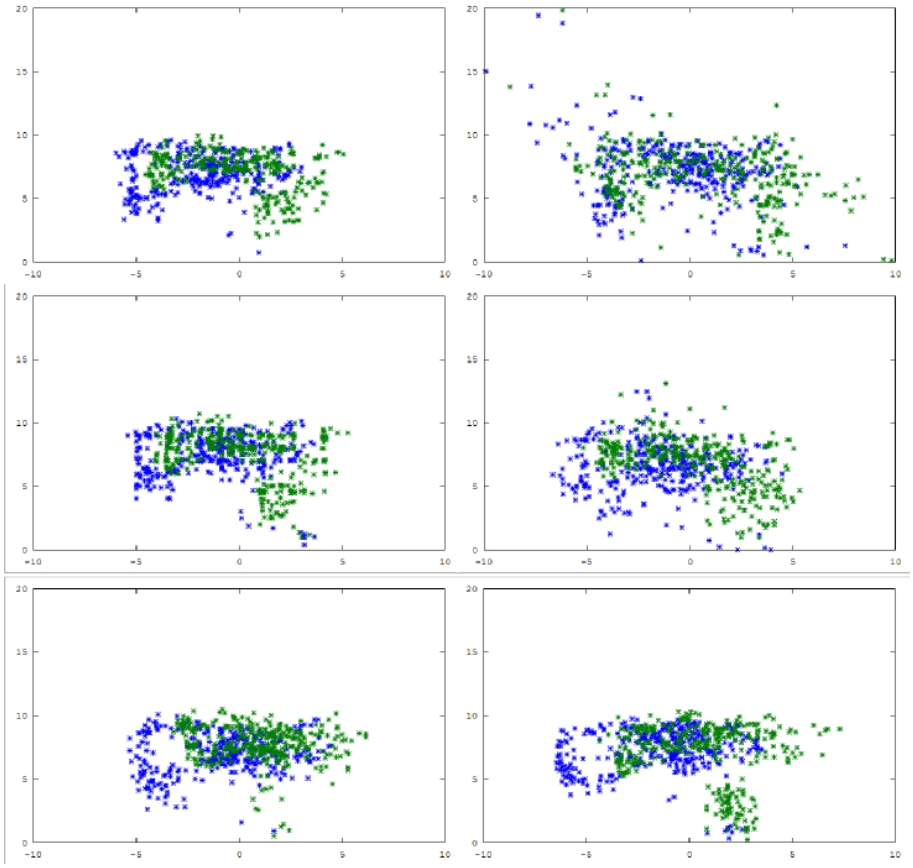

Bild 3.7: Ergebniswerte des Originalalgorithmus, des Algorithmus von Zhu und Yang, des Algorithmus von Poursaberi und Araabi, des Algorithmus von Perez et al., des Algorithmus von Ohno, Mukawa und Toshikawa und des Circle Approximation Algorithmus für eine Sequenz mit Punktverfolgung in Gnuplot (von links oben nach rechts unten gelistet)

Kapitel 4

Zusammenfassung und Ausblick

Im Rahmen dieser Studienarbeit wurden acht verschiedene Algorithmen unterschiedlichen Umfangs und Komplexität zur Pupillenmittelpunktssuche implementiert und im Vergleich mit dem Originalalgorithmus ausgewertet. Die Berechnung des Hornhautreflektionsmittelpunkts wurde modifiziert, so dass die Helligkeitswerte der Hornhautreflektion bei der Berechnung des Schwerpunkts gewichtet werden.
Bei der Auswertung wurde festgestellt, dass drei der acht Algorithmen, der Starburst-Algorithmus für hochauflösende Bilder, Daugmans Algorithmus für Aufnahmen bei sichtbarem Licht und der Average Coordinate Algorithmus von Daunys und Ramanauskas, Mängel in Zusammenhang mit dem gegebenen System aufweisen, so dass diese momentan nicht für die Mittelpunktssuche im Gazetracker geeignet sind. Möglich wäre allerdings eine Modifizierung der Algorithmen oder eine bessere Filterung des Bildmaterials vor Einsatz der Algorithmen, so dass die Hornhautreflektion keine Fehlerquelle mehr darstellen können. Allerdings sind die Algorithmen gegenüber den einfacheren Algorithmen immer noch mit mehr Aufwand verbunden und somit nicht für den Gazetracker zu empfehlen.
Die restlichen Algorithmen zeigten im grafischen Vergleich ähnlich gute Ergebnisse und wurden im Test verglichen, wobei der Algorithmus von Perez, Garcia, Mendez, Munoz, Pedraza und Sanches und der Algorithmus von Poursaberi und Araabi die besten Ergebnisse aufwiesen in Bezug auf Dichte der Punkte, Fehlerpunkte und Outlier. So wären diese als gute Alternative zur einfachen Schwerpunktberechnung zu sehen, die ebenso gute Er-

