HAW HAMBURG

BACHELOR THESIS
Hauke Simon Rösler

Modellbasierte Erkennung von Selektionsgesten

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE
WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Hauke Simon Rösler

Modellbasierte Erkennung von Selektionsgesten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang *Bachelor of Science Angewandte Informatik*
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Thomas Lehmann
Zweitgutachter: Prof. Dr. Kai von Luck

Eingereicht am: 27. März 2025

Hauke Simon Rösler

Thema der Arbeit

Modellbasierte Erkennung von Selektionsgesten

Stichworte

Dynamic Time Warping, Pose Estimation, Selektionsgesten, Gesten Erkennung

Kurzzusammenfassung

In dieser Bachelorarbeit wird ein System zur Erkennung von Selektionsgesten zur Gestensteuerung erstellt. Dazu wird die Bewegung des Nutzers mittels Pose Estimation aus RGB-Videos erfasst und Gesten mittels Dynamic Time Warping erkannt. Auch das Ziel der Selektion wird über geometrische Operationen bestimmt. Der entwickelte Prototyp demonstriert mit einer Accuracy von $0,7\overline{3}$ das Potential des Ansatz.

Hauke Simon Rösler

Title of Thesis

Model-based recognition of selection gestures

Keywords

Dynamic Time Warping, Pose Estimation, Selection Gestures, Gesture Recognition

Abstract

A system for the recognition of selection gestures was developed for this bachelor thesis. The system uses Pose Estimation to extract the Users Pose from RGB-videos. Gestures are classified in the pose data with the Dynamic Time Warping Algorithm. The targets of the selection is determined using geometric operations. The developed prototype shows the potential of the design with an accuracy of $0,7\overline{3}$.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Die Anzahl Haushalte, die Smart Home Technologien einsetzen, ist in den letzten Jahren angestiegen. Allein in der EU ist die Anzahl Nutzer zwischen 2024 und 2025 um 18,8% angestiegen [8]. Während die Funktionen eines Smart Home sich bereits nahtlos in die Umgebung einfügen, benötigt die Steuerung, abseits von Sprachassistenten, immer noch, dass Führen einer Fernbedienung um das Smart Home jederzeit zu steuern.
Gesten, d.h. Körperbewegungen, bieten die Möglichkeit, ähnlich wie verbale Sprache, das natürliche Kommunikationsverhalten des Menschen auszunutzen um eine intuitive Steuerung zu ermöglichen. Dazu wird lediglich benötigt, dass sich der Nutzer in Sensorreichweite befindet, was sich in der statischen Struktur von Wohnräumen gut flächendeckend herstellen lässt. Während Gestensteuerung eine ähnliche Funktion erfüllen wie Sprachassistenten, kann sie Menschen bedienen, die nicht zu verbaler Kommunikation fähig sind oder dieser anderweitig abgeneigt sind.
Je eine Geste mit einer Aktion zu verknüpfen funktioniert gut solange die Aktionen kontextfrei sind, d.h. solange immer alle Fenster zusammen geöffnet oder geschlossen werden sollen. Sobald jedoch nur einzelne Fenster beeinflusst werden sollen, muss der Nutzer angeben können welches. Während es möglich wäre je eine Geste für "öffne Fenster Aünd "öffne Fenster Bßu definieren, wird dies mit ansteigender Objektzahl zunehmend unintuitiv, da immer abstraktere Gesten verwendet werden müssen um die notwendige Anzahl individueller Gesten zu erreichen.
Sprachassistenten lösen dieses Problem indem das Objekt und die Aktion jeweils explizit genannt werden. Die Beispiele "öffne Fenster Aünd "öffne Fenster Bßeigen dies bereits deutlich. Die Aktion "öffneïst in beiden Fällen die selbe und trägt dementsprechend einen konstanten Betrag zur Komplexität bei. Unabhängig davon wie viele verschiedene Fenster es gibt.
Die Trennung der Selektion d.h. der Auswahl des Objekts und der Manipulation, d.h. der Aktion lässt sich auch auf Gestensteuerung übertragen. Das Ziel dieser Arbeit besteht darin, ein System zur Selektion via Gestensteuerung zu designen. Dazu soll es in der Lage sein den Output einer einzelnen RGB-Kamera in Echtzeit auszuwerten um darin Selek-

tionen zu erkennen und mit minimaler Zeitverzögerung an das größere Steuerungssystem weiterzugeben. Zu der Tatsache, dass eine Selektion stattgefunden hat soll ebenfalls das selektierte Objekt aus einer bekannten Menge stationärer Objekte identifiziert werden.

2 Analyse

In diesem Kapitel werden beschrieben aus welchen Teilproblem sich die Selektionserkennung zusammensetzt und wie diese bereits in anderen Arbeiten bewältigt wurden. Es beginnt damit mit welchen Gesten eine Selektion kommuniziert werden kann.

2.1 Selektions Gesten

Bei der Gestensteuerung, wie auch allen anderen Steuerungsmethoden, ist das Ziel dem System eine Absicht zu kommunizieren, damit diese ausgeführt wird. Die selbe Geste kann dabei aber abhängig vom Kontext unterschiedliche Absichten ausdrücken. In der zwischenmenschlichen/natürlichen Kommunikation existiert das Konzept der gemeinsamen Aufmerksamkeit [19, 20] wo die Aufmeksamkeit der Kommunikationsteilnehmer auf dem selben Objekt liegt und damit zum Kontext des Gesprächs beiträgt. Dieser Zustand wird über Blicke sowie deiktische Gesten und Ausdrücke hergestellt und koordiniert.
In Stukenbrocks *Model of the Interactional Organization of Deictic Reference and Joint Attention* [21] wird der Prozess der Herstellung gemeinsamer Aufmerksamkeit in 10 Bestandteile unterteilt. Für die angestrebte Gestensteuerung sind drei dieser von Bedeutung, da der Ansatz darin besteht über das natürliche Kommunikationsverhalten des Nutzers gemeinsame Aufmerksamkeit zwischen dem Nutzer und dem System herzustellen und so den Kontext für folgende Befehle zu bestimmen.
Der erste dieser Schritte ist dabei die Ausführung der deiktischen Geste. In dieser Arbeit handelt es sich spezifisch um die deiktische Geste des Zeigens, welche von Müller-Tomfelde [12] in drei Phasen unterteilt wird. In der ersten Phase wird das Zeigewerkzeug, in dieser Arbeit der Arm, auf das Ziel gerichtet, wo es in der zweiten Phase eine kurze Zeit verweilt. In der dritten Phase wird das Zeigewerkzeug wieder vom Ziel weg bewegt 2.1.
Der zweite relevante Schritt besteht darin, dass der Adressat die Menge plausibler Ziele bestimmt um den Suchraum für den nächsten Schritt zu begrenzen. Dieser letzte Schritt umfasst die Identifikation des Ziels über das geometrische Verhältnis des Zeigens zu den

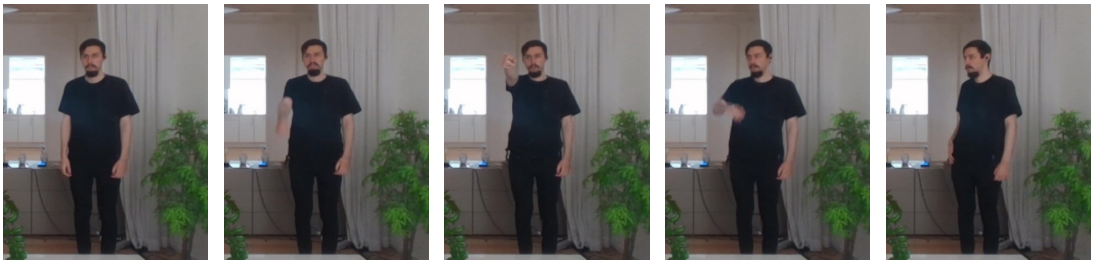

Abbildung 2.1: Beispielsequenz: Zeigen aus dem Arm

möglichen Zielen.

Zudem muss noch beachtet werden, dass sich Gesten statische und dynamische Gesten unterteilen lassen. Erstere sind durch eine charakteristische Haltung der beteiligten Körperteile definiert, die sich über den Zeitraum der Geste nicht verändern. Das heißt, dass statische Gesten auch auf Fotos identifizierbar sind. Dynamische Gesten hingegen definieren sich über einen charakteristischen Bewegungsablauf und lassen sich dementsprechend nicht auf eine einzelne Haltung reduzieren. Die Geste des Zeigen lässt sich neben der bereits beschriebenen dynamischen Variante auch statisch auffassen. Dafür wäre die Handhaltung bei der nur der Zeigefinger gestreckt ist, das charakteristischste Merkmal. Problematisch wäre jedoch die Abgrenzung zur Geste des Aufzeigens/Meldens, die die selbe Handhaltung aufweist. Dieser spezifische Fall bereitet natürlich nur Probleme, wenn sich direkt über dem Nutzer ein steuerbares Objekt befindet

2.2 Human Pose Estimation

„ Pose estimation refers to the process of estimating the configuration of the under- lying kinematic or skeletal articulation structure of a person.“ (Moeslund u. a. [11] S.105)

Pose Estimation beschreibt die Schätzung von Ort und Orientierung von Objekten und Personen im Raum basierend auf Bild und Video Daten. Die für diese Arbeit relevante Schätzung der Haltung von Personen wird begrifflich auch als Human Pose Estimation abgegrenzt. Die Haltung einer Person wird dabei über die Koordinaten einer Menge an Keypoints, die signifikante Punkte des Körpers markieren, angegeben. Die Anzahl und Zusammensetzung dieser Keypoints variiert dabei zwischen Pose Estimation Modellen. Meist wird jedoch die Struktur des menschlichen Skeletts ausgenutzt indem Keypoints

an den Gelenken platziert werden, da die Haltung zwischen diesen durch die statische Form der Knochen bestimmt ist. D.h. solange die Positionen des Schulter- und Ellenbogengelenks bekannt sind, ist klar, dass der Punkt bei der Hälfte des Oberarms mittig zwischen diesen Positionen liegt, da der Oberarmknochen gerade ist.
Manche Pose Estimation Modelle bestimmen die Positionen der Keypoints nur in der Bildebene während andere die kompletten dreidimensionalen Koordinaten im Raum bestimmen. Da Systeme, die nur auf einzelnen Videoquellen arbeiten, dabei noch Probleme mit Unklarheit in der räumlichen Tiefe haben Zheng u. a. [26], gibt es Ansätze, die zusätzliche Sensoren wie Tiefensensoren oder Inertial Measurement Unit (IMU) verwenden um die Genauigkeit der räumlichen Informationen zu erhöhen. Dementsprechend benötigen diese Ansätze jedoch spezielle Hardware, welche die Kosten erhöhen. Andere Ansätze beruhen darauf, weitere Videodaten in Form von Stereokameras oder multipler Monokameras in unterschiedlichen Perspektiven für Tiefeninformationen zu nutzen. Während die Verwendung von Stereokameras die selben Vor- und Nachteile wie bei Tiefensensoren hat, hat die Verwendung von multiplen Perspektiven den Vorteil, dass verdeckte Keypoints aus einer anderen Perspektive potentiell sichtbar sind und dementsprechend akkurater bestimmt werden können. Jedoch erfordert die Abstimmung der verschiedenen Perspektiven zueinander zusätzlichen Aufwand.

Es gibt verschieden Modelle zur Pose Estimation auf Bilddaten. Unterschieden wird bei diesen zwischen Top-Down und Bottom-Up Pose Estimation. Erstere erkennen dabei zuerst wo sich Personen im Bild befinden und bestimmen dann die Positionen der Landmarks in diesen Bereichen, während letztere zuerst alle Landmarks erkennen und im zweiten Schritt zu Personen zuordnen.

Zu den Modellen die ohne zusätzliche Sensoren auskommen, gehören MediaPipe Pose [1] und OpenPose [6]. OpenPose bietet die Option zur 3D Triangulation über zusätzliche Perspektiven, ist aber sonst auf 2D Pose Estimation von multiplen Personen im selben Bild ausgelegt.Das Bottom-Up Modell OpenPose verwendet dafür eine CNN Architektur, die die sogenannten Part Confidence Maps und Part Affinity Fields erstellt. Unter Part Confidence Map wird hier eine Heatmap für jeweils einen Keypoint verstanden, welche alle Vorkommnisse des Keypoints und deren Position im Bild ausdrückt. Die Part Affinity Fields stellen die Richtung der Gliedmaßen als Richtungsvektor pro Pixel dar. Sie werden pro Gliedmaße errechnet und dienen dazu Vermischung der Keypoints multipler

Personen zu vermeiden.

