Universität Regensburg

# Eyetracking mit mehreren Kameras in Smart Kitchens

Masterarbeit im Fach Medieninformatik am
Institut für Information und Medien, Sprache und Kultur (I:IMSK)


Vorgelegt von: Markus Guder

Adresse: Wacholderstraße 22, 92421 Schwandorf

E-Mail (Universität): markus.guder@stud.uni-regensburg.de

E-Mail (privat): markus.guder@gmail.com

Matrikelnummer: 1655429


Erstgutachter: Prof. Dr. Bernd Ludwig

Zweitgutachter: Prof. Dr. Niels Henze

Betreuer: Prof. Dr. Bernd Ludwig

Laufendes Semester: SS 2019

Abgegeben am: 30.9.2019

# Inhalt

## Abbildungsverzeichnis

## Tabellenverzeichnis



## Abkürzungsverzeichnis

| | | |
|---|---|---|
| CSV | - | Comma-separated values (Dateiformat) |
| AOI | - | Area of Interest |
| AOIs | - | Areas of Interest |
| CLNF | - | Conditional Local Neural Fields |
| JSON | - | JavaScript Object Notation (Dateiformat) |
| CPU | - | Central Processing Unit |
| CMU | - | Carnegie Mellon University (Pennsylvania, USA) |
| ARM | - | ARM-Architektur Prozessor Architektur |
| LAN | - | Local Area Network |
| CNN | - | Convolutional Neural Network |
| GPU | - | Grafikprozessor (englisch - graphics processing unit) |
| FPGA | - | Field Programmable Gate Array |

## Zusammenfassung

In der hier vorliegenden Arbeit wird untersucht, wie ein *Eyetracking System* in einer *smarten* Küche entwickelt werden kann, das Probleme bisheriger Systeme in diesem Themenfeld überwindet. Hierfür werden im ersten Schritt die Begriffe *smart* im Allgemeinen beziehungsweise *Smart Home* und *Smart Kitchen* im Spezifischen definiert. Für das bei derartigen Systemen auftretende Bedürfnis der Kontexterkennung der Aktivität der Nutzenden wird *Eyetracking* als geeigneter Kommunikationskanal und Problemlösungsansatz präsentiert. Nach der Vorstellung bereits existierender Blick basierter Systeme im Küchenumfeld, wird auf unterschiedliche verwandte Arbeiten eingegangen, die sich mit sogenannten *Active Eyetracking* beschäftigen. Auf Basis dieser Literaturrecherche wird die Motivation bzw. die Forschungsfrage vorgestellt. Im nächsten Kapitel wird die Systementwicklung beschrieben, in der auf Basis der Vorevaluierung eines ersten Systementwurfs und der zuvor durchgeführten wissenschaftlichen Analyse ein Anforderungskatalog für ein zu entwickelndes System definiert wird. Für die Systemimplementierung werden die einzelnen Komponenten (*Eyetracking* Einheit, Server, Webanwendung) anhand der Faktoren Hardware, Software, Schnittstellen und Einschränkungen im Detail beschrieben. Abschließend wird eine Übersicht über das gesamte System gegeben und mithilfe einer Flaschenhalsanalyse Schwachstellen des Systems aufgezeigt bzw. sich kritisch mit dem Entwurf der Gesamtarchitektur auseinandergesetzt. Im folgenden Abschnitt werden die Anforderungen und der Aufbau der Studie beschrieben, mit deren Hilfe das System evaluiert werden soll. Dabei wird neben den Eckdaten der Studie auch die Annotierung und Auswertung der Daten beschrieben. Anschließend werden die Ergebnisse, angepasst an die zuvor aufgestellten Anforderungen an die Studie, präsentiert. Diese werden wiederum im nächsten Teil diskutiert und Begründungen für auftretende Effekte vorgestellt. Im letzten Abschnitt werden weitere Einsatzmöglichkeiten für das System aufgezeigt und zudem auf nötige Veränderungen für eine mögliche Weiterentwicklung, basierend auf den Erkenntnissen aus der Studie, hingewiesen.

## Abstract

This paper examines how an eyetracking system can be developed in a smart kitchen that overcomes problems of previous systems in this area. In a first step, the terms smart in general and Smart Home and Smart Kitchen in specific will be defined. Eyetracking is presented as a suitable communication channel and problem-solving approach for the need of context recognition of the user's activity occurring in such systems. After the presentation of already existing view based systems in the kitchen environment, different related works will be discussed which deal with so called Active Eyetracking. On the basis of this literature research the motivation or the research question is presented. In the next chapter the system development is described, in which on the basis of the pre-evaluation of a first system draft and the before accomplished scientific analysis a requirement catalog is defined for a system which can be developed. For the system implementation, the individual components (eyetracking unit, server, web application) are described in detail on the basis of the factors hardware, software, interfaces and restrictions. Finally, an overview of the entire system is given and weak points of the system are pointed out with the help of a bottleneck analysis and further the design of the overall architecture is critically examined. The following section describes the requirements and structure of the study that will be used to evaluate the system. In addition to the key data of the study, the annotation and evaluation of the data is also described. Subsequently, the results are presented, adapted to the previously established requirements of the study. These will be discussed in the next part and reasons for effects will be presented. In the last section further application possibilities for the system are pointed out and in addition necessary changes for a possible advancement, based on the realizations from the study, are pointed out.

# 1 Einleitung

Als Teil des *Internet of Things (IOT)* kann auch die Entwicklung hin zu immer intelligenteren oder *smarten* Häusern und Wohnungen betrachtet werden (Li & Lu, 2011). Folgt man einer Umfrage der Gesellschaft für Konsumforschung (Gfk), dem größten deutschen Marktforschungsinstitut, glauben 43 Prozent der befragten deutschen Internetnutzerinnen und -nutzer, dass dieser technologische Trend in den nächsten Jahren den größten Einfluss auf ihr Leben haben wird (Dale, 2016). Dabei versteht man konkret unter dem Oberbegriff des *Smart Homes*, dass durch das technische Ausstatten von Wohnräumen mit unterschiedlichsten Sensoren bzw. der intelligenten Vernetzung von beispielsweise Haushaltsgeräten in der Küche, Lebens- und Wohnqualität, Sicherheit und Energienutzung der Bewohnerinnen und Bewohner einer so vernetzen Wohnung verbessern lassen (Solaimani & Keijzer-Broers & Bouwman, 2015).

Wie bereits angesprochen ist einer der primären Vorteile, die *Smart Homes* mit sich bringen, die Steigerung von Sicherheit und Komfort für die Bewohner. Diese beiden Vorteile treten dann noch einmal verstärkt in den Vordergrund, wenn man die steigende Anzahl an älteren Menschen betrachtet, die durch diese Unterstützung eine höhere Lebensqualität erhalten könnten. Das statistische Amt der Europäischen Union (Eurostat) geht aktuell davon aus, dass bis 2060 der Anteil der über 65-Jährigen von aktuell 17,1% auf 30% ansteigen wird. Ähnliche Entwicklungen lassen sich in noch extremerer Form beispielsweise in Japan beobachten. Dort geht man davon aus, dass bis 2050 39.6% der Bevölkerung über 65 sein wird (Blasco & Marco & Casas, 2014). Zählt man zu dem Umstand einer zunehmend alternden Gesellschaft noch den Fakt hinzu, dass sich die meisten Unfälle im Haushalt, spezifisch im Umfeld Küche, ereignen, macht die Entwicklung einer intelligenten Küche im besonderen Sinn (AXA Versicherung, 2019). Intelligente Assistenzsysteme, die zu jeder Zeit wissen was der oder die Nutzenden gerade tun und Hilfestellungen zu unterschiedlichst gearteten Situation geben, können hierbei Schlüsselpositionen einnehmen, um mehr Sicherheit im Umfeld Küche zu bieten.

Für derartige intelligente Assistenzsysteme, die auf Software fußend, den Kern für den eigentlichen Begriff *smart* bilden, ist es fundamental wichtig jederzeit erkennen zu können, was die tatsächlich Intention des Nutzenden,

abhängig von der jeweiligen Situation und Tätigkeit ist. Diese Kontaktaufnahme zwischen Mensch und Maschine wird beispielsweise bei Sprachassistenzsystemen meist über das Aussprechen eines initialen Trigger-Wortes gestartet (Gruber & Hills, 2014). Neben dem Umstand der Unnatürlichkeit dieses Vorgangs kommen hierzu noch Probleme der Erkennungs-Stabilität oder der benötigten Unterstützung diverser Sprachen (Hornig, 2017). Jedoch gibt es neben der Sprachinteraktion auch die Möglichkeit der Blickinteraktion. Diverse Studien haben bisher gezeigt, dass dem Blickverhalten eine immense Bedeutung im Kommunikationsprozess zukommt (Lapidot-Lefler & Barak, 2012). Durch das Anblicken eines Objekts wird der Kontakt zwischen System und Nutzenden hergestellt und daraus resultierend der Kontext erkannt, auf dem die Aufmerksamkeit des Nutzers gerade liegt.

Diese Blickerkennung ist aktuell mit unterschiedlichen Eyetracking Systemen möglich, aber in den aktuellen Ausprägungen in einem *Smart Environment* nicht praktikabel einsetzbar. Das hier vorliegende *Smart Environment* ist eine Küche, in der das permanente Tragen eines mobilen Eye Trackers, beispielsweise während des Kochens, auf nur wenig Akzeptanz aufgrund des mangelnden Tragekomforts stoßen würde (Risko & Kingstone, 2011). Stationäre Eyetracker haben hingegen den Vorteil, dass diese den Nutzenden nicht in seiner Tätigkeit beeinflussen würden. Herkömmliche stationäre Eyetracking-Systeme bergen aber den Nachteil, dass ein flexibles Erkennen bedingt durch die Tiefe des Raumes, wie dies in unterschiedlichen Küchen der Fall ist, nicht geleistet werden kann (Shell & Vertegaal, 2003). Hinzu kommt der Umstand, dass derzeitige stationäre Eyetracking Systeme kostspielig sind.

Ziel dieser Arbeit ist es daher zum einen ein kostengünstiges stationäres Eyetracking-System zu entwickeln, das die Nutzenden nicht während des Kochvorgangs behindert, trotzdem aber diese über den gesamten Raum der Küche hinweg erfassen kann. Eine anschließende Evaluierung soll hierbei herausfinden zu welchen Leistungen das System fähig ist und wo insbesondere dessen Grenzen liegen. Hierzu wird im folgenden Kapitel zuerst der nähere Forschungsbereich betrachtet, indem der Begriff *smart* näher definiert wird und warum bei derartigen smarten Systemen die Intentionserkennung der Nutzenden eine zentrale Rolle spielt. Anschließend wird dargelegt warum Eyetracking als Interaktionskanal zwischen Mensch und Maschine besonders

geeignet ist. Danach werden verwandte Arbeiten vorgestellt, die sich mit ähnlichen Problemstellungen, wie der hier vorliegenden, konfrontiert sehen. Am Ende dieses Kapitels wird die Forschungsfrage und die Motivation dieser Arbeit dargestellt. Das dritte Kapitel beschreibt die Systementwicklung, beginnend mit der Zusammenführung von Anforderungen an das System basierend auf der Literaturrecherche und der Vorevaluierung eines ersten Systementwurfs. Nachfolgend werden die jeweiligen Systemkomponenten einzeln hinsichtlich der Faktoren Hardware, Software, Schnittstellen und Einschränkungen näher betrachtet. Am Ende des Abschnitts wird die Gesamtarchitektur analysiert und diskutiert. Im Anschluss findet die Evaluierung des System statt. Dazu wird im ersten Schritt das Studiendesign beschrieben, anschließend die Ergebnisse präsentiert und abschließend diese diskutiert bzw. Erklärungsansätze präsentiert. Zuletzt soll ein Ausblick gegeben werden, inwiefern das System noch verbessert werden müsste und welche Möglichkeiten der grundsätzliche Ansatz des Systems bietet Eyetracking in die unterschiedlichsten Alltagssituation zu integrieren.

# 2 Stand der Technik und verwandte Arbeiten

Im folgenden Kapitel wird der Stand der Technik aufgezeigt. Um ein Verständnis für den Kontext der Arbeit zu schaffen werden daher im ersten Schritt die Begrifflichkeiten eines *Smart Environments* und einer *Smart Kitchen* definiert. Da das in der Arbeit zu entwickelnde System Blickerkennung als Interaktions-Methodik verwendet, wird anschließend der Begriff der *Gaze Interaction* und dessen Vorteile, primär im Küchen-Umfeld, besprochen. Am Ende dieses Kapitels wird außerdem die Forschungsfrage der Arbeit und der Entwicklungsstand verwandter Arbeiten im Bereich von *Hands-Free Eyetracking* Systemen vorgestellt und die sich daraus ergebenden Möglichkeiten in dem hier vorliegenden Kontext eingeordnet.

## 2.1 Smart Environment

Unter dem Begriff *Ubiquitous Computing* (allgegenwärtiges Rechnen) versteht man das Konzept der verbesserten Nutzung von Computern in der uns physisch umgebenden Umwelt (Weiser, 1991). Die damit einhergehende intelligente Verknüpfung untereinander und das praktische unsichtbar werden dieser Rechnersysteme für den Benutzenden selbst, stehen dabei im Vordergrund. Mark Weiser prägte und definierte diesen Begriff bereits Anfang der 1991 Jahre und prognostizierte eine Entwicklung weg von reinen Desktop-Rechner gestützten Systemen hin zu intelligent vernetzten Gegenständen. Die virtuelle Repräsentation physischer Objekte schafft somit die Brücke zwischen der physischen und der digitalen Welt (Weiser, 1991). Kevin Ashton bezeichnet dies erstmal 1999 als das sogenannte "Internet der Dinge", einen Überblick geben dazu Ramaswamy und Tripathi (Ramaswamy & Tripathi, 2015).

Aus diesen Begriffen heraus hat sich der Begriff *Smart Environment* entwickelt. In ihrem Buch "Smart Environments: Technology, Protocols and Applications" definieren Cook und Das den Begriff folgendermaßen:

> *"A smart environment is a small world where all kinds of smart devices are continuously working to make inhabitants' lives more comfortable. A definition of smart or intelligent is the ability to autonomously acquire and apply knowledge, while environments refers to our surroundings." (Cook & Das, 2005, S. 3)*

Hierbei geben sie eine treffende Definition für die beiden Begrifflichkeiten ab, die dieses Themenfeld ausmachen. Das sogenannte *Environment* ist hierbei immer abhängig von dem jeweilig vorliegenden Verwendungs-Szenario, auf welches für die hier vorliegenden Arbeit im nachfolgenden Punkt 2.2 näher eingegangen wird. Der Begriff *smart* steht für sie dabei für Wissen automatisiert und kontextabhängig zu erwerben und zielgerichtet anzuwenden. Poslad greift diese Definition von Cook und Das 2009 auf und erweitert diese, indem er sagt, dass diese intelligenten Umgebungen aus einer Reihe von untereinander vernetzten Geräten, mit einer Verbindung zur physikalischen Welt, bestehen (Poslad, 2009). Des Weiteren haben diese Vorrichtungen für ihn auf einen spezifischen Kontext hin ausgerichtete Aufgaben, die sie automatisch auf die Interaktion mit dem Nutzenden ausführen oder diesem implizit vorwegnehmen. Die sich daraus ergebende implizite *Human Computer Interaction* deckt sich daher mit einem der Grundgedanken von Mark Weiser, der das Unsichtbar-Werden von Rechnersystemen, im Zusammenhang mit *Ubiquitous Computing* betont (Weiser, 1991).

Will man nun den Begriff *smart* unter einem praktischen Gesichtspunkt betrachten, so ist es erst Software, die die intelligente Verknüpfung unterschiedlichster Sensoren und deren Auswertung hinsichtlich der Unterstützung der Nutzenden in unterschiedlichsten Situation ermöglicht. Susanne Biundo und Andreas Wendemuth bezeichnen in ihrem Buch "*Von kognitiven technischen Systemen zu Companion-Systemen*" diese intelligente Software als Begleiter, dessen Aufgabe es ist, in einem bestimmten Kontext zu unterstützen (Biundo & Wendemuth, 2010). Weiter führen sie an, dass die Intentionserkennung einer der zentralen Bestandteile eines solchen *Companion-Systems* sein muss. Darin liegt auch laut Ludwig einer der primären Herausforderung für derartige Systeme. In seinem Buch "Planbasierte Mensch-Maschine-Interaktion in multimodalen Assistenzsystemen" legt er dar, dass das Ableiten aus der Handlung der Nutzenden bzw. die Identifikation komplexer Ziele, als eine der wesentlichen Schwierigkeit bei interaktiver Assistenz zu sehen ist (Ludwig, 2015).

## 2.2 Smart Home und Smart Kitchen

Wie im vorangegangenen Punkt angesprochen, gibt es unterschiedlichste Szenarien (*Environments*), in denen diese intelligenten Systeme eingesetzt werden können, um den Nutzenden unterstützend zur Seite zu stehen. Eines dieser Szenarien ist das intelligente Zuhause, das als Wohnung definiert werden kann, die mit Computer- und Informationstechnologie ausgestattet ist und deshalb die Bedürfnisse seiner Bewohner vorausahnen und auf diese eingehen kann. Dabei sind Kompetenzen wie das Bereitstellen und Erhöhen von Komfort, Sicherheit oder Unterhaltung durch das Management innerhalb des Hauses und in die Welt hinaus zentraler Bestandteile einer als *Smart Home* zu bezeichnenden Umgebung. Diese aus dem Buch "*Inside the Smart Home*" aus dem Jahr 2003 von Richard Harper entliehene Definition beschreibt im Kern die Dualität derartiger Systeme, nämlich einerseits mit unterschiedlichsten Sensoren und Technologien ausgestattet zu sein, um den aktuellen Status des Hauses zu erfassen, und andererseits die Lebensqualität der Bewohner zu erhöhen (Harper, 2003).

Neben den unterschiedlichsten Sensoren, die ein *Smart Home* beinhalten kann und viele Wohnungen heute schon regulär beinhalten (z. B. Bewegungsmelder, Thermostate, Rauchmelder) bildet auch hier eine Software das zentrale Element, das diese unterschiedlichen Geräte und Sensoren miteinander verknüpft und die generierten Informationen zielgerichtet auswertet (Kazmierzak, 2011). Eine derartige *Ambient Intelligence*, die als weniger Hardware bezogener Begriff als *Ubiquitous computing* zu verstehen ist, muss unterschiedlichste Informationen heranziehen, um an sie gestellte Aufgaben, beispielsweise das Dimmen des Lichts zu einer gewissen Uhrzeit, da der Bewohner Fernsehen möchte, erfolgreich zu bewältigen (Augusto & Mccullagh, 2007). Das sogenannte Kontextquintett (wer, wo, was, wann und warum) kann hierfür herangezogen werden, um die Nutzenden und ihre Intentionen zu verstehen (Brooks, 2003). Auch hierbei wird wiederum die kontextabhängige Intentionserkennung der Nutzenden, ausgedrückt durch das "warum" im Quintett, als größte Herausforderung angesehen.

*Smart Home* bildet einen Oberbegriff für die unterschiedlichsten intelligent miteinander verknüpften Lebensbereiche in einer Wohnung. Im folgenden Abschnitt wird einer dieser Bereiche, die *Smart Kitchen*, eingehender betrachtet.

Dabei sollen die besondere Notwendigkeit von intelligenten Computersystemen in der Küche und die damit einhergehenden Vorteile herausgearbeitet werden.

Die im vorangegangen Abschnitt beschriebenen vernetzten Assistenzsysteme können vor allem im Küchen-Umfeld einen sinnvollen Einsatzzweck finden, da dort komplexe Arbeitsabläufe stattfinden und die Nutzenden multitaskingfähig[1] sein müssen, um beispielsweise Speisen zuzubereiten. Nimmt man den Umstand hinzu, dass sich in der Küche die meisten Unfälle im Haushalt ereignen, dann kann eine *smarte* Küche auch erheblich zu einem höheren Sicherheitsgefühl beitragen (Blasco & Marco & Casas, 2014). Dies spielt primär bei älteren Nutzenden eine wichtige Rolle, denen durch eine intelligente Assistenz, die sich jederzeit des Handlungskontextes der anwesenden Bewohner bewusst ist, Unabhängigkeit bei Aktivitäten des Alltags zurückgegeben werden kann. Zu einem anderen Aspekt, nämlich einer Verbesserung der Ernährung und zur Unterstützung bei Diäten, hat das *Massachusetts Institute of Technology* 2010 einen Artikel veröffentlicht. Dabei konnten die Forscher zeigen, dass das von Ihnen entwickelte System zur Erkennung von Portionsgrößen der gerade zubereiteten Speisen und den damit einhergehenden Kalorien, die Nutzenden maßgeblich dabei unterstützen konnte, ausgewogener und gesünder zu kochen. Dabei wurden unter anderem Gewichtssensoren unter und Kameras über den Arbeitsflächen genutzt (Chen & Huang, 2010). Andere Arbeiten versuchen das Problem der Kontexterkennung auf eine andere Art und Weise zu lösen. Dabei haben beispielsweise Forscher der Technischen Universität München ein Schneidebrett entwickelt, das aufgrund des Schneideverhaltens (z.B. Intensität und Dauer) erkennt, was gerade auf ihm geschnitten wird (Kranz & Rigoll, 2007). Dabei steht bei diesen Arbeiten immer die Identifikation der Handlung der Nutzenden im Vordergrund, um falls benötigt Hilfestellung leisten zu können.

## 2.3 Gaze Interaction

Unter dem Begriff der *Gaze Interaction* kann die Blickerkennung von Nutzenden verstanden werden, um durch die Auswertung von Blicken eine Schnittstelle zur Interaktion zwischen Benutzenden und Maschine zu schaffen (Møllenbach

[1] die Fähigkeit besitzend, mehrere Aufgaben gleichzeitig zu bewältigen (Duden, 2019)

& Lillholm, 2019). In diesem Abschnitt soll zuerst begründet werden, warum sich Blickerkennung als Kanal der Interaktion eignet und auf welche Art und Weise es als *attentive user interfaces* fungieren kann.

Andrew T. Duchowski gibt mit seinem Buch "*Eye Tracking Methodology*" einen ausführlichen Überblick über den aktuellen Forschungs- und Entwicklungsstand hinter dem Themenkomplex Eyetracking (Duchowski, 2007). Dabei identifiziert er zwei große Typengruppen an Blickerkennungs Systemen. Hierbei nennt er zuerst die sogenannten *Head-Mounted*-Systeme. Diese werden meist mit einer Halterung ähnlich der einer Brille am Kopf des Probanden angebracht. Dabei werden Videoaufnahmen des Auges, meist unter zuhilfenahme eines Infrarot Markers, ausgewertet und darüber die Blickrichtung bestimmt. Vorteile, die er für diese Art der Blickerkennung anführt, sind die relative Nicht-Invasivität in die zu untersuchenden Testaufbauten und deren Genauigkeit (Duchowski, 2007, S.109). Als zweite Gruppe sind sogenannte *Table-Mounted*-Systeme zu betrachten. Dabei sind moderne Systeme ebenfalls videobasiert und errechnen über die Reflexion der Hornhaut im Auge die aktuelle Blickrichtung. Wichtig ist hierbei, dass diese Blickerkenner nicht an der Versuchspersonen selbst, sondern direkt vor der Person, meist unterhalb eines Bildschirmmonitors, angebracht werden. Vorteile dieser Systeme sind, dass Sie den Nutzenden bei seiner Arbeit vor dem Bildschirm, im Gegensatz zu *Head-Mounted* Eyetrackern, nicht behindern und zudem leicht und nur einmalig aufzubauen sind. Ein großer Nachteil dieses Systems ist, dass meist nur der Bildschirmbereich und nichts darüber hinaus an Blicken erfasst werden kann. Dazu kommt noch, dass diese *Table-Mounted* Systeme meist in den zu untersuchenden Versuchsaufbau integriert werden müssen, um den Nutzenden zu erfassen (Duchowski, 2007, S.101).

Neben dem Umstand, dass Eyetracker häufig zur Erforschung von Mensch-Maschine-Schnittstellen und der allgemeinen Benutzerfreundlichkeit von Systemen benutzt werden, kann der Blick eines Nutzenden auch als Kanal zur Interaktion mit einem technischen System genutzt werden. Dies wird im Allgemeinen als *Gaze Interaction* bezeichnet. Als Grundlage dabei werden zwei Hypothesen formuliert, auf denen die Vorteile von Blickerkennungs-Systemen basieren. Dies ist zum einen die Auge-Geist-Hypothese (*eye-mind assumption*) von Just und Carpenter (Just & Carpenter, 1980). Diese besagt, dass Objekte nur dann angesehen werden, wenn ein Interesse an diesen besteht bzw. diese

kognitiv ausgewertet werden. Dazu kommt die sogenannte Unmittelbarkeits-Hypothese (*immediacy assumption*), welche festlegt, dass diese kognitive Auswertung unmittelbar mit der visuellen Aufnahme einhergeht (Just & Carpenter, 1980). Fasst man diese beiden Aussagen zusammen und vereinfacht sie, kann Folgendes aus ihnen geschlussfolgert werden. Erkennt man den Blick des Nutzenden auf ein Objekt, so kann man davon ausgehen, dass sich mit diesem gerade beschäftigt wird. Betrachtet man dies unter dem Gesichtspunkt des vorangegangenen Kapitels, in denen die Kontexterkennung von Nutzenden immer wieder betont wurde, erscheint Blickerkennung als Mittel der Interaktion sehr gut geeignet.

Unterschiedliche Studien haben zudem gezeigt, dass Interaktion über Blicke in vielen Anwendungsfällen schneller und genauer funktioniert als beispielsweise eine Interaktion über Sprache. In Studien wie "*Where is it? Event Synchronization in Gaze-Speech Input Systems*" sollten die Tester ein Objekt, aus einer Gruppe auf dem Bildschirm, per Sprachbefehl an eine andere Position bewegen. Dabei wurde festgestellt, dass die Fixierung des Objekts mit Augen im Durschnitt 630 Millisekunden vor dem Aussprechen des Wortes "*Move*" geschieht (Kaur & Mantravadi, 2003). Zu einem ähnlichen Ergebnis kommt auch die Untersuchung "*Gaze and Speech in Attentive User Interfaces*" aus dem Jahr 2000 in Beijing. Dabei wurden die Blicke der Probanden im Büro der Zukunft, in welchem die Interaktion mit Bildschirmen über Sprache funktioniert, parallel mit den Sprachbefehlen aufgenommen und untersucht. Es wurde festgestellt, dass die Nutzenden immer zuerst das anblicken, zu dem sie anschließend einen Sprachbefehl geben. Zudem ergeben sich aus den Daten, dass in 98% der Fälle die Blickdaten alleine ausreichend gewesen wären, um den Gegenstand, in diesem Fall das kontextabhängige Objekt im Büro, eindeutig ausmachen zu können. Abgesehen von den Nachteilen, die sprachbasierte Interaktionssysteme mit sich bringen, beispielsweise die ungenaue Aussprache durch die Nutzenden, so zeigt sich deutlich, dass Blickerkennung, vor allem bei der Erkennung des Kontextes, sowohl schneller als auch deutlich eindeutiger funktioniert (Maglio & Smith, 2000) .

Des Weiteren geben Päivi Majaranta und Andreas Bulling im dritten Kapitel "*Eyetracking and Eye-Based Human–Computer Interaction*" des Buches "*Advances in Physiological Computing*" einen sehr detaillierten Einblick über den aktuellen Forschungsstand im Bereich der Eyetracking-Technologie, zeigen

Herausforderungen auf diesem Gebiet auf und listen dabei unterschiedliche Benutzeranforderungen und Schlüsselfunktionen an derartige Systeme auf (Majaranta & Bulling, 2014). Darüber hinaus werden unterschiedliche Interaktions-Kategorien für Blickerkennungssysteme aufgelistet. Dabei ist für diese Arbeit besonders das *Attentive User Interface* interessant. Hierbei ist der Grundgedanke, dass der Nutzende nicht sein Blickverhalten ändern muss, um explizit Befehle zu erteilen. Informationen werden über die Bewegung der Augen im Hintergrund gesammelt und nutzbar gemacht. Dabei wird betont, dass der Nutzende nicht von seiner eigentlichen Aktivität abgelenkt wird und zudem den Überblick über seine Handlung behält (Majaranta & Bulling, 2014). Eine Alltagssituation, die genau dies von einem System fordert, findet sich im anschließenden Punkt .

## 2.4 Gaze Interaction in der Küche

In diesem Abschnitt soll zunächst der Themenbereich *Smart Kitchen* mit den Möglichkeiten der *Gaze Interaction* zusammengeführt werden und anschließend bereits existierende Bilderkennungssysteme in der Küche und die damit verbundenen Herausforderungen herausgearbeitet werden.