gebnisse liefert, aber stellenweise keinen Mittelpunkt findet, so dass abweichende Ergebnisse entstehen können. Ebenso hat der Circle Approximation Algorithmus von Daunys und Ramanauskas gute Ergebnisse geliefert, während die Algorithmen von Zhu und Yang und von Ohno, Yoshikawa und Mukawa bei einzelnen Sequenzen von starken Abweichungen belastet wurden.
In Zukunft könnte man die Algorithmen noch verbessern durch eine Entfernung der Hornhautreflektion vor der Mittelpunktssuche oder durch Einsetzen von Punkttrackern, die die Position der Pupillenmitte im nächsten Frame vorhersagen können. Eine Entfernung der Reflektion wäre beispielsweise denkbar durch einen erweiterten Medianfilter, der Punkte über einem bestimmtem Schwellwert nicht mehr in seine Berechnung einzieht. Ebenso denkbar wäre eine Bildreparatur mittels der Fouriertransformation. Als Tracker für den Mittelpunkt wäre die Implementation eines Partikelfilters denkbar, wie er beispielsweise in einem weiteren Paper von Hansen[HAP05] beschrieben wird. In weiterführender Literatur wird ebenso der Einsatz eines Extended Kalman Filters beschrieben, dessen Implementierung für den Gazetracker möglich wäre.

4.1 Anhang

Vergleich des Algorithmus von Ohno, Mukawa und Yoshikawa mit dem Originalalgorithmus. Die linke Abbildung entspricht dem Originalergebnis, die Rechte dem des Algorithmus' von Ohno, Mukawa und Yoshikawa.

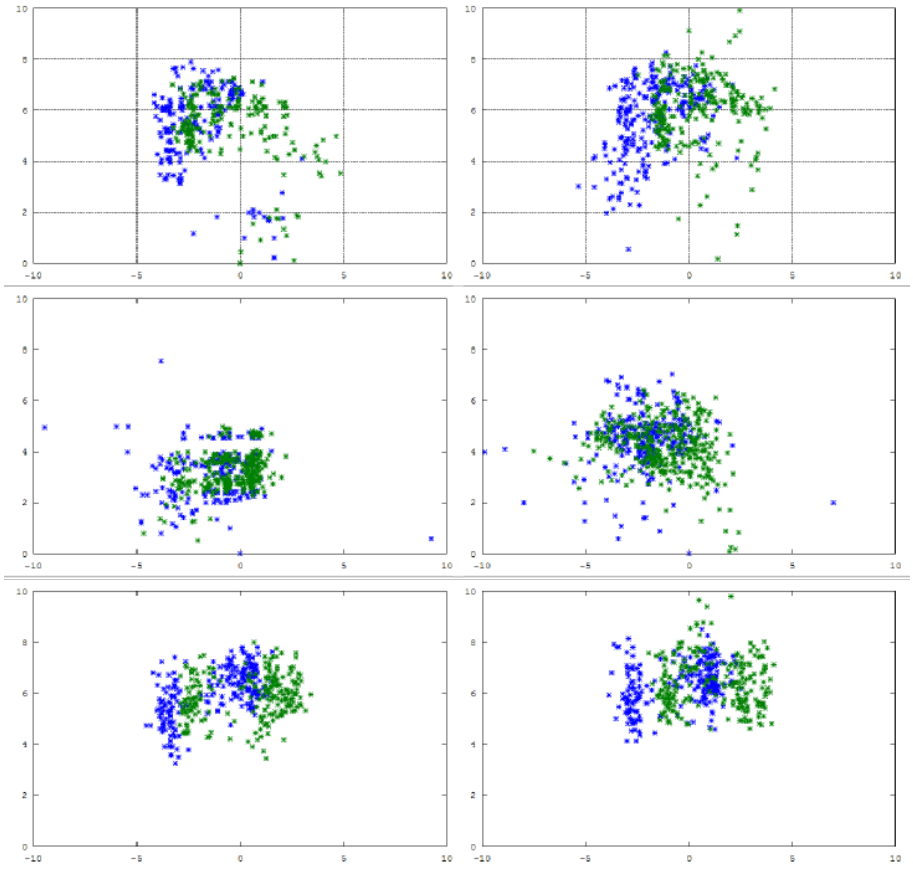

Bild 4.1: Von oben nach unten:
Serie 240108: Verfolgung eines Punktes auf dem Bildschirm
Serie 250108: Betrachten der Website der Universität
Serie 240108-3: Betrachten der Website der Universität