MediaPipe ist eine Sammlung von Opensource KI Bibliotheken von Google. Es enthält eine Reihe von Computer Vision Bibliotheken darunter auch die Pose Landmark Detection Bibliothek (im folgenden 'MediaPipe Pose'). Das Top-Down Modell MediaPipe Pose verwendet ein CNN, spezifisch eine Variante des BlazePose Modells Bazarevsky u. a. [2], welche GHUM Xu u. a. [25] zur 3D Modellierung verwendet [1].
Bei BlazePose handelt es sich um ein CNN-Modell dessen Fokus darauf liegt real-time Pose Estimation einzelner Personen auf mobilen Geräten auszuführen. Zu diesem Zweck verwendet es einen Ansatzt bei dem die Regression zu Keypoint-Koordinaten während des Trainings durch einen Heatmap/Offset-Zweig des Netzwerk unterstützt wird.
Um die Inferenz nur auf den Bildabschnitt indem sich die Person befindet zu reduzieren, wird ein separater 'Person Detector' verwendet, welcher mit dem BlazeFace Bazarevsky u. a. [3] Netzwerk zuerst das Gesicht erkennt. Basierend auf der Position des Gesichts wird die Bounding Box der Person bestimmt, welche den Input für das eigentliche Pose Estimation Netzwerk definiert. In den darauf folgenden Frames wird die Bounding Box stattdessen von dem Resultat des letzten Frames abgeleitet, bis keine Person mehr erkannt wurde und im nächsten Frame wieder mit dem 'Person Detector' gearbeitet wird. Dadurch besteht die Einschränkung, dass das Gesicht der Person zumindest initial sichtbar sein muss.
Erkannt werden 33 Keypoints 3.5, die die groben Features der Haltung beinhalten aber feinere Features wie die Haltung der Finger vernachlässigen. Wenn diese Informationen benötigt sind bietet MediaPipe die sogenannte "holistic landmark detection"welche in der Lage ist die Modelle für pose, face und hand landmark detection zu kombinieren. Zu den Koordinaten dieser Keypoints beinhaltet der Output auch einen visibility score, der angibt zu welcher Wahrscheinlichkeit der Keypoint im Input zu sehen ist. Das Koordinatensystem der Keypooints folgt den Bildschirmachsen wonach die x-Achse nach rechts und die y-Achse nach unten verläuft. Die z-Achse gibt die Tiefe an und verläuft entlang der Blickrichtung der Kamera.
Der Output besteht aus zwei Sets an Koordinaten, die jeweils alle 33 Keypoints beinhalten, den sogenannten Image und World Coordinates. Die Image Coordinates haben x und y Werte zwischen 0.0 und 1.0 , welche die relative Position zur Bildbreite/-höhe angeben. Die z-Achse hat ihren Nullpunkt hingegen beim Mittelpunkt der Hüfte der erkannten Person. Die Größeordnung der z-Achse orientiere sich dabei laut der Dokumentation an der x-Achse. Die World Coordinates haben den Nullpunkt des gesamten Koordinatensys-

tems im Mittelpunkt der Hüften. Die Werte dieser Koordinaten sind jedoch in Metern. In den Koordinatensequenzen findet sich Zittern der Koordinaten, das in seiner Intensität schwankt und besonders bei schlechter sichtbaren Keypoints stärker auftritt.
Die Skelett-Daten beinhalten einige Ungenauigkeiten mit unterschiedlicher Relevanz für diese Arbeit. Zu den weniger relevanten Verhalten zählt zum Beispiel, dass MediaPipe Pose annimmt, dass die Beine an den Torso angezogen sind, wenn sie nicht sichtbar sind. Auffällig ist auch, dass die Skellette eine Schräglage in Richtung der Kamera aufweisen. Diese variiert zwischen ungefähr 90° und 60° zum Boden und wird meist kleiner, d.h. der Winkel zum Boden nähert sich 90°, wenn mindestens ein Arm angehoben wird. Da diese Schräglage keinen erkennbaren Einfluss auf das entwickelte System hat, wurde ihr Winkel nicht genauer vermessen.
Die vielleicht relevanteste Verzerrung findet sich in den Image Coordinates. In diesen werden die Abstände zwischen den Keypoints in der Bildebene mit zunehmender Entfernung zur Kamera erwartungsgemäß kleiner. Entlang der z-Achse hingegen werden die Körperteile hingegen auf jeweils konstante Werte geschätzt, die nicht von der Größe der Person im Bild beeinflusst wird. Dies hat zum Effekt, dass z.B. die Schulterbreite der Person variiert während sie sich auf der Stelle dreht. Während die Schultern parallel zur Bildebene ausgerichtet sind, die Person also Richtung Kamera steht, entspricht die Schulterbreite ihrem perspektivischen Ausmaß. Wenn sich der Person aber dreht und die Schultern parallel zur Bildachse ausrichtet, entspricht die Breite einem konstanten Wert.

2.2.1 Ortsschätzung

Da der MediaPipe Pose Output keine Informationen zum Ort der Person im Raum beinhaltet, müssen diese separat errechnet werden. Die MediaPipe Sammlung beinhaltet MediaPipe Iris Vakunov und Lagun [24], ein Modell zur Erkennung der Iris des Auges in Bildern/Videos. Dieses ist auch in der Lage die Distanz zwischen der Person zur Kamera zu bestimmen indem der zweite Strahlensatz 2.1 auf der Größe der Iris angewendet wird. Mit dem 2. Strahlensatz lässt sich aus der Brennweite f sowie der realen Größe w_r und der Größe im Bild w_p eines Objekts die Entfernung d bestimmen. Dieser Ansatz erfordert natürlich, dass die reale Größe des verwendeten Referenzwerts bekannt ist und, dass der Referenzwert konstant im Bild sichtbar ist. Zudem beeinflussen Verzerrungen des Referenzwert die errechnete Entfernung.

$$\frac{f}{w_p} * w_r = d \tag{2.1}$$

2.3 Gestenerkennung

2.3.1 Beispielbasierte Methoden

Es gibt verschiedene Möglichkeiten Bewegungen zu erfassen und zu klassifizieren. Darunter zum Beispiel die Auswertung elektromyographischer Daten d.h. Informationen über die elektrische Muskelaktivität, über Support Vector Machines [4] oder CNN-LSTMs [9] zu verarbeiten. Auch Videodaten können mit CNNs ausgewertet werden.

2.3.2 Modellbasierte Methoden

Der Dynamic Time Warping Algorithmus [16], der ursprünglich zur Spracherkennung entwickelt wurde, dient zur Mustererkennung in sequentiellen Daten. Dabei wendet DTW eine nichtlineare Zeitnormierung an, um Muster trotz stellenweiser Geschwindigkeitsunterschiede erkennen zu können. Damit eignet sich der Algorithmus auch für die Erkennung von dynamischen Gesten, da diese ebenfalls zwischen Personen unterschiedlich schnell ausgeführt werden und sich diese Differenz nicht gleichmäßig über die gesamte Geste erstreckt.

Aus den bestehenden Ansätzen zur Gestenerkennung über DTW lassen sich eine Reihe von allgemeinen Problemen und ihren Lösungen ermitteln. Das erste dieser ist, dass die Keypoint Positionen oft relativ zum Ort des Sensors ermittelt werden. Damit weisen die Koordinaten innerhalb der selben Gesten große Unterschiede auf, wenn die ausführende Person an einem anderen Ort relativ zum Sensor steht. Indem die Keypoints in einen auf der Person zentrierten Rahmen versetzt werden lässt sich dies jedoch beheben [18, 5, 15]. Da Verschiebung durch Addition eines Vektors realisiert wird, kann der Ortsvektor eines Keypoints von allen anderen Keypoints subtrahiert werden um jeweils die relative Position zu diesem zu erhalten.
Schneider u. a. [18] skalieren die Skelette zusätzlich um den Einfluss der Körpergröße und der Distanz zu eliminieren. Dazu werden alle Koordinaten nachdem sie in Relation zur Person gestellt wurden durch die Distanz zwischen den Schultern geteilt werden womit

dieses Distanz 1 wird. Dabei ist anzumerken, dass die Schultern hier gewählt wurden, da die Breite der Schultern durch die Knochenstruktur konstant ist und angenommen wird, dass sie immer frontal zur Kamera ausgerichtet sind. Ansonsten würde die Distanz in der Projektion auf die Bildebene nicht gleichmäßig mit der Gesamtgröße variieren und eine akkurate Skalierung wäre nicht möglich.
Anstelle anzunehmen, dass die Person immer frontal zur Kamera ausgerichtet ist, stellen Bodiroža u. a. [5] ebenfalls Rotations-Invarianz her indem das Skelett rotiert wird, dass sie an der Hüfte gemessen in die selbe Richtung orientiert sind.
Das Zittern der Keypoints stellt ebenfalls eines der übergreifenden Probleme dar. Um die eigentlichen Bewegungen aus dem Zittern zu extrahieren werden Glättungsfilter wie der Gauß- [18, 23] oder der Butterworth-Filter [13] eingesetzt.
Da alle Koordinaten der Pose-Keypoints mindestens 2 Dimensionen haben und ein einzelner Keypoint nicht zwingend zur Klassifikation ausreicht, muss die Anwendung des DTW Algorithmus dies zumindest in der Distanzfunktion berücksichtigen. Die Euklidische Distanz stellt die Distanz zwischen einzelnen Punkten im Raum am akkuratesten dar und wird dementsprechend verwendet [13]. Sobald jedoch die Elemente der Zeitsequenzen aus den Koordinaten multipler Keypoints zusammengesetzt sind, verzerrt die Euklidische Distanz die Zusammenhänge zwischen den Elementen. Dazu kommt, dass der Informationsgehalt in den verschiedenen Dimensionen sich zwischen den Gesten unterscheidet. Um den Informationsgehalt zu berücksichtigen, kann eine dynamische Auswahl der berücksichtigten Dimensionen getroffen werden.
Für Ten Holt u. a. [23] ist diese Auswahl notwendig, da sie jede Dimension auf einen Mittelwert von Null und Einheitsvarianz normalisieren. Die Normalisierung führt jedoch auch dazu, dass wenn eine Dimension nur Rauschen enthält, dieses verstärkt wird. Ohne die Normalisierung könnte jedoch eine Dimension mit einem größeren Wertebereich die DTW-Distanz, welche mit der Manhattan Metrik über alle Dimensionen berechnet wird, dominieren. Deshalb werden alle Dimensionen, deren Varianz unter einem Grenzwert liegt, aussortiert.
Schneider u. a. [18] wenden den DTW Algorithmus hingegen auf alle Dimensionen separat an und bestimmen dann die durchschnittliche Distanz aller Dimensionen. Sie können somit die Komplexität verringern, indem sie die Anzahl verwendeter Dimensionen reduzieren. Die Auswahl basiert ebenfalls auf einem Varianz Grenzwert, den die Dimensionen entweder im Muster oder in der Sequenz überschreiten müssen. Eine dynamische statt einer statischen Auswahl hat den Vorteil, dass auch Gesten die sich Bewegungen teilen akkurat klassifiziert werden können. So könnte bei einer statischen Auswahl, die nur die Mustersequenz berücksichtigt, "Winken mit der linken Handëinen besseren Score er-

halten als "Winken mit der beiden Händenäuch wenn letzteres ausgeführt wird. Indem dynamisch ausgewählt wird, kann in diesem Beispiel auch im Vergleich mit der "Winken mit der linken HandMustergeste die Differenz zwischen der still gehaltenen und der winkenden rechten Hand einfließen.
Nach DTW lässt sich per 1-Nearest-Neighbour die zutreffendste Geste auswählen [23, 18].