Ein zentrales Thema einer intelligenten Küche ist die Intentionserkennung, dabei die Aktivitäten und der Bedürfnisse der Nutzenden in Echtzeit zu erfassen, an ein Assistenzsystem weiterzugeben und mithilfe dessen Hilfestellungen an den Nutzenden heranzutragen. Bei *eyeCOOK* beispielsweise handelt es sich um ein System, das über einen Eyetracker, der unter einem Bildschirm positioniert ist, erkennt, an welcher Stelle der Nutzende gerade bei der Zubereitung eines Rezepts ist und daraufhin die Zutaten und weiteren Arbeitsschritte laut vorliest (Shell & Vertegaal, 2003). Dieser Eyetracker fällt in die erste Klasse der von Duchowski beschriebenen *Table-Mounted* Systeme. Dabei wird in der wissenschaftlichen Arbeit, in der *eyeCOOK* vorgestellt wird, deutlich der Nachteil dieser Eyetracker aufgezeigt. Diese funktionieren nur, wenn der Nutzende direkt vor dem Bildschirm steht. Dies ist eine deutliche Limitierung der Interaktion per Blick in einer so komplexen Umgebung wie einer Küche. Betrachtet man hingegen den Forschungsstand was *Head-Mounted* Eyetracker in der Küche angeht, so lösen diese zwar das Problem der Mobilität, doch werden nicht als Mittel der Interaktion verwendet. Vielmehr werden sie zur Evaluierung

unterschiedlichster anderer Systeme in einer Küche eingesetzt. Der Grund hierfür mag darin begründet sein, dass es wenig zielführend erscheint Systeme zu entwickeln, die Nutzende nicht verwenden würde. Die Bereitschaft in einer Küche einen mobilen Eyetracker aufzusetzen, in der ein gewisses Maß an körperlicher Aktivität gefordert ist und der Wärmeentwicklung bei dem eigentlich Kochvorgang, dürfte gering ausfallen.

## 2.5 Verwandte Arbeiten

Der folgende Abschnitt stellt verwandte Arbeiten vor, die die Problematiken von konventionellen Eyetracking Systemen, eben im vorangegangen Punkt beschrieben, adressieren und Lösungsvorschläge liefern. Dabei werden unterschiedliche Arbeiten vorgestellt, die zwar nicht im Küchen-Kontext stattfinden, die sich aber mit einer ähnlichen Problemstellung konfrontiert sehen. Dabei sind zwei Voraussetzung für ein solches System nötig. Einerseits darf das System den Nutzenden nicht in seiner eigentlichen Handlung einschränken, wie es ein mobiler Eyetracker tun würde. Andererseits muss eine Möglichkeit geboten werden, den Nutzenden in der gesamten Küche zu erfassen, im Gegensatz zu einem stationären Eyetracking System.

Einen möglichen Lösungsansatz beschreibt Mohamed Khamis in seiner Arbeit "*EyeScout: Active Eyetracking for Position and Movement Independent Gaze Interaction with Large Public Displays*" unter dem Begriff des *Active Eye Tracking*. Darunter werden Systeme verstanden, die sich an die Augenposition des Benutzers anpassen, anstatt seine Kopf- oder Körperbewegungen einzuschränken. Khamis sieht sich dabei mit *EyeScout* einem ähnlichen Problem konfrontiert, nämlich der Erfassung des Nutzers entlang eines langen Displays. Seine Lösung setzt dabei auf ein vor dem Display montiertes Schienensystem, auf dem ein Eyetracker montiert ist. Die Bewegung bzw. die Position des Nutzers wird über einer *Microsoft Kinect One* erfasst und die Position des Eyetrackers auf der Schiene dementsprechend angepasst. Dabei wird eine neue Interaktionsmöglichkeit, das sogenannte "*Walk and Interact*", vorgestellt, welches von den Nutzern gut angenommen wurde (Khamis & Bulling, 2017).

Zusätzlich verweist Khamis in seiner Arbeit auf den Einsatz von *Active Eyetracking* Systemen im medizinischen Kontext. Hierbei ist der Verwendungszweck in der Augenchirurgie bei "Laser-in-situ-Keratomileusis" - Operationen angesiedelt. Ein gängiger Ansatz ist die Verwendung einer

einzelnen oder mehrerer Schwenkkameras zur Anpassung an die Kopfposition des Patienten. Um Gesichter zu erkennen und den Winkel eines Eye Trackers, der auf einem Schwenk- und Kippmechanismus montiert ist anzupassen, verwendeten Hennessey und Fiset ebenfalls eine *Kinect (*Mrochen & Hütz, 2001).

Die Arbeit "*Long range eye tracking: bringing eyetracking into the living room*" bewegt sich, ähnlich der hier vorliegenden Arbeit, in einem *Smart* Environment, nämlich einem Wohnzimmer. Dabei soll mithilfe eines Eye Trackers die Interaktion mit einem *Smart TV* ermöglicht werden. Ein auf einem Schwenk- und Kippmechanismus montierter Eye Tracker wird über einen Gesichtsverfolgungsmechanismus einer *Microsoft Kinect One* ständig an die wechselnde Position des Nutzenden neu angepasst. Die Forscher konnten in ihrer Arbeit zeigen, dass sie sowohl die Genauigkeit regulärer *Table-Mounted* Systeme erreichen konnten, als auch, dass die Systemleistung ausreichend für die Steuerung des *Smart TVs* ist (Hennessey & Fiset, 2012).

Mit einem ähnlichen Ansatz, wie die vorangegangene Arbeit, beschäftigt sich auch das wissenschaftliche Paper "*Development of an active gaze tracking system in unrestricted posture*" aus dem Jahr 2007. Jedoch wird hier nicht auf eine *Kinect*, welche erst 2010 veröffentlicht wurde, zurückgegriffen, sondern auf eine Überwachungskamera, um eine interaktive Schnittstelle zwischen Benutzendem und Computer zu schaffen. Dabei besteht die von dem Forscherteam entwickelte Software aus einem Gesichtsverfolgungsmodul, einem Eye Tracking-Modul, einem aktiven Steuermodul und Kontrollmodul für die Eingabe. Dabei wird der Blick des Nutzers, unabhängig von seiner Haltung und ohne aufwendige Kalibrierung geschätzt und auf einen Bildschirm übertragen (Chan Chao Ning & Chern-Sheng Lin, 2007).

Einen radikal neuen Ansatz stellen ebenfalls wieder Mohamed Khamis, Anna Kienle, Florian Alt und Professor Andreas Bulling mit ihrer wissenschaftlichen Arbeit "*GazeDrone: Mobile Eye-Based Interaction in Public Space Without Augmenting the User*" 2018 vor. Dabei greifen sie auf die Software *OpenFace* zurück, um die Blickdaten eines Nutzers, erzeugt von einer an einer Drohne befestigten Webcam, zu erfassen. In der Arbeit konnten sie zeigen, dass eine spontane bzw. kalibrierfreie blickbasierte Interaktion, aus beliebigen Positionen bzw. während sich der Nutzende bewegt, mit der Drohne möglich ist. Trotz der prinzipiell guten Leistung des Systems bezüglich der

Blickerkennung und -interaktion, wird von mehreren Testern betont, dass sie weniger bereit wären mit der Drohne zu interagieren, je größer diese wäre. Erschwerend kommt der Geräuschpegel der Drohnen hinzu, der verstärkt noch einmal im spezifischen Kontext eines geschlossenen Raumes auftritt (Khamis & Bulling, 2018).

Drei der vorangegangen Paper wurden unter der Leitung von Andreas Bulling erstellt. Auch hatte er maßgeblichen Anteil an der zuvor angesprochenen Eyetracking Software *OpenFace*. In der Arbeit "*Eye tracking for public displays in the wild*" (Zhang, Chong et al., 2015), an dem er ebenfalls mitwirkte, wird das sogenannte *GazeHorizon* System vorgestellt (Zhang & Gellersen, 2014). Diese Arbeit ist aus dem Grund besonders interessant, weil an einem praktischen Beispiel ein *Attentive User Interfaces* basierend auf dem Erkennen von Blicken, bereits in Punkt 2.3 beschrieben, mit einer großen Anzahl an Testpersonen (190 Testpersonen - von denen über 150 befragt wurden), vorgestellt wird. Es konnte gezeigt werden, dass spontane blickbasierte Interaktion im öffentlichen Raum, hier ein öffentlicher Bildschirm ausgestattet mit einer Webcam, möglich ist und dass der Interaktionskanal Blick das Erlebnis der Nutzenden mit öffentlichen Displays verbessert (Zhang & Gellersen, 2014).

Des Weiteren beschreibt Bulling, in Zusammenarbeit mit Khamis und Alt in der Arbeit "*Tackling Challenges of Interactive Public Displays using Gaze*" Probleme, die bei der Interaktion mit Blick auftreten können, spezifisch im Kontext mit öffentlichen Displays. Als eine der größten und zugleich wichtigsten Herausforderungen wird dabei das Erkennen der Aufmerksamkeit der Nutzenden genannt. Dabei wird erwähnt, dass der Blick auf das Display ein guter Indikator für die Erkennung der Aufmerksamkeit im Allgemeinen ist und in den meisten Fällen einer Interaktion mit dem System vorausgeht (Khamis & Bulling, 2015). Dies deckt sich ebenfalls mit den aus den Studien im Punkt 2.3 gewonnen Erkenntnissen. Abschließend werden noch zwei wichtige Einschränkungen der Eyetracking Systeme in diesem Zusammenhang angesprochen. Das ist einerseits die Abhängigkeit von Lichtverhältnissen, die im Gegensatz zu Tests in einem Labor-Szenario nicht immer optimal sein können. Andererseits wird betont, dass das System mehrere Nutzende gleichzeitig erkennen muss, da dieser Fall in realen Versuchsreihen öfter auftrat.

## 2.6 Motivation und Forschungsfrage

Das vorliegende Kapitel zeigt einerseits auf, dass unter dem Begriff *smart* Software zu verstehen ist und andererseits, dass dabei eine der primären Herausforderungen die Erkennung der Intention des Nutzenden ist. Auch kann gezeigt werden, warum sich Blickerkennung als Lösungsansatz hierfür besonders gut eignet. Welche Voraussetzungen im Umfeld einer *Smart Kitchen* dafür aber vorhanden sein müssen, z. B. das Tracken der Nutzenden über den gesamten Küchenbereich, ohne eine störende Eyetracking-Brille, wird ebenso angesprochen. Unterschiedliche verwandte Arbeiten zeigen abschließend auf, welche Lösungsansätze es bisher gibt, um eine stabile Blickerkennung zu ermöglichen. Dabei werden aber auch Limitationen und weitere Herausforderungen an derartige Systeme genannt.

Vor allem die Arbeit *GazeDrone* (Khamis & Bulling, 2018) zeigt deutlich , dass einfache Webcams, ausgestattet mit *smarter* Software, ausreichend sind, um Blickerkennung zu ermöglichen, ohne den Nutzenden in seinen eigentlichen Handlungen einzuschränken. Die bereits angesprochene Geräusch-Problematik von Drohnen in geschlossenen Räumen, oder der zusätzliche Bewegungssensor in Projekten wie *EyeScout* (Khamis & Bulling, 2017) machen eine einfache Übertragung dieser Lösungskonzepte auf den Use Case *Smart Kitchen* unmöglich, nicht zuletzt aufgrund der unterschiedlichen Formen von Küchen, und verlangen daher nach einem anderen Lösungsansatz. Diese Arbeit schlägt eine Lösung für das Spannungsfeld zwischen der Nicht-Behinderung der Nutzenden und der Erkennung der Nutzenden über den gesamten Küchenbereich vor.

Das Ziel ist es daher, ein System zu entwickeln, dass es ermöglicht mit kostengünstiger Hardware und *smarter* Software Blicke von Nutzern zu erfassen und daraus resultierend deren Handlungsabsicht für ein mögliches Assistenzsystem abzuleiten. Dabei ist die grundsätzliche Idee aus den eben genannten Gründen mit günstigen Webcams, die über eine Küchenzeile verteilt sind, die Blicke der Nutzenden in einer Küche zu erfassen. Eine anschließende Evaluierung des Systems soll klären, wie gut das System arbeitet und wo seine Limitierungen liegen. Der Entstehungsprozess dieser Idee, deren Implementierung (Punkt 3 - Systementwicklung) und die Evaluierung des Systems (Punkt 4 - Studie, Punkt 5 - Ergebnisse und Punkt 6 - Diskussion) werden in den folgenden Kapiteln näher beleuchtet.

# 3 Systementwicklung

In diesem Kapitel wird auf die Entwicklung des Systems eingegangen. Dabei wird aufgezeigt, wie der konzeptionelle Entwurf aussieht und wie sich die Entwicklung zu diesem finalen Entwurf ergeben hat. Hierfür wird zuerst die Vorevaluierung 3.1 vorgestellt, bei der erstmals eine Webcam mit der verwendeten Software erprobt wurde. Die daraus gewonnen Ergebnisse flossen, in Kombination mit den in Punkt 2 erfassten Erkenntnissen, anschließend in die Anforderungsanalyse 3.2 ein, in der das eigentliche Problem- und Lösungskonzept vorgestellt wird. Unter Punkt 3.3 Systemimplementierung wird dann die Umsetzung der gesamten Soft- und Hardware vorgestellt. Abschließend wird eine Flaschenhalsanalyse durchgeführt und die Gesamtarchitektur des Systems kritisch hinterfragt.

## 3.1 Vorevaluierung

Bevor auf den eigentlichen Versuchsaufbau und den Hintergrund der Vorevaluierung eingegangen werden kann, muss zunächst das grundsätzliche technische Konzept grob beschrieben werden. Details zur Implementierung folgen zu jeder der hier angesprochenen Komponenten im Detail unter Punkt 3.3 Systemimplementierung. Die Vorevaluierung ist ein wesentlicher Teil der Anforderungserhebung und muss daher vor dieser besprochen werden, um in diese einfließen zu können.

Die grundsätzliche Idee basiert auf der Verwendung der *Open Source* Software *OpenFace 2.0*. *OpenFace* wurde ursprünglich von Tadas Baltrušaitis in Zusammenarbeit mit dem *CMU MultiComp Lab* unter der Leitung von Prof. Louis-Philippe Morency entwickelt (Baltrušaitis & Morency, 2016). Morency forscht dabei im Rahmen des *CMU* an den Grundlagen, um Computer mit der Fähigkeit zu auszustatten, um subtile menschliche kommunikative Verhaltensweisen während sozialer Interaktionen zu analysieren, zu erkennen und vorherzusagen (MultiComp Lab, 2019). Einige der ursprünglichen Algorithmen wurden durch die *Rainbow Group* der *Cambridge University* erstellt, welche seit den 1960ern im Bereich der Computergrafik und digitalen Bildverarbeitung forscht (Robinson, 2016). Diese Software wurde unter anderem in dem zuvor angesprochenen *GazeDrone* Projekt verwendet. Dabei werden die von einer Kamera erfassten Bilddaten durch die Software analysiert und die

Augenposition, sowie die Blickrichtung des Nutzers in Form einer *CSV*-Datei ausgegeben (Baltrušaitis & Morency, 2016).

Bei der eigentlichen Versuchsreihe der Vorevaluierung wurden von 13 Testpersonen, welche auf fünf Tage verteilt waren in der Künstlergarderobe der Universität Regensburg unterschiedliche Gerichte zubereitet. Während des gesamten Experiments waren die Probanden dazu angehalten, ihre Vorgehensweisen und Überlegungen laut mitzusprechen (*think aloud* Prinzip). Die Rezepte wurden den Probanden Schritt für Schritt auf einem vor ihnen platzierten Tablet angezeigt und per Sprachausgabe vorgelesen. Dies fand im Rahmen des Kurses "Algorithmen für Mensch-Maschine-Interaktion" statt. Parallel zur eigentlichen Untersuchung wurde eine Webcam im linken oberen Eck der Versuchsküche, mit Blick auf die Arbeitsfläche, positioniert. Anschließend wurden die gesammelten Daten von einer Gruppe des Kurses ausgewertet (Bosek & Büttner & Hauser, 2019).

In diesem frühen Stadium des Experiments wurde die Kamera mit Klebeband an der Wand fixiert, was, bedingt durch täglichen Auf- und Abbau der Experimentausrüstung, eine exakte nachträgliche Bestimmung der Kamera im Raum erschwerte. Diese exakten gleichbleibenden Faktoren (Position, Neigung, Drehung) sind allerdings entscheidend, um die durch *OpenFace* erfassten Blickdaten richtig auf das Küchenmodell umzurechnen, (vgl. 3.3.1 *OpenFace* und Client). Daher ist eine stabile Haltevorrichtung zwingend notwendig, um eine korrekte Umrechnung der Daten zu ermöglichen.

Des Weiteren wurde festgestellt, dass nur relative Zeitstempel für die Messdaten verfügbar waren. Dies hängt damit zusammen, dass bei der Versuchsreihe die Software auf einen *Raspberry Pi Model 3 B+* ausgeführt wurde. Das dort verwendete Raspbian *Linux OS* hatte keine Verbindung zum Internet und konnte somit keine korrekte Systemzeit liefern, was dazu führte, dass die erfassten Blickdaten nicht in Beziehung zu anderen erfassten Datensätzen, wie beispielsweise den Audioaufnahmen gestellt werden konnten.

Primär kann aber die schlechte Erkennungsrate von *OpenFace* genannt werden, die mit der Position der Kamera im Raum zusammenhängt, was nachfolgende Abbildung verdeutlicht.

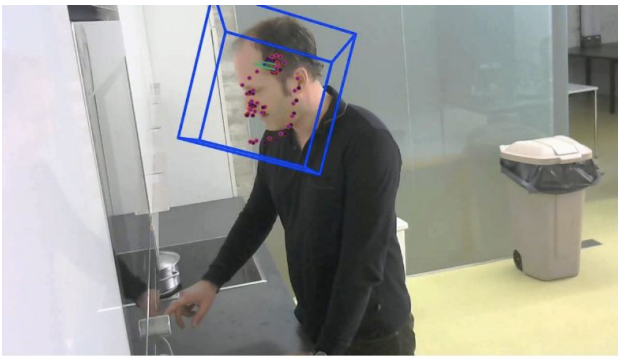

**Abbildung 1: *OpenFace* Aufnahme während der Vorevaluierung**

Durch die seitliche Position wird zwar die gesamte Küchenzeile erkannt, aber dadurch ist eine Erfassung des Gesicht nur sehr bedingt möglich und somit eine Blickerkennung fast unmöglich. Erschwerend kommt hinzu, dass die Augen, wenn der Nutzende am anderen Ende der Küche steht, aufgrund der zu niedrigen Auflösung, nicht mehr erkannt werden können.

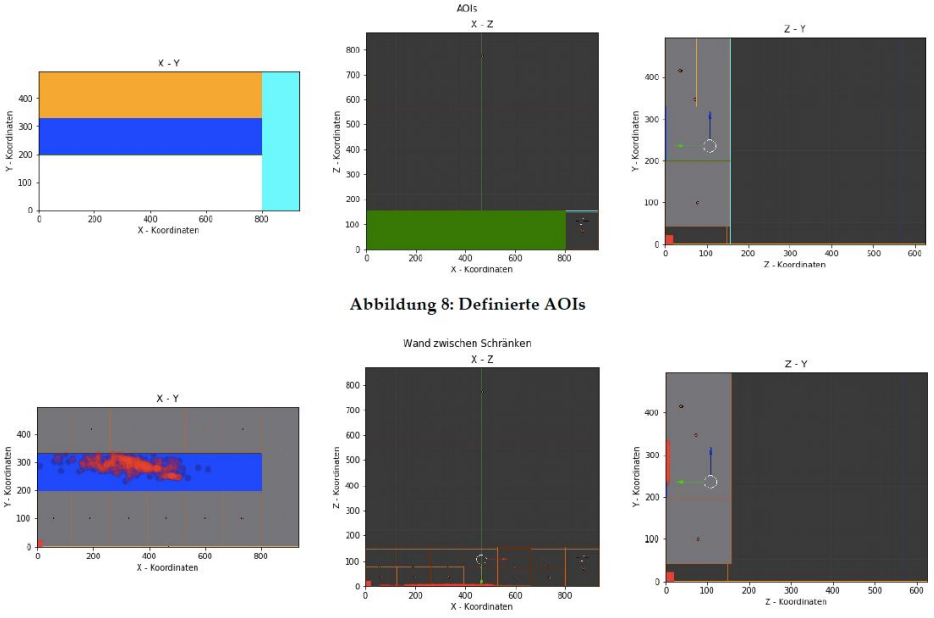


**Abbildung 8: Definierte AOIs**

**Abbildung 2: Visualisierung der Blickpunkte der Vorevaluierung**

Dies spiegelt sich deutlich in den Daten wieder. Diese sollten erwartungsgemäß auf das Tablet oder die Arbeitsfläche in der Mitte der Küche auftreten, da dort das Rezept mit den jeweiligen Arbeitsschritten präsentiert wurde. Jedoch werden primär Blicke zur Kamera hin, nach links oben, erkannt. Dadurch

erkennt das System nur Blickpunkte auf die Wand zwischen der Arbeitsfläche und den oberen Küchenschränken (blaue Fläche in Abbildung 2). Dort gibt es für den Nutzenden keine relevanten Arbeitsgegenstände. Diese Punkte ergeben sich teils aus der ungenauen Position der Webcam, andererseits an ihrer beschränkten Fähigkeit nur in diesem Bereich wirklich Blicke zu erfassen. Die aus dieser Vorarbeit gewonnenen Erkenntnisse fließen in die nachfolgende Anforderungsanalyse und dem finalen Konzept des Systems mit ein (Bosek & Hauser, 2019, S. 26).

## 3.2 Anforderungsanalyse und Problemlösungskonzept

In dem nun folgenden Kapitel werden die sich aus der Literatur und der Vorevaluierung ergebenden Anforderungen aufgelistet und ein adäquates Lösungskonzept abgeleitet.

### 3.2.1 Anforderungen aus der Literatur

**LF1 - Barrierefreiheit:** Betrachtet man die Ergebnisse der Literatur so zeigt sich, dass klassische Eyetracker für den Kontext Küche nur sehr bedingt geeignet sind. Zudem sind *Table-Mounted* Systeme sind nicht flexibel genug, den Nutzenden in der Tiefe des Raumes zu erfassen (Shell & Vertegaal, 2003). Daher muss das zu entwickelnde System, sowohl den Nutzenden über den gesamten Bereich der Küche erfassen, ohne ihn beim Interagieren mit der Küche zu behindern.

**LF2 - Kosten:** Bei regulären Eyetracking Systemen kommt der Kostenfaktor erschwerend hinzu. Dr. Bryn Farnsworth gab im Juli 2019 auf der Internetseite der Firma *iMotions* einen Überblick über aktuell auf dem Markt verfügbare Eyetracking Systeme. Dabei listete er die günstigen Systeme beginnend ab 169€ (*Tobii Eye Tracker 4C*) auf, bemängelt aber, dass diese Systeme zusätzlich noch einmal Kosten für lizenzierte Software der Hersteller verursachen bzw. die Daten nur in Kombination mit dieser speziellen Software genutzt werden können. Somit muss das zu entwickelnde System mit günstiger Hard- und Software ausgestattet sein (Farnsworth, 2019).

**LF3 - Echtzeit:** Die Hauptabsicht in der Erfassung von Blickdaten ist es, die Intention des Nutzers ableiten zu können. Das was der Nutzende anblickt, ist

das, mit dem er gerade interagiert, wodurch sich die Forderung nach einer Auswertung der Daten in Echtzeit ergibt. Das System muss zeitnah feststellen können, worauf der Nutzende gerade blickt.

### 3.2.2 Anforderungen aus der Vorevaluierung

**VF1 - Halterung:** Um die erkannten Blickdaten mit der Raumarchitektur abgleichen und bestimmen zu können, auf welches Objekt der Nutzende in der Küche blickt, muss eine gleichbleibende Ausrichtung und Positionierung der Kameras im Raum gegeben sein. Die sich daraus ergebende indirekte Kalibrierung des Systems ist entscheidend für dessen Genauigkeit.

**VF2 - Zeitstempel:** Assistenzsysteme können unterschiedliche Eingabequellen verarbeiten, um die Aktivität der Nutzenden zu bestimmen. Um diese unterschiedlichen Daten in ein Verhältnis zueinander setzen zu können, muss eine exakte zeitliche Zuordnung der Daten ermöglicht werden.

**VF3 - Position:** Als einer der wichtigsten Punkte ist die Position der Kamera im Raum zu nennen. Die Vorevaluierung zeigt deutlich, dass eine einzelne Kamera in einer Ecke der Küche die Nutzenden nicht ausreichend gut erfasst. Es muss daher ein vernünftig begründbares Kamerapositionierungskonzept vorgeschlagen werden, um die Nutzenden über die ganze Küchenzeile hinweg zuverlässig erfassen zu können.

**VF4 - Konfigurierbarkeit:** Ein weiterer zu berücksichtigender Aspekt sind die Anordnung von Küchenelementen einer jeden Küche. Diese können in einer einfachen Form wie in 3.1 beschrieben, vorliegen, bei welcher das von Khamis entwickelte *EyeScout* System, mit seinem auf Schienen gelagerten Eyetracker funktionieren könnte (Khamis & Bulling, 2017). Diese Systeme stoßen allerdings dann an ihre Grenzen, wenn beispielsweise die Küche in Form eines U's angeordnet ist. Das zu entwickelnde System muss sich folgerichtig an den Aspekt, der freien Anordnung von Küchenelementen anpassen und unabhängig von deren Architektur funktionieren.

### 3.2.3 Problemlösungskonzept

Das Konzept des hier in der Arbeit zu entwickelnden Systems versucht, die zuvor angesprochenen Anforderungen aufzugreifen und eine gesamtheitliche Lösung dafür zu präsentieren.

Die Lösungsidee basiert im Kern darauf, teure und den Nutzenden behindernde, Eyetracking-Systeme zu vermeiden und mit der *Open Source* Software *OpenFace* die Blicke der Nutzenden zu erfassen. Hierfür werden vier günstige Web-Kameras im gleichen Abstand entlang der Küchenzeile mit, im 3D Druck gefertigten, Halterungen positioniert. Jede Kamera ist mit ihrer eigenen Recheneinheit ausgestattet, auf der *OpenFace* installiert ist. Hierdurch ist der Aufbau um eine beliebige Zahl an Eyetracking Einheiten erweiterbar, angepasst an den Aufbau der Küche.

Jede Einheit sendet die von ihr erfassten Daten in Echtzeit an eine zentrale Recheneinheit. Diese trägt einerseits die Verantwortung dafür, die empfangenen Daten zeitlich genau zu persistieren, abhängig von der jeweiligen Sendeeinheit den Schnittpunkt der Blicke mit der Küche festzustellen und diese Ergebnisse einer nachgelagerten Anwendung zur Verfügung zu stellen.

Bei der nachgelagerten Anwendung handelt es sich um eine Webanwendung, die Daten einerseits in Echtzeit visualisiert und andererseits die Konfiguration der einzelnen Eyetracking Einheiten ermöglicht.

## 3.3 Systemimplementierung

In diesem Teil der Arbeit wird auf den Implementierungsaspekt näher eingegangen. Hierbei gliedert sich der Systemaufbau in drei wesentliche Komponenten auf. Dies ist zum einen die Eyetracking Einheit (3.3.1), die zentrale Recheneinheit, welche die erfassten Daten zusammenfasst (3.3.2) und die Webanwendung (3.3.3), welche jene Daten visualisiert. Dabei wird bei jeder Komponente explizit deren Funktion und Aufbau, aus Software- und Hardwarsicht, erklärt. Es werden einerseits die Schnittstellen zu anderen Komponenten beschrieben, als auch Grenzen und mögliche Probleme des jeweiligen Entwurfs aufgezeigt. Am Ende dieses Kapitels wird noch einmal eine Gesamtübersicht über das System gegeben, welche beschreibt, wie die einzelnen Komponenten untereinander kommunizieren und es wird mithilfe einer Flaschenhals Analyse untersucht, wo sich Schwachstellen des Systems

hinsichtlich des geforderten Echtzeitaspekts befinden. Zudem wird an unterschiedlichen Stellen mit Hilfe der Kürzel LF (Anforderungen aus der Literatur) und VF (Anforderungen aus der Vorevaluierung) auf die Anforderungen verwiesen, die das System an dieser Stelle zu erfüllen versucht.

### 3.3.1 Eyetracking Einheit

Die primäre Funktion der Eyetracking Einheit ist die Erfassung der Blicke des Nutzers mithilfe der *OpenFace* Software und die Weiterleitung der Daten an die zentrale Server-Einheit.