Vergleich des Algorithmus von Zhu und Yang mit dem Originalalgorithmus. Die linke Abbildung entspricht dem Originalergebnis, die Rechte dem des Algorithmus' von Zhu und Yang.

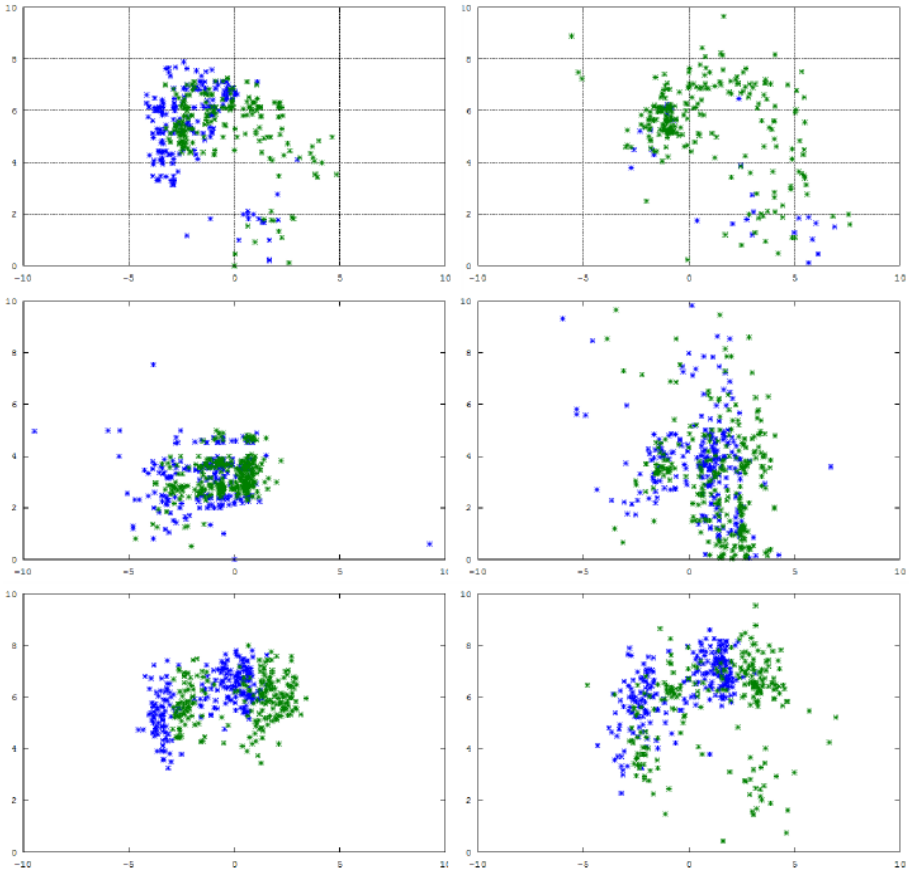

Bild 4.2: Von oben nach unten:
Serie 240108: Verfolgung eines Punktes auf dem Bildschirm
Serie 250108: Betrachten der Website der Universität
Serie 240108-3: Betrachten der Website der Universität

Vergleich des Circle Approximation Algorithmus von Daunys und Ramanauskas mit dem Originalalgorithmus. Die linke Abbildung entspricht dem Originalergebnis, die Rechte dem des Algorithmus' von Daunys und Ramanauskas.