2.4 Identifikation des selektierten Objekts

Das Wissen, das eine Selektion stattgefunden hat, ist allein von geringen Nutzen, wenn nicht ebenfalls bekannt ist welches Objekt selektiert wurde. Da das Ziel des Zeigens sich vom Zeigenden aus in der durch das Zeigewerkzeug angegebenen Richtung befindet, handelt es sich hier um ein Problem der Geometrie. Die Position des Zeigenden muss dafür mit den Positionen der möglichen Zielobjekte und der Zeigerichtung in Relation gebracht werden.
In VR Systemen besteht ebenfalls das Problem den Nutzer zwischen verschiedenen Objekten in 3D Umgebungen wählen zu lassen, wofür verschiedene Cursor entwickelt wurden. Während VR Cursor hilfreiche Metriken zur Identifikation des beabsichtigten Ziels beinhalten, lassen sie sich nicht vollständig auf Smart-Home-Gestensteuerung übertragen, da ohne erheblichen extra Aufwand keine 3D Visualisierung möglich ist. Lu u. a. [10] vergleichen verschiedene VR Cursor mit ihrer Implementation eines dreidimensionalen Bubble Cursor. Zu beachten ist, dass die VR Cursor nicht nur in der Existenz einer Visualisierung vom Anwendungsgebiet dieser Arbeit abweichen, sondern auch darin, dass ein Tastendruck auf einem Controller anstelle einer Geste verwendet wird um die Selektion auszulösen. So kann der Nutzer zuerst sicherstellen, dass das korrekte Objekt im Auswahlbereich ist bevor die Selektion ausgelöst sind. Der Bubble Cursor ist mechanisch auch darauf ausgelegt diese Verifizierung zu unterstützen indem der Auswahlbereich angepasst wird um optisch immer nur ein Objekt umschließt. Indem so die Verifizierung erleichtert wird, wird auch die Selektion im gesamten gestärkt. Die Ergebnisse lassen jedoch Rückschlüsse darauf zu, welche Auswahl-Metriken die Erwartungshaltung des Nutzers am besten widerspiegeln.
Während der Naive Ray bereits die höchste Fehlerrate aller Verglichener Techniken aufweist, ist davon auszugehen, dass diese noch weiter ansteigt, wenn der Nutzer ohne Visualisierung nicht mehr dazu in der Lage ist den spezifischen Strahl zu sehen. Die Techniken,

die das Objekt wählen welches im kleinsten Volumen um den Zeigestrahl liegt weisen geringere Fehlerraten und auch bessere subjektive Bewertung der Nutzer auf. Zwischen den zwei Varianten des Bubble Cursor, von denen je eine die euklidische Distanz und eine die Winkeldistanz zwischen Zeigestrahl und Zielobjekt verwendet, ist die Performance in fast allen Metriken zwar ähnlich jedoch leicht zugunsten der Winkeldistanz gewichtet. In einem Testszenario indem die Ziele in verschiedenen Distanzen grob hintereinander angeordnet waren, bevorzugte die euklidische Distanz die vorne gelegenen Zielobjekte, womit sich die Winkeldistanz in ähnlichen Situationen besser eignet.

2.5 Zielsetzung

Das in dieser Arbeit entwickelte Selektionserkennungs-System sollte folgende Eigenschaften erfüllen:

- Selektionen sollen zuverlässig erkannt werden.
- Um den Nutzers wahrzunehmen soll eine einzelne RGB-Kamera verwendet werden.
- Die Selektionserkennung soll im Rahmen eines Smart Home einsetzbar sein.
- Die Selektionserkennung soll in Echtzeit laufen.
- Eine Selektion sollte innerhalb einer Sekunde erkannt und registriert werden.

Folgende Annahmen werden gemacht:

- Zwischen der Selektionserkennung und der restlichen Gestensteuerung besteht eine zuverlässige Verbindung.
- Alle steuerbaren Objekte sind so positioniert sind, dass der Nutzer nie nach hinten zeigen muss, da die Richtung des Zeigens sonst vom Körper des Nutzers verdeckt wird.
- Bei den steuerbaren Objekten handelt es sich um Objekte in der Größenordnung eines Fensters. D.h. ihr optischer Schwerpunkt befindet sich zwischen 0,8m und 1,6m über dem Boden mit mindestens 1,5m zwischen zwei Objekten.
- Aus technischen Einschränkungen 3.5.1 wird davon ausgegangen, dass der Nutzer beim Ziegen frontal zur Kamera steht.

3 Lösungsansatz

In diesem Kapitel wird das Design des Lösungsansatzes beschrieben. Zuerst wird das Gesamtkonzept beschrieben, gefolgt von der jeweiligen Methodik um die Teilprobleme Gestenerkennung, Identifikation des Ziels und Pose Estimation zu lösen. Zuletzt wird auf im Vorfeld notwendige Setup Operationen eingegangen.

3.1 Gesamtkonzept

In Relation zu seiner Umwelt 3.1, hat das System zwei Kontaktpunkte. Auf der einen Seite ist der Nutzer, der per Kamera wahrgenommen wird und eine Selektion per Geste kommuniziert. Auf der anderen Seite steht der `Manipulation Controller`, der das System beschreibt, welches den Manipulations-Befehl des Nutzers ausführt und dafür den Kontext aus der Selektion benötigt. Da sich diese Arbeit auf die Erkennung der Selektion beschränkt, ist der `Manipulation Controller` außerhalb des Scopes und wird dementsprechend gemockt.

Nachdem alle notwendigen Daten geladen wurden, erfolgt die eigentliche Selektions-Erkennung in einem zyklischen Ablauf 3.2. Um Selektionen in Echtzeit zu erkennen werden kontinuierlich Bilder von der Kamera gelesen und verarbeitet. Aus diesen Bildern

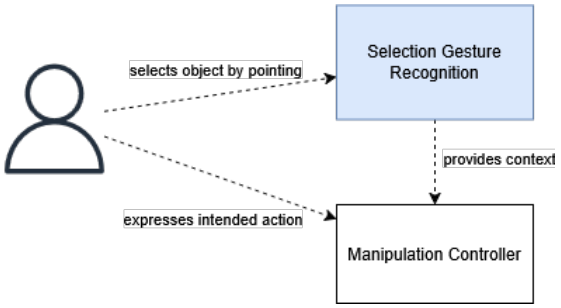

Abbildung 3.1: Systemkontextdiagramm

wird zuerst die Haltung des Nutzers mittels Pose Estimation als Keypoints extrahiert. Da mit DTW ein Klassifikationsalgorithmus verwendet wird, der vollständige Sequenzen statt je einzelnen Elementen benötigt, müssen die Keypoints zwischengespeichert werden. Sobald die Sequenz im Buffer die Ziellänge erreicht hat, kann klassifiziert werden ob eine Selektionsgeste enthalten ist oder nicht.
Da davon auszugehen ist, dass die Gesten nicht erkannt werden, wenn nur ein Bruchstück in der Sequenz ist, sollte der Buffer nach der Klassifikation nicht vollständig entleert werden. Stattdessen sollten nur die ältesten Elemente entfernt werden. Indem also überlappende Sequenzen klassifiziert werden, kann sichergestellt werden, dass jede Selektion mindestens einmal vollständig in der Sequenz enthalten ist. Womit die Chance, dass sie korrekt erkannt wird maximiert wird. Die Ziellänge der Sequenz sollte sich an der durchschnittlichen Länge der Zeigegesten orientieren, diese kann zusammen mit der Mustergeste im Setup ermittelt werden. Die Anzahl der Elemente, die jeweils nach der Klassifikation entfernt werden, kann variiert werden. Eine größere Anzahl verringert die Anzahl Klassifikationen, die in der selben Zeit ausgeführt werden müssen und reduziert damit den Rechenaufwand. Der Preis dafür ist jedoch, dass die Chance, dass die Selektionen je nie vollständig in einer Sequenz auftauchen ansteigt. Eine kleinere Anzahl hat dementsprechend den gegenteiligen Effekt.
Wenn die Klassifikation keine Selektionsgeste findet, kann mit dem nächsten Frame fortgefahren werden. Ansonsten muss ermittelt werden welches Objekt selektiert wird.
Bei der Identifikation des Zielobjekts muss berücksichtigt werden, dass auch wenn die Gestenerkennung perfekt funktioniert nicht jede erkannte Zeigegeste zur Steuerung gedacht ist. Da die Klassifikation nicht in der Lage ist diese Fälle auszusortieren, werden weitere Kriterien benötigt. Eines dieser Kriterien ist, dass der Nutzer auf keines der steuerbaren Objekt zeigt. Ebenfalls lässt sich davon ausgehen, dass die Geste nicht zur Steuerung gedacht ist, wenn auf sie kein Manipulationsbefehl folgt. Da die Manipulation jedoch erst im `Manipulation Controller` berücksichtigt wird, kann dieses Kriterium erst in diesem verwendet werden. Indem zum Beispiel die Selektion nach einem Timeout ohne Manipulation wieder verfällt.
Wenn auch ein Zielobjekt erkannt wird, muss zuletzt noch die Selektion an den `Manipulation Controller` übergeben werden, bevor wieder mit dem nächsten Frame fortgefahren wird.

Aus den drei großen Arbeitsschritten leiten sich drei der sieben Komponenten 3.3 direkt ab. `targeting` umfasst die Identifikation des Zielobjekts und `pose_estimation` die Pose Estimation aus den Bilddaten. Während die `classifier` für die Klassifikation

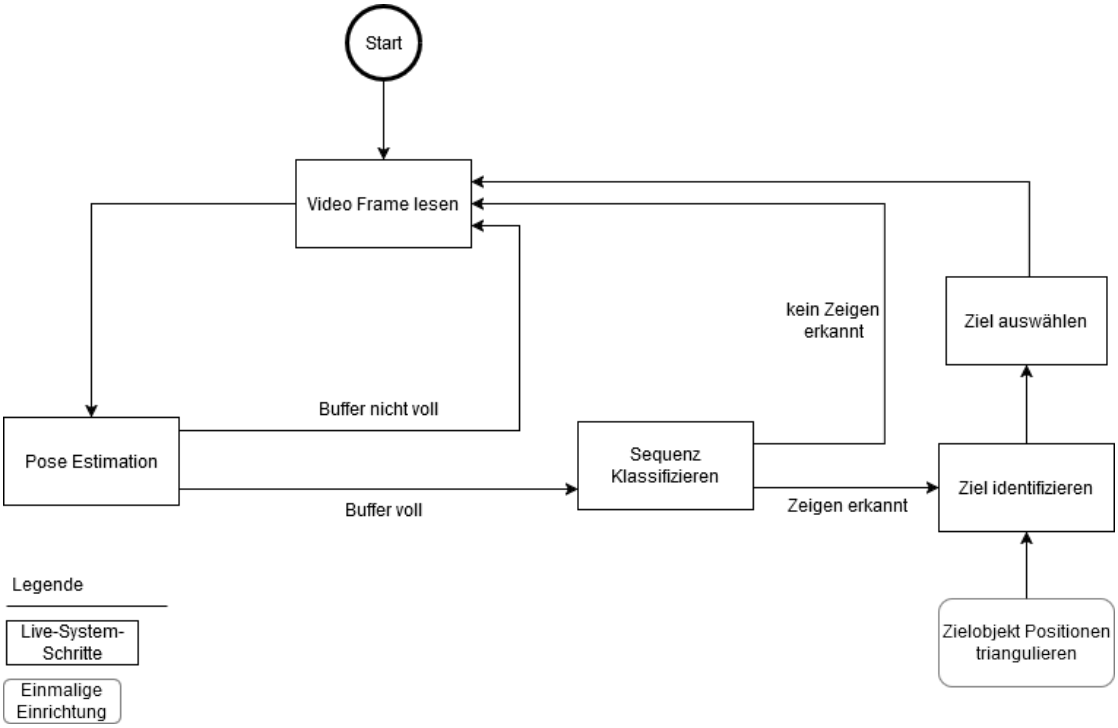

Abbildung 3.2: Flow Chart des primären Schleife

der Sequenzen verantwortlich ist, nehmen die dazu notwendigen Vorverarbeitungsschritte mit `pre_processing` eine eigene Komponente ein. Die Komponente `setup` umfasst die Schritte im Vorfeld der Selektionserkennung um notwendige Daten wie die Orte der steuerbaren Objekte zu ermitteln. `selection_recognition` beinhaltet das `main`-Script und ist damit für die in 3.2 beschriebene Schleife zuständig. Sie verknüpft die Funktionalitäten der restlichen Komponenten zur vollständigen Selektionserkennung. In `data_writer` finden sich Funktionen um relevante Informationen in Dateien zu speichern. Diese umfassen sowohl Performance Daten zur Auswertung als auch Pose Daten in PoseViz (.pose) Dateien. Letztere vereinfachen die Entwicklung, da Pose Informationen aus diesen zu lesen bedeutend schneller ist als sie jedes mal per Pose Estimation aus Videos zu extrahieren. Diese beiden Zwecke tragen nicht zur eigentlichen Selektionserkennung bei, weshalb die Komponente als nicht-essentiell gilt.