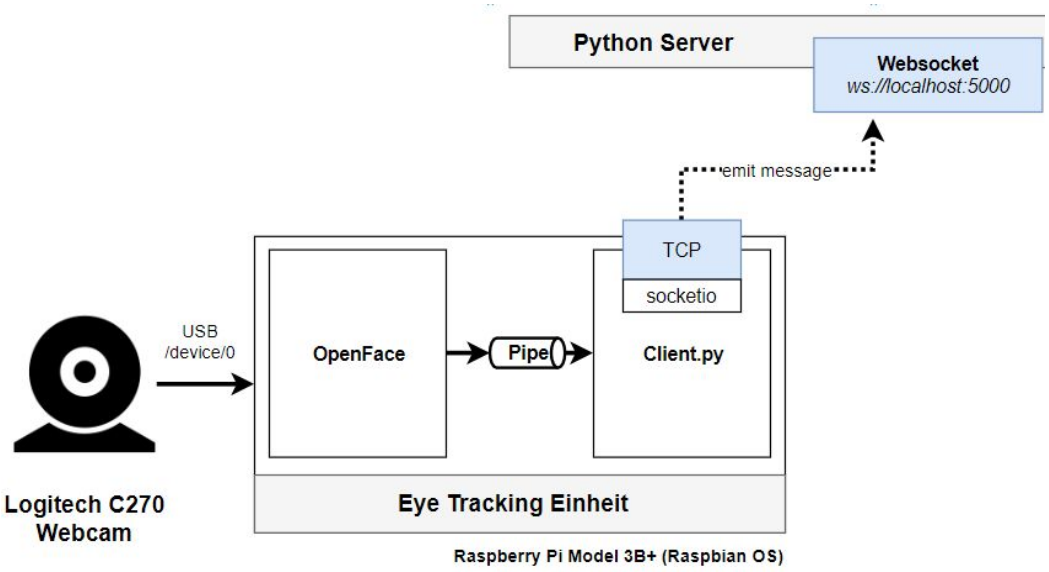


**Abbildung 3: Aufbau der Eye Tracking Einheit**

#### 3.3.1.1 Hardware

Im ersten Schritt wird auf den Aufbau der Hardware näher eingegangen. Dieser besteht im Wesentlichen aus drei Komponenten. Eine Komponente ist die *Logitech C270 HD Webcam*, welche bereits zu einem Preis von 21,99€ zu erhalten ist. Mit einer Auflösung von 720p/30FPS und einem Sichtfeld von 60°, bietet sie das beste Preis-Leistung-Verhältnis, zu vergleichbaren Systemen der Firma Logitech (*C525 - 47,99€; C920 - 62,90€*), oder anderen Herstellern. Zudem waren zwei dieser Kameras bereits am Lehrstuhl der Informationswissenschaften der Universität Regensburg vorhanden (LF2). An dem Gehäuse der Webcam ist ein Gelenk verbaut, an dem die serienmäßige Kamerahalterung montiert ist. Dieses Gelenk wurde genutzt, um eine Halterung an die Webcam anzubringen. Bei der

*C270* handelt es sich, bedingt durch den günstigen Preis, um eine der am weitest verbreiteten *Webcam* Systeme. Daraus ergibt sich, dass unterschiedliche Konstruktionen für Haltemechanismen im Internet gefunden werden können. Die Seite *Thingiverse*, deren Aufgabe darin besteht, durch Benutzende der Seite erstellte digitale Design-Entwürfe zu sammeln und auszutauschen, stellt somit eine bereits fertige, für die Arbeit passende, Konstruktion einer Halterung bereit (VF1).

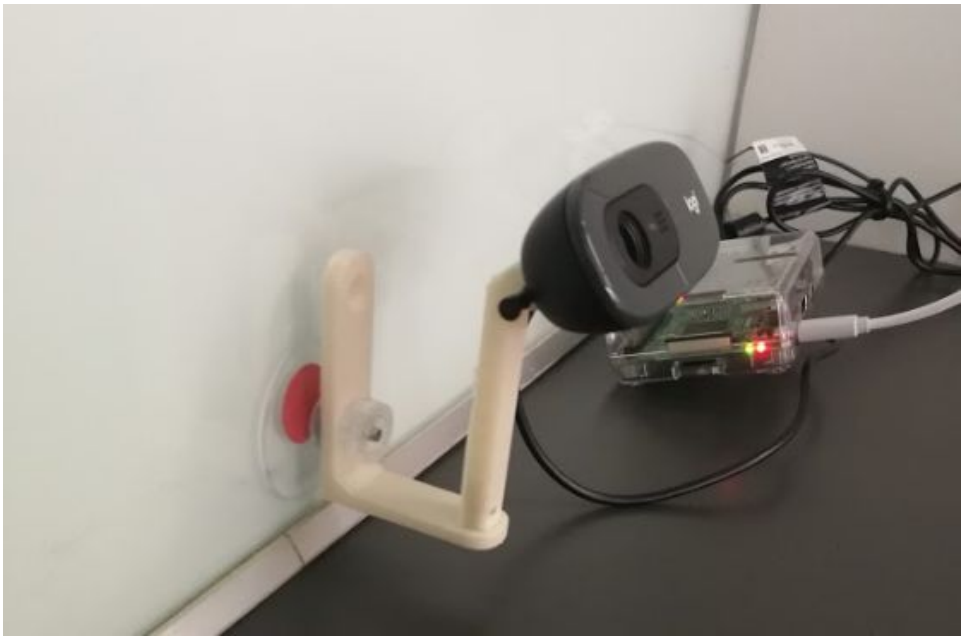

**Abbildung 4: Kamerahalterung an der Küchenwand**

In der Abbildung 4 ist die Halterung zu sehen, die aus zwei Teilen besteht. In der vorliegenden Küche gibt es einen Spritzschutz aus Glas, der es ermöglicht das Kamera Setup mit einem Saugnapf an der Wand zu befestigen. Dies ist durch das geringe Gewicht der Webcam (36g) und der Konstruktion (8g) möglich. Die Befestigung in einer regulären Wand, wäre jederzeit mit einer Schraube ebenfalls möglich, sollte keine glatte Oberfläche für den Saugnapf vorhanden sein (VF3).

Für die Verarbeitung der Bilddaten wurde ein *Raspberry PI 3 Model B+* verwendet. Mit 34,99 € handelt es sich dabei um eine vergleichbar günstige Recheneinheit (LF2), welche Entwicklern einen leichten Zugang zum System ermöglicht. Es ist einerseits möglich den Rechner, verwendet man das Raspbian Betriebssystem, über eine grafische Schnittstelle zu bedienen und andererseits ist standardmäßig ein *Wireless LAN* Modul verbaut, welches erlaubt

Softwareänderungen einzuspielen beziehungsweise Daten über das Netzwerk zu verteilen und zu empfangen. Zudem besitzt das Modell *3 B+* einen leistungsfähigen ARM-Vierkern Prozessor, welcher eine Taktfrequenz von 1,5 Ghz besitzt, was für die Analyse der Bilddaten entscheidend ist. Mögliche Alternativen zu dem *Raspberry PI* werden unter Einschränkungen 3.3.1.4 und im Punkt 3.3.4 bei der Flaschenhals Analyse besprochen.

#### 3.3.1.2 Software

Sowie die Hardware ist auch die Software in drei Subkomponenten unterteilt. Dies ist zum einen das Betriebssystem *Raspbian OS*, welches ein Debian-basiertes Betriebssystem ist. Dies wird einerseits vom Hersteller der *Raspberry PIs* zur Verwendung empfohlen (Raspberry PI Foundation, 2019), zum anderen empfehlen auch die Entwickler der Software *OpenFace* selbst ein Linux basiertes Betriebssystem (*OpenFace* Github Wiki, 2018). Wie bereits erwähnt ermöglicht die mit Raspian OS mitgelieferte grafische Oberfläche eine recht einfache Entwicklungstätigkeit, indem beispielsweise das *Remote VNC*[2] Protokoll unterstützt wird.

Bei der zweiten Komponente handelt es sich mit um eine der Kernkomponenten des gesamten Systems. *OpenFace* (Version: 2.2.0) ist eine Software, die für Forscher im Bereich des Maschinellen Sehens und des maschinellen Lernens gedacht ist, die interaktive Anwendungen auf der Grundlage der Gesichts Verhaltensanalyse entwickeln möchten. *OpenFace* könnte neben dem Erfassen der Blicke, dessen Funktionalität in diesem Forschungsprojekt hauptsächlich genutzt wird, auch die Kopfhaltung schätzen oder Gesichts Aktionseinheiten erkennen in Echtzeit (OpenFace Github, 2019). Dazu kommt noch, dass *OpenFace* als *Open Source* Software auf *GitHub* zur Verfügung steht. Entwickelt wurde das System ursprünglich von Tadas Baltrušaitis in Zusammenarbeit mit dem *CMU MultiComp Lab* unter der Leitung von Prof. Louis-Philippe Morency (OpenFace Github, 2019).

[2] Virtual Network Computing (VNC) ist ein Remote-Protokoll mit dem der Bildschirminhalt eines Remote-Rechners auf einem lokalen Bildschirm dargestellt und von diesem aus Bedienungsoperationen im entfernten Rechner ausgeführt werden können (itwissen, 2011)

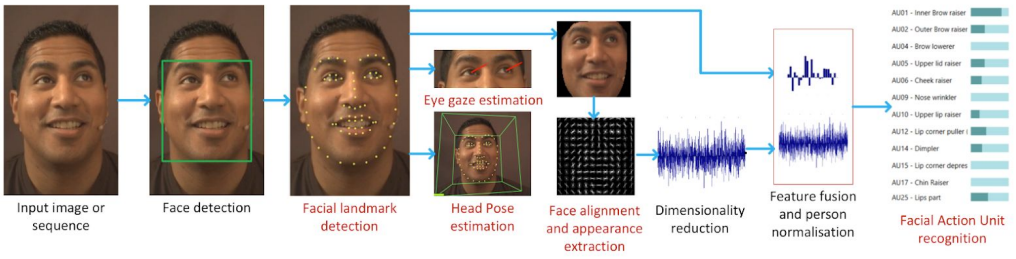


**Abbildung 5: *OpenFace* Pipeline (Baltrušaitis & Morency, 2016)**

Abbildung 5 zeigt die Gesichts Verhaltensanalyse Pipeline von *OpenFace*. Alle relevanten Entwicklungs Details beschreibt Tadas Baltršaitis in seiner wissenschaftlichen Arbeit "OpenFace: an open source facial behavior analysis toolkit". Der Verarbeitungsablauf gestaltet sich wie folgt, dass zuerst mit Hilfe des *Conditional Local Neural Fields (CLNF)* Framework versucht wird, mit 68 Landmarken (56 davon für die Augen) das Gesicht der Person vor der Kamera zu erkennen (Peng & Xu, 2009). Anschließend findet die Schätzung des Blicks, der Kopfposition, sowie die Ausrichtung des Gesichts und Aussehens Extraktion bzw. die Gesichts Aktionseinheiten Erkennung parallelisiert statt. Im hier vorliegenden Fall wird ausschließlich die Schätzung des Blicks und der Kopfposition genutzt. Das *CLNF Framework* erkennt zudem Landmarken im Bereich der Augen, was Augenlider, Iris und Pupille mit einschließt. Ist die Position des Auges bekannt wird individuell für jedes Auge der Blickrichtungsvektor berechnet. In der Arbeit wird zudem *OpenFace* gegen vergleichbaren System evaluiert, wobei sich zeigt, dass *OpenFace* die niedrigste Fehlerrate beim Schätzen des Blickrichtungsvektors aufweist (Baltrušaitis & Morency, 2016).

*OpenFace* stellt zudem unterschiedliche Verarbeitungsmodi des aufgenommenen Bildmaterials bereit. Hier wird der sogenannte *FeatureExtraction* Modus verwendet, der in Echtzeit die Bilddaten auswertet und in einer *CSV* Datei ablegt (LF3). Dabei ergibt sich aber das Problem, dass der aktuelle Entwicklungsstand der Software keine Echtzeit Weiterleitung dieser Daten ermöglicht. Daher wurde von hier für diese Arbeit ein *Fork* des auf Github befindlichen Quellcodes von *OpenFace* erstellt und die Datei *RecorderOpenFace.cpp* um eine Konsolen-Ausgabe erweitert. Dabei werden nun die erkannten Blickdaten nicht nur in die *CSV* Datei geschrieben, sondern jeder

neu analyse *Frame* im *JSON* Format zuvor auf der Konsole ausgegeben. Als Frame ist hier das von *OpenFace* gerade zu verarbeitende Bild zu verstehen.

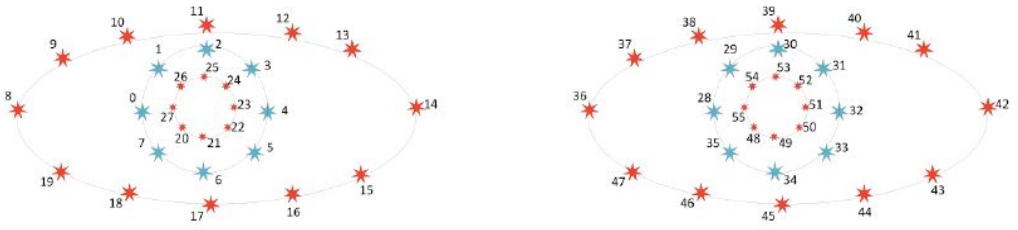

**Abbildung 6: 56 Augenlandmarken**

Dieser besteht zum einen aus den Metadaten von *OpenFace* wie *frame*, *face_id*, *timestamp*, *confidence* und *success (*Frame, erkannte Gesichts-ID, Zeitstempel, Wahrscheinlichkeitsgrad der geglückten Gesichtserkennung, erfolgreiche Erkennung der Blickrichtung). Zum anderen besteht er aus Richtungsvektoren, sowohl für das rechte *(gaze_0_x*, *gaze_0_y*, *gaze_0_z*), als auch für das linke (*gaze_1_x*, *gaze_1_y*, *gaze_1_z*) Auge. Dazu kommen noch alle, wie in der vorangestellten Abbildung zu sehen, 56 Augenlandmarken (*eye_lmk_X_i*, *eye_lmk_Y_i, eye_lmk_Z_i)* (OpenFace Github Wiki, 2018).

An dieser Stelle greift nun die dritte Softwarekomponente ein. Es handelt sich dabei um ein *Python Script*, das die jeweils einzelnen erkannten Frames aus der Konsole aufnimmt, auf Richtigkeit prüft und an den Server weiterleitet. Zudem wird jede an den Server weitergeleitete Nachricht mit einer *Client-ID* versehen. Dies ermöglicht es dem Server mehrere Clients voneinander zu unterscheiden und die Daten korrekt zu interpretieren (VF4). Ermöglicht wird dieses Auslesen der Konsolenausgabe durch den *Pipe* Operator, der unter Linux Systemen existiert. Dabei wird in der Befehlszeile zuerst *OpenFace* aufgerufen und direkt danach mittels *Pipe*-Operator das *Python Script* aufgerufen was dafür sorgt, dass die Daten nicht in die Konsole geschrieben werden, sondern an das *Python Script* weitergeleitet werden.

```
./FeatureExtraction -tracked -q -device 0 -gaze | python3
        ../../client/client.py 192.168.188.20 cam_2
```

Neben den Parametern für *OpenFace*, nur die Blickdaten auszuwerten und die zu verwendende Webcam, werden dem *Python-Script* (*client.py*) noch die Server

Adresse im Netzwerk und seine eigene *Client-ID* mitgegeben (VF4). Nach der Initialisierung der *ClientGazeLogger*-Klasse wird die Methode *watch_data_stream(self)* aufgerufen. Diese liest, den durch den *Pipe* Operator an sie weitergeleiteten, *Input Stream* aus und wartet auf den Schlüsselwert "*relevant_entry*", auf den dann das *JSON-Objekt* folgt, welches die Blickdaten beinhaltet. Diesem werden noch die aktuelle *Client-ID* (VF4) und der aktuelle Systemzeitstempel (VF2) hinzugefügt und daraufhin die Daten, mithilfe der Methode *push_to_server(self, message)* an den Server weitergeleitet.

#### 3.3.1.3 Schnittstellen

Wie bereits angesprochen, verarbeitet jeder Eyetracking Einheit ihre Daten nicht selbst, sondern reicht diese an die im Punkt 3.3.2 beschriebene zentrale Servereinheit weiter. Dies geschieht über das *TCP*-basierte *Websocket* Protokoll (Fette & Melnikov, 2011). Dabei wird von der Servereinheit ein *Socket* auf dem TCP-*Port 5000* im Netzwerk bereitgestellt, auf welchen sich initial beim Start der *client.py* verbunden wird. Dies ist daher möglich, da sowohl Server, als auch die Clients sich im gleichen Netzwerk befinden. Bei allen Aufbauten des Systems wurde dafür eine *AVM Fritzbox 7490* verwendet und allen verbundenen Geräten (Server/Clients) wird stets die gleiche IP-Adresse zugewiesen. Hintergründe bzw. Einschränkungen des *Websocket* Protokolls finden sich im Abschnitt 3.3.2.2 und 3.3.2.3. Primärer Vorteil für den Client an dieser Stelle ist aber die Entkopplung vom Server selbst, da nach dem Absetzen der Nachricht nicht auf eine Antwort von diesem gewartet werden muss und sofort die nächsten Blickdaten weiter geleitet werden können, die in der Konsole auftreffen.

#### 3.3.1.4 Einschränkungen

Betrachtet man den Softwareaspekt näher, so müsste *OpenFace* um das hier mit *Python* und *Pipe*-Operatoren realisierte Echtzeit Feature erweitert werden. Hierfür müsste die *RecorderOpenFace.cpp* erweitert werden um die eingehenden Daten direkt an eine Servereinheit weiterzuleiten. *C++* Bibliotheken wie *boost* können dies ermöglichen. Während der Entwicklung gelang es jedoch nicht, diese in die bestehende Projektstruktur mit einzugliedern. Zudem müsste eine Parametrisierung mit der Serveradresse und *Client-ID* zusätzlich möglich sein.

Hinsichtlich Auswahl der Hardware handelt es sich um ein prototypisches Konzept bei dem als kritischer Punkt der *Raspberry PI* zu betrachten ist. Mit teurer, bzw. auf den Anwendungsfall besser zugeschnittener Hardware, hätten schneller Blicke erfasst werden könnten. Das aktuelle System schafft die Auswertung von 1-2 *Frames* pro jede 5 Sekunden, bei einer *CPU* Auslastung von im Schnitt 75%. Dies ergibt sich daraus, dass *OpenFace* nur drei der vier verfügbaren Prozessorkerne unter Vollauslastung stellen kann. Im Punkt 3.3.4 wird das Konzept der hier vorliegenden dezentralen Lösung, dem einer zentralen Verarbeitung aller Daten, im Zuge der Flaschenhals Analyse, gegenübergestellt.

### 3.3.2 Server

Die Hauptaufgabe der zentralen Servereinheit ist die Annahme der von den Clients generierten Daten, deren Persistierung, die Client abhänige Transformation der Blickdaten und deren Schnittpunkt Berechnung mit den Küchenelementen. Anschließend müssen die errechneten Daten wiederum einem potentiellen Assistenzsystem bereitgestellt werden. In dieser Arbeit wird hierfür eine Visualisierungskomponente genutzt, die im Punkt 3.3.3 beschrieben wird.

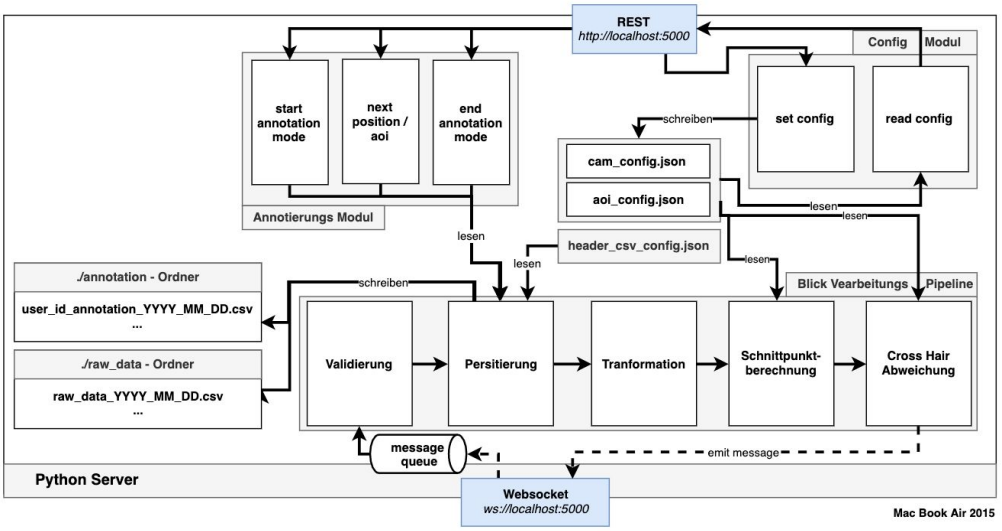


**Abbildung 7: Aufbau der Server Einheit**

#### 3.3.2.1 Hardware

Die beiden Anforderungen an die Hardware sind, dass einerseits auf dem Betriebssystem *Python 3* installiert sein muss und sich andererseit die

Servereinheit per *LAN* oder *WLAN* mit dem zuvor beschriebenen Netzwerk verbinden kann. In allen Testläufen wurde hierfür ein *Macbook Pro 2015* genutzt. Hierfür könnte jedoch auch eine durchaus schwächere Recheneinheit verwendet wären, wie beispielsweise ein weiterer *Raspberry PI.*

#### 3.3.2.2 Software

Die gesamte Serverlogik wurde in *Python 3* entwickelt. Dabei bilden Erweiterungen der Python Bibliothek *Flask* mit *Flask Restful* und *Flask Socketio* die *REST* bzw. die *Websocket* Verbindungen ab. Bibliotheken wie *numpy* werden für nahezu alle mathematischen Befehle genutzt. Darin liegt unter anderem die Designentscheidung *Python* zu nutzen, begründet. Im Laufe dieses Abschnitts wird auf mathematische Operationen noch näher eingegangen, die sich einfacher, als möglicherweise in anderen Programmiersprachen, umsetzen lassen und von sich aus viele benötigte Operationen bereits standardmäßig besitzen. Das auf *Python* basierende *Flask* macht es zudem möglich eine zuverlässige, *multithread*-fähige Server Architektur zu entwerfen, die ohne größeren Aufwand sowohl multiple *REST*, als auch *Websocket* Endpunkte unterstützt. Auf diese wird im nachfolgenden Punkt 3.3.2.3 näher eingegangen.

Das zentrale Kernstück des Servers ist die Pipeline zur Verarbeitung von Blickinformationen. Im Bereich des *Software Engineerings* versteht man unter dem Begriff Pipeline, dass die Ausgabe eines Befehls zugleich als Eingabe für den nachfolgenden Befehl dient (Buschmann, 2000, S.54), was in diesem Fall zwingend notwendig ist. Dabei nehmen zusätzlich das Annotierungs Modul bzw. das Konfigurations Modul Einfluss auf die Art der Verarbeitung in der Pipeline.

Treffen von den Clients neue Nachrichten auf dem *Websocket* des Servers ein werden diese im ersten Schritt validiert. Dabei wird sichergestellt, dass die erhaltenen Daten dem geforderten Datenformat entsprechen und für eine weitere Verarbeitung geeignet sind.

Daraufhin gibt es zwei parallel laufende Pipeline Abschnitte. Der eine der beiden ist für die Persistierung der Daten verantwortlich. Sobald der Server gestartet ist, wird auf dem Dateisystem ein *CSV* Datei mit entsprechenden Zeitstempel abgelegt, die somit als Log-Datei aller aktueller eingehender Ereignisse zu verstehen ist. Dieser Persistierungsprozess ist von außen steuerbar. Das bedeutet, dass der Annotierung Modus, der im Kapitel 4.1 näher

beschrieben wird, den Server dazu anhalten kann, parallel zu der standardmäßig erzeugten *CSV* Datei, eine sitzungsbasierte Datei für den aktuellen Tester während der Systemevaluierung anzulegen. Beim Ablegen der Dateien in beiden Modi wird zudem darauf geachtet den Systemzeitstempel *server_timestamp*, die Daten aus *OpenFace* und die *Client-ID* mit abzulegen, um eine nachträgliche Auswertung zu ermöglichen (VF2).

Der zweite Pipeline-Abschnitt ist für die Transformation der erhaltenen Blickdaten verantwortlich. Die entsprechende Mathematik und die Grundlage für den Quellcode hierzu, ebenso wie zu der nachgelagerten Schnittpunkt-Berechnung, stammt von Markus Bosek, Johannes Büttner und Tobias Hauser aus dem Kurs "*Algorithmen für Mensch-Maschine-Interaktion*" am Lehrstuhl für Informationswissenschaften der Universiät Regensburg (Bosek & Büttner & Hauser, 2019). Damit festgestellt werden kann, welche Objekte in der Küche der Nutzende tatsächlich anblickt, müssen die durch *OpenFace* erzeugten Daten in das Koordinatensystem des 3D Modells umgewandelt werden. Hierfür wurde unter anderem mit Blender ein 3D Modell erzeugt und mit *LibreCAD* eine technische Zeichnung der Küche (*side, top, front*) erstellt. Die Grundlage der Transformation bildet hierbei eine Transformationsmatrix, deren mathematischen Details sich im Abschnitt 4.3 Koordinatentransformationen in der zuvor erwähnten Arbeit nachlesen lassen. Dabei ist im Besonderen bei der Umwandlung wichtig, dass der Ursprung des Koordinatensystems bekannt ist, in das umgewandelt werden soll. Als Ursprung kann die Position der Kamera verstanden werden. Auf Grundlage der Kameraposition und deren Rotation im Raum kann nun die Transformationsmatrix erstellt werden. Aufgrund des Konfigurations Aspekts sind Kameraposition beziehungsweise deren Rotation für jeden bekannten Client in der der *cam_config.json* Datei abgelegt, welche per Webanwendung über das *REST Interface* bearbeitbar ist. Jeder Client benötigt aufgrund unterschiedlicher Positionen entlang der Küchenzeile eine andere Transformationsmatrix. Daher ist es auch wichtig, dass sich jeder Client gegenüber dem Server mit seiner *Client-ID* identifiziert, damit dieser bei der Transformation weiß, welche Matrix verwendet werden muss, um die Daten korrekt zu interpretieren (VF4). Hierbei wird noch einmal die zentrale Bedeutung einer stabilen Kamerahaltung deutlich. Diese stellt sicher, dass die Ausrichtung und Positionierung der Kamera durchgehend bzw. nach einem Ab- bzw. erneuten Aufbau gleich bleibt (VF1) (Bosek & Büttner & Hauser, 2019).

Die transformierten Blickdaten werden nun an den nächsten Verarbeitungsschritt der Schnittpunktberechnung weitergegeben. Dabei werden sogenannte *Areas of Interest* (AOIs) in der Küche, als dreidimensionale Rechtecke definiert (Bosek & Büttner & Hauser, 2019). Jede AOI besteht daher aus drei Punkten und ihrem Mittelpunkt (*Cross Hair Point*), der für die spätere Evaluierung benötigt wird. Diese AOIs lassen sich beliebig definieren. Für die spätere Evaluierung wurden hierfür 26 unterschiedliche AOIs festgelegt, die sich an realen Küchengegenständen orientieren. Die transformierte Augenposition und der Blickrichtungsvektor, erzeugt durch *OpenFace*, werden dann mit der durch die AOI definierten Ebene geschnitten. Dies wird für beide Augenpaare durchgeführt. Dabei wird derjenige Schnittpunkt als Endpunkt definiert, der die geringste Distanz zu der AOI an sich hat. Folglich kann somit die geschnittene AOI als angeblickte AOI definiert werden (Bosek & Büttner & Hauser, 2019). Alle AOIs werden in der *aoi_config.json* definiert, welche ebenfalls per Webanwendung über die *REST* Schnittstelle bearbeitbar ist.

Im letzten Schritt der Pipeline wird die euklidische Distanz des eben berechneten Schnittpunkts zu dem AOI abhängigen *Cross-Hair* Punkt berechnet. Hierbei handelt es sich um einen Punkt der in der Mitte der AOI angebracht wurde, dessen Funktion im Punkt 4.1 genauer erläutert wird.

Abschließend werden die von den Clients erhaltenen Daten, die transformierten Daten, der Schnittpunkt mit der AOI, die getroffene AOI selbst und die Distanz zum *Cross-Hair* Punkt in einem *JSON-Objekt* zusammengefasst und auf dem *Websocket* veröffentlicht.

#### 3.3.2.3 Schnittstellen

Die Servereinheit besitzt zwei Schnittstellen, dies ist zum einen der *Websocket*, über den Daten der Clients angenommen bzw. Daten für die Webanwendung bereitgestellt werden. Zum anderen existiert ein *REST Interface*, das es ermöglicht die Konfiguration der Eyetracking Einheiten und der Areas of Interests abzuändern und andererseits Befehle für den Annotierungs Modus bereitstellt.

In der folgenden Abbildung sind alle *REST* Endpunkte mithilfe einer Swagger Dokumentation abgebildet. Dies gliedert sich wie bereits angesprochen in die beiden Blöcke *annotation* und *configuration* auf. Die *REST*

*Interfaces* werden ausschließlich von der Webanwendung genutzt und dienen primär der einfacheren System Bedienbarkeit.

