HAW HAMBURG

BACHELOR THESIS
Jonas Albers

Entwicklung einer Template-basierten Gestensteuerung zur Interaktion in vernetzten Smarthome-Umgebungen

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

Jonas Albers

Entwicklung einer Template-basierten Gestensteuerung zur Interaktion in vernetzten Smarthome-Umgebungen

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. Kai von Luck
Zweitgutachter: Dr. Susanne Draheim

Eingereicht am: 18. September 2025

Jonas Albers

Thema der Arbeit

Entwicklung einer Template-basierten Gestensteuerung zur Interaktion in vernetzten Smarthome-Umgebungen

Stichworte

Dynamic Time Warping, Pose Estimation, Gestenerkennung, Smarthome, Human-Computer Interaction

Kurzzusammenfassung

Diese Arbeit untersucht die Entwicklung und Evaluation einer template-basierten Gestensteuerung zur Interaktion in vernetzten Smarthome-Umgebungen. Grundlage ist die Kombination von präziser 3D-Pose-Erfassung mittels Stereokamera und einer Gestenerkennung auf Basis von Dynamic Time Warping. Der entwickelte Prototyp ermöglicht eine intuitive und natürliche Interaktion durch Zeigegesten zur Objektauswahl und anschließende Manipulationsgesten zur Steuerung von Geräten. Die Evaluation bestätigt die Eignung des Ansatzes im Kontext der Human-Computer Interaction (HCI), zeigt aber auch Optimierungspotenzial bei der Robustheit einzelner Gesten.

Jonas Albers

Title of Thesis

Development of a Template-based gesture control for interacting in connected Smarthome-enviroments

Keywords

Dynamic Time Warping, Pose Estimation, Gesture Recognition, Smarthome, Human-Computer Interaction

Abstract

This thesis presents the development and evaluation of a template-based gesture control system for interaction within connected Smarthome environments. The approach combines precise 3D pose estimation using a stereo camera with gesture recognition based on Dynamic Time Warping. The prototype enables intuitive and natural interaction through pointing gestures for object selection followed by manipulation gestures for device control. The evaluation confirms the suitability of the approach in the field of Human-Computer Interaction (HCI), while also revealing optimization potential for robustness and gesture differentiation.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

AABB Axis-Aligned Bounding Box.

CSTI Creative Space for Technical Innovation Labor an der HAW.

DTW Dynamic Time Warping.

HAW Hochschule für Angewandte Wissenschaften Hamburg.

HCI Human-Computer Interaction.

IoT Internet of Things.

1 Einleitung

„A gesture is a motion of the body that contains information. Waving good-bye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on its way to hitting the key is neither observed nor significant.“
Kurtenbach und Hulteen [16] – 1990

Gesten sind eine der natürlichsten Formen menschlicher Kommunikation und dienen seit jeher als Ausdrucksmittel in sozialen Interaktionen. Schon lange bevor Sprache oder technische Werkzeuge zur Verfügung standen, nutzten Menschen Bewegungen von Händen, Armen oder dem gesamten Körper, um Informationen auszutauschen. Auch heute noch sind Gesten ein zentraler Bestandteil zwischenmenschlicher Kommunikation, sei es beim Zeigen, Winken oder dem Unterstreichen gesprochener Worte durch Handbewegungen.

Im Bereich der Mensch-Maschine-Interaktion (Human-Computer Interaction, HCI) haben Gesten seit den 1980er-Jahren einen festen Platz als Forschungsgegenstand gefunden. Einer der frühesten Ansätze beschäftigte sich dabei zum Beispiel schon 1986 mit der Entwicklung eines Sensor-Handschuhs, welcher die Bewegung der Hand auf den Computer übertragen konnte [27]. Der Reiz dieser Steuerungen liegt darin, dass sie eine intuitive und unmittelbare Form der Steuerung technischer Systeme ermöglichen. Anders als klassische Eingabemethoden wie Tastaturen, Fernbedienungen oder Touchscreens erfordern Gesten keine physische Schnittstelle, sondern nutzen den menschlichen Körper selbst als Interaktionsmedium. Dadurch entfällt nicht nur die Abhängigkeit von Eingabegeräten, sondern es eröffnet sich auch die Möglichkeit, Technik unauffällig und nahtlos in den Alltag zu integrieren.