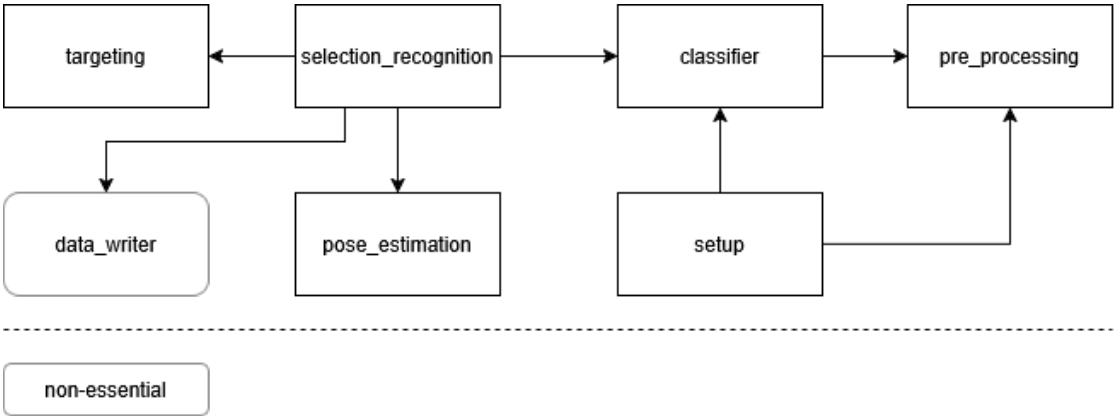

Abbildung 3.3: Komponentendiagramm
Von den sieben Komponenten ist `data_writer` als nicht-essentiell gekennzeichnet, da sie nur Funktionen enthält die die Entwicklung und Auswertung vereinfachen, nicht jedoch zur Selektionserkennung notwendig sind.

3.1.1 Implementation

Der Prototyp wird in Python 3.10 implementiert. Das `main`-Script `selection_recognition/selection_recognition.py` implementiert die primäre Schleife 3.2.
Diese Schleife arbeitet bis das Ende der Datenquelle erreicht ist. Da das System auch damit umgehen können muss, dass keine Person im Sichtfeld der Kamera ist, muss dies berücksichtigt werden. Der Dynamic Time Warping (DTW) Algorithmus kann nur mit leeren Frames umgehen, wenn eine Distanzfunktion verwendet wird, die explizit darauf ausgelegt ist nit diesen umzugehen. Stattdessen setzt die Schleife die Gestenerkennung aus, wenn keine Person erkannt wird. Der verbliebene Inhalt des Buffers wird dafür klassifiziert und entleert sobald die Person das Sichtfeld verlässt.
Über das gesamte System wird eine gleichmäßige Repräsentation der Pose Sequenzen verwendet. Dabei handelt es sich um drei dimensionale Arrays. Die erste Dimension umfasst die einzelnen Frames der Sequenz. Die zweite Dimension beinhaltet die Keypoints und die letzte Dimension die Achsen der Koordinaten.

3.2 Manipulation Controller Mock

Die einzige Funktion, die der Mock bieten muss, besteht darin eine Selektion anzunehmen. Da es sich um ein Echtzeit System handeln soll, kann angenommen werden, dass der

einzige relevante Zeitpunkt einer Selektion "jetztïst. Somit lässt sich der Zeitpunkt der Selektion implizit durch den Zeitpunkt des Funktionsaufruf angeben. Welches Objekt Selektiert wird ist der zweite Bestandteil einer Selektion und kann per Objekt ID als Argument übergeben werden.

3.3 Gestenerkennung

Zur Erkennung der Zeigegeste wird ein modellbasierter Ansatz anstelle eines Beispielbasierten verwendet, da es sich bei letzteren um Blackboxen handelt. Modellbasierte Ansätze erlauben es deutlich einfacher die Performance nachzuvollziehen und gezielt anzupassen. Spezifisch wird er DTW Algorithmus verwendet, weil bereits gezeigt wurde, dass sich dieser gut zur Gestenerkennung auf Pose Daten eignet 2.3.2.
Anstatt DTW auf den Koordinaten der Keypoints auszuführen, werden Winkel verwendet. Dabei werden sowohl Winkel zwischen Körperteilen und den Achsen des Raums verwendet, welche die Orientierung dieser Körperteile angeben, als auch Winkel an den Gelenken zwischen den Körperteilen. Da Knochen steif sind, lässt sich die Haltung des gesamten Körpers durch die Ausrichtung der Knochen zueinander, d.h. den Winkeln der Gelenke, beschreiben. Der Vorteil der Winkel besteht darin, dass bei der selben Haltung die Winkel gleich sind auch wenn zwei unterschiedlich große Personen sie einnehmen. Auch der Standort und Rotation, die die Koordinaten beeinflusst haben keinen Effekt auf die Winkel. Während die Winkel also anders als die Koordinaten separat berechnet werden müssen, sind sie von Natur aus Translation, Scale, und Rotation Invariant, was die notwendige Vorverarbeitung reduziert.
Zur Erkennung der Geste liefert nur eine begrenzte Teilmenge der 33 MediaPipe Keypoints relevante Informationen. Anders als bei Schneider u. a. [18] sollen nur Zeigegesten erkannt werden, womit die involvierten Körperteile im Verhältnis uniform sind. Anstelle einer dynamischen Auswahl relevanter Keypoints wird deshalb auf statische Filter gesetzt, da immer die selben Keypoints relevant sind.
Sowohl der Butterworth als auch der Gauß Filter kommen zur Glättung in Frage. In Tests 3.4 hat sich gezeigt, dass beide Filter sehr ähnliche Ergebnisse in der Glättung zeigen. Der Butterworth Filter fügt jedoch eine kurze Verzögerung von $\approx \frac{1}{6}s$ hinzu, weshalb der Gaußsche Filter verwendet wird.

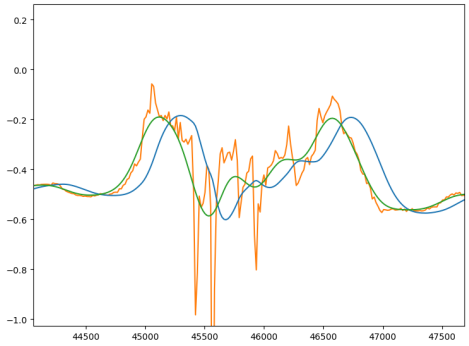

Abbildung 3.4: Visualisierung der Glättungsfilter
Graph der x-Koordinate des rechten kleiner-Finger-Keypoint (Nr. 18 in MediaPipe). Der unveränderte Mediapipe Output ist in Orange, in Blau ist die mit Butterworth gefilterte Sequenz und in Grün ist die Gaußsche gefilterte Sequenz

Da mehr als ein Winkel im DTW verwendet werden, ist die Sequenz multidimensional. Die einzelnen Winkel haben keine signifikante Abhängigkeit zueinander und sollten dementsprechend als absolute Differenz pro Winkel in die Distanz einfließen. Deshalb wird die Manhattan Distanz als Distanzfunktion verwendet.
Während es für die Gestenerkennung ebenfalls möglich wäre, wie bei Schneider u. a. [18] auf allen Winkeln separat DTW zu verwenden und die durchschnittliche Distanz zu bestimmen, erhält man bei dieser Methode keinen eindeutigen Warping Path, sondern je einen unterschiedlichen pro Winkel. Da jedoch der Warping Path benötigt wird um zu ermitteln in welchem Frame auf das Ziel gezeigt wird, ist dieser Ansatz keine Option.

Zur Gestenerkennung wird die `DTWClassifier` Klasse in der `classifer` Komponente definiert. Sie hat abseits des Konstruktors eine öffentliche Methode.

- `__init__(gestures, kp_filter, dist_function, smoothing_filter, threshold)` Der Konstruktor hat fünf Parameter. In `gestures` wird eine Liste mit allen Mustersequenzen übergeben. Auf die Mustersequenzen werden die selben Vorverarbeitungsschritte angewendet, die auch während der Klassifikation verwendet werden, um die Vergleichbarkeit zu gewährleisten. `kp_filter` beschreibt die statische Auswahl Keypoints die in den Vergleich einfließen. `dist_function` ist die verwendete Distanzfunktion und `smoothing_filter` der verwendete Glät-

tungsfilter. `threshold` ist der Distanz-Grenzwert über dem davon ausgegangen wird, dass keine Geste enthalten ist.

- `classify(sequence)` Die Kernmethode der Gestenerkennung, klassifiziert die Sequenz und gibt die Id der Mustergeste mit der geringsten DTW Distanz zur Sequenz, eine Liste mit den Distanzen zu allen Mustergesten und den Warping Path der Geste mit der geringsten Distanz zurück. Wenn zu keiner der Mustergesten die Distanz kleiner als `threshold` ist, wird sowohl für die Gesten Id als auch den Warping Path `None` zurückgegeben. Die Distanzen zu allen Mustergesten werden zurückgegeben um das Sammeln von Evaluationsdaten zu erleichtern.

Buffer

Da DTW nur vollständige Sequenzen vergleichen kann, der Kamerafeed aber nur einzelne Frames nacheinander liefert, müssen diese in einem Buffer zwischengespeichert werden bis eine ausreichend Große Sequenz erreicht ist. Um die Frames weiterhin in chronologischer Reihenfolge zu bearbeitet, muss der Buffer dem FIFO Prinzip folgen. Von einer simplen Queue unterscheidet sich der Buffer daher, dass statt dem ersten Element die ersten n Elemente als Sequenz gelesen werden. Zudem werden nicht zwingend alle n gelesenen Elemente aus dem Buffer entfernt, sondern nur die ersten m damit die Sequenzen überlappen können. Die Klasse `WindowBuffer` hat mit dem Konstruktor die folgenden fünf Methoden.

- `__init__(window_size, step)` Mit `window_size` wird die Länge der gelesenen Sequenz n beschrieben also die Fenstergröße. `step` beschreibt die Schrittweite, also die Anzahl Elemente die beim Lesen entfernt werden m.
- `add(item)` Diese Methode fügt das Element `item` am Ende des Buffers hinzu.
- `get_window()` Über diese Methode wird vom Buffer gelesen. Wie beschrieben werden die ersten n Elemente zurückgegeben und die ersten m entfernt. Wenn weniger als n Elemente im Buffer enthalten sind, werden diese zurückgegeben. Dies beinhaltet eine leere Sequenz, wenn der Buffer leer ist.
- `has_full_window()` Diese Methode gibt zurück ob der Inhalt des Buffers mindestens so lang ist wie die Fenstergröße n ist. Wenn dies nicht der Fall ist wird beim lesen eine Sequenz zurückgegeben die kürzer als n ist.

- `clear()` Diese Methode entleert den Buffer vollständig ohne zu lesen.

3.3.1 Vorverarbeitung

Die vier Funktionen zur Vorverarbeitung finden sich in der `pre_processing` Komponente.

- `prepare(sequence, kp_filter, smoothing_filter)` Diese Funktion bündelt alle folgenden Vorverarbeitungs-Funktionen. Zuerst werden die irrelevanten Keypoints entfernt. Die Winkel werden dann auf den geglätteten Keypoints berechnet
- `filter_keypoints(kp_filter, sequence)` Diese Funktion gibt eine Kopie von `sequence` zurück in der nur die in `kp_filter` angegebenen Keypoints enthalten sind.
- `smooth_sequence(sequence, smoothing_filter)` Diese Funktion gibt eine mit dem Glättungs Filter `smoothing_filter` geglättete Kopie von `sequence` zurück. Die Glättung werden dabei nur innerhalb der einzelnen Dimensionen geglättet, auch wenn verwendete Filter, wie der Gauß Filter, mehrdimensionale Glättung ermöglichen.
- `calculate_angles(sequence)` Für jedes Element von `sequence` werden die Winkel zwischen den Keypoints des Elements errechnet und als neue Sequenz zurückgegeben. Dabei wird angenommen, dass jeweils die Gelenkwinkel einer Gliedmaße berechnet wird. Gliedmaßen lassen sich als Reihe von verbunden Gelenken verstehen. In der Keypoint Repräsentation einer derart abstrahierten Gliedmaße, sind die der erste und letzte Keypoint jeweils die Anfang und Ende der Gliedmaße. Jeder Keypoint dazwischen beschreibt eines der Gelenke. Der Gelenkwinkel für das Gelenk bei Keypoint K_i ist gegeben durch den Winkel zwischen den Vektoren von Keypoint K_i zu den Keypoints K_{i+1} und K_{i-1}

3.3.2 Implementation

Im Prototyp wird die Implementation des DTW Algorithmus aus dem fastDTW Python Package [22] verwendet. Während es sich hier eigentlich um eine Implementation des FastDTW Algorithmus von Salvador und Chan [17] handelt, kann diese so konfiguriert