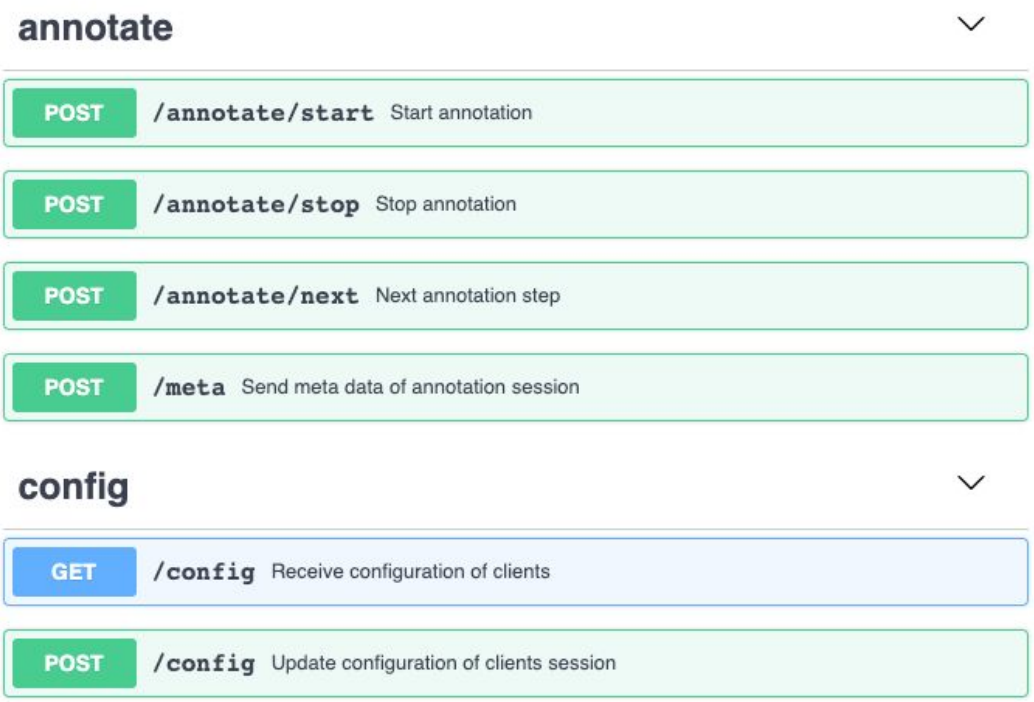


**Abbildung 8: Swagger Definition der Schnittstellen**

Die zweite Schnittstelle ist hier der *Websocket*. Durch die Verwendung des *Python Frameworks Flask* und dessen Erweiterung *Flask SocketIO*, sind sowohl die Websocket als auch die *REST-Schnittstelle* über einen gemeinsamen *Port* erreichbar.

Zuerst muss der Websocket aus Sicht des Clients betrachtet werden. Zu Beginn der Implementierung wurde versucht, die Kommunikation zwischen Client und Server ebenfalls mit einem *POST* basierten *REST* Endpunkt zu lösen. Dabei trat das Problem auf, dass der Server zwischen 2-5 Sekunden, bei nur zwei gleichzeitig Daten erzeugenden Clients, benötigte, um auf eine *REST* basierte Anfrage zu antworten, was die Verarbeitung weiterer Daten auf dem Client blockierte. Dies liegt zum einen daran, dass *REST* auf dem *HTTP* Protokoll basiert und daher auf eine Antwort des Servers wartet. Zum anderen sind die Endpunkte bei einer laufenden Anfrage zeitweise durch gerade zur Verarbeitung anstehenden Anfragen blockiert. Daher ist es nötig diese direkte Kopplung zwischen Client und Server aufzulösen, um nicht unnötige Wartezeiten zu erzeugen. Somit wurde eine *Websocket* Schnittstelle als

naheliegendste Lösung, einfach in das *Flask Framework* eingliederbar, herangezogen. Das Websocket Protokoll erwartet keine Antwort vom Server. Der Client veröffentlicht seine Daten auf dem vom Server bereitgestellten *Socket*. Dieser *Emit* findet sich in der *client.py* und wird bei jedem neu eintreffenden Datensatz, der von *OpenFace* erzeugt wird aufgerufen.

```
50        def push_to_server(self, message):
51            self.sio.emit('message', message)
```

**Abbildung 9: Client Methode zum Versenden der Daten**

Die vom Client veröffentlichten Nachrichten werden serverseitig angenommen und in eine *Nachrichten-Queue* geschoben, durch die eine sequentielle Abarbeitung möglich ist (Black & Johnson, 1999). Die Methode *handle_message(message)* ist einerseits für die weitere Verarbeitung der Daten und andererseits für die Weitergabe der bereits verarbeiteten Daten verantwortlich, indem diese wiederum auf einen *Websocket* veröffentlicht werden.

```
13  @socketIO.on('message')
14  def handle_message(message):
15      print('received message: ' + str(message['client_id']))
16      parsed_data = mainClass.main_method(message)
17      emit_data_object = {
18          'raw': message,
19          'parsed_data': parsed_data
20      }
21      emit('event', emit_data_object, broadcast=True)
```

**Abbildung 10: Server Methode zur Annahme der Daten**

Aus der Sicht der Webanwendung ist es auch sinnvoll einen *Websocket* als Kommunikationsschnittstelle zu verwenden. Eine dem Server nachgeschaltete Anwendung weiß nicht wann Daten vom Server geschickt werden, müsste also in einem gewissen zeitlichen Abstand den Server aktiv nach neuen Daten fragen, auch wenn keine neuen Blickdaten zur Verfügung stehen würden. Websockets bringen hier den Vorteil mit, dass nachgelagerte Anwendungen über neue Nachrichten benachrichtigt werden. Dazu kommt noch, dass

möglicherweise bei einem zukünftigen Anwendungs-Szenario nicht nur eine Anwendung Daten abgreifen möchte. Andere Dienste oder Anwendungen könnten aufgrund des *Socket* Ansatzes ebenfalls einfach mit Daten versorgt werden.

#### 3.3.2.4 Einschränkungen

Eine der primären Einschränkungen des Systems ist, wie viele Clients das System gleichzeitig verarbeiten kann. In dem vorliegenden Anwendungsfall wurde das System mit maximal vier Clients gleichzeitig unter Last gesetzt. Tatsächlich wäre es in Zukunft sehr interessant die Server Komponente mit Lasttests, abhängig von der Hardware (*Raspberry Pi* oder Desktop PC) auf der sie gerade läuft, zu testen. Als kritisch ist neben der eigentlichen Verarbeitungsgeschwindigkeit auch das Protokoll zu bewerten, mit welchem Server und Client miteinander kommunizieren. In diesem Fall wurde sich unter anderem auch für das *Websocket* Protokoll entschieden, da es als asynchrones Protokoll gemeinsam mit dem *Flask REST Framework* zu implementieren ist. Trotz seiner im vorangegangen Punkt geschilderten Vorteile gibt es doch Probleme, die möglicherweise andere Protokolle besser lösen könnten. Dem zuvor angesprochenen Last-Problematik könnten das *gRPC* Protokoll positiv entgegenwirken. *gRPC* wurde mit dem Einsatzzweck im Themenfeld der *Internet of things* von *Google* entwickelt, in welchem wir uns mit mit der Smart Kitchen befinden. Die Clients sind hierbei als *Things* zu verstehen, die mit Hilfe von *gRPC* extrem effizient komprimierte Nachrichten an den Server übermitteln könnten. Dabei würde die eigentliche Nachricht in ein Programmiersprachen-unabhängiges *Protobuf* umgewandelt werden. Bedenkt man, dass nicht der *Raspberry Pi*, sondern vermutlich spezielle Hardware im finalen Produkt verwendet werden, so könnte das *gRPC* Protokoll, speziell für extrem stromsparende Situationen entwickelt, das besser geeignete Kommunikationsprotokoll sein (Hempel, 2017).

Neben dem Umstand, dass Nachrichten effizient gesendet werden müssen, kommt noch der Aspekt der Zuverlässigkeit der Kommunikation hinzu. Aktuell besteht die Problematik, dass ein erneutes Verbinden zwischen Client und Server aufgrund nach einem Verbindungsabbruch nur schwierig wieder herzustellen ist, was unter anderem an dem *Websocket* Protokoll liegt. Zudem gehen hierbei in der Zwischenzeit Nachrichten verloren. Protokolle wie

beispielsweise *MQTT*, die Mithilfe eines *Brokes* Nachrichten vorhalten und sicherstellen, dass diese übermittelt werden, würden dieses Problem lösen (Hempel, 2017).

### 3.3.3 Web Anwendung

Wie bereits angesprochen ist die Webanwendung nur eine Zusatzkomponente, ohne die das System trotzdem nutzbar wäre. Die öffentlich im Netzwerk verfügbare Schnittstelle der Servereinheit, auf der fortlaufend neue Daten veröffentlicht werden, könnte beispielsweise auch von einem Assistenzsystem als Eingabequelle für Blickdaten genutzt werden. Nichtsdestotrotz erfüllt sie im Kern drei wichtige Aufgaben. Die Darstellung der Daten in Echtzeit, das einfache Konfigurieren der Clients und die Annotierung von Echtzeitdaten für die spätere Evaluierung des Systems. Der Punkt Hardware ist in diesem Kapitel bewusst ausgegliedert, da es an diese keine besonderen Anforderungen gibt. Jedes System, das fähig ist, sich mit einem *WIFI Netzwerk* zu verbinden und über einen *Browser* verfügt, der *JavaScript ES5* kompatibel ist, kann die Webanwendung nutzen. Das System wurde in allen hier vorliegenden Anwendungsszenarien mit einem *MacBook Air 2015* und der aktuelle Version des *Chrome Browsers* (Version: 77) genutzt. Die folgende Abbildung zeigt den Aufbau der Webanwendung und sowohl deren Verwendung der *REST*, als auch der *Websocket* Schnittstelle des Servers.

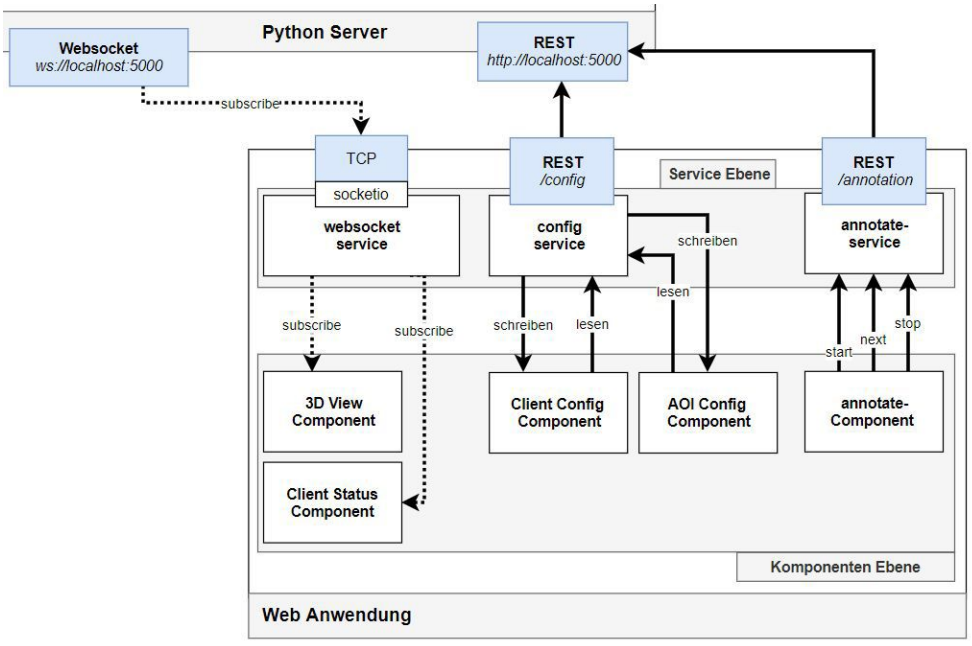


**Abbildung 11: Aufbau der Webanwendung**

#### 3.3.3.1 Software

Es wurde sich für die Implementierung einer Webanwendung aus zwei Gründen entschieden. Es war zum einen bereits vor der Entwicklung Wissen über die Konzeption derartiger Anwendung mit dem *Framework Angular 7* vorhanden, was die Entwicklung in diesem Fall vereinfachte. Zudem bietet Angular einige Vorteile, auf die im Verlauf dieses Kapitels noch eingegangen wird. Zum anderen sind Webanwendungen plattformunabhängig, wie bereits im vorangegangen Absatz erwähnt.

Die Webanwendung besteht dabei im Wesentlichen aus zwei Ebenen. Dies ist zum einen die Komponentenebene und Serviceebene, auf die im nachstehenden Punkt 3.3.3.2 näher eingegangen wird. Im *Angular Framework* können die einzelnen Seiten in Komponenten aufgegliedert werden, wobei diese wieder in anderen Komponenten verschachtelt wiederverwendet werden können. In der hier vorliegenden Anwendung steht jede Komponente für eine Unterseite des Systems.

Eine Ausnahme hierbei bildet die client-status.component. Diese findet sowohl in der *3d-view-component*, als auch *annotate-component* Verwendung. Diese Komponente erhält ihre Daten über den Websocket des Servers und zeigt in Echtzeit an, welche Daten von welchem Client eintreffen. Dabei wird als optisches Erkennungsmerkmal ein Client anstatt mit Rot mit Grün markiert. Erhält die Anwendung zu einem Client keine Daten mehr, wird dieser nach zehn Sekunden wieder mit Rot markiert. Somit ist dessen Inaktivität einfach zu erkennen.

Die *3d-view-component* nutzt *WebGL*, um die vom Server erhaltenen Daten in einer 3D Ansicht der Küche einzusetzen und zu visualisieren. Dabei ist es möglich sich frei mit hilfe der Maus um das 3D-Objekt zu bewegen. Zudem wird immer, sollte ein Blick auf ein Objekt erkannt werden, diese Information in der rechten oberen Ecke in Textform angezeigt.

Die *aoi-config-component* bzw. die *cam-config-component* bekommen jeweils die aoi-config.json und cam-config.json, die der Server verwendet. Beide erstellen abhängig von der jeweiligen JSON Datei eine Eingabemaske, diese JSON zu verändern und mit dem Klick auf einen *Save Button* wieder auf dem Server zu persistieren. Beide geben zudem die Möglichkeit sowohl neue Clients, als aus AOIs, durch das Klicken des *Add Buttons* anzulegen.

Eine Besonderheit bildet die annotation-component durch ihren Verwendungszweck. Dienen alle anderen Komponenten entweder der Visualisierung der Daten oder der Konfiguration des Servers, so ist die Notwendigkeit dieser Annotierungs-Seite aufgrund der Evaluierung des Systems entstanden. Aus diesem Grund wird auf ihre Anforderungen und Funktionalitäten im Punkt 4.1 näher eingegangen.

#### 3.3.3.2 Schnittstellen

Neben der eben beschriebenen Komponenten Ebene gibt es auch eine Service Ebene. Dies in Angular als Singletons erdachten Services sind als Schnittstelle Ebene zu unterschiedlichsten Diensten gedacht und können gleichzeitig von mehreren Komponenten genutzt werden. Dabei stellt die *config-service.ts* die *GET* und *SET* Methoden für die jeweiligen Konfigurationsdateien zur Verfügung und spricht somit direkt mit der *REST*-Schnittstelle des Servers. Exakt die gleiche Funktion hat der *annotation-service.ts*. Dieser übermittelt per *POST*-Aufruf den aktuellen Befehl bzw. den Status im Annotierung Vorgang an den Server.

Eine besondere Bedeutung kann dem *websocket-service.ts* zugeschrieben werden. Dieser abboniert mit Hilfe des *socket.io* Frameworks auf den *Websocket* des Servers und horcht auf diesen, um transformierte Blickdaten bzw. angeblickte Objekte zu erhalten.

```
11      public getMessage(): Observable<any> {
12          return this.socket.fromEvent("event").pipe(map( data => data ));
13      }
```

**Abbildung 12: Webanwendungs Methode zur Annahme der Daten**

Jede Komponente, die diese Echtzeitdaten erhalten will, abonniert ihrerseits wiederum die Methode *getMessage()* dieses Dienstes. Deren Rückgabe ist ein Datenfluss in Form eines Observables. Dabei wird kommt ein sogenanntes reaktives Programmierschema (*reactive coding*) zum Einsatz, das durch die Verwendung von *Websockets* impliziert wird (Lohmüller, 2016). Dies macht hier besonders Sinn, da man eben nicht möchte, dass die Komponente als Konsument der Daten permanent anfragen muss, ob neue Daten vorhanden sind, sondern eventbasiert und automatisch über neue Informationen

benachrichtigt wird. Daraus resultiert zudem eine einfache Erweiterbarkeit des Systems, da jede weitere neue Komponente diese Methode nutzen kann, um informiert zu werden, ohne dass der *Service* hierfür diese Komponente kennen muss.

#### 3.3.3.3 Einschränkungen

Eine der Einschränkungen, die die Webanwendung mit sich bringt ist, dass sie aktuell noch nicht für die mobile Nutzung angepasst ist. Das verwendete *CSS* Framework der Anwendung (*materialize.css*) würde dies aber möglich machen.

Der im Punkt 3.3.2.4 disktuierte Einsatz von anderen Protokollen zur Kommunikation mit dem Server macht hier nur bedingt Sinn. Die Kommunikation über *REST*, für das Übermitteln einfacher *JSON* basierter Daten, ist als ein etablierter Standard anzusehen. Eine Alternative zu dem hier vorliegenden *Websocket* Protokoll könnten *Server-Sent Events (SSE)* sein. *SSE* ist ein Mechanismus, der es dem Server ermöglicht, die Daten asynchron an den Client zu übertragen, sobald die Client-Server-Verbindung hergestellt ist. Dies würde sich anbieten, da die *Websocket* Verbindung durch die Webanwendung nur unidirektional genutzt wird. Darüber hinaus verfügen Server-Sent Events über eine Vielzahl von Funktionen, die *WebSockets* fehlen, wie z.B. das automatische Wiederverbinden, *Ereignis-IDs* und die Möglichkeit, beliebige Events zu senden.

### 3.3.4 Gesamtübersicht

Im folgenden Abschnitt soll nun das System in seiner Gesamtheit betrachtet werden, nachdem in den vorangegangen Kapiteln die jeweilige Komponenten getrennt voneinander im Detail betrachtet wurden. Dabei wird noch einmal kurz die Skizzierung der Systemübersicht erläutert und anschließend eine Flaschenhals Analyse durchgeführt.

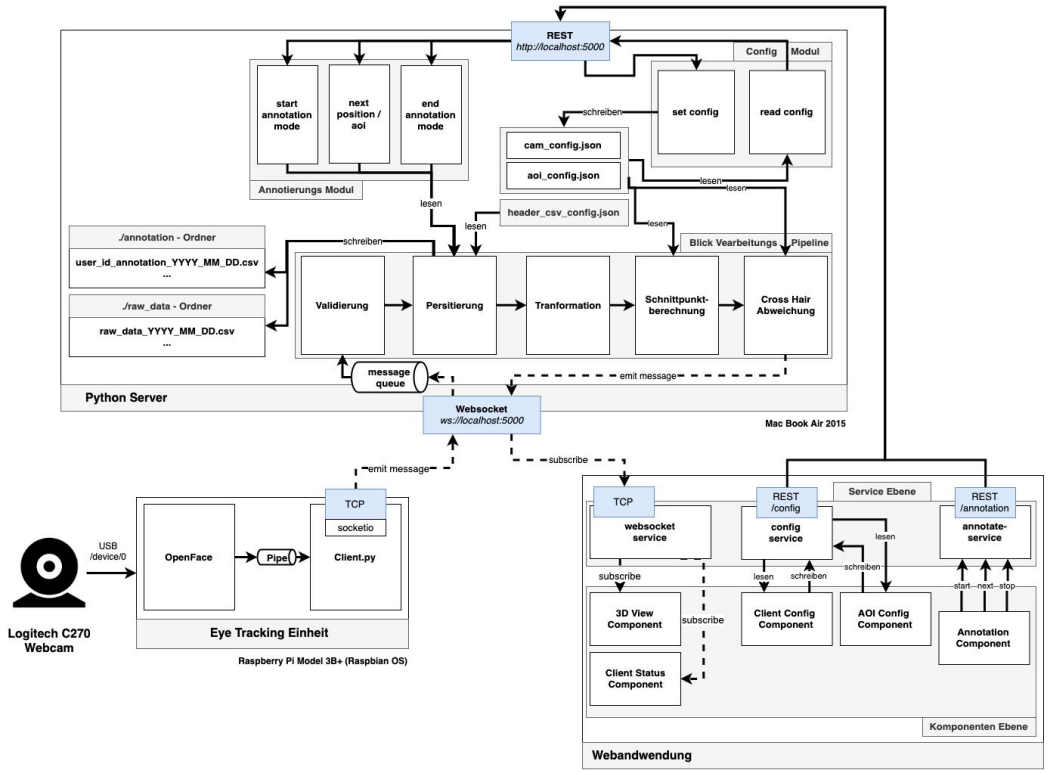


**Abbildung 13: Aufbau des gesamten Systems**

Die Gesamtübersicht verdeutlicht den Fluss der Daten durch das System. Die Daten werden von der Eyetracking Einheit erzeugt und asynchron (gestrichelte Linie) an die Websocket-Schnittstelle weitergeleitet. In der Servereinheit selbst sind die einzelnen Module abgebildet und an welchen Stellen und auf welche Art das Konfigurations- und das Annotierung-Modul Einfluss auf die Verarbeitungs-Pipeline nehmen. Zudem wird verdeutlicht welche Teile der Webanwendung mit welche Modulen des Servers verbunden sind.

### 3.3.4.1 Flaschenhals Analyse

Im Oberseminar vom 1.7.2019 der Lehrstuhl für Medieninformatik an der Universität Regensburg wurde in einer Diskussionsrunde die Erstellung einer Flaschenhals Analyse angeregt, um potentielle Schwachstellen des System zu erkennen. Hierfür wurden alle Verarbeitungsschritte vom Erzeugen des Eingangsbilds durch die Webcam auf dem Client, bis hin zur Visualisierung der Daten in der Webanwendung identifiziert und mit Systemzeit Logausgaben versehen. Das System wurde so aufgebaut, dass sich alle Webcams direkt nebeneinander befanden und zur selben Zeit ein Gesicht gleichzeitig erfassen konnten. Dann wurden über 142 Minuten die Zeit-Logausgaben gesammelt und

ausgewertet, wobei 67.392 Datensätze erzeugt und anschließend deren zeitlicher Abstand zueinander bestimmt wurden. Die nachfolgende Abbildung zeigt alle Verarbeitungsschritte und die jeweiligen Dauer der einzelnen Schritte.

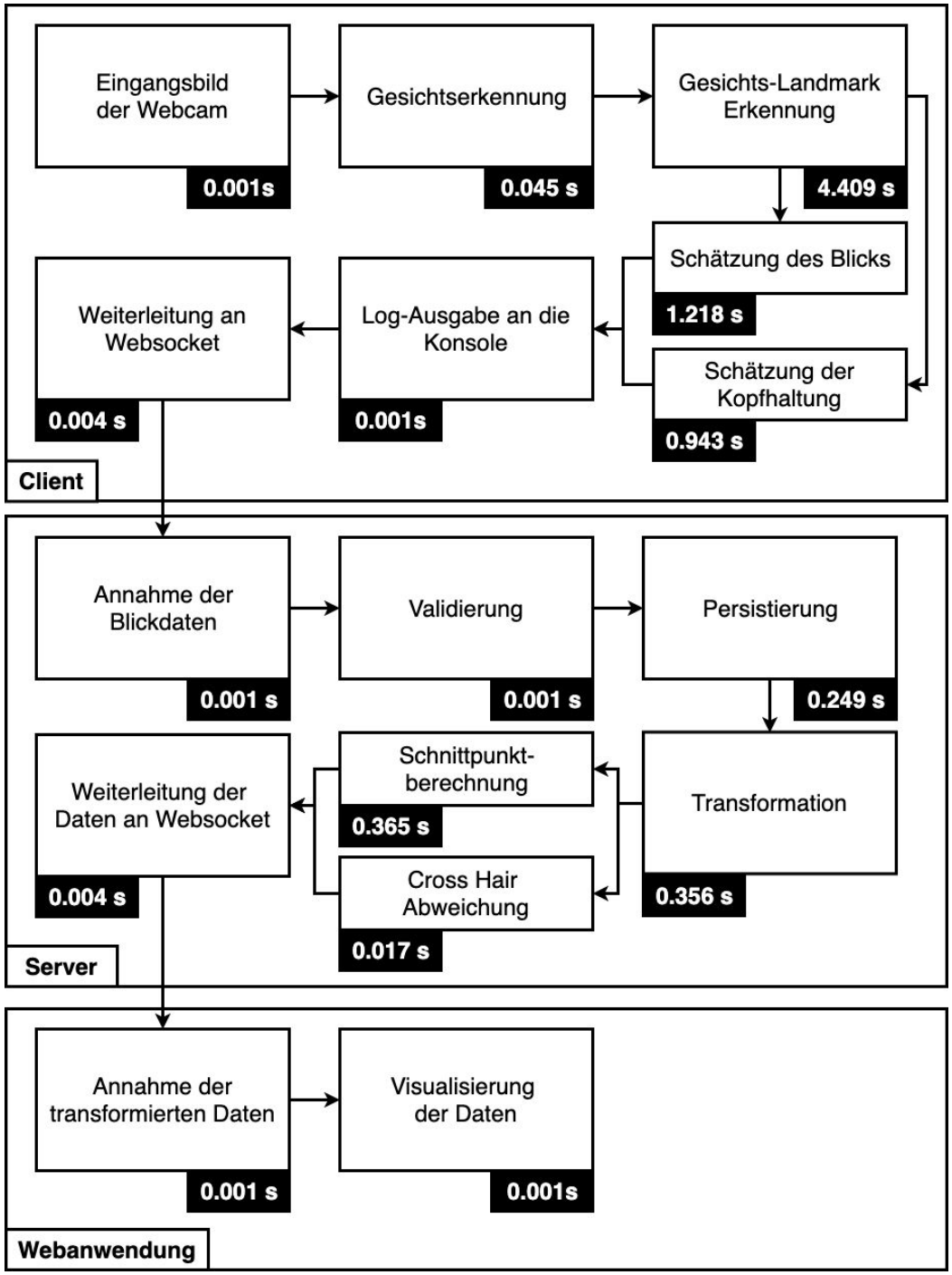


**Abbildung 14: Verarbeitungsschritte des System mit den durschnittlichen Verarbeitungszeiten**

Hierbei wird deutlich, dass sich der eigentliche Flaschenhals des Systems in der Software *OpenFace* selbst zu finden ist. Die "Gesichts-Landmark-Erkennung" ist der mit Abstand am zeitintensivsten Arbeitsschritt. Deutlich wird dies noch einmal durch grafische Auswertung der Zeiten.

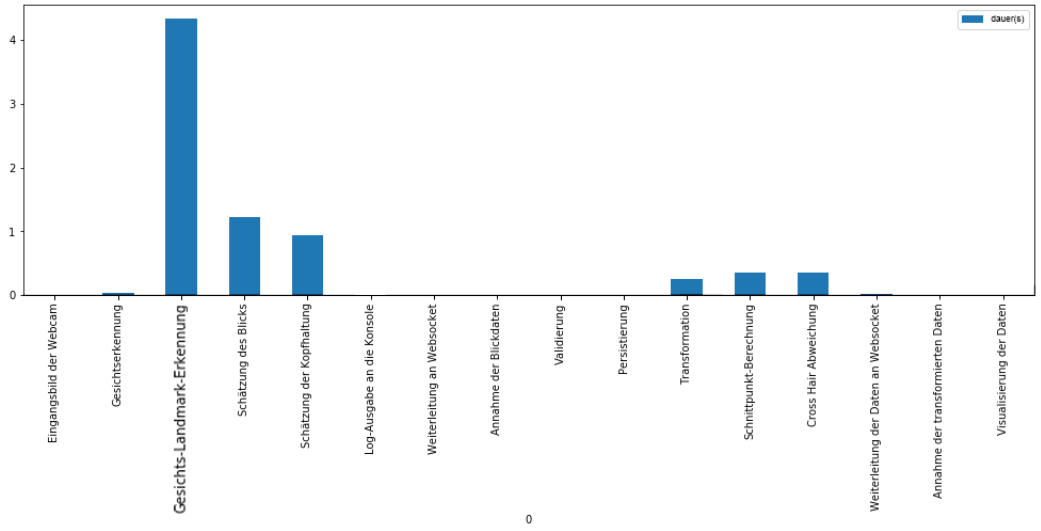


**Abbildung 15: Visualisierung der Zeiten der Verarbeitungsschritte**

Die "Gesichts-Landmark-Erkennung" dauert mit 4,409 Sekunden mit Abstand am längsten. Ebenso benötigt die Berechnung der Augenposition am zweitlängsten mit 1,218 Sekunden. Dieses Ergebnis kann auf die *CPU* Leistung des *Raspberry PIs* zurückgeführt werden, dessen Auslastung konstant bei 75% lag, wenn das System unter Last stand. Was bedeutet, dass *OpenFace* nur drei der vier verfügbaren Prozessorkerne beansprucht. Doch muss erwähnt werden, selbst wenn *OpenFace* den vierten Kern mit nutzen würde, muss bedacht werden, dass speziell für den Anwendungsfall zugeschnittene Hardware für den Echtzeit Aspekt des Systems unerlässlich ist.