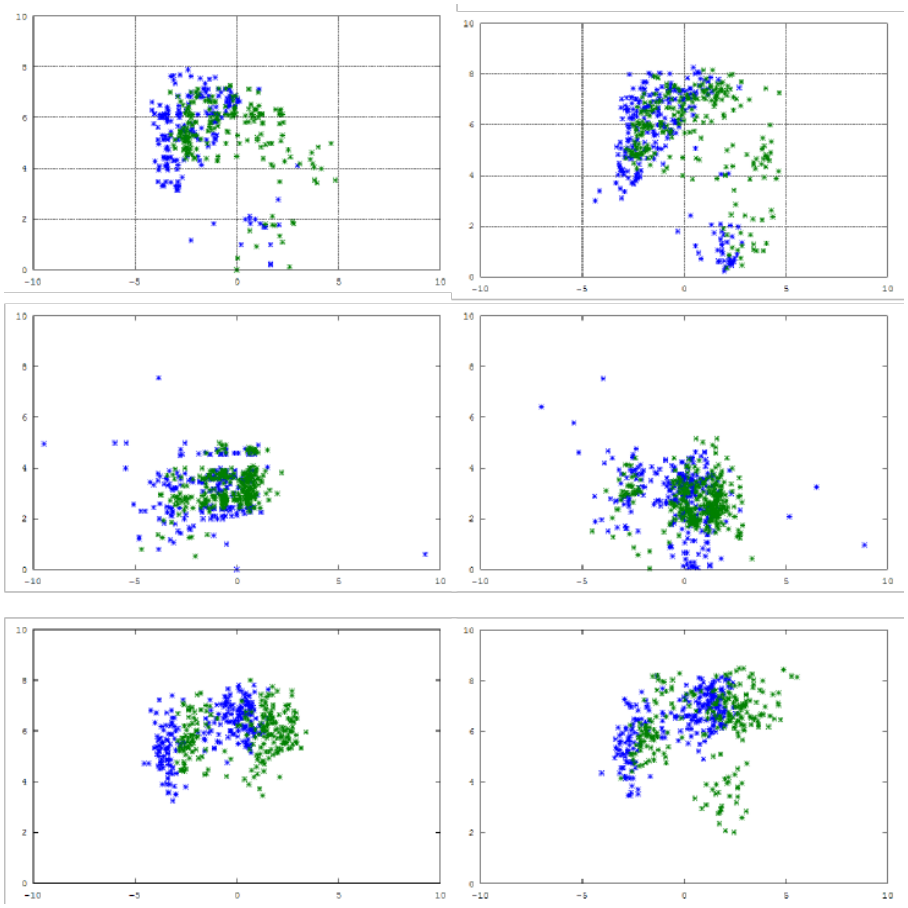

Bild 4.3: Von oben nach unten:
Serie 240108: Verfolgung eines Punktes auf dem Bildschirm
Serie 250108: Betrachten der Website der Universität
Serie 240108-3: Betrachten der Website der Universität

Vergleich des Algorithmus von Poursaberi und Araabi mit dem Originalalgorithmus. Die linke Abbildung entspricht dem Originalergebnis, die Rechte dem des Algorithmus' von Poursaberi und Araabi.