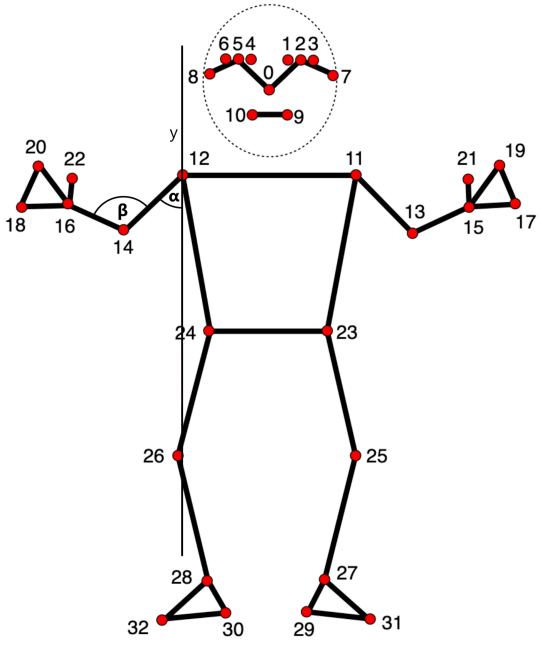

Abbildung 3.5: MediaPipe Körper Key Points
In den Keypoints der MediaPipe Pose Estimation sind die verwendeten Winkel eingezeichnet

werden, dass reguläres DTW ausgeführt wird. Diese Implementation wurde gewählt, da sie leicht erlaubt eigene Distanzfunktionen zu verwenden und sie das Potential bietet auf den FastDTW Algorithmus zu wechseln, falls eine schnellere Performance benötigt wird. Die verwendeten Keypoints und dementsprechend Winkel, werden werden der Config Datei importiert. Der Prototyp verwendet hier die Schulter, den Ellenbogen und die Hand. Um neben dem Ellenbogenwinkel noch den Hebewinkel des Oberarms zu verwenden, wird senkrecht unter dem Schulter-Keypoint ein Pseudokeypoint hinzugefügt 3.5. Der Pseudokeypoint ist eine Kopie des Schulterkeypoints mit erhöhtem y Wert. Die Winkel der Gelenke werden wie bei der Identifikation des Ziels mit der Formel 3.1 berechnet.

3.4 Identifikation des Ziels

Nachdem erkannt wurde, dass eine Zeigegeste zur Selektion getätigt wurde, muss erkannt werden auf welches Objekt gezeigt wurde. Dafür könnte man unterschiedliche Zeige-Mustersequenzen definieren, die jeweils das Ziel beinhalten um im selben Schritt die

Geste und das Ziel zu erkennen. Damit dieser Ansatz funktioniert muss jedoch die relative Position zwischen Subjekt und Objekt konstant sein. Dazu müsste das Subjekt immer am selben Ort stehen oder man müsste für jede Kombination aus Ort und Ziel eine Mustersequenz verwenden.
Stattdessen lässt sich die Identifikation des Objekts separat umsetzten indem basierend auf der Position des Subjekts und der Zeigerichtung errechnet werden, welches der möglichen Objekte gemeint ist.
Dafür muss zuerst identifiziert werden, an welchem Punkt in der Sequenz die volle Zeigehaltung eingenommen wurde und der Arm auf das Ziel zeigt. Ein Ansatz dafür besteht darin den Frame der Sequenz zu wählen, der die maximale Armstreckung aufweist. Dabei wäre jedoch problematisch, dass der in Ruheposition hängende Arm ebenfalls nahezu maximal gestreckt ist und, dass dies für alle Frames gilt sollte der gestreckte Arm von der Ruheposition aus der Schulter in die Zeigehaltung rotiert werden. Letzteres ist der Grund wieso ein mindest-horizontalitäts Grenzwert nicht ausreicht um die Frames der Zeigehaltung korrekt zu extrahieren. Ein solcher Grenzwert könnte den überlappenden Bereich zwischen *"leicht nach unten zeigen"* und *"kurz davor gerade zu zeigen"* nicht korrekt trennen.
Stattdessen lässt sich der beim DTW errechnete Warpingpath zur Identifikation verwenden. Der Warpingpath enthält die Information welches Element der Sequenz dem Element in der Mustersequenz entspricht, in dem die Zeigehaltung erreicht ist. Dies erfordert, dass dieser Punkt in der Mustersequenz bekannt ist, was pro Mustersequenz einmalig ermittelt werden muss und in dieser Arbeit manuell gemacht wird.
Um aus einem Frame das Ziel des Zeigens zu errechnen, können der Ausgangspunkt und die Richtung des Zeigen durch einen Strahl definiert werden, dessen Ausgungspunkt und Richtung durch je einen Pose Keypoint gegeben sind. Beim Zeigen mit dem gestreckten Arm, ist dies von der Schulter durch die Hand 3.6. Wenn andere Zeigevarianten verwendet werden, müssen andere Keypoints verwendet werden. Das könnte z.B. vom Ellenbogen durch die Hand oder auch vom Handgelenk zur Spitze des Zeigefingers sein. Solange nur Zeigestrahlen benötigt sind, die sich durch existierende Pose Keypoints definieren lassen, ist es möglich, die mit der erkannten Zeigevariante assoziierten, Keypoints dynamisch auszuwählen um immer den korrekten Zeigestrahl zu verwenden.

Wenn der Zeigestrahl bestimmt ist, kann mit diesem das Ziel identifiziert werden. Dazu könnte man prüfen ob der Zeigestrahl durch die Bounding Boxen der Ziele verläuft und dann das Ziel selektieren, das sowohl durchlaufen wird als auch dem Ursprung des Zei-

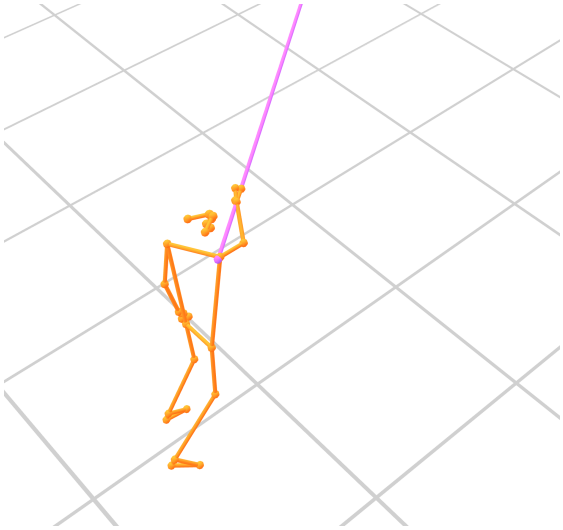

Abbildung 3.6: Visualisierung des Zeigestrahls von der Schulter durch die Hand

gestrahls am nächsten liegt.
Da jedoch alle räumlichen Daten aufgrund der Limitationen der Pose Estimation ungenau sind, wäre es besser eine Methode zu verwenden die dies berücksichtigt. Eine Option hierfür ist es nicht mit dem reinen Zeigestrahl selbst auf Kollisionen zu prüfen, sondern einen Kegel um den Zeigestrahl zu spannen und zu prüfen welche der Objekte sich in diesem befinden. Die Spitze und Achse des Kegels sind dabei durch den Ursprung und die Richtung des Zeigestrahls gegeben. Anders als bei der Verwendung eines Zylinders oder eines anderen Prisma als prüfendes Volumen berücksichtigt die Verwendung eines Kegels, dass Abweichungen in der Richtung des Strahls von der Richtung zum Ziel bei größeren Distanzen zu einer größeren Entfernung zum Strahl führen 3.7.

Wenn mit einem Volumen auf Kollision geprüft wird, ist es möglich die Zielobjekte auf Punkte zu abstrahieren. Das hat den Vorteil, dass weder ein komplexes Mesh noch die Bounding Box des Objekts benötigt wird. Während ein Rückgang der Genauigkeit möglich ist, wird in den meisten Fällen ungefähr auf die optische Mitte des Objekts gezeigt. Daraus ergibt sich ein Zeigeschwerpunkt des Objekts, dass wenn der Kegel ausreichend weit ist immer in diesem liegt. Die Abstraktion auf einen Punkt kann also ausgeglichen werden indem der Kegel weiter geöffnet wird.
Ob sich ein Punkt P in einem Kegel befindet lässt sich dadurch bestimmen ob der Winkel zwischen der Kegelachse und dem Vektor von der Spitze des Kegels zu P kleiner ist als der halbe Öffnungswinkel des Kegels. Bei einem Winkel mit begrenzter Höhe müsste

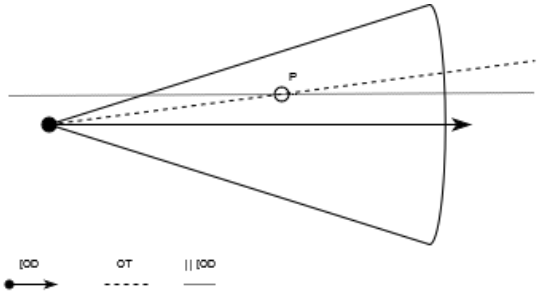

Abbildung 3.7: Zeigestrahl-Kegel Diagramm
Auf 2 Dimensionen reduzierte Abbildung des Zeigestrahls [OD und dessen umspannednen Kegels, sowie der Richtung zum Ziel OT. Die Parallele ∥ [OD ist eingezeichnet, um zu visualieren, dass ab P das Ziel bei Verwendung eines Zylinders nicht mehr erkannt werden würde, während der Abweichungswinkel zwischen OT und [OD gleich bleibt.

zwar noch bestimmt werden, ob der Punkt von der Spitze aus hinter der Grundfläche liegt, aber da von einem Zeigestrahl und keiner Zeigestrecke ausgegangen wird, ist dieses Kriterium unwichtig. Der halbe Öffnungswinkel ist bei dieser Methode die Stellschraube um den Grad der Toleranz an die bestehende Ungenauigkeit anzupassen.

Zur Zielerkennung wird die Klasse `Targeting` definiert, die neben dem Konstruktor eine Methode hat.

- `__init__(targets, angle)` Da die Menge der steuerbaren Objekte während der Laufzeit konstant bleibt, kann sie in `targets` dem Konstruktor übergeben und als Attribut gespeichert werden. `angle` beschreibt den halben Öffnungswinkel des Kegels, der um den Zeigestrahl gespannt wird.
- `find_target(origin, direction)` Diese Methode erhält den Zeigestrahl in Form von zwei Punkten. Der Strahl beginnt am Punkt `origin` und verläuft durch `direction`. Die Methode gibt die ID des Zielobjekts zurück, welches sowohl im Kegel um den Zeigestrahl liegt und den kleinsten Winkel zu diesem aufweist. Wenn keines der Objekte im Kegel liegt, wird `None` zurückgegeben.

3.4.1 Implementation

Der Winkel zwischen der Kegelachse und dem Vektor von der Spitze des Kegels zum Punkt, lässt sich wie alle Winkel zwischen zwei Vektoren mit der Formel 3.1 berechnen. Da diese Formel sowohl zur Identifikation des Ziels als auch in der Gestenerkennung verwendet wird, wird die Funktion `calculate_angle(vector_a, vector_b, get_degrees` außerhalb der beiden Komponenten in der Datei `util.py` implementiert. `get_degrees` gibt an ob der Winkel als Cosinus Wert oder in Grad zurückgegeben wird.

$$\cos\alpha = \frac{\vec{a} \cdot \vec{b}}{|\vec{a}| \times |\vec{b}|} \tag{3.1}$$

3.5 Pose Estimation

Während die 2D Pose Estimation von Modellen wie OpenPose zur Gestenerkennung ausreicht, benötigt die Identifikation des Ziels die dreidimensionale Haltung zur geometrischen Analyse. OpenPose benötigt zur 3D Pose Estimation mehr als eine Kameraperspektive und ist damit nicht für dieses Projekt geeignet. MediaPipe gibt die Haltung in 3D an, vernachlässigt jedoch die Position im Raum. Obwohl die Position noch zusätzlich errechnet werden muss, erfüllt MediaPipe die meisten Anforderungen und wird deshalb verwendet.
Der MediaPipe Landmarker hat drei Betriebsmodi. Einen für Bilder und je einen synchronen und einen asynchronen für Videos. Der in dieser Arbeit erstellte Prototyp verwendet den synchronen Videobetriebsmodus, da sich dieser simpler in den Programmfluss einfügen lässt.
Um Haltungsdaten aus Videos und PoseViz-Dateien zu lesen sowie in der Lage zu sein das verwendete Pose Estimation Modell zu wechseln ohne Änderungen an der restlichen Projektstruktur vorzunehmen, wird das `pose_loader` Interface definiert. Dieses besteht aus fünf Funktionen.