Tadas Baltrusaitis, der Entwickler von *OpenFace*, stellt sein Verfahren zur "Gesichts-Landmark-Erkennung" in seinem Paper "Constrained Local Neural Fields for robust facial landmark detection in the wild" 2013 vor (Baltrusaitis & Morency, 2013). Dabei wird eine neuronales Netzwerk verwendet, um die Punkte zu bestimmen. Es wird mehrfach betont, dass die Robustheit bei der Erkennungsgenauigkeit wichtiger von ihm angesehen wird, als eine mögliche längere Rechenzeit.

Eine mögliche Lösung des Problems ließe sich dadurch erreichen, dass die *Raspberry PIs* durch schnellere bzw. besser für den Verwendungszweck

geeignete Hardware ersetzt werden könnten. GPUs oder FPGAs eignen sich beispielsweise besonders gut, Operationen mit einem neuronalen Netzwerk auszuführen.

Eine weitere Möglichkeit dieses Problem zu beheben wäre die Verlagerung des Konzepts weg von einer dezentralen Lösung von mehreren *Raspberry PIs*, die sich die Analyse der Webcams teilen, hin zu einer zentralen Verarbeitungseinheit. Dies könnte ein Server im *Smart Home* selbst sein, an welchen die Kamerabilder gestreamt werden und dieser könnte schneller, da mit besserer Hardware bestückt, die Blickdaten auswerten. Auch eine *Cloud* basierte Lösung ist denkbar, bei der *OpenFace* als Service angeboten wird. Beide Alternativvorschläge bergen allerdings unterschiedliche Probleme. Zum einen könnte der Datenschutzaspekt bei einer *Cloud*-basierten Lösung problematisch sein, da eine unbestimmte Anzahl an Kameras Bildmaterial auf einen externen Server streamen. Zum anderen ist eine zentrale serverbasierte Lösung mit der Anschaffung teurer Hardware verbunden, die mit einer unbestimmten Anzahl an Clients umgehen muss.

In dieser Arbeit wurden vier Kameras verwendet, um die Küchenzeile abzudecken, was allerdings je nach Art der Küchenform nicht ausreichend sein mag. Hier gilt zu berücksichtigen, dass bei Erhöhung der Clientanzahl, auch der Server, gemessen an der Leistung, mitskaliert werden sollte.

# 4 Studie

Im vierten Kapitel dieser Arbeit soll auf das Design der Studie näher eingegangen werden, die als Ziel die Evaluierung des im Punkt 3 beschrieben Systems hat. Dabei wird zuerst auf den methodischen Aufbau der Studie eingegangen und welche Absichten dabei verfolgt werden. Anschließend wird der Versuchsaufbau hinsichtlich der Eckpunkte, Aufgabe der Probanden, Ort des Experiments, Akquise von Probanden und Anforderungen an diese beschrieben. Im nächsten Punkt soll sowohl auf den Annotierungsmodus der Webanwendung näher eingegangen werden, als auch welche Auswirkung dieser auf die in Punkt 3.3.2.2 beschriebene "Blick Verarbeitungs Pipeline" des Servers hat. Abschließend soll auf die Vorgehensweise bei der Datenauswertung näher eingegangen werden. Im darauffolgenden Kapitel werden dann die Ergebnisse der Studie vorgestellt und in Punkt 6 diskutiert.

## 4.1 Grundlegende Anforderungen an die Studie

Die grundsätzliche Frage, die zum Aufbau der Studie gestellt werden muss, welches Ziel die Studie verfolgen soll. Dabei muss sich klar gemacht werden, dass es sich bei dem hier entwickelten System um einen ersten prototypischen Entwurf handelt.

Vorweg ist ein qualitativer Charakter der Studie auszuschließen. Wie bereits im Kapitel 2 "Stand der Technik" angesprochen, handelt es sich bei dem hier vorliegenden System, um ein nicht invasives System, mit dem der Nutzende nicht direkt kommuniziert. Auch ist es nicht Ziel der Arbeit, ein Assistenzsystem zu entwickeln, mit dem der Nutzende interagieren könnte. Das Ziel ist ein von Majaranta und Bulling beschriebenes *Attentive User Interface* zu schaffen, das aber für den Nutzenden nicht direkt sichtbar ist (Majaranta & Bulling, 2014). Ein Fragebogen beispielsweise zur qualitativen Bewertung des Systems ist daher nicht sinnvoll. Hierzu reiht sich ein weiterer Ausschluss an das Design mit ein. Der Gedanke, die Tester in der Küche kochen zu lassen, während parallel dazu das System läuft, wäre tatsächlich ein richtiger Schritt. Dieser wäre aber im aktuellen Entwicklungsstadium nicht möglich. Wie eben beschrieben, gibt es noch kein Assistenzsystem, das die Erkenntnisse des hier entwickelnden entwickelten Systems nutzen kann. Daraus ergibt sich, dass die

Festlegung einer Grundwahrheit, was sieht der Nutzende gerade tatsächlich ansehen, in einem realen Koch Szenario nicht möglich ist. Zudem ist noch nicht bekannt, welche Einschränkungen es gibt bzw. wie gut das System generell funktioniert.

Somit steht die Evaluierung des Systems hinsichtlich spezifischer Aspekte im Mittelpunkt, die aus dem Nichtwissen um das Verhalten des Systems resultieren. Diese sollen einerseits im Folgenden aufgezählt und begründet werden, andererseits die sich daraus ergebenden Entscheidungen für das Studien-Design stützen.

Aus der genannten Unwissenheit resultiert, dass eines der zentralen Ziele der Studie sein muss, wie das allgemeine Systemverhalten ist. Da bereits vorweg ein qualitativer Aufbau begründet ausgeschlossen wurde, ergibt sich daraus die quantitative Analyse der während der Studie erhobenen Daten. Dies ermöglicht, mithilfe der Datenauswertung mögliche Abweichungen von erwarteten Ergebnissen zu erkennen.

Der Kernaspekt des Systems ist es, zu erkennen, welche Objekte der Nutzende in der Küche gerade anblickt. Daher muss diese flächendeckend in Areas of Interest (AOIs) aufgeteilt werden. Dabei kann sich an den Objekten wie Küchenschränken, Herd oder Spüle orientiert werden. Hierbei sind die Unterschiede hinsichtlich der Häufigkeit bzw. Genauigkeit, zwischen bestimmten AOIs besonders interessant.

Ein weiterer wichtiger Punkt ist herauszufinden, wie genau das System funktioniert. Daher werden auf jeden der zuvor beschrieben AOIs sogenannte Cross Hair Points mittig angebracht. Wie in Punkt 3.3.2.2. besitzt jede AOI hierfür einen Sollwert, zu dem das System die euklidische Distanz berechnen kann. Daher ist es von zentraler Bedeutung, dass die Nutzenden während des Tests angewiesen werden, diese Cross Hair Points für eine gewisse Zeit zu fixieren, um über die Abweichung von diesem Sollwert eine Aussage über die Genauigkeit des Systems treffen zu können.

Ein weiterer wichtiger Aspekt ist, wie das System mit unterschiedlichen Testern zurecht kommt. Körperliche Merkmale der Personen hinsichtlich der Körpergröße, ob ein Bart getragen wird oder ob derjenige Brillenträger ist, sind wichtig, um mögliche Einschränkungen des Systems klassifizieren zu können.

Hierzu kommt noch der Faktor der Standpunkte der Nutzenden in der Küche. Dabei ist interessant, ob ein größerer oder geringerer Abstand von der

Küche Auswirkungen auf das System hat. Durch den Anstellwinkel der Kamera nach oben kann dies zusätzlich mit der Körpergröße in Relation gestellt werden. Auch kann neben dem Erkennen von Anomalien im Abstand, auch Unterschiede entlang der Küchenzeile erkannt werden. Hierbei kann die Frage im Raum stehen, ob AOIs vor einer Position besser erkannt werden, als jene, die weiter von einer AOI entfernt sind. Diese Fragen nach einer vertikalen (Abstand) und horizontalen (AOIs) Ausrichtung der Standpunkte in Küche soll die Studie beantworten.

Zuletzt ist es wichtig, die zuvor bereits angesprochene Grundwahrheit festlegen zu können. Dies bedeutet, dass während der Studie in Echtzeit annotiert werden kann, welche AOI, welcher Nutzer, zu welchem Zeitpunkt an welcher Position gerade ansehen sollte. Der Vorgang der Annotierung und wie dieser sich im Detail in dem Experimentaufbau eingegliedert, wird im Punkt 4.3 im Detail beschrieben.

## 4.2 Versuchsaufbau und Durchführung der Experimente

Im folgenden Abschnitt soll auf die Rahmenbedingungen und den Ablauf der Studienexperimente näher eingegangen werden. Hierbei wird auf die Festlegung der Areas of Interest, sowie auf die der Positionen, an denen die Tester während der Studie stehen, erklärt. Die Position der Kameras wurde bereits im Teil 3.3.1.1 näher erläutert und basierend auf den Erkenntnissen der Vorevaluierung begründet.

### 4.2.1 Ort des Experiments

Alle Experimente fanden verteilt auf drei Tage in der Küche in der Künstlergarderobe des Audimax der Universität Regensburg statt. Hierbei handelt es sich um eine reguläre Küchenzeile, die alle gängigen Merkmale (Herd, Mikrowelle, Spüle, Kühlschrank) einer Küche erfüllt. Die Küche ist insgesamt 408cm breit 220cm hoch, dabei ist der untere Teil der Küche 69cm und die obere Schrankwand 33cm tief. Nachfolgend findet sich ein Bild der Küchenzeile (Abbildung 16).

**Abbildung 16: Panoramabild der Versuchsküche**

### 4.2.2 Positionierung der Kameras

Wie bereits in der Vorevaluierung festgestellt, ist eine seitliche Positionierung der Kamera ungünstig, um die Blicke der Nutzenden zu erfassen. Daher wurde sich für einen frontalen Aufbau der Kameras entlang der Küchenzeile entschieden. Nutzer, die sich in der Küche befinden und auf diese blicken, werden somit frontal erkannt. Um die gesamte Küchenzeile mit Kameras abdecken zu können, die ein Sichtfeld von 60° besitzen, sind in der aktuell vorliegenden Küche vier Kameras notwendig.

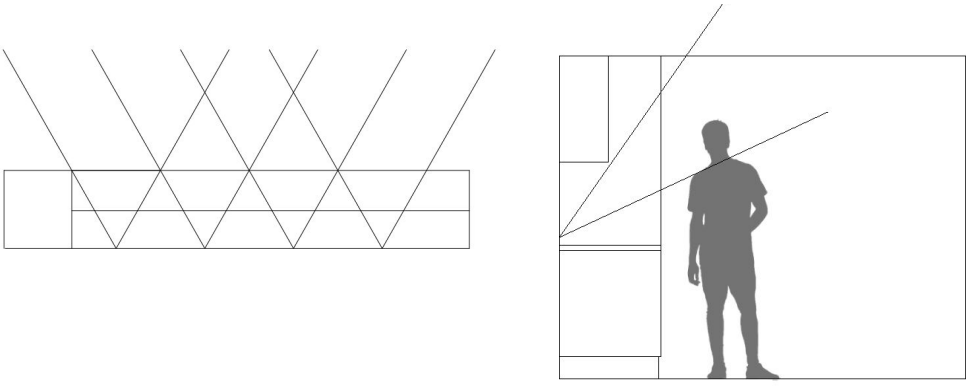

**Abbildung 17: Sichtfeld und Positionierung der Kameras; Topansicht (links); Seitenansicht (rechts)**

Abbildung 17 zeigt links eine technische Zeichnung der Küche von oben. Dabei befindet sich links der Kühlschrank und in offener Dreiecksform die jeweiligen Sichtfelder der einzelnen Kameras. Eine fünfte Kamera am rechten Rand ist nicht nötig. Dort befindet sich ein gemauerter Sockel, der es Nutzern der Küche

nicht ermöglicht, sich dort zu positionieren. Zudem sind die Kameras, wie in der vorletzten Abbildung 17 rechts zu sehen ist, mit einem 30°-Winkel nach oben und mit 4cm Abstand zur Arbeitsplatte ausgerichtet. Dies ermöglicht es, dass die Nutzer, abhängig von ihrer Körpergröße, in einem Bereich von 50-100 cm vor der Küche von den Kameras erfasst werden können. Die Abbildung 17 rechts soll dies verdeutlichen, indem sie die Seitenansicht der Küche zeigt. Dort ist zudem eine Person eingezeichnet, die bei einer Körpergröße von 175cm einen Abstand von 30 cm zur Küche hat.

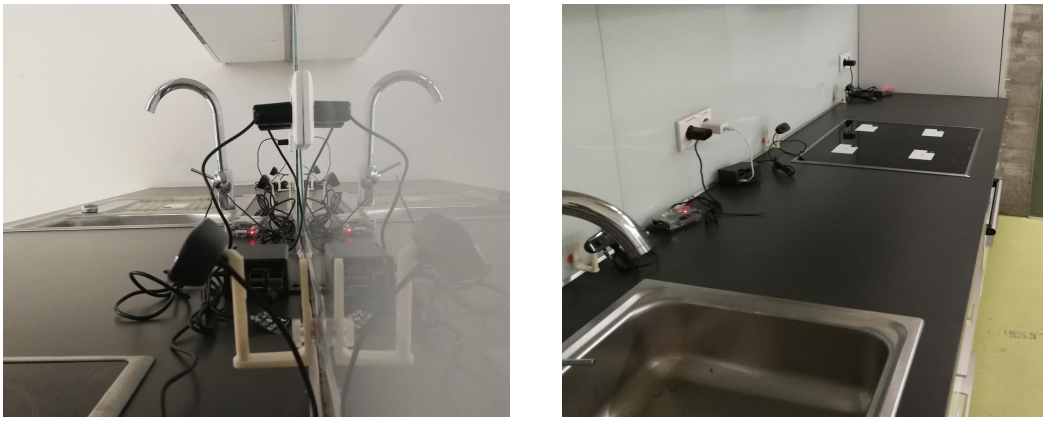

**Abbildung 18: Aufbau der Kameras in der Küche während der Studie**

Die Testküche hat den Vorteil, dass sie großzügig entlang der Küchenzeile mit Steckdosen ausgestattet ist. Dies wird besonders durch die vorangestellten Abbildung 18 deutlich. Pro Eye Tracking-Einheit wird davon eine genutzt, um den *Raspberry Pi* mit Strom zu versorgen. Ein wichtiger Hinweis hierbei ist die Verwendung von Netzteilen von mindesten 2.0 Ampere Stromstärke. Bei schwächeren Netzteilen kommt es bei längeren Hardware-Belastungen zu Ausfällen. Dies ist eine Erkenntnis, die ebenfalls aus der Vorstudie gewonnen werden konnte.

### 4.2.3 Definition der Areas of Interest

Wie in Punkt 4.1 gefordert, wurden die AOIs flächendeckend auf nahezu allen größeren Objekten der Küche angebracht. Dabei wurden sogenannte *Crosshair* Punkte auf 7 x 3,6 cm große weiße Briefetiketten gedruckt. Dabei befindet sich in deren Mitte das *Crosshair* (schwarzes Kreuz) und in der rechten unteren Ecke in schwarz die AOI Nummer. Nachfolgend findet sich eine technische

Zeichnung der Küche, mit den insgesamt 26 definierten AOIs und den dazugehörigen *Crosshair* Punkten.

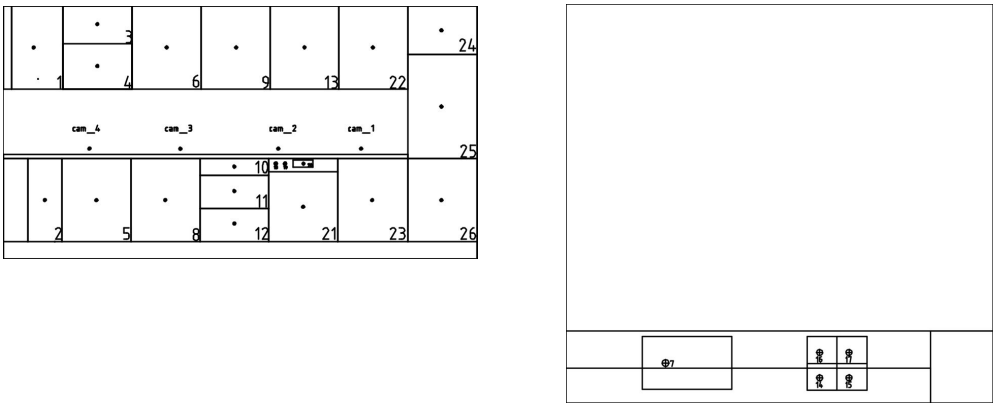


**Abbildung 19: Definition der AOIs; Frontansicht (rechts); Topansicht (links)**

Das bedeutet, dass jedes größere Schrankelement, aber auch Elemente wie Mikrowelle oder Herd, mit einer AOI versehen sind. Dem Herd mit seinen unterschiedlichen Elementen wurden mehrere AOIs zugewiesen. Neben dem Temperaturregler, um das Verhalten des Systems auf möglichst kleine Elemente zu testen, wird auch auf jedem Ceran-Kochfeld des Herdes eine AOI definiert. Einer der unteren Küchenschränke wurde gesondert in drei Teile eingeteilt. Zudem ist in der Spüle eine weitere AOI definiert. Hierbei sind dem Nutzenden die genauen Abmessungen der AOI während des Versuchs nicht bekannt. Um später aber die Abweichung von dem jeweiligen Crosshair Punkt bestimmen zu können, muss der Nutzende diesen fokussieren. Desweiteren ist es während des Versuchsablauf praktikabler, dem Nutzenden anzuweisen, einen Punkt zu fokussieren, als ihn auf einen beliebigen Punkt einer Fläche blicken zu lassen.

### 4.2.4 Definition der Positionen

Der Forderung nach einer vertikalen bzw. horizontalen Ausrichtung entlang der Küchenzeile wird mit fünf unterschiedlichen Positionen entsprochen. Dabei liegen drei der Positionen im Abstand von 30 cm hintereinander vor einer der Kameras (Position 2,4,3). Die Positionen 3, 1 und 5 haben den gleichen Abstand zur Küche (xcm), liegen aber entlang deren X-Achse verteilt vor den Kameras 1, 2 und 3. Ähnlich wie die Crosshair Punkte im vorherigen Punkt, wurden die Positionen in Form eines schwarzen Kreises mit der Positionsnummer auf Briefetiketten gedruckt und anschließend auf dem Boden angebracht.

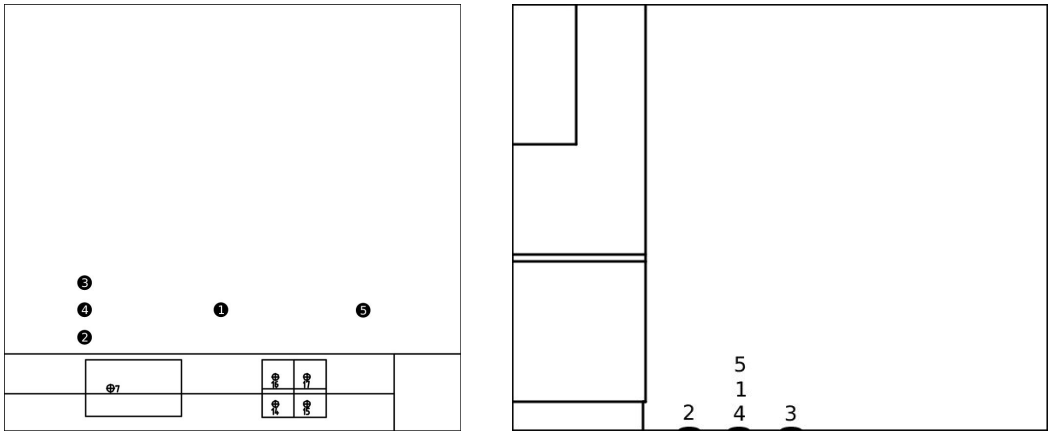


**Abbildung 20: Definition der Positionen; Topansicht (rechts); Seitenansicht (links)**

### 4.2.5 Ablauf des Experiments

Jeder Testvorgang mit den Probanden lief immer gleich ab. Diesen wurde zuerst erklärt, dass dieses Experiment im Rahmen einer Masterarbeit des Lehrstuhls für Informationswissenschaft an der Universität Regensburg stattfindet. Dabei wurden sie darauf hingewiesen, dass die Dauer der Versuche 15 Minuten sei und dass sie während dieser Zeit von Kameras gefilmt werden, und die dabei erhobenen Daten gespeichert werden. Alle Probanden mussten nach einer kurzen Erklärung zum Ablauf des Experiments eine Einverständniserklärung unterzeichnen, deren genauer Wortlaut im Anhang eingesehen werden kann. Hierauf wurden zusätzliche weitere personenbezogene Daten wie Körpergröße (vor Ort war die Möglichkeit einer Messung vorhanden), das Tragen einer Brille oder von Gesichtsbehaarung erfasst.

**Abbildung 21: Versuchsaufbau; Testperson steht vor der Küchenzeile; Testleiter sitzt an Tisch vor Laptop hinter Testperson**

Anschließend wurden die Personen durch den Testleiter angewiesen, sich auf eine der Positionen in der Küche zu stellen.

*"Bitte stellen Sie sich auf Position X und blicke auf Feld X"*

Die Situation während des Experiments ist in Abbildung 21 zu finden. Dabei befindet sich der Testleiter an einem Laptop sitzend hinter der Versuchsperson. Auf dem Laptop selbst befindet sich der Annotierungsmodus der Webanwendung (3.3.3), der im Detail im nachfolgenden Punkt 4.3 beschrieben wird. Dieser gibt die aktuelle Position und die anzusehende AOI für den Nutzenden an. Nach fünf Sekunden wird der Nutzende angewiesen, auf die nächste AOI zu blicken.

*"Bitte blicke nun auf Feld X"*

Dies geschieht für alle 26 AOIs. Sind diese alle nacheinander angeblickt worden, wird die Versuchsperson wieder durch den Testleiter gebeten, sich auf die nächste Position zu stellen. Dies geschieht für alle fünf Positionen. Nach Ende des Tests wird sich beim Tester Probanden statt Tester? für seine Teilnahme bedankt und ihm ein kleines Dankeschön übergeben.

## 4.3 Annotierung

Im Laufe dieser Arbeit wurde der sogenannte Annotierungsmodus bereits mehrfach angesprochen. In diesem Abschnitt soll nun auf seine genaue Funktion während der Evaluierung des Systems eingegangen werden. Grundsätzlich muss für die spätere Auswertung ein Sollzustand definierbar sein, um diesen in Relation mit dem Istzustand zu setzen, der den aufgezeichneten Daten entspricht. Dieser muss sich daher aus dem Wissen über den aktuellen Tester, dessen Position und die anzusehende AOI zusammensetzen. Diese Anforderungen wurden mit Hilfe des Annotierung Modus der Webanwendung umgesetzt. Ein Screenshot dieser Anwendung findet sich in der nachgestellten Abbildung.

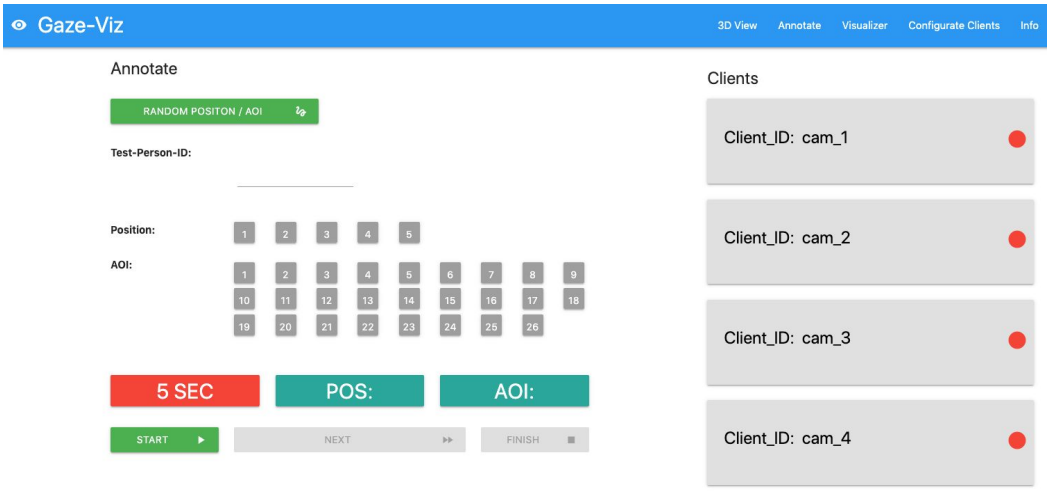


**Abbildung 22: Annotierungs Modus der Webanwendung**

Der erste obere *Button* "*Random Position / AOI*" der Seite generiert eine zufällige Reihenfolge der Positionen und der AOIs. Dies soll vermeiden, dass zufällige Fehler einzelne Positionen oder AOIs beeinflussen.

Wie bereits beschrieben, werden zu Beginn jedes Tests die persönlichen Daten der Nutzenden erfasst. Dabei wird eine chronologisch aufsteigende Probandennummer vergeben. Diese kann in dem ersten Feld "*Test-Person-ID*" der Webanwendung eingetragen werden. Der Testleiter kann nun auf den unteren rechten "*Start*" *Button* klicken, gleichzeitig sagt er die erste Position bzw. AOI laut an. Nach fünf Sekunden, als Hilfe läuft hierzu eine Zeitanzeige

ab, kann der Testleiter auf die nächste AOI schalten und diese dem Probanden ansagen. Sind alle 26 AOIs einmal durchlaufen, springt das System auf die nächste Position und die erste angezeigte AOI, welche wieder durch den Testleiter angesagt wird. Nach der letzten Position bzw. AOI beendet sich der Annotierung Modus automatisch und springt wieder auf seinen Ausgangszustand zurück.

Zusätzlich ist auch auf der Annotierung Seite der Webanwendung die Client-Status-Komponente eingebaut. Dies hat den Hintergrund, dass ein möglicher Ausfall eines Clients während des Testens über die farbliche Anzeige schnell bemerkbar wäre.

Wie bereits in Punkt 3.3.3.1 aufgezeigt, spricht dieser Teil der Webanwendung über eine *REST* Schnittstelle mit dem Server. Dabei wird diesem beim Klicken des "*Start*" *Buttons* mitgeteilt, dass jetzt eine Annotierung beginnt und dieser neu eintreffende Daten parallel zum regulären Log abspeichern soll. Das Drücken des "*End*" *Buttons* beenedet diese zusätzliche Aufzeichnung wieder. Sowohl beim Befehl "*Start*", als auch bei dem des "*Next*" wird immer die aktuelle Nutzernummer, Position und AOI im *JSON* Format an den Server weitergegeben. Dieser reichert mit diesen Informationen jeden neuen Eintrag in der Annotierungs-*CSV*-Datei zusätzlich an.

## 4.4 Datenauswertung

Als Grundlage für die Auswertung wurden die durch den Annotierungs Modus erzeugten *CSV* Dateien genutzt. Der gesamte Datensatz besteht noch aus deutlich mehr Daten. Eine Aufschlüsselung bzw. Verlinkung zum Download findet sich im Anhang. Dies ist einerseits die von *OpenFace* aufgezeichnete Log Datei bzw. die einzelnen Bild-Frames, die durch *OpenFace* ausgewertet wurden.

Die gesamte Auswertung fand unter der Verwendung von *Jupyter Notebook* und *Python* statt. Die Wahl fiel auf *Python*, da bereits die Transformation und Schnittstellen-Logik des Servers damit umgesetzt wurden und daher leicht wieder verwendet werden konnten. *Jupyter Notebook* vereinfacht die Auswertung der Daten stark durch den Umstand, dass einzelne Quellcodes in Zellen separiert werden können. So kann sehr einfach mit unterschiedlichen Datentransformationen oder Visualisierungen experimentiert werden, um passende Lösungen für Probleme und Fragestellungen der Auswertung zu finden.

Die Datenauswertung gliedert sich daher in insgesamt vier Notebooks, die unterschiedliche Aufgaben bei der Auswertung erfüllen. Das hier als erstes zu nennende ist das "*0_Vorverarbeitung_der_Daten.ipynb*" Notebook, in dem die aufgezeichneten Annotierungs-CSV-Dateien vorverarbeitet werden. Hier werden alle Einträge entfernt, in denen keine Blick erkannt wurden oder deren Erkennungsgenauigkeit unter 90% liegt. Diese werden dann wiederum einzeln getrennt nach Nutzenden im Ordner *1_Daten/2_processed* abgelegt. Dies ermöglicht eine schnelle Weiterverarbeitung in den folgenden Notebooks.

Das zweite Notebook "*1_Verteilung_der_Daten.ipynb*" zeigt die allgemeine Verteilung der Daten. Wie viele Daten sind pro Position, Kamera und AOI vorhanden? Wie verhält es sich mit die Trefferquoten abhängig von dem jeweiligen Probanden? Oder wie ist das Verhältnis von "Erkannten" zu "Nicht-Erkannten" Frames?