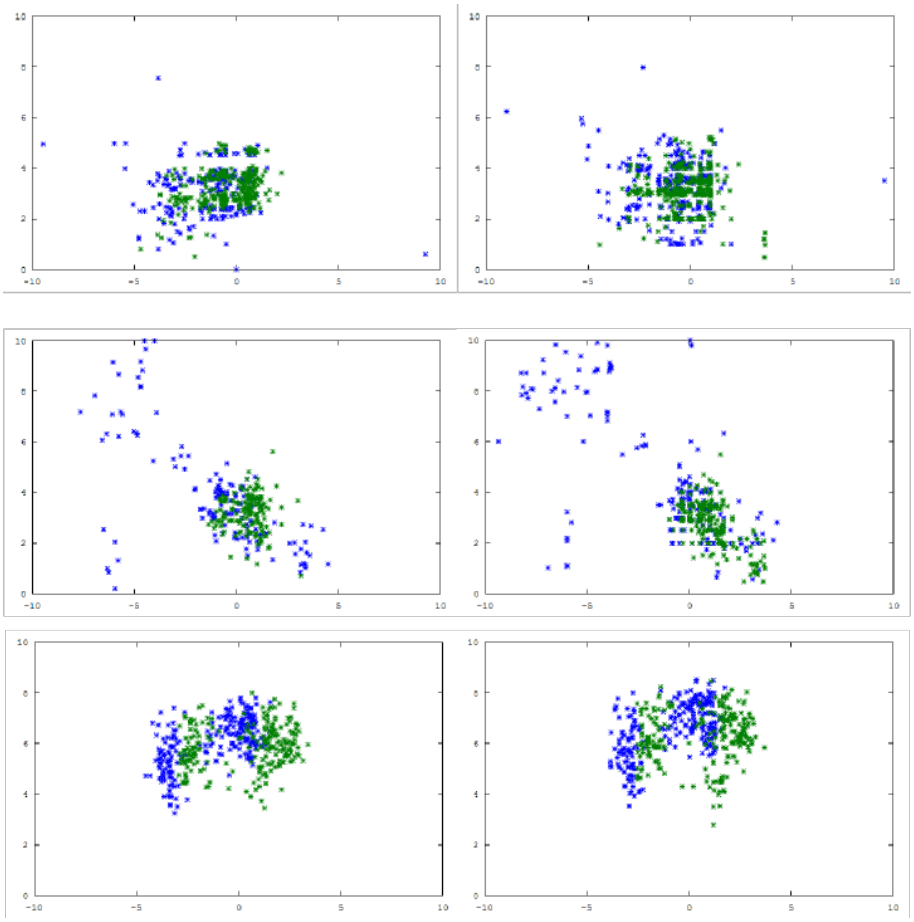

Bild 4.4: Von oben nach unten:
Serie 250108: Betrachten der Website der Universität
Serie 240108-2: Verfolung eines Punktes
Serie 240108-3: Betrachten der Website der Universität

Vergleich des Algorithmus von Perez et al. mit dem Originalalgorithmus. Die linke Abbildung entspricht dem Originalergebnis, die Rechte dem des Algorithmus' von Perez et al..

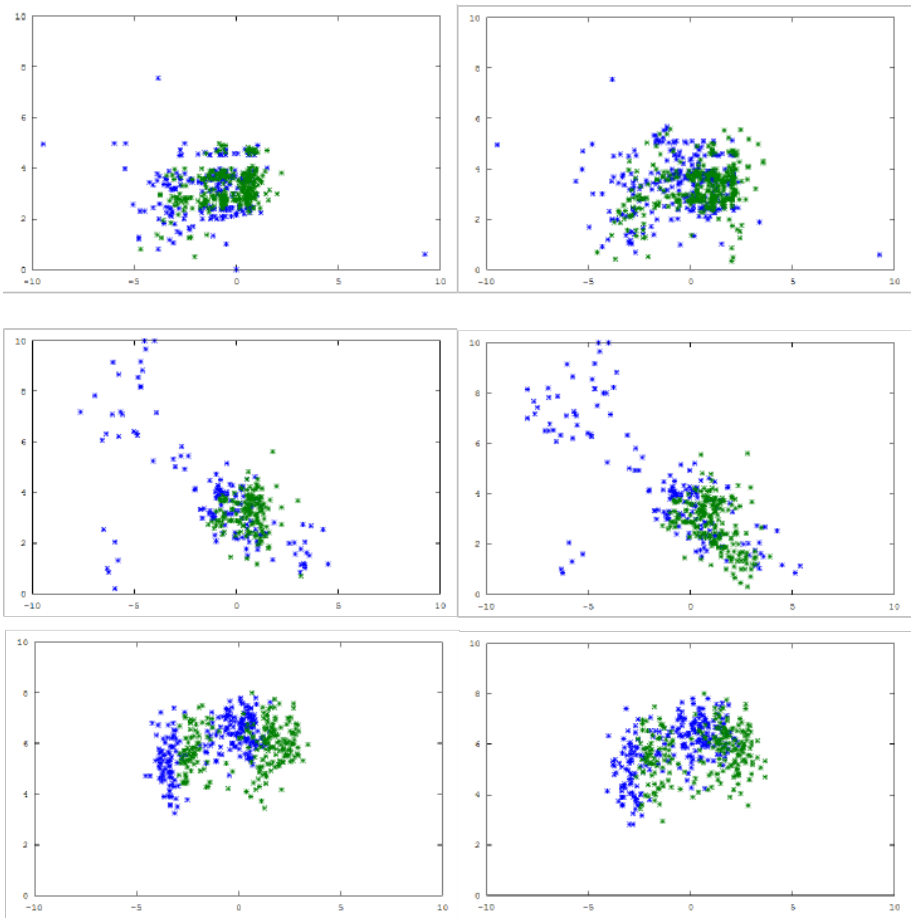

Bild 4.5: Von oben nach unten:
Serie 250108: Betrachten der Website der Universität
Serie 240108-2: Verfolung eines Punktes
Serie 240108-3: Betrachten der Website der Universität