- `get_pose_data_source(source)` Diese statische Funktion identifiziert die Art der Quelle und gibt eine initialisierte Instanz der zugehörigen Implementation zurück.

- `get_next_frame()` Diese Methode gibt die Haltung der Person im nächsten Frame in Form einer Mx3 Matrix zurück. Die Größe de Matrix ergibt sich aus der Anzahl Keypoints der verwendeten Pose Estimation und den drei räumlichen Achsen der Koordinaten.
- `is_opened()` Diese Methode gibt zurück ob von der Quelle weitere Frames gelesen werden können. Sie gibt False zurück, wenn entweder das Ende der Quelle erreicht ist oder sie geschlossen wurde. Sie wird als Ausstiegskondition in der Primären Schleife verwendet.
- `close()` Diese Methode schließt die Quelle um verknüpfte Ressourcen wie Kameras wieder freizugeben.
- `get_ms_per_frame()` Diese Methode gibt den Zeitabstand zwischen den Frames in Millisekunden zurück.

In der Implementation wird die eigentliche Pose Estimation in `get_next_frame()` über eine Instanz der `MPLandmarker` Klasse ausgeführt. `MPLanddmarker` ist ein Wrapper für die MediaPipe eigenen Objekte um die Umwandlung zwischen MediaPipe und Projekt Datentypen zu kapseln.

3.5.1 Ortsschätzung

Der Ort der Person kann aus der Größe und Position im Bild bestimmt werden. Diese lassen sich aus MediaPipes Image Coordinates errechnen. Wie bereits in der Analyse 2.2.1 beschrieben, kann der 2. Strahlensatz dazu verwendet werden die Distanz der Person zur Kamera d_{cp} zu bestimmen. Diese Distanz beschreibt die z Koordinate des Orts der Person. Dazu wird eine bekannte Größe benötigt, die in der realen Welt konstant groß bleibt und zumindest zum Zeitpunkt der Selektion sichtbar ist. Es wurde keine Größe gefunden die immer gut sichtbar ist. Die Körpergröße/Höhe scheidet aus, weil die Füße leicht hinter Möbeln verdeckt sind. Die Arme die zwar beim Zeigen zwingend sichtbar sein müssen, sind oft Richtung Kamera gestreckt und so maximal perspektivisch verkürzt. Die Wahl fällt deshalb auf die Schulterbreite, da diese zumindest wenn die Person zum Zeigen frontal zur Kamera steht, gut sichtbar sind.
Da Wohnräume in der Regel einen ebenen Boden haben, wird angenommen, dass sich der Nutzer auf einer konstanten Höhe bewegt, die nicht näher geschätzt werden muss.
Zuletzt wird noch die links/rechts Abweichung von der Bildachse, also die x Koordinate

des Orts benötigt. Gunerli [7] bestimmt diese indem zuerst das Verhältnis d_{pp} zwischen Pixeln im Bild und der realen Entfernung bei der errechneten Distanz d_{cp} bestimmt. Sie wird durch die Funktion 3.2 bestimmt. Dazu wird ebenfalls der FOV-Winkel der Kamera FOV und die Bildbreite in Pixeln w_i benötigt.

$$d_{pp} = \frac{tan(\frac{FOV}{2}) * 2 * d_{cp}}{w_i} \tag{3.2}$$

d_{pp} kann dann mit der horizontalen Distanz im Bild zwischen der Person und dem Mittelpunkt des Bildes in Pixeln multipliziert werden um die reale horizontale Verschiebung d_{ap} zu bestimmen.
Der resultierende Ortsvektor $\begin{pmatrix} d_{ap} \\ 0 \\ d_{cp} \end{pmatrix}$ kann auf die World Coordinates addiert werden um Haltung und Position zu vereinen. Die World Coordinates sind dafür besser geeignet, da sie anders als die Image Coordinates nicht die Bildposition enthalten und sie unabhängig der Distanz der Person gleich groß sind.

3.5.2 Implementation

Die Bilder werden mit der opencv-python Bibliothek geladen und mit der 'light' Variante des MediaPipe Landmarkers verarbeitet. Die Ortsschätzung verwendet die Implementation von Gunerli [7]. Anstatt die Brennweite aus der Auflösung und dem Sichtfeld (FOV) zu berechnen, wurde sie per Kamera Kalibrierung ermittelt.

3.6 Setup

Die Identifikation des Ziels benötigt die Koordinaten der möglichen Ziele und die Gestenerkennung benötigt Mustergesten, die die Gesten jeweils gut repräsentieren. Diese zu ermitteln ist Auffgabe der Kompnente `setup`.

3.6.1 Koordinaten der Zielobjekte

Die simpelste Methode um zu ermitteln wo sich die Zielobjekte befinden besteht darin den realen Raum und damit die Koordinaten manuell auszumessen. Dieser Ansatz funk-

tioniert jedoch nur, wenn die Pose Estimation die räumlichen Dimensionen zuverlässig abbildet. Wenn aber der virtuelle Raum eine andere potentiell unbekannte Größe besitzt, schlägt dies fehl. Es ist jedoch auch gar nicht notwendig die realen Koordinaten der Objekte zu kennen, da es ausreicht repräsentative Koordinaten zu haben, die bei der Identifikation des Ziels korrekte Ergebnisse liefern. Es kann also aus einer Menge Zeigestrahlen zu einem Objekt eine repräsentative Koordinate trianguliert werden, indem der Punkt, an dem sich die Strahlen am nächsten kommen, berechnet wird. Dazu werden zuerst die nächsten Punkte zwischen allen Paaren der Zeigestrahlen bestimmt. Der Mittelpunkt dieser nächsten Punkte wird als repräsentative Koordinate gewählt.
Wenn zwei Strahlen gegeben sind von denen einer von Punkt A in Richtung $\vec{a}$ und der andere von Punkt B in Richtung $\vec{b}$ verläuft, lässt sich der nächste Punkt zwischen diesen mit den Formeln 3.3 errechnen [14]. $\vec{c}$ beschreibt hier den Vektor zwischen A und B. D und E beschreiben jeweils den Punkt auf dem jeweils ersten und zweiten Strahl an dem dieser dem anderen am nächsten kommt. Der nächste Punkt zwischen den Strahlen ist dementsprechend $\frac{D+E}{2}$.

$$\begin{aligned} D =& A + \vec{a}\frac{-(\vec{a}\cdot\vec{b})(\vec{b}\cdot\vec{c}) + (\vec{a}\cdot\vec{c})(\vec{b}\cdot\vec{b})}{(\vec{a}\cdot\vec{a})(\vec{b}\cdot\vec{b}) - (\vec{a}\cdot\vec{b})(\vec{a}\cdot\vec{b})} \\ E =& B + \vec{b}\frac{(\vec{a}\cdot\vec{b})(\vec{a}\cdot\vec{c}) - (\vec{b}\cdot\vec{c})(\vec{a}\cdot\vec{a})}{(\vec{a}\cdot\vec{a})(\vec{b}\cdot\vec{b}) - (\vec{a}\cdot\vec{b})(\vec{a}\cdot\vec{b})} \end{aligned} \tag{3.3}$$

3.6.2 Optimale Mustergeste

Der Zweck einer Mustergeste besteht darin durch ihre Ähnlichkeit zu anderen Instanzen der selben Geste zu bestimmen ob es sich bei einer Sequenz um die selbe Geste handelt. Dementsprechend ist eine gute Mustergeste allen anderen Instanzen der Geste möglichst ähnlich.
Die Ähnlichkeitsmetrik ist dabei die selbe, wie während der Klassifikation also die DTW Distanz. Aus einer Menge möglicher Mustergesten kann die beste dadurch bestimmt werden, dass DTW zwischen allen Mustergesten der selben Geste durchgeführt wird. Die beste Mustergeste ist dann die mit der kleinsten durchschnittlichen Distanz.

4 Evaluation

4.1 Versuchsaufbau

Um das System zu testen wurden in einem Versuchsaufbau einige Selektionen ausgeführt. Die Selektionserkennung wurde dabei auf einem HP Envy x360 Laptop mit einer 2.40GHz Intel i5 CPU und 16GB RAM ausgeführt. Die Webcam des Geräts wurde dabei als Kamera verwendet. Dazu wurde das Gerät auf einem 120cm hohen Sockel aufgestellt und so ausgerichtet, dass die Kamera horizontal in den Raum zeigt. Die Kamera nimmt mit einer Auflösung von 720p und einer Framerate von 30fps auf.
Der Versuchsaufbau 4.1 umfasst dabei drei Ziele LW, RW und FT. LW und RW sind zwei 115cm breite und 155cm hohe Fenster. Sie befinden sich 90cm über dem Boden und sind 70cm von einander entfernt. FT ist ein ungefähr 40cm x 30cm x 30cm großes Objekt, das 3m von RW entfernt auf einem 120cm hohen Sockel steht.
Der Versuchsaufbau wurde im Living Place Labor der HAW-Hamburg erstellt. Bei diesem handelt es sich um ein Testlabor für Smart Home Anwendungen womit es die Bedingungen eines Smart Home widerspiegelt.
Zur Triangulation der repräsentativen Koordinaten 3.6.1 der Zielobjekte wurde auf jedes Objekt von vier Standpunkten aus gezeigt. Für alle Objekte wurden die selben Standpunkte verwendet. Der Mittlere der Standpunkte ist 300cm vom Sockel der Kamera entfernt. Die weiteren Standpunkte befinden sich in Relation zu diesem je 120cm links, 60cm rechts und 120cm näher an der Kamera.
Innerhalb des Versuchsaufbaus wurden auch sechs mögliche Mustergesten aufgenommen. Bei der Aufnahme dieser wurde nicht explizit auf die Zielobjekte gezeigt um sicherzustellen, dass die Mustergesten möglichst allgemein sind. Die Mustergesten wurden so geschnitten, dass sie mit der Bewegung anfangen und enden.
Um mehr als einen Test auf den selben Daten ausführen zu können, wurden die Selektionen als Video aufgenommen. In diesem wird aus fünf verschiedenen Standpunkten jedes Ziel einmal selektiert. Damit wurden innerhalb von 128s 15 Selektionen ausgeführt.

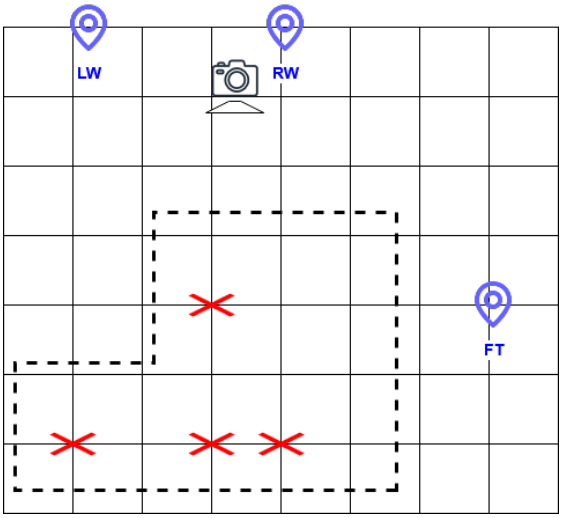

Abbildung 4.1: Versuchsaufbau
Diagramm der Draufsicht des Versuchsaufbau. Eingezeichnet sind die Positionen der Zielobjekte (blaue Marker), die Triangulations-Standpunkte (rote Kreuze), der Bereich aus dem die Selektionen ausgeführt wurden (grün gestrichelt) die Kamera, die gerade nach unten zeigt. Die Skala des Diagramms beträgt 60cm pro Kästchen.

Um die Auswertung zu vereinfachen wurden die Ziele jeweils in der selben Reihenfolge selektiert. Die Reihenfolge ist FT, LW und zuletzt RW. Damit zwischen den Selektionen wieder die Ruheposition eingenommen wurde, wurde nach jeder Selektion kurz gewartet. Zwischen den Selektionen wurden andere Bewegungen wie Strecken oder Winken ausgeführt. Sowohl die Mustergesten als auch die Selektionen wurden mit Zeigegesten aus dem ganzen Arm 2.1 ausgeführt.
Für alle Test wurde die Fenstergröße des Buffers basierend auf der Länge der zwei besten Mustersequenzen auf 65 Frames gesetzt.