Motivation und Relevanz

Mit dem Aufkommen von Smart Environments, also Umgebungen, in denen verschiedene vernetzte Geräte und Systeme nahtlos zusammenarbeiten, wächst der Bedarf an

flexiblen, leicht erlernbaren und kontextsensitiven Interaktionsformen. Sprachsteuerung, die in den letzten Jahren große Verbreitung gefunden hat [14], stößt hier teilweise an ihre Grenzen: Sie ist störanfällig in lauten Umgebungen, erfordert eine klare sprachliche Artikulation und wird von vielen Menschen in öffentlichen oder gemeinschaftlich genutzten Räumen als unnatürlich oder störend empfunden. Gestensteuerungen stellen in diesem Zusammenhang eine wertvolle Ergänzung dar, da sie ohne Sprache funktionieren, diskret eingesetzt werden können und eine Form der Interaktion darstellen, die vielen Menschen bereits aus alltäglichen Situationen vertraut ist.

Darüber hinaus bietet die Gestensteuerung Vorteile in Situationen, in denen Hände oder Aufmerksamkeit anderweitig gebunden sind. Ein Beispiel ist die Steuerung von Geräten in einer Werkstatt oder Küche, in denen physische Eingabegeräte nicht immer leicht erreichbar sind. Auch im Bereich der Barrierefreiheit eröffnen Gesten neue Möglichkeiten, da sie für Personen mit eingeschränkter Mobilität eine alternative Eingabemethode darstellen können.

Ansatz dieser Arbeit

Der in dieser Arbeit verfolgte Ansatz untersucht die Möglichkeit, durch Körpergesten Objekte in einer Umgebung direkt zu adressieren und deren Auswahl zuverlässig zu steuern. Im Vordergrund steht dabei nicht die Komplexität der Gesten, sondern deren Robustheit, Verständlichkeit und Übertragbarkeit auf verschiedene Nutzungsszenarien. Eine einfache, aber zuverlässige Erkennung von Gesten ermöglicht es, Systeme alltagstauglich einzusetzen, ohne dass lange Trainingsphasen oder eine hohe technische Vorkenntnis erforderlich sind.

Ein wesentliches Ziel des entwickelten Systems ist es daher, dass es sowohl an unterschiedliche räumliche Gegebenheiten als auch an verschiedene Nutzerinnen und Nutzer ohne hohen Aufwand angepasst werden kann. So lässt sich das System schnell in neue Räume übertragen oder an individuelle Ausführungsstile der Nutzer anpassen.

Ein weiterer Vorteil des vorgestellten Ansatzes liegt in der Nutzung weit verbreiteter Schnittstellen und standardisierter Technologien. So basiert die Datenübertragung zwischen Kamerasystem und Erkennungsmodul auf dem MQTT-Protokoll, einem etablierten Standard in der IoT-Kommunikation. Dadurch wird nicht nur eine hohe Interoperabilität

mit bestehenden Smarthome- oder Smart-Office-Systemen gewährleistet, sondern auch eine spätere Anbindung an reale Geräte erheblich erleichtert. Das System ist somit nicht auf eine bestimmte Hardware oder Softwareumgebung beschränkt, sondern bewusst offen und erweiterbar gestaltet.

Das zentrale Forschungsinteresse dieser Arbeit liegt darin, zu zeigen, wie Gestenerkennung in praktischen Szenarien zur Selektion von Objekten in realen Umgebungen eingesetzt werden kann.

Die Besonderheit liegt darin, dass nicht nur einzelne Gesten erkannt werden, sondern dass deren räumlicher Bezug zu Objekten im Raum eine zentrale Rolle spielt. Damit unterscheidet sich der Ansatz von rein gestenbasierten Steuerungen, bei denen Befehle losgelöst von der Umgebung ausgeführt werden. Stattdessen entsteht hier eine kontextsensitive Interaktion.