Im dritten Notebook "*2_Bestimmung_der_Transformationsmatrizen. ipynb*" werden die Transformationsmatrizen für die Umwandlung der Blick-Koordinaten festgelegt. Dieses Notebook soll sicherstellen, dass die verwendeten Transformationswerte richtig gesetzt wurden. Dabei zeigt es eine erste Visualisierung aller Daten in der Küche.

Im letzten Notebook "*3_AOI_Auswertung.ipynb*" wird betrachtet, welche Blicke wirklich eine AOI schneiden. Dabei wird dies abhängig von Position und Kamera ausgewertet.

## 4.5 Einschränkungen

Bevor im nächsten Kapitel die Ergebnisse präsentiert und anschließend im Kapitel 6 diskutiert werden, müssen die Einschränkungen dieser Studie vorab besprochen werden. Diese Einschränkungen resultieren aus der Auswertung der Daten und sollen dabei erklären, wie und warum gewisse Aspekte nachgelagert betrachtet werden müssen.

Eine Problematik, die erst bei der Auswertung der Daten auffiel, ist die Latenz des Systems. In Punkt 3.3.4.1 Flaschenhalsanalyse wurde diese dargelegt und begründet. Das Problem, das nun damit während der Annotierung einhergeht, gestaltet sich so, dass der Zeitpunkt der Annotierung (Testleiter schaltet auf nächste AOI) und der Auswertung der Daten durch *OpenFace* nicht nachträglich synchron zusammengeführt werden können. Das bedeutet, dass der aktuelle Frame meist eine bis fünf Sekunden dem eigentlichen realen

Geschehen hinterherhinkt. Das hat zur Folge, dass nachgelagert nicht mehr festgestellt werden kann, was der eigentlich Sollwert, also die durch den Tester anzublickende AOI ist. Punkte wie 5.5 und 6.6 können zeigen, dass dieses Problem bei groben Veränderungen wie einem Positionswechsel nicht ins Gewicht fallen. Dies gilt jedoch nicht bei den zeitlich sehr nahe beieinander liegenden AOIs, die sich mit einem Abstand von 5 Sekunden ändern. Dies hat zur Folge, dass keine wirkliche Aussage über die Genauigkeit des System getroffen werden kann. Eine mögliche Lösung, welche mittlerweile auch in den Annotierungs Modus implementiert ist, wäre den Zeitpunkt des Wechsels auf eine neue AOI in den schon protokollierten Metadaten des Testlaufs abzuspeichern.

# 5 Ergebnisse

In diesem Teil der Arbeit werden die Ergebnisse der Studie vorgestellt, im darauf folgenden Kapitel 6 sollen für diese erklärt und interpretiert werden. Dieses Kapitel orientiert sich bei seinem Aufbau an den grundsätzlichen Anforderungen, die in Punkt 4.1 aufgestellt wurden. Dabei soll auf Trefferquoten und Erkennungsgenauigkeiten abhängig von der jeweiligen AOI, Position und der Nutzenden eingegangen werden. Wenn im weiteren Verlauf von Trefferquote gesprochen wird, ist die Zahl an erkannten Blicken gemeint.

## 5.1 Stichprobenbeschreibung

An der Studie nahmen insgesamt 49 Personen teil. Hierbei liegt der Median für das Alter der Probanden und Probandinnen bei 26 Jahren (Q1/Q3 = 22/29, Range = 20-38 Jahre). Insgesamt gab es 7 Bartträger und 19 Brillenträger. Die Verteilung zwischen den Geschlechtern tendiert mit 19 Frauen und 30 Männern zu einem Überschuss an männlich Probanden.

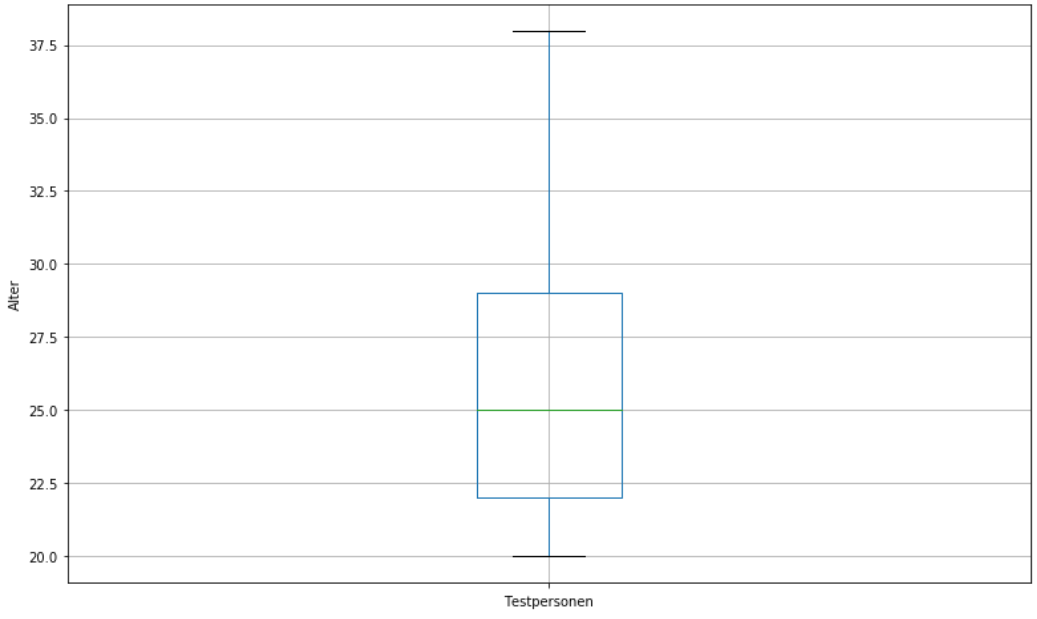


**Abbildung 23: Altersverteilung der Probanden**

Die vorangegangene Abbildung zeigt eine sehr geringe Streuung des Alters. Dies lässt sich auf die sogenannten *Snowball-Sampling*-Methode zurückführen

(Biernacki & Waldorf, 1981). Der Verfasser dieser Arbeit hat bei der Erhebung seiner Testpersonen primär auf Lehrstuhlmitarbeiter, Freunde und Bekannte zurückgegriffen, die zum Teil noch Studierende sind oder gerade am Anfang ihres beruflichen Werdegangs stehen. Daher ist der größte Teil der Nutzenden zwischen 21 und 29 Jahre alt.

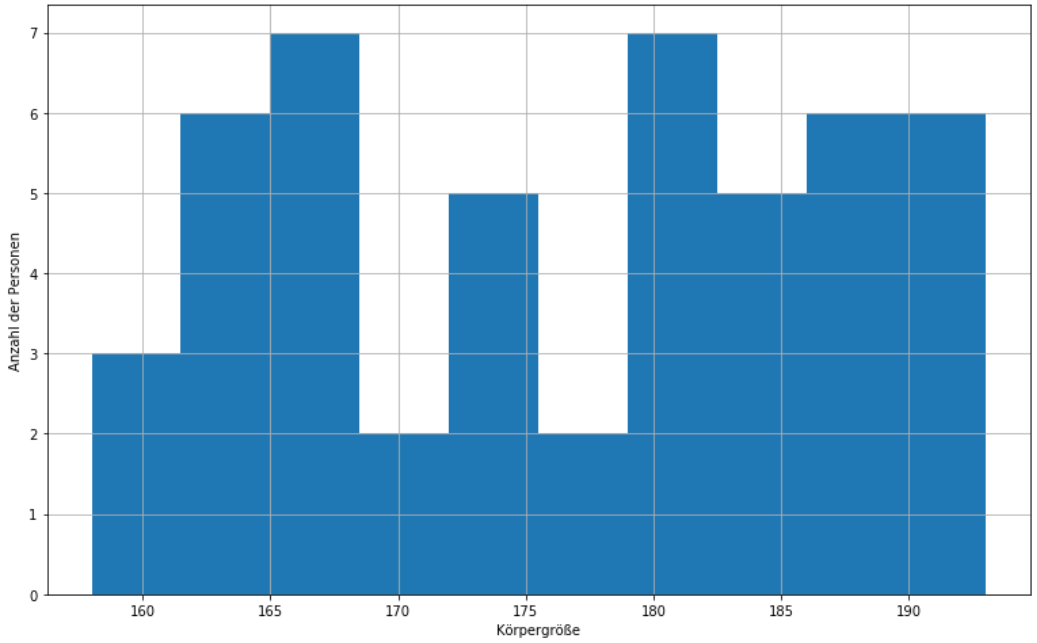


**Abbildung 24: Größenverteilung der Probanden**

Die Abbildung zeigt mit Hilfe eines Histogramms (Bin Größe 10) die Größenverteilung der 49 Versuchspersonen. Dabei beträgt die Körpergröße des kleinsten Nutzenden 158cm und die des größten 193cm. Der Median liegt insgesamt bei einer Körpergröße von 176cm (Deutscher Durschnitt 172cm). Männliche Teilnehmer sind hierbei im Durchschnitt 183cm (Deutscher Durschnitt 179cm), weibliche Teilnehmer hingegen 166cm (Deutscher Durschnitt 165cm) groß (Stephen Matthews, 2016).

## 5.2 Allgemeine Verteilung der Daten

In diesem ersten Punkt wird der Verteilung der gesammelten Daten hinsichtlich den Faktoren “Frames ohne Gesicht” und “Frames mit Gesicht” und die erkannten Treffer über alle Probanden hinweg betrachtet.

Insgesamt wurden während aller Tests 712.216 einzelne Frames aufgezeichnet. Von diesen enthalten 16.654 (2,34%) ein durch *OpenFace* erkanntes Gesicht. Für die hier vorliegende Auswertung wurden aber nur Blicke verwendet, bei denen ein *Confidence* Wert von mehr oder gleich 90% vorlag. In Abbildung 25 sind zwei Fälle zu sehen, in denen von *OpenFace* ein Gesicht zwar erkannt wird, aber keine verlässlichen *Confidence* Werte vorliegen. Die Zuordnung der Gesichts Landmarken weicht deutlich von der tatsächlichen Position des Gesichts ab.

**Abbildung 25: Fehlpositionierung der Landmarken durch OpenFace**

Daraus resultiert eine weitere Reduktion des Datensatzes, sodass für die Auswertung 11.120 (1,56%) Blick übrig bleiben. Betrachtet man anschließend die Blicke, die tatsächliche eine AOI schneiden, so sind dies noch 5092 (0,32%) Frames. Die nachfolgende Grafik zeigt deutlich dieses ungleiche Verhältnis von "Frames mit Blick" zu "Frames ohne Gesicht" pro Versuchsperson.

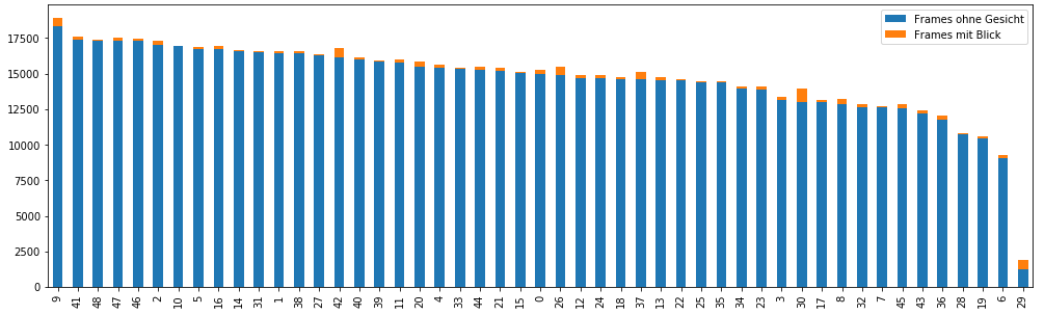


**Abbildung 26: Frames mit Bick (orange) versus Frames ohne Gesicht (blau)**

Im weiteren Verlauf dieser Ergebnispräsentation werden trotzdem alle verwertbaren Blickpunkte mit einem *Confidence* Wert über 90% herangezogen,

um Aussagen über die Trefferquoten machen zu können. Bei Punkten die im besonderen das Thema der AOIs betreffen wird dann, unter Berücksichtigung der Limitationen aus 4.5, gesondert auf die 5092 AOI schneidenden Punkte eingegangen.

## 5.3 Verteilung der Trefferquoten pro Position

Im folgenden Abschnitt werden die Trefferquoten pro Position explizit betrachtet. Diese fallen je nach Position unterschiedlich aus. Position 2 hat hierbei mit 1328 erkannten Blicken die geringste, Position 5 auf den ersten Blick mit 2165 Blicken die höchste Erkennungsrate.

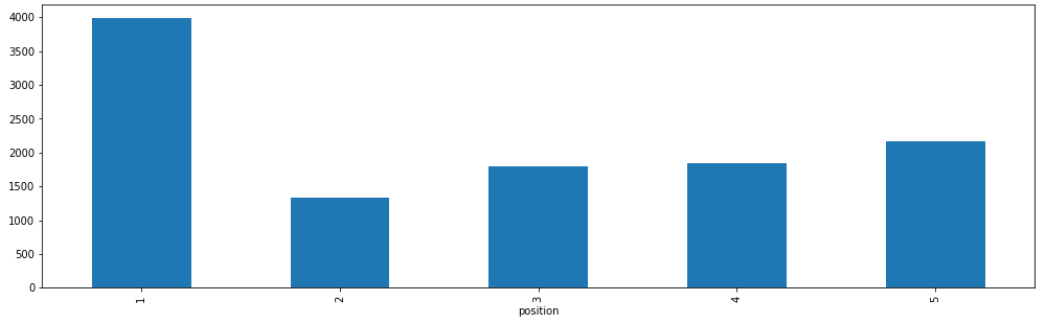


**Abbildung 27: Anzahl der Treffer pro Position**

Gesondert muss hierbei Position 1 betrachtet werden, diese wird von Kamera 2 und 3 gleichzeitig erfasst. Die nachfolgende Abbildung zeigt deutlich, dass durch Kamera 2 an Position 1 deutlich mehr Blicke (2513) erkannt wurden, als durch Kamera 3 (1564), obwohl die Position 1 gleichweit von beiden Kameras während der Versuche entfernt war.

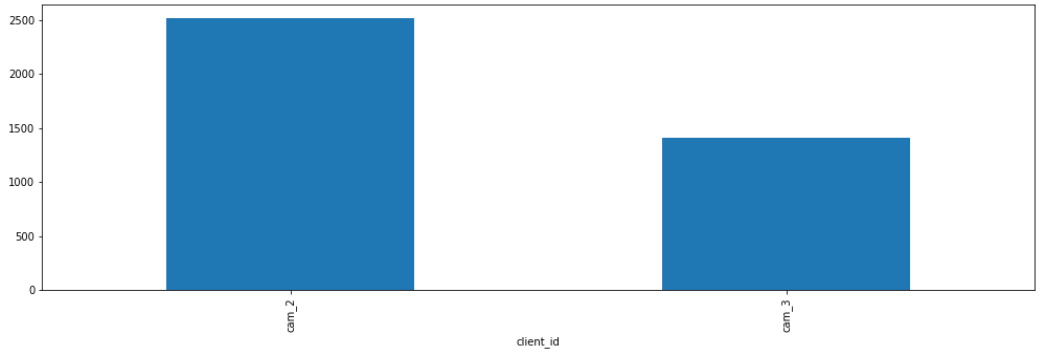


**Abbildung 28: Anzahl der Treffer für Position 1 pro Kamera**

## 5.4 Verteilung der Trefferquoten pro Testperson

Die Trefferquote fällt je nach Testperson extrem unterschiedlich aus. Dies gilt sowohl für die Erkennung des Gesichts, als auch für die Blicke der Probanden, die tatsächlich eine der AOIs schneiden. Mit 931 erkannten Blicken schneidet Proband 14 beispielsweise am besten, Proband 39 mit nur 37 erkannten Blicke am schlechtesten ab. Der Median liegt hier auf alle 49 Probanden bezogen bei 227 erkannten Frames mit Gesicht. Abbildung 29 zeigt diese deutlichen Schwankungen und liefert somit die Begründung für eine nähere Betrachtung einzelner Tester abhängig von deren körperlichen Merkmalen.

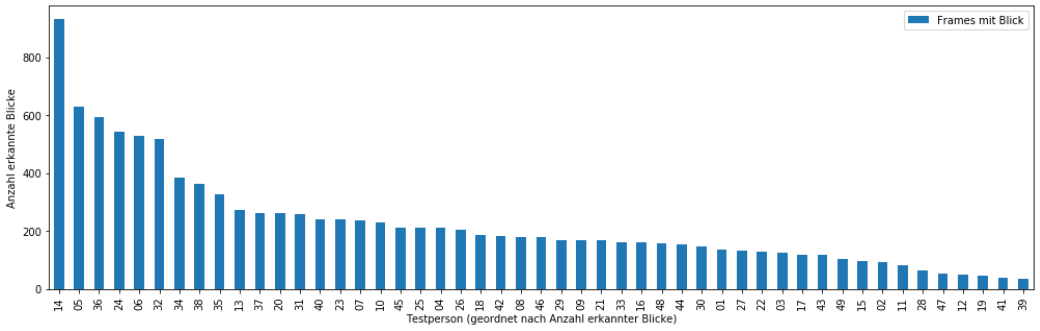


**Abbildung 29: Trefferquoten pro Testperson**

Zunächst wird die Körpergröße der einzelner Tester als zu untersuchendes Merkmal betrachtet. Dabei zeigt die nachfolgende Abbildung die einzelnen Trefferquoten der Nutzenden gruppiert in 5cm Intervalle. Hierbei zeigt sich, dass die Intervalle (165, 170], (175, 180] und (180,185] mit einem Median von 276 über dem zuvor benannten Median über alle Testprobanden liegen. Alle andere Intervalle sind ebenso hoch wie dieser Gesamt Median oder deutlich kleiner.

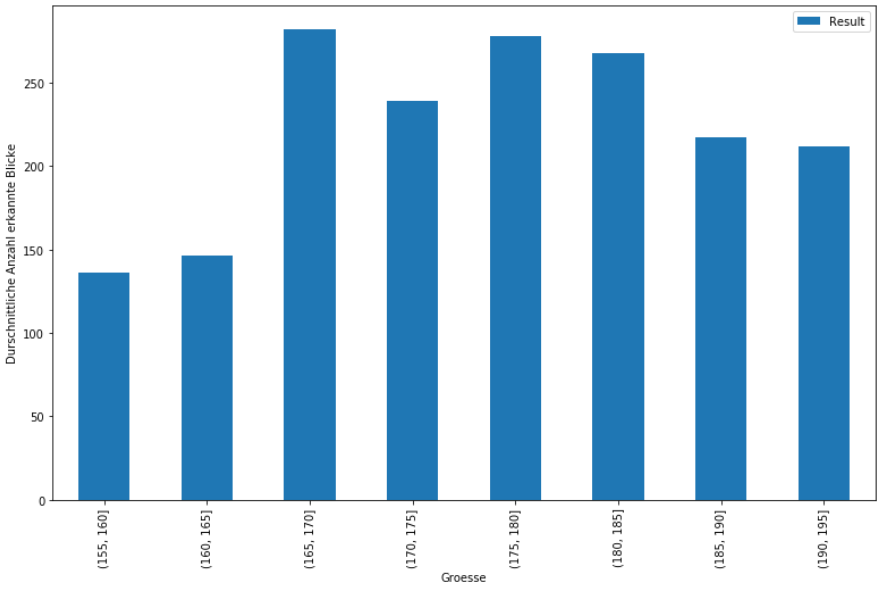


**Abbildung 30: Durchschnittliche Trefferquoten abhängig von der Körpergröße (Intervallgröße: 5cm)**

Zudem wurde untersucht, ob sich eine Korrelation zwischen Trefferquote und Körpergröße feststellen lässt. Hierbei wurde mit Hilfe des Pearson-Korrelationskoeffizienten[3] versucht den linearen Zusammenhang zwischen diesen beiden Variablen festzustellen. *r* liegt hierbei auf die gesamten Datensatz (s. Abbildung links) betrachtet bei 0,021. Somit liegt hier eine sehr schwache Korrelation vor. Unterteilt man nun den Datensatz in Körpergrößen über und unter 180cm, so tritt ebenfalls eine schwache Korrelation mit $r$=0,03 für Werte über 180cm. Für Werte darunter tritt eine mittlere Korrelation von $r$=0,12 auf.

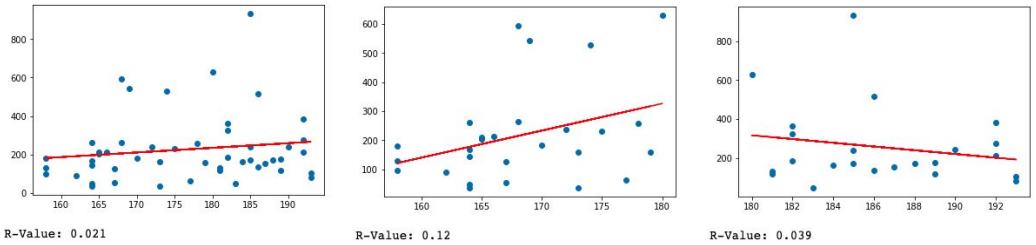


**Abbildung 31: Regressionsgeraden für alle Körpergrößen (links), für kleiner 180cm (mitte) und größer 180cm (rechts)**

[3] Die Korrelation ist eine Möglichkeit, den Zusammenhang zwischen zwei Variablen zu beschreiben (Brückler, 2017)

Auch wurde untersucht, ob es Unterschiede bei der Trefferquote abhängig von der Größe der Versuchsperson und deren Abstand zur Kamera gibt. Hierfür wurden gesondert das kleinste bzw. das größte Körpergrößen-Intervall betrachtet und die Position 2 als näheste und Position 3 als am weitesten Entfernung zu Kamera 4 herangezogen. Tabelle 1 präsentiert hierfür die durschnittlichen Trefferquoten und die prozentuale Differenz zwischen den beiden Körpergrößen-Intervallen.

| **Position** | **(155, 165]** | **(185, 195]** | **Differenz (%)** |
|---|---|---|---|
| Position 2 | 224 | 594 | 37% |
| Position 3 | 79 | 1018 | 7,76% |

**Tabelle 1: Median Trefferquoten für Position 2 und 4 für die Körpergrößen-Intervalle (155, 165] und (185, 195]**

Wie bereits in Punkt 5.3 gezeigt, fällt auch hier die durchschnittliche Trefferquote für Position 2 schlechter aus. Doch schneiden Personen im Intervall (155, 165] mit nur 79 Treffern an der am weitesten entfernten Position 3, um ein vielfaches schlechter ab, als Probanden im größten Körpergrößen-Intervall.

Des Weiteren wurde die Auswirkungen auf die Trefferquote durch das Tragen einer Brille bzw. das Tragen eines Bartes, bei männliche Probanden, betrachtet. Hierbei schneiden Brillenträger deutlich schlechter ab (Median: 169), als Träger ohne Brille (Median: 264). Auch liegt die Trefferquote bei Trägern eines Bartes (Median: 124) nur halb so hoch, wie unter Nicht-Bartträgern (Median: 244). Dieser Effekt lässt sich auch bestätigen, wenn man Versuchspersonen mit Brille und Bart betrachtet. Hier liegt die Trefferquote für Tester mit Bart und Brille bei einem Median von 118, bei nur Brillenträgern ohne Bart bei 192. Unter den Versuchspersonen waren nur die männlichen Teilnehmer mit einem Bart vertreten.

## 5.5 Verteilung der Trefferquoten pro Area of Interest

Zunächst muss der bereits am Beginn dieses Kapitels beschriebene Umstand betrachtet werden, dass von den insgesamt 11120 erkannten Blicken 2602 tatsächlich eine AOI schneiden. Dies liegt zuerst an der im Punkt 4.5 beschriebenen Limitation nicht der mehr eindeutigen Zuordenbarkeit der Blickpunkte zu der annotierten AOI.

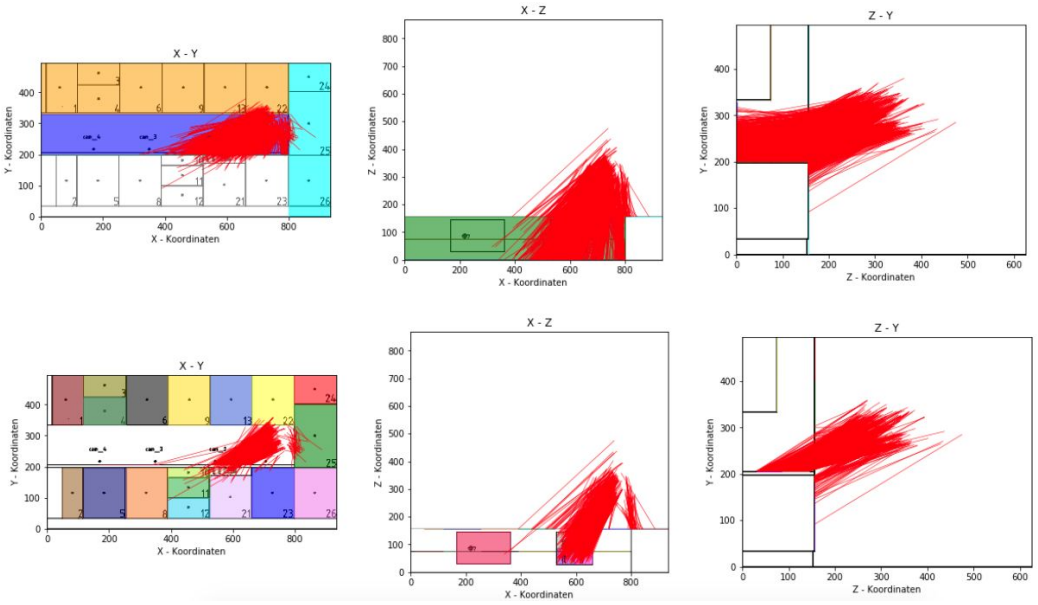


**Abbildung 32: Visualisierung der Blickdaten; Oben alle Blickdaten; Blickdaten die AOI schneiden)**

Desweiteren trifft der Großteil der Blicke die hintere Wand bzw. die nicht mit AOIs versehenen Stellen der Arbeitsfläche. Die vorangestellte Visualisierung der Blickdaten (Oben für alle Flächen, Unten nur die 26 definierten AOIs) verdeutlicht dies exemplarisch für Position 5. Auch die folgende Abbildung 33 zeigt die eben beschrieben Verteilung der Daten auf die Arbeitsfläche und die Wand, an der die Kameras befestigt sind, deutlich.

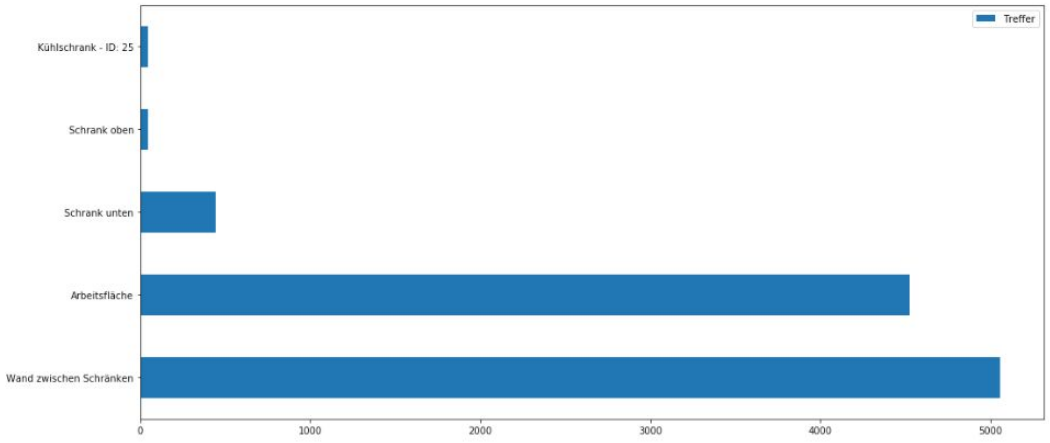


**Abbildung 33: Verteilung aller erkannten Blickdaten auf große Küchenflächen**

Betrachtet man die Verteilung der Blicke, die eine der 26 AOIs schneiden, so kristallisieren sich einzelne AOIs mit einer hohen Trefferquote heraus. Abbildung 34 stellt diese einzelnen geschnitten AOIs mit deren Trefferquoten dar.

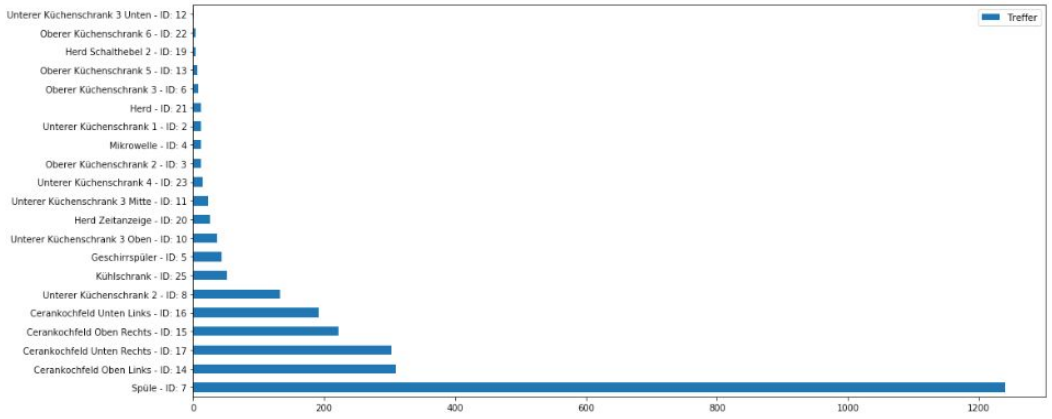


**Abbildung 34: Verteilung aller erkannten Blickdaten auf große Küchenflächen**

Hierbei zeichnet sich ein Bild, ähnliches wie in Abbildung 33 bereits angedeutet, ab. Der Großteil der Blicke (1278) schneidet die Spüle der Küche. Die fünf weiteren AOIs mit der höchsten Trefferquote liegen ebenfalls auf der Arbeitsplatte. Hierbei handelt es sich um die Cerankochfelder des Herdes.