4.2 Laufzeitmessungen

Die erste untersuchte Metrik ist die Laufzeit. Damit die Selektion in Echtzeit läuft ist es notwendig, dass die Schleife in der die Daten verarbeitet werden, diese mindestens so schnell verarbeitet wie sie entstehen. Das Laden der Mustergesten und anderer Daten vor der Verarbeitungsschleife haben keinen Einfluss auf die Verarbeitungsgeschwindigkeit. Die Zeitmessung beginnt deshalb mit der Schleife.

	Pose Estiamtion	+Gestenerkennung	+Zielerkennung
Mittelwert	16,25	45,314	45,745
Standartabweichung	0,294	0,668	0,277

Tabelle 4.1: Laufzeit Messergebnisse in ms
Alle Messungen wurden auf dem im Versuchsaufbau aufgenommenen Video je drei mal ausgeführt. Die Ergebnisse in dieser Tabelle sind auf die dritte Nachkommastelle gerundet. Die Werte beschreiben die Laufzeit pro Frame des Videos in Millisekunden.

Indem die Zeit bis das gesamte Video verarbeitet wird gemessen und durch die Anzahl Frames geteilt wird, kann die durchschnittliche Verarbeitungszeit pro Frame bestimmt werden. Bei einer Framerate von 30fps vergehen zwischen jedem Frame 33,333ms, die genutzt werden können bevor die Verarbeitung langsamer als die Kamera wird.
Um den Einfluss der einzelnen Komponenten auf die Laufzeit beurteilen zu können, wurden drei Konfigurationen vermessen, die progressiv weiter Komponenten verwenden. In der ersten wird nur die Pose Estimation mit Ortsschätzung ausgeführt. In der zweiten wird zusätzlich die Gestenerkennung ausgeführt. In der letzten wird auch die Identifikation des Ziels ausgeführt. Bei Gestenerkennung werden hier zwei Mustergesten verwendet. Die Schrittweite des Buffers ist hier 1. Das heißt, dass die Gesten- und Zielerkennung für jeden Frame ausgeführt werden.

4.2.1 Ergebnisse

Aus den Differenzen der Laufzeit 4.1 zwischen den Konfigurationen lassen sich die Laufzeiten der einzelnen Komponenten errechnen. Die Pose Estimation die für sich Vermessen wurde benötigt $16,25ms$. Die Gestenerkennung benötigt demnach $45,314ms - 16,25ms = 29,064ms$ und die Identifikation des Ziels $45,745ms - 45,314ms = 0,431ms$.
Die Vollständige Verarbeitungszeit von 45,745ms ist über der Echtzeitgrenze von 33,333ms und muss dementsprechend gesenkt damit das System in Echtzeit laufen kann. Der Einfluss der Pose Estimation kann nur reduziert werden indem die Framerate der Aufnahme gesenkt wird. Zu beachten ist auch, dass die Identifikation des Ziels während der Selektionserkennung, anders als bei dieser Messung nur eingesetzt wird wenn Erfolgreich eine Geste erkannt wird, was in einem Bruchteil der Frames der Fall sein sollte. Damit sinkt ihr bereits minimaler Einfluss auf die Laufzeit. Zudem macht es keinen Sinn die Häufigkeit de Zielerkennung allein künstlich zu senken, da eine entdeckte Geste ohne das Ziel der Selektion nutzlos ist. Stattdessen kann der Einfluss der Gestenerkennung

reduziert werden. Da die verwendeten Mustersequenzen ungefähr gleich lang sind, kann davon ausgegangen werden, dass sie gleichmäßig zur Laufzeit beitragen. Die Laufzeit der Gestenerkennung kann also halbiert werden indem entweder nur eine der Mustergesten verwendet wird, oder, wenn beide notwendig sind, indem die Schrittweite auf 2 verdoppelt wird. Eine Schrittweite von 2 würde verursachen, dass die Gestenerkennung halb so oft eingesetzt wird, was auch die Häufigkeit der Zielerkennung reduzieren würde. Die Laufzeit der Gestenerkennung auf $17,532$ zu halbieren würde die Gesamtlaufzeit bereits auf $28,213$ reduzieren. Damit kann festgehalten werden, dass das System in Echtzeit läuft, wenn die Schrittweite der Anzahl Mustergesten entspricht. Zumindest wenn die restlichen Einflussfaktoren wie die Fenstergröße des Buffers oder die Länge der Mustergesten unverändert bleiben.

4.3 Zuverlässigkeit der Selektionserkennung

Die Zuverlässigkeit der Selektionserkennung ergibt sich aus der Zuverlässigkeit ihrer Teile, also der Gestenerkennung und der Identifikation des Ziels. Beide Systeme funktionieren mit dem Prinzip den 1-Nearest-Neighbour aus einer Menge zu identifizieren. Dieser wird basierend auf je einer Metrik mit einem Cutoff-Threshold gewählt.
Da der der optimale Threshold experimentell bestimmt werden muss, wird die Zuverlässigkeit der Gestenerkennung daran gemessen, ob ein Threshold so definiert werden kann, dass alle Zeigegesten und nur diese den Threshold erfüllen. Da bereits angenommen wird, dass eine Selektion ausgeführt wird, wenn das Ziel identifiziert wird, ist der Threshold weniger wichtig, als dass das korrekte Ziel den besten Wert in der Metrik erhält.
In beiden Fällen sind die Werte der jeweiligen Metrik zur Bewertung notwendig. Da die Laufzeit beim Sammeln dieser Daten keine Rolle spielt, wird die Schrittweite des Buffers auf 1 gesetzt um die Werte der Metriken möglichst oft zu bestimmen. Zudem wird das Zielobjekt nach jeder Gestenerkennung bestimmt, unabhängig vom Resultat letzterer. So wird der Einfluss der Gestenerkennung auf die Bewertung der Zielerkennung minimiert. Ein gewisser Einfluss bleibt natürlich, da der Warping Path der Gestenerkennung zur Zielerkennung verwendet wird.

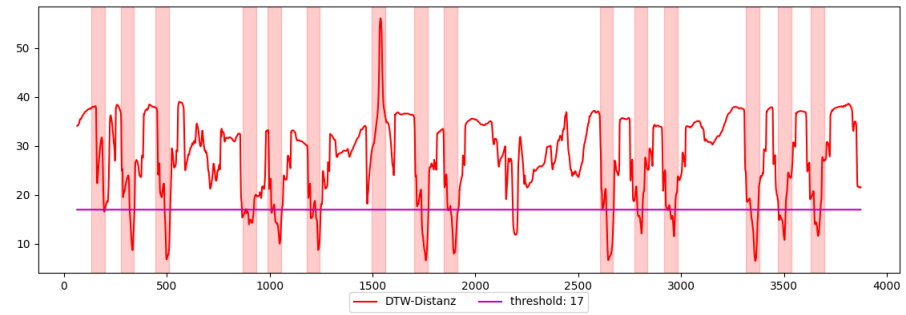

Abbildung 4.2: Graph der DTW Distanz aus der Gestenerkennung
Dieser Graph zeigt die DTW Distanz über die gesamte Aufnahme aus dem Versuchsaufbau. Die x Achse beschreibt den Zeitpunkt in der Aufnahme in Frames. Um zu markieren wann eine Selektionsgeste ausgeführt wurde, wurde die Enden der Gesten manuell bestimmt und die Länge eines Buffer Fensters vor den Enden leicht rot eingefärbt.

4.3.1 Gestenerkennungs Ergebnisse

Da die DTW Distanz kleiner wird je mehr zwei Sequenzen sich ähneln, ist es positiv zu beobachten, dass die Distanz bei 11 der zwölf Selektionen merklich sinkt 4.2. Die Zeigegeste bei Frame 1530, die eine ungewöhnlich hohe Distanz verursacht hat, kennzeichnet sich dadurch, dass die zweite Phase des Zeigens 2.1 deutlich länger gehalten wurde als in den anderen Fällen. Die niedrige Distanz ohne Zeigen bei Frame 2200 trifft auf als gewunken wurde. Während sich Zeigen und Winken darin ähneln, dass der Arm aus der Schulter gehoben wird. Die fürs Winken charakteristische Seitwärtsbewegung ändert weder den Hebewinkel des Oberarms noch den Winkel am Ellenbogen und wird dementsprechend in der Gestenerkennung nicht wahrgenommen. Während zusätzliche Winkel, die die Orientierung des Arms in der horizontalen Ebene angeben, die Rotationsinvarianz widersprechen würden, könnte in zukünftiger Weiterentwicklung untersucht werden, ob die Verwendung der Änderungsrate der Winkel statt ihrer absoluten Werte diesen Widerspruch beheben könnte.

Während ein niedrigerer Threshold von 11,8 einen False-Positve in diesem Fall verhindern könnte, ist die Distanz so klein wie bei einigen korrekten Gesten, sodass ein geringerer Threshold neue False-Negatives 4.3 verursachen würde. Wenn es im Anwendungsfall nicht besonders wichtig ist die Anzahl False-Positives zu minimieren sollte statt einem Threshold von 11,8 mit einer Accuracy von 0,75 ein Threshold von 17 verwendet

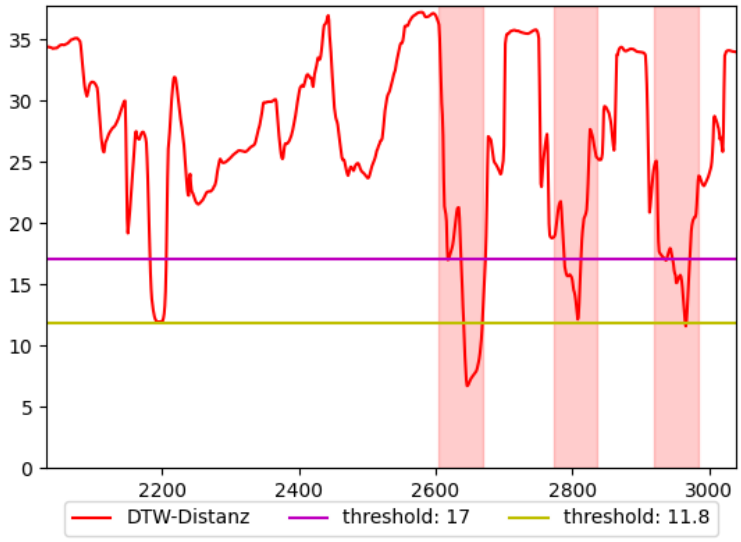

Abbildung 4.3: DTW Distanzen mit möglichen Grenzwerten
In diesem Ausschnitt des Distanz Graphs 4.2 werden zwei mögliche Cutoff Thresholds gezeigt. Ein Threshold von 17 würde je einen False-Positve und False-Negative sowie 11 True-Positives erzeugen. Ein Threshold von 11,8 würde den False-Positive entfernen. Es würde jedoch auch die Anzahl False-Negatives auf Kosten der True-Positives erhöhen.

werden, der eine Accuracy von 0,875 hat.
Es ist ebenfalls zu beobachten, dass die Fälle in denen der Tiefpunkt der Distanz 10 Frames oder weniger vor dem manuell bestimmten Ende der Geste auftritt, zu geringeren Distanzen tendieren als die Fälle in denen der Tiefpunkt früher erreicht wird. Eine mögliche Erklärung dafür ist, dass bei letzteren die Geste langsamer ausgeführt wurde und somit immer mindestens ein Ende der Geste nicht Teil der verglichenen Sequenz ist. Bei einem Threshold von 17 wird dieser in allen korrekten Fällen vor Ende der Geste unterschritten und für mindestens 5 Frames am Stück. Das heißt, dass die gestenerkennung alle fünf Frames ausgeführt werden könnte ohne Selektionen zu verlieren und, dass die Selektion aus Wahrnehmung des Nutzers ohne Zeitverzögerung registriert wird.