2 Analyse

Die Intention des nachfolgenden Kapitels besteht in der Erörterung der einzelnen Teilprobleme einer Gestensteuerung. Im Fokus der Untersuchung steht dabei auch die Bewältigung dieser Teilprobleme in vorangegangenen Arbeiten.

2.1 Gesten

Eine Geste ist eine Bewegung des Körpers, die Informationen enthält. Diese Aussage trafen Kurtenbach und Hulteen bereits 1990 in ihrer Ausarbeitung „Gestures in Human-Computer Communication“[16]. Diese Informationen an einen Computer zu übertragen ist mit traditionellen Steuerungsmechanismen wie der Tastatur nicht möglich. Gestensteuerung ermöglicht in der Human-Computer Interaction somit eine intuitive Steuerung durch Körpersprache, insbesondere durch Bewegungen der Hände und Arme.

Im Kontext dieser Arbeit werden Gesten in zwei funktionale Kategorien unterteilt: Selektionsgesten, die der Auswahl eines Objekts dienen, und Manipulationsgesten, die eine direkte Aktion auf ein bereits ausgewähltes Objekt ausführen.

2.1.1 Selektionsgesten

Selektionsgesten zielen darauf ab, ein Objekt in der Umgebung zu identifizieren und auszuwählen. Ein typisches Beispiel ist das Zeigen auf ein Objekt mit dem ausgestreckten Arm oder Finger. Diese Geste ist für den Menschen intuitiv und bietet eine direkte visuelle Referenz für das Zielobjekt. Lange bevor Kinder sprechen können, drücken sie sich bereits durch Zeigen aus [5], wobei das spezifische Zeigen mit ausgestrecktem Zeigefinger mit steigendem Alter schnell zunimmt [11].

Für die automatische Erkennung von deklarativem Zeigen als Selektionsgeste werden nach Müller-Tomfelde [18] folgende drei Schritte ausgewertet:

1. Orientierung des Zeigewerkzeugs: Als Werkzeug versteht sich im Kontext dieser Arbeit der Arm. Die Richtung des Unterarms oder die Linie zwischen Schulter und Hand zeigt auf ein Ziel im Raum.
2. Stabiles Halten: Die Geste muss über einen bestimmten Zeitraum (z. B. 2 Sekunden) stabil gehalten werden, um ungewollte oder flüchtige Bewegungen auszuschließen.
3. Ende der Geste: Das Zeigewerkzeug wird wieder abgesenkt und die Geste ist damit beendet.

2.1.2 Manipulationsgesten

Nach erfolgter Selektion eines Objekts erlauben Manipulationsgesten eine unmittelbare Interaktion mit diesem. Die Erwartung des Nutzers besteht darin, dass das System im Anschluss eine erkennbare Reaktion zeigt. Menschen verbinden dabei auf intuitive Weise bestimmte Gesten mit konkreten Aktionen. Woher diese Zuordnungen stammen, ist jedoch oft unklar. Sie können von vagen Assoziationen bis hin zu Erfahrungen mit früheren gestenbasierten Interfaces reichen. Bernat et al. [2] untersuchten im Rahmen eines agilen Entwicklungsansatzes, welche Gesten Nutzer spontan mit bestimmten Aktionen wie dem Einschalten von Licht verknüpfen. Allein für diese einfache Funktion wurden sieben unterschiedliche Gestenvorschläge gemacht [2, S. 5, Tabelle II]. Ein Hinweis darauf, wie vielfältig und subjektiv gestische Interaktion interpretiert werden kann. Dennoch lässt sich beobachten, dass für jede einzelne vorgeschlagene Geste eine Bewegung der Arme und/oder Hände genutzt wurde. Typische Handbewegungen: zur Manipulation Systemen sind dabei unter anderem:

- Greifen und Loslassen
- Wischen oder Schieben
- Rotationen
- Winken
- Heben oder Senken