## 5.6 Verteilung der Trefferquoten pro Position pro Area of Interest

In diesem Punkt soll das Verhältnis von Position zu getroffener Area of Interest eingegangen werden. Aus diesem Grund werden hierbei nur Blickpunkte benutzt, die tatsächlich eine AOI schneiden.

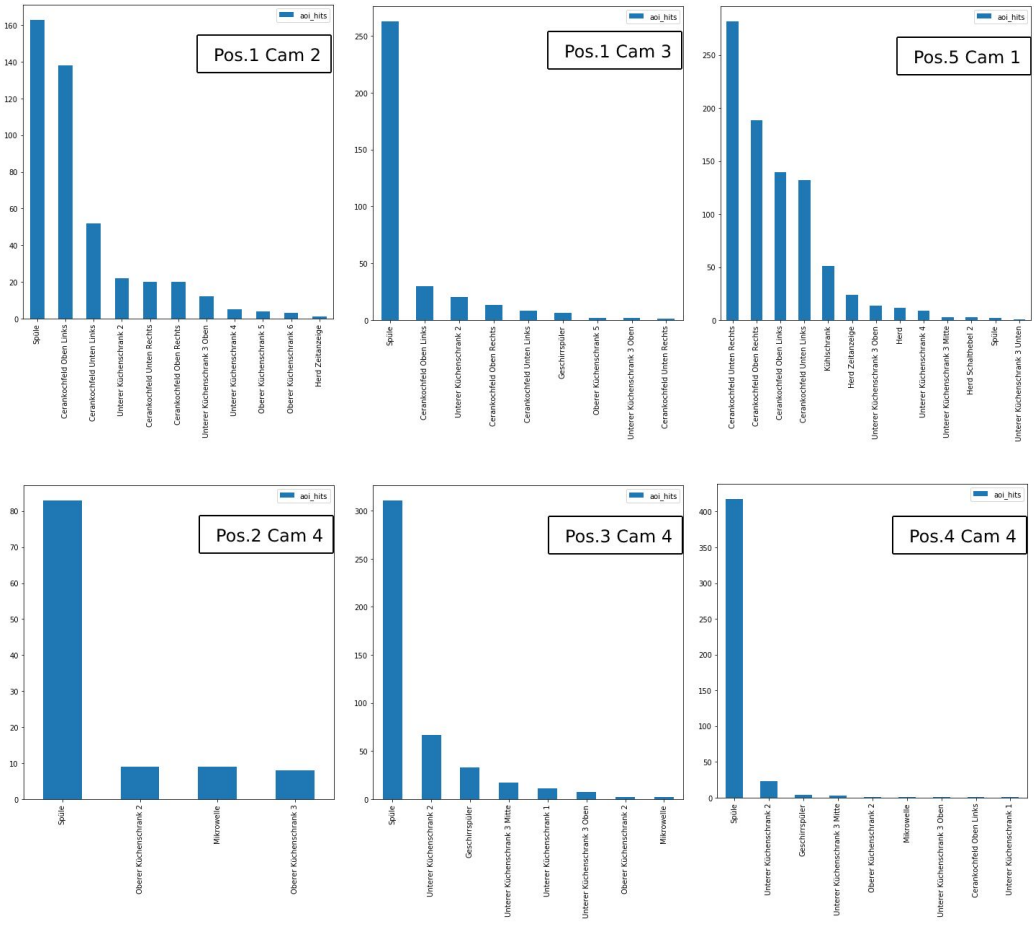


**Abbildung 35: AOIs abhängig von Position und Kamera**

Die Abbildung 35 zeigt die Trefferquoten der einzelnen AOIs, abhängig von der jeweiligen Kamera und Position (Position 1 wird von Kamera 2 und Kamera 3 erfasst). Hierbei zeigt sich wie bereits in Punkt 5.5 angesprochen, dass AOIs auf der Arbeitsfläche wesentlich häufiger getroffen werden, als die oberen oder unteren Schränke der Küche. Es zeichnet sich für Position 1 Kamera 3, Position 2 Kamera 4, Position 3 Kamera 4 und Position 4 Kamera 4 deutlich ab, dass die Spüle mit einem Median von 308 Treffern am häufigsten erkannt wird. Position 5 Kamera 1 zeigt hierbei eine breite Streuung auf die vier Cerankochfelder,

wobei die beiden rechten Felder mit einem Median von 225 doch häufiger erkannt werden, als die Felder links (Median: 132). An Position 1 Kamera 2 werden primär das linke Cerankochfeld (140 Treffer) und die Spühle (165 Treffer) erkannt.

# 6 Diskussion

In diesem Teil der Arbeit soll nun eine Interpretation der Ergebnisse erfolgen. Diese werden mit den Erwartungen an das System in Relation gesetzt und Erklärung hierfür mit Wissen über das System und der Literatur hergeleitet. Zudem soll klar dargelegt werden, wo sowohl Probleme, als auch Limitationen der Studie und des Systems selbst liegen und in welcher Weise diese zukünftig überwunden werden können.

## 6.1 Überschuss an nicht verwertbaren Daten

Der erste Punkt der Ergebnispräsentation 5.2 zeigt, dass schlussendlich nur 1,56% der gesendeten Daten überhaupt ausgewertet werden konnten. Dies liegt insgesamt an zwei unterschiedlichen System und Studien Gegebenheiten. Zum einen steht der Proband nur vor einer Kamera, mit Ausnahme von Position 1, die restlichen Clients erzeugen trotzdem in dieser Zeit weiter Datenpunkte, obwohl kein Gesicht erkannt wurde. Zum anderen kommt der Umstand hinzu, dass solange kein Gesicht, nach Start der Client Software, vor der Kamera war, eine Auswertungs Rate von 24 Frames pro Sekunde vorliegt. Dies hat zur Folge, dass in der Zeitspanne, bis ein Gesicht vor der Kamera ist und die Auswertungsrate auf 1-2 Frames je 5 Sekunde fällt, eine Großzahl der Frames ohne Gesicht erzeugt werden.

Für eine Weiterentwicklung des Systems wäre es daher sinnvoll diese nicht verwertbaren Daten bereits auf Client Seite heraus zu filtern und nicht an den Server weiterzuleiten, um diesen somit zu entlasten. Dieses Ergebnis stützt die Designentscheidung einer dezentralen Datenerzeugung, gegenüber einer im Punkt 3.4.1 diskutierten zentralen Lösung, in der eine derartige Vorfilterung nicht möglich wäre.

## 6.2 Unterschiede zwischen den Positionen

Die in 5.3 aufgezeigte Verteilung der erkannten Gesichter pro Position legt nahe, dass die Position der Probanden in der Küche einen Einfluss auf die Erkennungsleistung des Systems hat.

Position 2, die sich sehr nahe an der Küche befindet schneidet hierbei deutlich schlechter ab, als andere Positionen. Dies liegt zum Teil am Aufbau der

Studie. Probanden müssen sich, für weit von Position 2 entfernte AOIs wie 25 oder 26, unnatürlich nach hinten lehnen oder strecken, um diese zu sehen. Da der Winkel ihrer Position und der Position dieser AOIs, bedingt durch die Länge der Küchen, kleiner ausfällt, als an anderen Positionen, die weiter von der Küche entfernt sind. Das daraus resultierende Wegdrehen von der Kamera ermöglicht nur eine bedingte Erkennung der Probanden.

Des Weiteren fällt auf, dass Position 5 und Position 1 (Kamera 2), mit einem Median von 2339, eine höhere Erkennungsrate vorweisen können. Ein Erklärungsversuch hierfür können die Lichtverhältnisse in der Küche sein. Sowohl Bulling und Khamis, als auch Tadas Baltrušaitis beschreiben den Einfluss von Licht als entscheidenden Faktor bei der Erkennung durch rein bildgestützte Eye Tracking Verfahren (Khamis & Bulling, 2015). Baltrusaitis versucht mit seinem *Constrained Local Neural Fields* Ansatz auch explizit dieses Problem zu lösen (Baltrusaitis, 2016). Nichtsdestotrotz hat der Faktor Licht einen Einfluss auf die Erkennungsrate, wie die Daten zeigen.

**Abbildung 36: Versuchsküche mit Leuchtstoffröhre (rechts oben)**

Die Abbildung 39 zeigt die Versuchsküche, in der sich in der rechten oberen Ecke eine Leuchtstoffröhre befindet. Dies war während der gesamten Versuchsreihe eingeschaltet. Nach der Auswertung der Daten und einer ersten Vermutung, dass Licht ein Einflussfaktor in betracht gezogen werden muss, wurde mit Hilfe der Android Anwendung *Lux Light Meter* die

Beleuchtungsstärke für die jeweiligen Positionen 1,5,4 gemessen. Diese haben den gleichen Abstand von der Küche und sind somit vergleichbar.

| Position | Beleuchtungsstärke (lux) | Erkannte Blicke | Abstand Wand rechts (cm) |
|---|---|---|---|
| **1** | 395 | 2513 | 240 |
| **5** | 245 | 2165 | 130 |
| **4** | 140 | 1837 | 380 |

**Tabelle 2: Beleuchtungsstärke und Trefferquote von Position 1,5,4**

Tabelle 2 zeigt deutlich, dass ein Zusammenhang zwischen Beleuchtungsstärke und den erkannten Blicken pro Position besteht. Es ist zudem zu vermuten, dass die höhere Beleuchtungsstärke an Position 1, obwohl weitere von der Lichtquelle entfernt, mit dem Abluftrohr zwischen der Position und der Leuchtstoffröhre zusammenhängt. Es sollte jedoch in weiteren Versuchen ein geeichtes Lichtmessgerät verwendet werden.

Zudem kann gezeigt werden, dass das System bestimmen kann, an welcher Position sich der Nutzende im Raum befindet. Neben dem Faktor, dass *OpenFace* die Position des Kopfes der Testers relativ zur Kamera ausgibt, ermöglicht der Einsatz von multiplen Kameras die ungefähre Positionsschätzung.

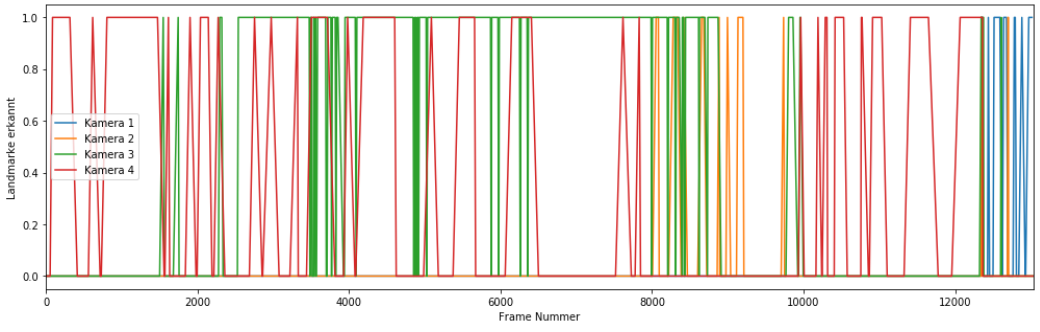


**Abbildung 37: Erfolgreiche Landmarken Erkennungsrate aufgetrennt nach Kamera für Proband 14**

Abbildung 37 zeigt dies exemplarisch für Versuchsperson 14. Während der Versuche wurde durch den Annotierungs Modus die Reihenfolge der AOIs bzw. der Positionen in einer Meta *JSON* Datei aufgezeichnet. Wie bereits in Punkt 4.5 Limitationen erwähnt leider nicht der Zeitpunkt, wann der Wechsel vorgenommen wurde. Für Proband 14 war die Positions Reihenfolge 3, 2, 1, 4, 5. Dies spiegelt Abbildung 37 dadurch wieder, dass unterschiedliche Kameras den Nutzenden erkennen. Beginnend mit Kamera 4, da der Nutzende sich an Position 3 und 2 befindet. Anschließend wird der Wechsel auf Position 1 durch das Erkennen durch Kamera 2 und 3 belegt. Der Wechsel auf Position 4 aktiviert wiederum Kamera 4 und abschließend erkennt Kamera 1 den Tester auf der letzten Position 5. Durch diesen Umstand kann das System zu jedem Zeitpunkt feststellen, ob sich der Nutzende beispielsweise vor dem Herd oder Spüle befindet, ohne seine Blicke auswerten zu müssen, und kann somit schon eine erste Einschätzung über die Intention des Nutzenden treffen.

## 6.3 Auswirkungen von körperlichen Merkmalen

Ergebnis Punkt 5.4 zeigt deutlich, dass körperliche Merkmale einzelner Probanden Auswirkungen auf das Erkennen von Gesichtern haben. Hierbei offenbart sich als erstes, dass es eine quasi optimale Körpergröße gibt, bei der das System überdurchschnittlich gut die Gesichter der Probanden erkennt. Diese Größe dürfte, betrachtet man die Korrelation zwischen Größe und erkannten Gesichtern, bei 180cm liegen. Die Regressionsgeraden aus Abbildung 31 verdeutlichen dies, indem bei kleinen Körpergrößen diese zuerst eine positive Steigung, anschließend bei größeren Personen ab 180cm eine negative Steigung aufweisen. Eine Begründung hierfür findet sich exemplarisch in Tabelle 1. Diese zeigt auf, dass kleinere Personen deutlich schlechter erkannt werden, umso weiter sie von der Kamera entfernt stehen. Dieser Umstand fußt auf dem Sichtfeld der Kameras. Steht nun eine kleinere Person weiter entfernt von der Kamera, kann es dazu kommen, dass das Sichtfeld über diese hinweg ausgerichtet ist. Sobald das Gesicht nicht mehr vollständig erfasst wird, ist eine Auswertung des Blicks nicht mehr möglich. Abbildung 38 zeigt dies. Beide Personen befinden sich auf Position 3. Die Person rechts (193cm) wird noch von der Kamera erkannt und ihre Blickrichtung kann ausgewertet werden. Die Person links (165cm) ist zu klein und kann somit nicht mehr vollständig erfasst werden.

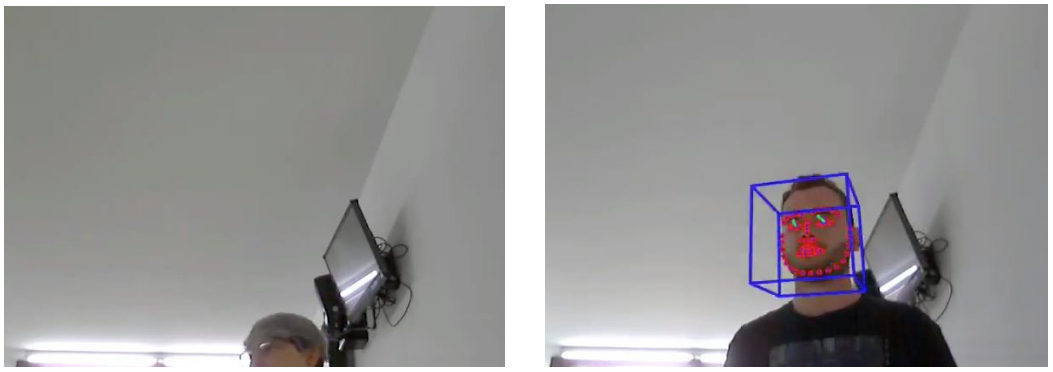

**Abbildung 38: Kleine Person (links) versus große Person (rechts) Position 3**

Mit Punkt 5.4 wird deutlich, dass Merkmale wie eine Brille oder ein Bart sich deutlich auf die Erkennungsgenauigkeit des Gesichts auswirken. Wie schon in der technischen Implementierung unter 3.3.1.2 beschrieben nutzt *OpenFace* ein *Conditional Local Neural Fields (CLNF),* um mit den sich daraus ergebenden Landmarken die Blickrichtung der Augen zu berechnen. Die Vorteile dieser Methode beschreibt Tadas Baltrusaitis in seiner Arbeit "Constrained Local Neural Fields for robust facial landmark detection in the wild" detailliert (Baltrusaitis & Morency, 2013). Dabei wird aber zur initialen Erkennung der Landmarken ein gängiges *Active Shape Model* benutzt. Laut der Arbeit "Semi self-training beard/moustache detection and segmentation simultaneously", in der sich explizit mit Landmarken Erkennung bei Gesichtern mit Bart fokussiert wurde, wird betont, dass die Verwendung von *Active Shape Models* problematisch sein kann (Le & Savvides, 2017). Die Forscher verwenden daher ein *Modified Active Shape Model*, das wesentlich zuverlässiger bei der Erkennung funktioniert, da statt nur 68 Landmarken 161 verwendet werden. In der folgenden Abbildung 39 finden sich Tester, die eine niedrige Erkennungsrate haben und mit den Faktoren Brille oder Bart ausgestattet sind. Was aber die Analysezeit noch einmal verlängern würde (s. Flaschenhals Analyse 3.3.4.1) .

**Abbildung 39: Probanden 11, 12, 27, 39 (von rechts)**

## 6.4 Trefferquoten abhängig von der AOI

Punkt 5.5 hat deutlich gezeigt, dass ein Großteil der Blicke auf die Arbeitsfläche bzw. an die Wand, an der die Kameras befestigt sind, trifft. Blicke auf Gegenstände oberhalb bzw. unterhalb dieser Zone werden nahezu nicht registriert. Eine Begründung hierfür kann im Datensatz gefunden werden, den *OpenFace* nutzt, um die Blickdaten der Nutzenden auszuwerten. Dabei baut das zugrundeliegende Modell auf dem *MPIIGaze* Datensatz auf. Hierfür wurden durch Xucong Zhang, wieder unter der Leitung von Professor Andreas Bulling, 213.659 Laptop Webcam Bilder von 15 Probanden über mehrere Monate hinweg gesammelt (Zhang & Bulling, 2017). Die Probanden mussten auf einen grauen Kreis mit einem weißen Punkt in der Mitte klicken, der an unterschiedlichen Positionen des Bildschirms auftauchte. Hierdurch konnten die durch die Webcam des Laptops aufgenommen Bilder mit den Soll-Blickrichtungen annotiert werden, welche wiederum in die Erstellung eines multimodalen konvolutionäre neuronalen Netzes einflossen. Dieses ist mit eines der Kernelemente der Gaze Berechnung von *OpenFace*, das schlussendlich den Blick Richtungsvektor ausgibt.

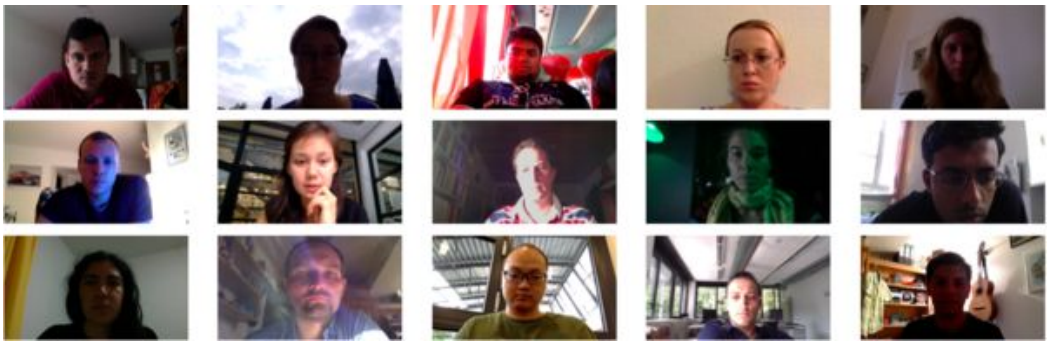

**Abbildung 40: Bilder des MPIIGaze Datensatzes**

Abbildung 40 zeigt die exemplarisch die dabei entstanden Bilder des *MPIIGaze* Datensatzes. Dabei zeigt sich, dass diese Bilder nicht in dem gleichen Winkel wie des hier in der Arbeit vorliegenden Versuchsaufbaus aufgenommen worden. Der Winkel zwischen Kamera und Kopf ist in diesem Fall wesentlich flacher und nicht so steil von Unten wie in der Versuchsküche. Daraus resultiert, dass zur Verbesserung der Erkennungsrate von Elemente im oberen bzw. unteren Bereich der Küche ein neuer Datensatz trainiert werden muss, der *MPIIGaze* ergänzt oder ersetzt.

## 6.5 Trefferquote der AOI abhängig von der Position

Zwar ist durch die im Punkt 4.5 beschriebene Annotierungslimitation nicht zu sagen, welche AOI der Nutzende tatsächlich angeblickt hat, doch lässt sich durchaus eine Aussage über die Verteilung der Blicke abhängig von der Position auf die jeweilige AOI treffen.

Bereits in Punkt 5.5 konnte gezeigt werden, dass die Position einen Einfluss auf die getroffenen AOIs hat. Die Spüle wird hierbei insgesamt überproportional oft getroffen, was daran liegt, dass sie die am nächstgelegene AOI für drei Positionen ist. In diesem Fall wären das Position 2, 3 und 4. Dazu kommt noch der Umstand, dass ihre Fläche mit 0,45qm deutlich größer ist, als die der einzelnen Cerankochfelder.

Der Punkt 5.6 zeigt, auf welche Art und Weise sich die Position auf die erkannten AOIs auswirken. Während an Positionen 2, 3 und 4, die alle im linken Teil der Küche liegen, primär die Spüle erkannt wird, so werden an Position 5, im rechten Teil der Küche, eher die Cerankochfelder erkannt. An Position 1 werden mit einer vergleichbaren Häufigkeit Cerankochfelder und Spüle getroffen, da eine ähnliche Distanz zu beiden AOIs vorhanden ist. Die Spüle wird aufgrund ihrer Größe häufiger getroffen. Zudem fällt auch, dass neben diesen AOIs auf der Arbeitsplatte primär AOIs getroffen werden, die in der Nähe der jeweiligen Position liegen. Bei der Position 2, 3 und 4 sind dies immer Elemente wie der Geschirrspüler oder die Mikrowelle. Die nachfolgende Abbildung visualisiert diesen Effekt für Position 4. Primär wird hierbei die Spüle, der Geschirrspüler und die Küchenschränke 5 und 8 getroffen. Die Cerankochfelder werden nahezu nicht erkannt.

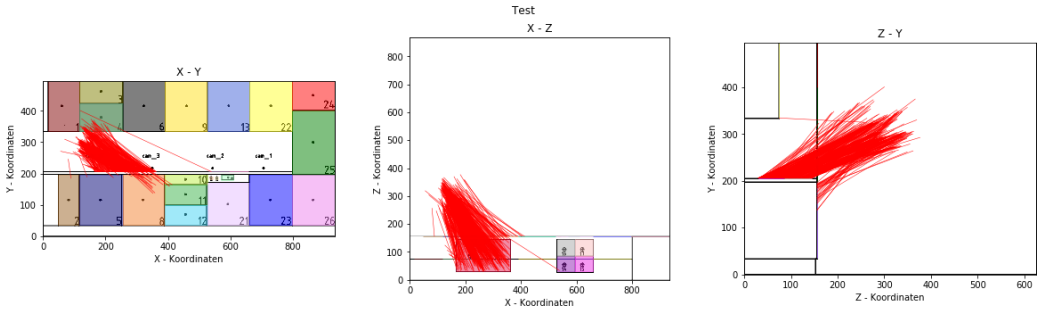


**Abbildung 41: Blick Visualisierung für Position 4**

Abbildung 41 zeigt nun eine Visualisierung für Position 5. Hier werden die Cerankochfelder, der Kühlschrank sowie weitere Elemente des Herdes primär getroffen. Blicke zur Spüle wird hierbei nahezu nicht erkannt.

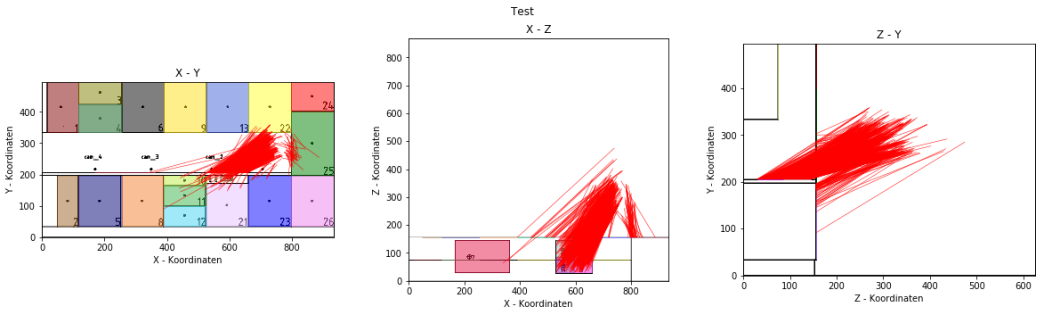


**Abbildung 42: Blick Visualisierung für Position 5**

Eine Begründung für den Umstand, dass Elemente im unmittelbaren Umfeld der Position erkannt werden, kann darin gefunden werden, dass ab einer gewissen Drehung des Kopfes von der Kamera weg, eines der Augen für die Kamera nicht mehr sichtbar ist.

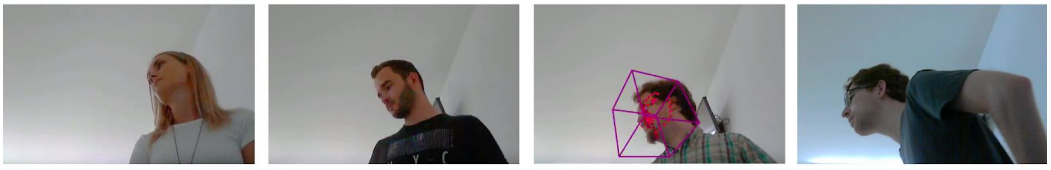

**Abbildung 43: Nichterkennung der Probanden bei weit entfernten AOIs**

Abbildung 43 zeigt mehrere dieser problematischen Fälle. An dieser Stelle muss diskutiert werden, ob dieser Umstand tatsächlich ein Problem darstellt. Dies basiert auf der Überlegung, dass Nutzende in einer Küche in den meisten Fällen situationsbezogen arbeiten. Das bedeutet, dass der Interaktionsbereich meist direkt vor ihnen liegt. Dieser Umstand konnte auch während der Studie beobachtet werden. Fast alle Nutzende merkten im persönlichen Gespräch nach den Versuchen für Position 2 an, dass diese für sie zu unangenehmen Verrenkungen führte, um weit entfernte AOIs zu betrachten. Dies führte wiederum dazu, dass die Drehung des Kopfes so stark ausgeprägt ist, dass eine Blickerkennung unmöglich ist. Es müssten weitere Untersuchungen durchgeführt werden, ob Nutzende in einem freien, nicht Positions gebundenen Versuchsaufbau, vor das betrachtete Objekt gehen und es dann direkt

betrachten. Hierbei wäre, durch den Einsatz mehrere Kameras, eine Erkennung der Blicke wiederum wahrscheinlicher.

## 6.6 Bewertung hinsichtlich der Forschungsfrage

In diesem letzten Abschnitt der Diskussion soll ein Fazit unter die Studie gezogen und bewertet werden, inwiefern die Ergebnisse eine Relevanz für die Forschungsfrage haben. Die Forschungsfrage gliedert sich in zwei Teile (s. 2.6). Dies ist zum einen die Frage danach, wie ein Eyetracking Systeme in der Küche mit mehreren Kameras entwickelt werden kann. Darauf wird in Kapitel 3 ausführlich Antwort gegeben. Die andere Frage, die gestellt wurde, ist die nach den Limitation und Problemen des Systems, die mithilfe der evaluierenden Studie (s. Kapitel 4, 5 und 6) versucht wird zu beantworten.

Eine der primären Probleme des Systems, dies macht der Abschnitt 6.4 deutlich, ist der nicht angepasste Datensatz zur Bestimmung des Blickrichtungsvektors der Augen. Für den aktuellen Aufbau der Kameras und den Blicken in den oberen und unteren Bereich der Küche, ist der *MPIIGaze* Datensatz nur bedingt geeignet. Es müsste daher ein spezifischer auf den Usecase zugeschnittener Bilddatensatz trainiert werden.

Eine weitere Problematik zeigt sich mit Hinblick auf die Lichtverhältnisse. Diese sind teilweise von der Beleuchtung vor Ort abhängig, teils vermutlich auch von den Wetterverhältnissen, sollten große Fenster vorhanden sein. Diese Eventualitäten wurden in der Studie nur im Ansatz aufgezeigt. Daher ist es nötig in weiteren Untersuchungen herauszufinden, wo genau in diesem Bereich Probleme und Limitationen des Systems liegen.