Literaturverzeichnis

[DAU02] DAUGMAN, John: How Iris Recognition Works, IEEE Conference on ICIP, 2002

[DAR04] DAUNYS, G. and RAMANAUSKAS, N. :The accuracy of eye tracking using image processing. In Proceedings of the Third Nordic Conference on Human-Computer interaction (Tampere, Finland, October 23 - 27, 2004). NordiCHI '04, vol. 82. ACM, New York, NY, 377-380, 2004

[FIH06] FUNABIKI, Nobuo ; ISOGAI, Megumi ; HIGASHINO, Teruo ; ODA, Masashi: An Eye-Contour Extraction Algorithm from Face Image using Deformable Template Matching. In: Memoirs of the Faculty of Engineering, Okayama University, Vol.40, pp.78-82, January, 2006

[GEI07] GEIER, Thorsten: Gaze-Tracking zur Interaktion unter Verwendung von Low-Cost-Equipment, Universität Koblenz-Landau, Institut für Computervisualistik, Diplomarbeit, 2007

[HAP05] HANSEN , Dan Witzner ; PECE , Arthur E. C.: Eye tracking in the wild. In: Computer Vision and Image Understanding, v.98 n.1, p.155-181, April 2005

[HAP03] HANSEN, D. W. ; PECE, A. E. : Iris Tracking with Feature Free Contours. In: Proceedings of the IEEE international Workshop on Analysis and Modeling of Faces and Gestures (October 17 - 17, 2003). AMFG. IEEE Computer Society, Washington, DC, 2003 , p.208

[KAR02] KAWAGUCHI, T; RIZON, M: Iris detection using intensity and edge information. In: Pattern Recognition. Vol. 36, no. 2, pp. 549-562 , Feb. 2002

[LWP05] LI, Dongheng ; WINFIELD, David ; PARKHURST, Derrick J.: Starburst: A hybrid algorithm for video-based eye tracking combining feature-based and model-based approaches. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Bd. 3, S. 79 , 2005

[LWP06] LI, Dongheng ; PARKHURST, Derrick J.: Open-Source Software for Real-Time Visible-Spectrum Eye Tracking. In: Proceedings of the COGAIN 2006, September 4-5, 2006

[MUL95] MULLIGAN, Jeffrey B.: Image Processing for improved eye- tracking accuracy. In: Behavior Research Methods, Instruments and Computers, 29 (1), 54-65 , 1997

[OMY02] OHNO, Takehiko ; MUKAWA, Naoki ; YOSHIKAWA, Atsushi: FreeGaze: A Gaze Tracking System for Everyday Gaze Interaction. In: Proceedings of the symposium on ETRA 2002: eye tracking research and applications symposium, p.125-132 , 2002

[PCG03] PEREZ, A ; CORDOBA, M. L. ; GARCIA, A. ; MENDEZ, R. ; MUNOZ, M.L ; PEDRAZA, J.L. ; SANCHEZ, F.: A Precise Eye-Gaze Detection and Tracking System. In: Proceedings of the 11th International Conference in Central Europe of Computer Graphics, Visualisation and Computer Vision, 2003

[PCR05] PENG, Kun ; CHEN, Liming ; RUAN, Su ; KUKHAREV, Georgy: A Robust Algorithm for Eye Detection on Gray Intensity Face without Spectacles. In: Journal of Computer Science and Technology,2005

[POA05] POURSABERI, Ahmad; ARAABI, Babak: A Novel Iris Recognition System Using Morphological Edge Detector and Wavelet Phase Fea-

tures.In: ICGST International Journal on Graphics, Vision and Image Processing Volume 5, 2005

[PRA06] PROENCA, H. ; ALEXANDRE L.A.: Iris segmentation methodology for non-cooperative recognition, In: IEE Proceedings Vision, Image and Signal Processing, April 2006, 199- 205

[TZL03] TAN, Huachun ; ZHANG, Yu-Jin ; LI, Rui: Robust Eye Extraction Using Deformable Template and Feature Tracking Ability. In: Information, Communications and Signal Processing, 2003 and the Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on Volume 3, Issue, p. 1747-1751 , October 2003

[ZHY02] ZHU, Jie ; YANG, Jie: Subpixel Eye Gaze Tracking, Proceedings of the Fifth IEEE International Conference on Automatic Face and Gesture Recognition. p.131, May 20-21, 2002