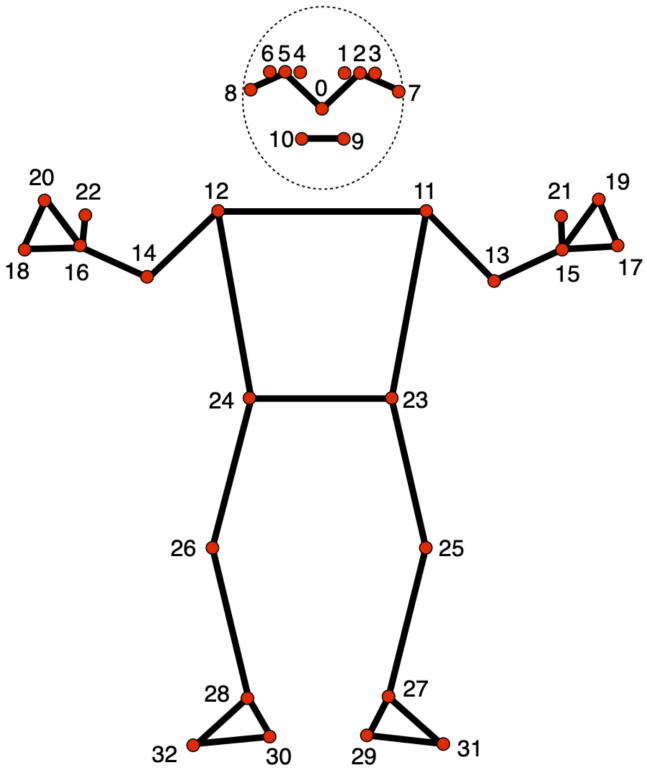


Abbildung 2.1: Index der Pose-Keypoints in MediaPipe. Quelle: [9]

2.2 Pose Estimation

„Human Pose Estimation aims to locate the human body parts and build human body representation (e.g., body skeleton) from input data such as images and videos.“[26, S.1]

Human Pose Estimation bezeichnet die Erkennung und räumliche Lokalisierung signifikanter Körperpunkte (sogenannte Keypoints) einer Person anhand von Bild- oder Videodaten. Die so ermittelten Keypoints werden typischerweise an Gelenken wie Schultern, Ellenbogen oder Knien positioniert und bilden zusammen ein vereinfachtes Skelettmodell. Das Ziel besteht darin, eine möglichst präzise Repräsentation der Körperhaltung im Raum zu erhalten. Dies ist unter anderem für die Analyse und Interpretation von Bewegungsabläufen und Gesten von Relevanz.

Die Anzahl und Auswahl der Keypoints variiert je nach Modell und Anwendungsfall. Die einzelnen Punkte des Modells sind oft durch eine feste skelettale Struktur miteinander

verbunden, wodurch sich auch die Bewegungsmöglichkeiten (z. B. Rotation in Gelenken) logisch einschränken lassen.

Ein zentrales Unterscheidungskriterium aktueller Pose-Estimation-Ansätze ist die Dimensionalität der erkannten Positionen:

- 2D-Pose Estimation lokalisiert Keypoints ausschließlich in der Bildebene (x- und y-Koordinaten).
- 3D-Pose Estimation erweitert die Lokalisierung um die Tiefeninformation (z-Koordinate), was insbesondere für Interaktionen mit der physischen Umgebung essenziell ist.

Da monokulare Bildquellen keine direkte Tiefeninformation liefern, ist die 3D-Pose Estimation mit einzelnen Kameras deutlich herausfordernder. Hier werden Verfahren eingesetzt, die auf Trainingsdaten basierende Rekonstruktionen durchführen. Für präzisere Ergebnisse kommen alternativ Methoden mit zusätzlicher Tiefeninformation zum Einsatz, beispielsweise:

- RGB-D-Kameras, die Tiefendaten direkt mit Infrarotsensorik erfassen (z. B. Microsoft Kinect 2),
- Mehrkamerasysteme, die mehrere synchronisierte Einzelbilder aus unterschiedlichen Winkeln zur Rekonstruktion nutzen, darunter auch Stereokameras wo sich zwei oder mehr Kameras leicht versetzt in einem einzelnen Gehäuse befinden

Während rein visuelle Verfahren weniger Sensorik erfordern, sind sie anfälliger für Perspektivprobleme und verdeckte Gelenke. Multisensor-Systeme dagegen bieten höhere Robustheit, setzen aber spezialisierte Hardware und Kalibrierung voraus.