Es konnte mit der Studie zudem gezeigt werden, dass der Umstand wie gut das System funktioniert stark von der Versuchsperson und deren Position in der Küche abhängig ist. Bedingt durch den Aufbau der Kameras und deren Ausrichtung entlang der Küchenzeile, ist festzustellen, dass es eine grünen Bereich vor der Küche gibt, indem mit einer hohen Wahrscheinlichkeit ein Großteil der Nutzenden, unabhängig von ihrer Körpergröße, erkannt wird. Wobei auch klar gezeigt wurde, dass kleine Personen den Nachteil haben, umso weiter sie von der Kamera entfernt stehen, umso schlechter sie erkannt werden.

Zudem zeigt die Studie, dass das System Blicke auf Objekte, die weit vom Nutzenden entfernt liegen, bei denen der Winkel zwischen Kopf und Objekt sehr groß ist, nur bedingt erkennen kann. Der Punkt 6.5 zeigt, dass umso

näher Objekte an der Position des Nutzenden liegen, das System eine Blick auf diese erkennt.

# 7 Ausblick

In dieser Arbeit wurde gezeigt, wie ein System aussehen kann, dass in einer *smarten* Küche die Blicke von Nutzern erkennt und wo dessen Limitierungen und Probleme liegen. Die beiden primären Vorteile des System sind einerseits seine Skalierbarkeit und der kostengünstige Aufbau durch Hard- als auch Software.

An dieser Stelle muss eine Frage von Professor Dr. Niels Henze gestellt beziehungsweise beantwortet werden, welche sich auf die universelle Einsetzbarkeit des Systems bezieht. Die Vorstellung ein kostengünstiges System einfach und flexibel in den unterschiedlichsten Szenarien zu platzieren und damit die Blicke von Nutzern zu erfassen ist durchaus interessant. Die Einsatzszenarien gehen definitiv über den Bereich einer *Smart Kitchen* oder eines *Smart Homes* hinaus. Seien es nun öffentliche Displays, wie in der Forschung Bullings, oder beispielsweise im Bereich der Erfassung des Kundenverhaltens vor Regalen in Supermärkten. Die beiden Hauptanforderungen für alle diese Anwendungsszenarien sind Probleme mit denen das hier vorliegende System, das konnte die Studie zeigen, auch zu kämpfen hat. Dies wäre einerseits die Erstellung eines Datensatzes, der auf den Anwendungsfall spezifisch zugeschnitten sein muss, um eine stabile Blickerkennung durch *OpenFace* gewährleisten zu können. Zum anderen ist Erkennungsrate bedingt durch die aktuelle Verwendung des *Raspberry Pi* ein Problem. In dieser Arbeit wurde dargelegt, wo die Vorteile einer dezentrale Architektur liegen, doch sollte diese in Zukunft auf das jeweilige Szenario zugeschnitten sein.

Für weitere Arbeiten wäre es zudem interessant zu untersuchen wie genau das System die Blicke der Nutzenden in unterschiedlichsten Anwendungsfällen erfassen kann. Die Evaluierung des System konnte zeigen, dass beispielsweise Faktoren wie die Lichtverhältnisse durchaus Einfluss auf die Erkennungsrate haben. Dies ist im Besonderen interessant, wenn man den über den Anwendungsbereich Küche hinaus denkt.

Abschließend muss dennoch gesagt werden, dass die Interaktion per Blick enormes Potential bietet, um *smarte* Systeme, die die Menschen im Alltag oder darüber hinaus unterstützen, zu entwickeln.

## Literaturverzeichnis


Xu Li, Rongxing Lu, X. L. (2011). Smart Community: An Internet of Things Application. Abgerufen von www.homeplug.

Dale, R. (2016). GfK Future of Smart Home Study. Abgerufen von https://www.gfk.com/fileadmin/user_upload/dyna_content/GB/documents/Innovation_event/GfK_Future_of_Smart_Home__Global_.pdf

Solaimani, S., Keijzer-Broers, W., & Bouwman, H. (2015). What we do – and don't – know about the Smart Home: An analysis of the Smart Home literature. http://doi.org/10.1177/1420326X13516350

Blasco, R., Marco, Á., Casas, R., Cirujano, D., & Picking, R. (2014). A Smart Kitchen for Ambient Assisted Living. Sensors, 14, 1629–1653. http://doi.org/10.3390/s140101629

AXA Versicherung. (2019). Unfälle zu Hause vermeiden: Sicherheits-Tipps⁄ AXA. Abgerufen 19. September 2019, von https://www.axa.de/das-plus-von-axa/haus-mieten-wohnen/haus-unfall/unfaelle-im-haus-checkliste

Gruber T. & Hills, E. (2014). Voice trigger for a digital assistant. Abgerufen von https://patentimages.storage.googleapis.com/7c/2d/8f/ea11a2c87cda3a/US20140222436A1.pdf

Nico Hornig. (2017). Die Probleme von Alexa, Siri und Google Assistant. Abgerufen 19. September 2019, von https://www.wiwo.de/unternehmen/it/die-probleme-von-alexa-siri-und-google-assistant-warum-spracherkennung-so-schwierig-ist/20327384.html

Mark Weiser. (1991). The Computer for the 21st Century. Abgerufen 19. September 2019, von https://web.archive.org/web/20070630213952/http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html

Ramaswamy, S., & Tripathi, R. (2015). Internet of Things (IoT): A Literature Review. Journal of Computer and Communications, 3, 164–173. http://doi.org/10.4236/jcc.2015.35021

Cook, D. J., & Das, S. K. (2005). Smart environments : technologies, protocols, and applications. John Wiley. Abgerufen von https://books.google.de/books?hl=de&lr=&id=fZ5gfxMLw-oC&oi=fnd&pg=PR7&dq=mart+Environments:+Technology,+Protocols+and+Applications.+Wiley-Interscience&ots=OUi0FAQFj0&sig=P9yw_ouQRGtMdp7cOklfY03Yeic#v=onepage&q&f=false

Poslad, S. (2009). Ubiquitous Computing Ubiquitous Computing: Smart Devices, Environments and Interactions. Abgerufen von http://www.eecs.qmul.ac.uk/~stefan/publications/2009-poslad-Ubiquitous Computing-chapter1.pdf

Biundo, S., & Wendemuth, A. (2010). Von kognitiven technischen Systemen zu Companion-Systemen. KI - Künstliche Intelligenz, 24(4), 335–339. http://doi.org/10.1007/s13218-010-0056-9

Ludwig, B. (2015). Interaktive Assistenzsysteme (S. 5–46). http://doi.org/10.1007/978-3-662-44819-9_2

Harper, R. (2003). Inside the Smart Home. Abgerufen von http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.661.3611&rep=rep1&type=pdf

Kazmierzak, F. (2011). Smart Home Environment-Concepts and Solutions. Abgerufen von https://www.snet.tu-berlin.de/fileadmin/fg220/courses/WS1112/snet-project/smart-home-environments-kazmierzak.pdf

Augusto, J. C., & Mccullagh, P. (2007). Ambient Intelligence: Concepts and Applications. Abgerufen von https://pdfs.semanticscholar.org/2b64/fae8054ebffd3e41ebe0cf7eb6c70f4725f9.pdf

Brooks, K. (2003). The Context Quintet: Narrative Elements Applied to Context Awareness. Abgerufen von http://alumni.media.mit.edu/~brooks/storybiz/Brooks-context_quintet.pdf

Chen, J.-H., Pei-Yu Chi, P., Chu, H.-H., Chia-Hui Chen, C., & Huang, P. (2010). A Smart Kitchen for Nutrition-Aware Cooking. Abgerufen von www.computer.org/pervasive

Kranz, M., Schmidt, A., Maldonado, A., Rusu, B., Beetz, M., Hörnler, B., & Rigoll, G. (2007). Context-Aware Kitchen Utilities. Abgerufen von http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.92.8416&rep=rep1&type=pdf

Møllenbach, E. ; Hansen, J. P., & Lillholm, M. (2019). Eye Movements in Gaze Interaction. Journal of Eye Movement Research (Bd. 6). APA. Abgerufen von https://orbit.dtu.dk/files/120332523/Eye_Movements_in_Gaze_Interaction.pdf

Andrew T. Duchowski. (2007). Eye Tracking Methodology. London: Springer London. http://doi.org/10.1007/978-1-84628-609-4

Csanyi, Gottfried; Reichl, Franz; Steiner, A. (2012). Eye Tracking in Forschung und Lehre. Möglichkeiten und Grenzen eines vielversprechenden Erkenntnismittels. Abgerufen von http://www.waxmann.com

Kaur, M., Tremaine, M., Huang, N., Wilder, J., Gacovski, Z., Flippo, F., & Mantravadi, C. S. (2003). Where is it? Event Synchronization in Gaze-Speech Input Systems. Abgerufen von http://immi.inesc-id.pt/immi04/works/week6/p151-kaur.pdf

Maglio, P. P., Matlock, T., Campbell, C. S., Zhai, S., & Smith, B. A. (2000). Gaze and speech in attentive user interfaces. In International Conference on Multimodal Interfaces (pp. 1-7). Springer, Berlin, Heidelberg.

Majaranta, P., & Bulling, A. (2014). Eye Tracking and Eye-Based Human-Computer Interaction. http://doi.org/10.1007/978-1-4471-6392-3_3

Shell, J. S., Bradbury, J. S., Knowles, C. B., Dickie, C., & Vertegaal, R. (2003). eyeCOOK: A Gaze and Speech Enabled Attentive Cookbook. Abgerufen von https://s3.amazonaws.com/academia.edu.documents/3246720/Shell_Bradbudy_Knowles_Dickie_Vertegaal_2003.pdf?response-content-disposition=inline%3B filename%3DEyeCOOK_A_Gaze_and_Speech_Enabled_Attent.pdf&X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKI

Khamis, M., Hoesl, A., Klimczak, A., Reiss, M., Alt, F., & Bulling, A. (2017). EyeScout: Active Eye Tracking for Position and Movement Independent Gaze Interaction with Large Public Displays, 155–166. http://doi.org/10.1145/3126594.3126630

Mrochen, M., Eldine, M. S., Kaemmerer, M., Seiler, T., & Hütz, W. (2001). Improvement in photorefractive corneal laser surgery results using an active eye-tracking system. Journal of cataract and refractive surgery, 27(7), 1000–6. http://doi.org/10.1016/s0886-3350(00)00884-1

Solomon, K. D., Fernández de Castro, L. E., Sandoval, H. P., Biber, J. M., Groat, B., Neff, K. D., … Lindstrom, R. L. (2009). LASIK World Literature Review. Ophthalmology, 116(4), 691–701. http://doi.org/10.1016/j.ophtha.2008.12.037

Hennessey, C., & Fiset, J. (2012). Long range eye tracking. In Proceedings of the Symposium on Eye Tracking Research and Applications - ETRA '12 (S. 249). New York, New York, USA: ACM Press. http://doi.org/10.1145/2168556.2168608

Chan Chao Ning, Oe Shunichiro, & Chern-Sheng Lin. (2007). Development of an active gaze tracking system in unrestricted posture. In 2007 International Conference on Control, Automation and Systems (S. 1348–1353). IEEE. http://doi.org/10.1109/ICCAS.2007.4406547

Khamis, M., Kienle, A., Alt, F., & Bulling, A. (2018). GazeDrone: Mobile Eye-Based Interaction in Public Space Without Augmenting the User, 66. http://doi.org/10.1145/3213526.3213539

Perceptual User Interfaces group (2019). Prof. Dr. Andreas Bulling – Perceptual User Interfaces. Abgerufen 19. September 2019, von https://perceptual.mpi-inf.mpg.de/people/andreas-bulling/

Zhang, Y., Uller, J. M. ¨, Chong, M. K., Bulling, A., & Gellersen, H. (2014). GazeHorizon: Enabling Passers-by to Interact with Public Displays by Gaze. http://doi.org/10.1145/2632048.2636071

Khamis, M., Bulling, A., & Alt, F. (2015). Tackling Challenges of Interactive Public Displays using Gaze. http://doi.org/10.1145/2800835.2807951

Just MA, & Carpenter PA. (1980). A theory of reading: from eye fixations to comprehension. Psychological review, 87(4), 329–54.

Risko, E. F., & Kingstone, A. (2011). Eyes wide shut: implied social presence, eye tracking and attention. Attention, Perception, & Psychophysics, 73(2), 291-296.

Baltrušaitis, T., Robinson, P., & Morency, L. P. (2016, March). OpenFace: an open source facial behavior analysis toolkit. In 2016 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1-10). IEEE.

Bosek, M., Büttner, J., & Hauser, T. (2019). Eyetracking in der Küche, 1–32., Universität Regensburg

Farnsworth, B. (2019). Eye Tracker Prices - An Overview of 20+ Eye Trackers. Abgerufen von https://imotions.com/blog/eye-tracker-prices/

Raspberry PI Foundation. (2019.). Raspberry Pi Downloads - Software for the Raspberry Pi. Abgerufen 19. August 2019, von https://www.raspberrypi.org/downloads/

OpenFace Github Wiki (2018). Unix Installation. Abgerufen 19. August 2019, von https://github.com/TadasBaltrusaitis/OpenFace/wiki/Unix-Installation

OpenFace Github (2019). Abgerufen 19. August 2019, von https://github.com/TadasBaltrusaitis/OpenFace

I. Fette, A. Melnikov. (2011). RFC 6455 - The WebSocket Protocol. Abgerufen von http://www.rfc-editor.org/info/rfc6455.

Pipeline (software) - Wikipedia. (o. J.). Abgerufen 19. September 2019, von https://en.wikipedia.org/wiki/Pipeline_(software)

Buschmann, F. (2000). Pattern-orientierte Software-Architektur : ein Pattern-System. Addison-Wesley. Abgerufen von https://books.google.de/books?hl=de&lr=&id=o2nuK0qpo3QC

Black, G., & Johnson, P. (1999). U.S. Patent No. 5,878,056. Washington, DC: U.S. Patent and Trademark Office.

Wolfram Hempel. (2017.). An overview of realtime protocols. Abgerufen 19. September 2019, von https://deepstreamhub.com/blog/an-overview-of-realtime-protocols/

Jan Carsten Lohmüller. (2016+). Reactive Programming – Mehr als nur Streams und Lambdas | Informatik Aktuell. Abgerufen 19. September 2019, von https://www.informatik-aktuell.de/entwicklung/programmiersprachen/reactive-programming-mehr-als-nur-streams-und-lambdas.html

Peng, J., Bo, L., & Xu, J. (2009). Conditional neural fields. In Advances in neural information processing systems (pp. 1419-1427).

Peter Robinson. (2016). Computer Laboratory – Research in the Rainbow Group: Research. Abgerufen 21. September 2019, von https://www.cl.cam.ac.uk/research/rainbow/research/

MultiComp Lab. (2019). MultiComp | MultiComp Lab's mission is to build the algorithms and computational foundation to understand the interdependence between human verbal, visual, and vocal behaviors expressed during social communicative interactions. Abgerufen 21. September 2019, von http://multicomp.cs.cmu.edu/

Biernacki, P., & Waldorf, D. (1981). Snowball sampling: Problems and techniques of chain referral sampling. Sociological methods & research, 10(2), 141-163.

Stephen Matthews. (2016). Netherlands men top height table at just under 6ft while Guatemalan women are shortest | Daily Mail Online. Abgerufen 23. September 2019, von https://www.dailymail.co.uk/health/article-3706923/How-tall-country-Men-Netherlands-table-just-6ft-women-Guatemala-stand-4-10.html

Baltrusaitis, T., Robinson, P., & Morency, L. P. (2013). Constrained local neural fields for robust facial landmark detection in the wild. In Proceedings of the IEEE international conference on computer vision workshops (pp. 354-361).

Le, T. H. N., Luu, K., Zhu, C., & Savvides, M. (2017). Semi self-training beard/moustache detection and segmentation simultaneously. Image and Vision Computing, 58, 214-223.

Zhang, X., Sugano, Y., Fritz, M., & Bulling, A. (2017). Mpiigaze: Real-world dataset and deep appearance-based gaze estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 41(1), 162-175.

IT Wissen. (2011). VNC (virtual network computing) :: ITWissen.info. Abgerufen 24. September 2019, von https://www.itwissen.info/VNC-virtual-network-computing.html

Brückler, F. M. (2017). Geschichte der Mathematik kompakt: Das Wichtigste aus Analysis, Wahrscheinlichkeitstheorie, angewandter Mathematik, Topologie und Mengenlehre. Springer-Verlag.

Lapidot-Lefler, N., & Barak, A. (2012). Effects of anonymity, invisibility, and lack of eye-contact on toxic online disinhibition. Computers in human behavior, 28(2), 434-443.

# Anhang

## Anhang A: Einverständniserklärung

Wir laden Sie ein, an der von Markus Guder initiierten und durchgeführten experimentellen Studie Machbarkeitsstudie Zur Erkennung Des Blickpunkts In Einer Frei Konfigurierbaren Küche teilzunehmen. Die Forschung wird von Prof. Dr. Bernd Ludwig an der Universität Regensburg betreut. Bitte beachten Sie:

- Ihre Teilnahme ist vollkommen freiwillig.
- Die experimentelle Studie dauert ca. 10 Minuten.
- Wir werden während der Sitzung Fotos machen.
- Wir werden während der Sitzung Videos aufnehmen.
- Wir verfolgen Ihre Augen- und Kopfbewegungen während der Sitzung.
- Wir können unsere Ergebnisse aus dieser und anderen Sitzungen in unseren Berichten veröffentlichen, aber alle diese Berichte sind vertraulich und enthalten nicht Ihren Namen.

Wenn Sie Fragen oder Beschwerden zum gesamten Einwilligung Prozess dieser Forschungsstudie oder zu Ihren Rechten als Human Researcher haben, wenden Sie sich bitte an Markus Guder (E-Mail: markus.guder@stud.uni-regensburg.de) oder Prof. Dr. Bernd Ludwig. Sie sollten die folgenden Informationen sorgfältig lesen. Bitte nehmen Sie sich soviel Zeit, wie Sie benötigen, um die Einwilligungserklärung zu lesen.

### 1. Zweck und Ziel dieser Forschung

Ist eine kostengünstige Blickverfolgung in der Küche möglich. Entwickeln Sie ein System, das in der Lage ist, den Blick der Benutzer ohne Eyetracker zu verfolgen. Ihre Teilnahme wird uns helfen, dieses Ziel zu erreichen. Die Ergebnisse dieser Forschung können auf wissenschaftlichen oder fachlichen Veranstaltungen präsentiert oder in wissenschaftlichen Arbeiten und Zeitschriften veröffentlicht werden.

### 2. Teilnahme und Vergütung

Ihre Teilnahme an dieser experimentellen Studie ist völlig freiwillig. Du wirst eine von etwa 50 Personen sein, die für diese Forschung getestet werden. Sie erhalten keine Entschädigung. Sie können die Teilnahme jederzeit ohne Strafe widerrufen und beenden. Wenn Sie die Teilnahme ablehnen oder sich von der experimentellen Studie zurückziehen, wird niemand auf dem Campus darüber informiert. Sie können sich weigern, Fragen zu beantworten, die Sie nicht beantworten möchten. Der Prüfer kann Sie von dieser Forschung zurückziehen, wenn Ihr Arzt uns mitteilt, dass eine fortgesetzte Teilnahme Ihre Gesundheit schädigen kann.

## 3. Vorgehensweise

Nachdem Sie Ihre Einwilligung nach Aufklärung bestätigt haben, werden Sie dies tun:

*1. Der Benutzer geht zu verschiedenen Positionen.*

*2. Der Benutzer sieht sich verschiedene Punkte in der Küche an.*

Der gesamte Ablauf dieser experimentellen Studie wird ca. 10 Minuten dauern.

## 4. Risiken und Nutzen

Mit dieser experimentellen Studie sind keine Risiken verbunden. Unannehmlichkeiten oder Unannehmlichkeiten sind gering und werden wahrscheinlich nicht auftreten. Wenn Unannehmlichkeiten zu einem Problem werden, können Sie Ihre Teilnahme beenden. Durch die Teilnahme an dieser experimentellen Studie werden Sie nicht direkt profitieren. Wir hoffen, dass die Informationen, die wir aus Ihrer Teilnahme erhalten haben, dazu beitragen können, die Forschung in diesem Bereich voranzubringen.

## 5. Datenschutz und Vertraulichkeit

Der Forscher wird Sie in Berichten, die Informationen aus dieser experimentellen Studie verwenden, nicht mit Ihrem wirklichen Namen identifizieren, und dass Ihre Vertraulichkeit als Teilnehmer an dieser experimentellen Studie sicher bleibt. Alle Daten, die Sie in dieser experimentellen Studie angeben, werden anonymisiert veröffentlicht und gemäß der Allgemeinen Datenschutzverordnung (GDPR) der Europäischen Union (EU) vertraulich behandelt. Die anschließende Nutzung von Datensätzen und Daten unterliegt einheitlichen Richtlinien zur Datennutzung, die die Anonymität von Personen und Institutionen schützen. In allen Fällen unterliegt die Verwendung von Datensätzen und Daten der GDPR. Fakultät und Verwaltung des Campus haben keinen Zugang zu Rohdaten oder Transkripten. Diese Vorsichtsmaßnahme verhindert, dass Ihre individuellen Kommentare negative Auswirkungen haben. Wir machen Fotos, nehmen Videos auf, zeichnen Ihre Augenbewegungen auf und machen möglicherweise Notizen während der experimentellen Studie. Alle Aufzeichnungen dürfen von niemandem außerhalb dieses Forschungsprojekts eingesehen werden, es sei denn, wir lassen Sie ein separates Genehmigungsformular unterschreiben, das uns die Nutzung gestattet (siehe unten). Die Aufzeichnungen werden nach Beendigung der Recherche, wie von der Förderorganisation gefordert, vernichtet oder wenn Sie sich mit dem Forscher in Verbindung setzen, um sie sofort zu vernichten oder zu löschen; wie bei jeder Veröffentlichung oder Online-Tätigkeit ist das Risiko einer Verletzung der Vertraulichkeit jederzeit möglich. Nach dem GDPR werden die Forscher den Teilnehmer informieren, wenn eine Verletzung vertraulicher Daten festgestellt wurde. Wir machen Fotos, nehmen Videos auf, zeichnen Ihre Augenbewegungen auf und machen möglicherweise Notizen während der experimentellen Studie.

## 6. Identifizierung von Prüfern

Wenn Sie Fragen oder Bedenken bezüglich der Forschung haben, können Sie sich gerne an uns wenden:

**Prof. Dr. Bernd Ludwig**
**Lehrstuhl für Informationswissenschaft**
**Universitätsstraße 31**
**93053 Regensburg, Germany**

## 7. Informierte Zustimmung und Vereinbarung

**1** Ich verstehe die mir gegebene Erklärung. Ich habe eine Kopie dieses Formulars erhalten. Ich habe alle meine Fragen zu meiner Zufriedenheit beantwortet bekommen, und ich stimme freiwillig der Teilnahme an dieser experimentellen Studie zu.

2 Ich bin damit einverstanden, dass meine Datensätze und Messungen während der Studie im Rahmen dieser Forschung veröffentlicht werden können.

3 Ich stimme zusätzlich zu, dass Fotos während dieser Studie gemacht und veröffentlicht werden können.

4 Ich stimme zusätzlich zu, dass Videos während dieser Studie aufgezeichnet und veröffentlicht werden können.

Mir ist bekannt, dass die Daten anonymisiert werden und nicht mit meinem Namen in Verbindung gebracht werden können. Aus der Zustimmung zur Veröffentlichung leite ich keine Rechte (z.B. Vergütung oder Mitautorschaft) ab. Diese Einwilligungserklärung ist im Sinne der GDPR jederzeit widerruflich. Im Falle eines Widerrufs werden die veröffentlichten Aufnahmen, soweit sie der Disposition der Forscher unterliegen, entfernt. Weitere Aufnahmen werden nach dem Widerruf der Veröffentlichung nicht mehr verwendet oder veröffentlicht.

## Erklärung zur Urheberschaft

Ich habe die Arbeit selbständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie alle Zitate und Übernahmen von fremden Aussagen kenntlich gemacht.

Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt.

Die vorgelegten Druckexemplare und die vorgelegte digitale Version sind identisch.

Von den zu § 27 Abs. 5 der Prüfungsordnung vorgesehenen Rechtsfolgen habe ich Kenntnis.

| | |
|---|---|
| Ort, Datum | Unterschrift |

## Erklärung zur Lizenzierung und Publikation dieser Arbeit

**Name:** Markus Guder

**Titel der Arbeit:** Eyetracking mit mehreren Kameras in Smart Kitchens

In der Regel räumen Sie mit Abgabe der Arbeit dem Lehrstuhl für Medieninformatik nur zwingend das Recht ein, dass die Arbeit zur Bewertung gelesen, gespeichert und vervielfältigt werden darf. Idealerweise liefern Seminararbeiten, Projektdokumentationen und Abschlussarbeiten aber einen Erkenntnisgewinn, von dem auch andere profitieren können. Wir möchten Sie deshalb bitten, uns weitere Rechte einzuräumen, bzw. idealerweise Ihre Arbeit unter eine freie Lizenz zu stellen.

Die in unseren Augen praktikabelsten Lösungen sind vorselektiert.

Hiermit gestatte ich die Verwendung der **schriftlichen Ausarbeitung** zeitlich unbegrenzt und nicht-exklusiv unter folgenden Bedingungen:

☐ Nur zur Bewertung dieser Arbeit
☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
☒ Unter einer Creative-Commons-Lizenz mit den folgenden Einschränkungen:
- ☒ BY – Namensnennung des Autors
- ☐ NC – Nichtkommerziell
- ☐ SA – Share-Alike, d.h. alle Änderungen müssen unter die gleiche Lizenz gestellt werden.

(An Zitaten und Abbildungen aus fremden Quellen werden keine weiteren Rechte eingeräumt.)

Außerdem gestatte ich die Verwendung des im Rahmen dieser Arbeit erstellen **Quellcodes** unter folgender Lizenz:

☐ Nur zur Bewertung dieser Arbeit
☐ Nur innerhalb des Lehrstuhls im Rahmen von Forschung und Lehre
☐ Unter der CC-0-Lizenz (= beliebige Nutzung)
☒ Unter der MIT-Lizenz (= Namensnennung)
☐ Unter der GPLv3-Lizenz (oder neuere Versionen)

(An explizit mit einer anderen Lizenz gekennzeichneten Bibliotheken und Daten werden keine weiteren Rechte eingeräumt.)

Ich willige ein , dass der Lehrstuhl für Medieninformatik diese Arbeit – falls sie besonders gut ausfällt - auf dem Publikationsserver der Universität Regensburg veröffentlichen lässt.

Ich übertrage deshalb der Universität Regensburg das Recht, die Arbeit elektronisch zu speichern und in Datennetzen öffentlich zugänglich zu machen. Ich übertrage der Universität Regensburg ferner das Recht zur Konvertierung

zum Zwecke der Langzeitarchivierung unter Beachtung der Bewahrung des Inhalts (die Originalarchivierung bleibt erhalten).

Ich erkläre außerdem, dass von mir die urheber- und lizenzrechtliche Seite (Copyright) geklärt wurde und Rechte Dritter der Publikation nicht entgegenstehen.

☒ Ja, für die komplette Arbeit inklusive Anhang
☐ Ja, für eine um vertrauliche Informationen gekürzte Variante (auf dem Datenträger beigefügt)
☐ Nein

☐ Sperrvermerk bis (Datum):

Ort, Datum

Unterschrift

## Inhalt des beigefügten Datenträgers

| | |
|---|---|
| **/1_Ausarbeitung** | Die schriftliche Ausarbeitung als PDF und DOC |
| **/2_Code** | Quellcode und kompilierte Anwendung des Prototypen |
| **/3_Studie/Design** | Script und Überlegungen für die Benutzerstudie |
| **/3_Studie/1_Daten** | Rohdaten der Studie im CSV-Format, inkl. Beschreibung der Felder |
| **/3_Studie/2_Notebooks** | Mit Jupyter Notebook erstellte Auswertung |
| **/3_Studie/3_Lib** | Python Logik zur Transformation der Koordinaten |
| **/3_Studie/4_Bilder** | Technischen Zeichnungen der Küche (Libre CAD) |
| **/3_Studie/5_Ergebnisse** | Jupyter Notebooks exports als HTML Datei |
| **/3_Studie/6_Flaschenhals** | Daten und Auswertung der Flaschenhalsanalyse |
| **/4_Quellen** | Alle in der Arbeit zitierten Quellen |
| **/4_Quellen/Internetquellen** | Screenshots der zitierten Webseiten |
| **4_Quellen/Paper** | PDFs der zitierten Paper |
| **/5_Bilder** | Alle selbst erstellten und aus anderen Quellen übernommenen Bilder |
| **/6_Vorträge** | Folien von Antritts- und Abschlussvortrag im PDF-Format |
| **/7_Sonstiges** | Notizen aus Besprechungen |