Bei den Winkel zu den jeweiligen Zielobjekten während der Zielerkennung 4.4 fällt auf, dass sich die Winkel sprunghaft von einem Frame Frame zum nächsten verändern. Dabei wird von 90° was in der Ruheposition erwartet wird, direkt auf unter 20° gesprungen. Wenn der Zeigestrahl naiv jeweils aus dem nächsten Frame erstellt wird, wäre davon auszugehen, dass die Winkel gleichmäßig mit dem Anheben des Arm sinken. Es kann folglich davon ausgegangen werden, dass die Auswahl des Zeige-Frames über den Warping Path wie geplant funktioniert. Wenn sie funktioniert würde schließlich erwartet werden, dass sobald der korrekte in der Sequenz ist dieser erkannt wird und der Winkel springend sinkt. Der kleine Winkel würde dann konstant bleiben, bis der korrekte Zeigeframe die Sequenz wieder verlässt. Dieses Verhalten wird im Graph gezeigt.
Ein Winkel Threshold von 25° würde alle korrekt erkannten Ziele umfassen. Dieser Threshold würde ebenfalls verhindern, dass das Winken bei Frame 2200 zu einer Selektion führt auch wenn der Threshold von 17 bei der Gestenerkennung verwendet wird.
Während die Zielobjekte FT und LW immer die kleinsten Winkel aufweisen, wenn sie selektiert werden sollen, hat RW nur in 40% der Fälle den kleinsten Winkel. In den restlichen Fällen wird fälschlicherweise LW ausgewählt. Auch in der Triangulation der Koordinaten fiel auf, dass die meisten der Zeigstrahlen zu RW sich bereits von ihrem Ursprung aus voneinander entfernt haben. Eine mögliche Erklärung dafür ist, dass das die Dimensionen in der Ortsschätzung und der Haltung aus MediaPipe nicht gleich groß sind. Dies könnte dazu führen, dass die Strahlen nicht zu einem Punkt konvergieren. Dass RW am nächsten hinter der Kamera liegt unterscheidet es am stärksten von den seitlicher gelegenen Objekten FT und LW und ist wahrscheinlich ein Teil des Grundes.
Die Identifikation des Ziels hat also mit drei Fehlern bei RW und insgesamt 15 Selektionen eine Accuracy von 0,8.

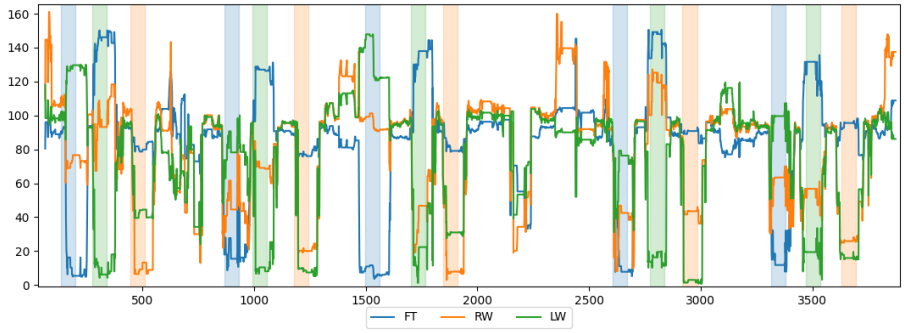

Abbildung 4.4: Graph der Winkel zwischen den Zeigestrahlen und den Zielobjekten Gezeigt werden die Winkel zwischen den Zeigestrahlen und den Zielobjekten über die gesamte Aufnahme aus dem Versuchsaufbau. Die Winkel sind an der y Achse in Grad angegeben, die Position in der Sequenz an der x Achse in Frames. Die vertikal markierten Bereiche zeigen die manuell bestimmte Position der Selektionen. Sie sind in der selben Farbe des korrekten Objekts eingezeichnet.

In Kombination würden mit einem Gestenerkennungs Threshold von 17 und einem Winkel Threshold von 25 von den 15 Selektionen 11 korrekt erkannt. 3 würden das falsche Objekt selektieren und eine würde gar nicht registriert werden. Das ergibt eine gesamte Accuracy von $0,7\overline{3}$.

4.4 Fazit

Der Entwickelte Prototyp demonstriert, dass der Lösungsansatz Potential hat. Die Selektion funktioniert mit einer Accuracy von $0,7\overline{3}$. 20% der Fehler treten bei der Identifikation eines Zielobjekts auf, sodass wenn das unterliegende Problem ermittelt und behoben wird die Accuracy $0,9\overline{3}$ steigen würde. Die Ergebnisse zeigen auch, dass der Prototyp bis zu 5 verschiedene Mustergesten in Echtzeit erkennen könnte. Das Ziel einer Reaktionszeit von unter einer Sekunde wurde ebenfalls erfüllt, da die Selektion bereits erfasst ist, wenn der Nutzer wieder die Ruheposition erreicht hat.

5 Ausblick

Wie bereits im Fazit erwähnt, sollte in weiterführender Arbeit die Ursache für falsche Identifikation des Zielobjekts ermittelt werden. Sollte die Ursache wirklich in einer Differenz zwischen Pose Estimation und Ortsschätzung liegen, könnte dies ein anderes Pose Estimation Modell erfordern. Das neue Modell müsste in dem Fall native Ortsschätzung enthalten. Sollte dies mit einzelnen RGB-Kameras nicht möglich sein, sollte die Verwendung spezilaisierter Hardware das Problem beseitigen.
Um die Ortsschätzung zu verbesssern, könnte auch untersucht werden ob multiple Refernzwerte gleichzeitig verwendet werden können. Dafür könnte der Refernzwert dynamisch aus einer Menge ausgewählt werden, der zur Bildebene am parallesten orientiert ist. Dies könnte die Bedingung, dass der Nutzer gerade zur Kamera steht unnötig machen.
Wenn die Selektionserkennung eine ausreichende Qualität erreicht hat, liegt es nahe eine Manipulationsgestenerkennung zu entwickeln um die Gestensterúerung zu vervollständigen.

Literaturverzeichnis

[1] *Pose landmark detection guide.* – URL https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker. – [Online; Zugriff 6.11.2024]

[2] Bazarevsky, Valentin ; Grishchenko, Ivan ; Raveendran, Karthik ; Zhu, Tyler ; Zhang, Fan ; Grundmann, Matthias: *BlazePose: On-device Real-time Body Pose tracking.* 2020. – URL https://arxiv.org/abs/2006.10204

[3] Bazarevsky, Valentin ; Kartynnik, Yury ; Vakunov, Andrey ; Raveendran, Karthik ; Grundmann, Matthias: *BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs.* 2019. – URL https://arxiv.org/abs/1907.05047

[4] Benalcázar Palacios, Marco E. ; Aguas, Xavier ; Valdivieso Caraguay, Ángel L. ; Barona López, Lorena I. ; Nogales, Rubén ; Guilcapi, Jaime ; Benalcázar, Freddy: An interactive system to improve cognitive abilities using electromyographic signals. In: *Proceedings of the 5th International Conference on Advances in Artificial Intelligence.* New York, NY, USA : Association for Computing Machinery, 2022 (ICAAI '21), S. 84–90. – URL https://doi.org/10.1145/3505711.3505723. – ISBN 9781450390699

[5] Bodirožа, Saša ; Doisy, Guillaume ; Hafner, Verena V.: Position-invariant, real-time gesture recognition based on dynamic time warping. In: *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction*, IEEE Press, 2013 (HRI '13), S. 87–88. – ISBN 9781467330558

[6] Cao, Zhe ; Hidalgo, Gines ; Simon, Tomas ; Wei, Shih-En ; Sheikh, Yaser: *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields.* 2019. – URL https://arxiv.org/abs/1812.08008

[7] Gunerli, John: *Coordinate distance estimation using MediaPipe.* 2023. – URL https://github.com/johngunerli/LIDARLAB_CDE

[8] Insights, Statista Technology M.: *Number of users of smart homes in Europe from 2019 to 2028 (in millions) [Graph].* 2023. – URL https://www.statista.com/forecasts/1283780/smart-home-users-in-europe

[9] Ji, Penghui ; Cao, Chongli ; Zhang, Hang ; Li, Qi: Multi-Scale Convolution Attention Neural Network for Gesture Recognition. In: *Proceedings of the 2024 3rd International Conference on Cryptography, Network Security and Communication Technology.* New York, NY, USA : Association for Computing Machinery, 2024 (CNSCT '24), S. 421–425. – URL https://doi.org/10.1145/3673277.3673350. – ISBN 9798400716959

[10] Lu, Yiqin ; Yu, Chun ; Shi, Yuanchun: Investigating Bubble Mechanism for Ray-Casting to Improve 3D Target Acquisition in Virtual Reality. In: *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, 2020, S. 35–43

[11] Moeslund, Thomas B. ; Hilton, Adrian ; Krüger, Volker: A survey of advances in vision-based human motion capture and analysis. In: *Computer Vision and Image Understanding* 104 (2006), Nr. 2, S. 90–126. – URL https://www.sciencedirect.com/science/article/pii/S1077314206001263. – Special Issue on Modeling People: Vision-based understanding of a person's shape, appearance, movement and behaviour. – ISSN 1077-3142

[12] Müller-Tomfelde, Christian: Dwell-Based Pointing in Applications of Human Computer Interaction. In: Baranauskas, Cécilia (Hrsg.) ; Palanque, Philippe (Hrsg.) ; Abascal, Julio (Hrsg.) ; Barbosa, Simone Diniz J. (Hrsg.): *Human-Computer Interaction – INTERACT 2007.* Berlin, Heidelberg : Springer Berlin Heidelberg, 2007, S. 560–573. – ISBN 978-3-540-74796-3

[13] Ottenheym, Thore: *Graph Convolutional Network based Gesture Interpretation of Sekeleton Data.* 2024. – URL https://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/ottenheym.pdf

[14] Palitri: *Closest point between two rays.* Accessed: 23.03.2025. – URL https://palitri.com/vault/stuff/maths/Rays%20closest%20point.pdf

[15] Riofrío, Santiago ; Pozo, David ; Rosero, Jorge ; Vásquez, Juan: Gesture Recognition Using Dynamic Time Warping and Kinect: A Practical Approach. In: *2017 International Conference on Information Systems and Computer Science (INCISCOS)*, 2017, S. 302–308

[16] Sakoe, H. ; Chiba, S.: Dynamic programming algorithm optimization for spoken word recognition. In: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26 (1978), Nr. 1, S. 43–49

[17] Salvador, Stan ; Chan, Philip: Toward accurate dynamic time warping in linear time and space. In: *Intell. Data Anal.* 11 (2007), Oktober, Nr. 5, S. 561–580. – ISSN 1088-467X

[18] Schneider, Pascal ; Memmesheimer, Raphael ; Kramer, Ivanna ; Paulus, Dietrich: Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping. In: Chalup, Stephan (Hrsg.) ; Niemueller, Tim (Hrsg.) ; Suthakorn, Jackrit (Hrsg.) ; Williams, Mary-Anne (Hrsg.): *RoboCup 2019: Robot World Cup XXIII*. Cham : Springer International Publishing, 2019, S. 281–293. – ISBN 978-3-030- 35699-6

[19] Shteynberg, Garriy: Shared Attention. In: *Perspectives on Psychological Science* 10 (2015), Nr. 5, S. 579–590. – URL https://doi.org/10.1177/1745691615589104. – PMID: 26385997

[20] Striano, Tricia ; Reid, Vincent M. ; Hoehl, Stefanie: Neural mechanisms of joint attention in infancy. In: *European Journal of Neuroscience* 23 (2006), Nr. 10, S. 2819–2823. – URL https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1460-9568.2006.04822.x

[21] Stukenbrock, Anja: Deixis, Meta-Perceptive Gaze Practices, and the Interactional Achievement of Joint Attention. In: *Frontiers in Psychology* 11 (2020). – URL https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2020.01779. – ISSN 1664-1078

[22] Tanida, Kazuaki: *fastDTW 0.3.4*. 2019. – URL https://pypi.org/project/fastdtw/

[23] Ten Holt, Gineke A. ; Reinders, Marcel J. ; Hendriks, Emile A.: Multi-dimensional dynamic time warping for gesture recognition. In: *Thirteenth annual conference of the Advanced School for Computing and Imaging* Bd. 300, URL https://www.researchgate.net/profile/Marcel-Reinders/publication/228740947_Multi-dimensional_dynamic_time_warping_for_gesture_recognition, 2007, S. 1

[24] Vakunov, Andrey ; Lagun, Dmitry: *MediaPipe Iris: Real-time Iris Tracking Depth Estimation*. – URL https://research.google/blog/mediapipe-iris-real-time-iris-tracking-depth-estimation/. – [Online; Zugriff 7.11.2024]

[25] Xu, Hongyi ; Bazavan, Eduard G. ; Zanfir, Andrei ; Freeman, William T. ; Sukthankar, Rahul ; Sminchisescu, Cristian: GHUM GHUML: Generative 3D Human Shape and Articulated Pose Models. In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, S. 6183–6192

[26] Zheng, Ce ; Wu, Wenhan ; Chen, Chen ; Yang, Taojiannan ; Zhu, Sijie ; Shen, Ju ; Kehtarnavaz, Nasser ; Shah, Mubarak: Deep Learning-based Human Pose Estimation: A Survey. In: *ACM Comput. Surv.* 56 (2023), August, Nr. 1. – URL https://doi.org/10.1145/3603618. – ISSN 0360-0300

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original