2.2.1 Top-Down vs. Bottom-Up Verfahren

Die meisten Pose-Estimation-Modelle lassen sich algorithmisch in zwei Hauptkategorien einteilen:

- Top-Down Verfahren erkennen zuerst die Personen im Bild (z. B. über ein Bounding-Box-Verfahren) und führen dann eine Einzelpersonen-Pose-Erkennung innerhalb dieses Bereichs durch.

- Bottom-Up Verfahren erkennen zuerst alle Keypoints im Bild, unabhängig von der Anzahl der Personen, und gruppieren diese anschließend zu einzelnen Skeletten.

2.2.2 Bekannte Modelle

Mehrere bekannte Frameworks und Bibliotheken für Pose Estimation haben sich bereits durch ihre verschiedenen Stärken etabliert:

- OpenPose [7]: Ein Bottom-Up-Ansatz, der sogenannte Part Confidence Maps für einzelne Keypoints sowie Part Affinity Fields zur Verknüpfung dieser Punkte zu Skeletten erzeugt. Durch diese beiden Vorgänge werden bestimmte Körperteile zuerst erkannt und dann zu unterschiedlichen kompletten Körpern gruppiert. Ebenfalls erwähnenswert ist an dieser Stelle die auf Einzelpersonen spezialisierte Weiterentwicklung von Open Pose mit dem Namen EfficientPose. [4]
- MediaPipe Pose: Auch Pose Landmarker genannt, ist ein Top-Down-Modell auf Basis des BlazePose-Netzwerks [1]. Das Modell erkennt 33 Körperpunkte pro Körper. Dafür werden zwei Modelle seriell ausgeführt, zuerst das „Pose detection model “, welches erkennt wo sich eine Person im Bild befindet und anschließend das „Pose landmarker model “, welches die 33 Körperpunkte zuordnet. [9]
- HRNet, PoseNet, DensePose: Weitere populäre neuronale Netzwerke, die entweder durch besonders hohe Präzision oder Rechenleistung auffallen. HRNet [23] beispielsweise erreicht hohe Genauigkeit durch parallele Verarbeitung von Hoch- und Niedrigauflösungen (erzeugt durch Convolution) mit ständigem Informationsaustausch zwischen diesen Netzwerken.

2.2.3 Herausforderungen und Limitierungen

Human Pose Estimation ist mit mehreren Herausforderungen verbunden:

- Verwechslung bei Mehrpersonen-Szenen: Bottom-Up-Modelle müssen Keypoints korrekt Personen zuordnen. [13]
- Verdeckte Körperteile: Wenn Körper teilweise verdeckt sind ist die Rekonstruktion der Keypoints schwieriger. Insbesondere in Kombination mit dem vorherigen

Punkt, wenn mehrere Körper sich gegenseitig verdecken wird die korrekte Erkennung erheblich schwieriger. Dazu führen zum Beispiel Song–Hai Zhang, Ruilong Li, Xin Dong et al. in ihrem Artikel „Pose2Seg: Detection Free Human Instance Segmentation “[21] den „Occluded Human (OCHuman)“-Maßstab für die korrekte Erkennung von semi verdeckten Personen ein.

- Perspektivische Verzerrung: Ohne Tiefeninformationen ist die korrekte Rekonstruktion der Körperhaltung erschwert. Ein Problem das von monokularen Kameras nur schwer bewältigt werden kann. Schlechte oder verzerrte Bilder sorgen dann für Missinterpretation der 3D Ebene.
- Instabilität der Daten: Pose-Daten schwanken häufig stark zwischen einzelnen Frames – besonders bei schlechter Bildqualität oder Bewegung. Zur Kompensation dieser Probleme kommen häufig Filtertechniken zum Einsatz, wie z. B. der Butterworth-Filter [6] oder gleitende Mittelwerte. Auch zeitliche Zusammenhänge können genutzt werden, um durch Rekurrente Netze oder temporale Glättung stabilere Posen zu generieren.

2.3 Algorithmische Gestenerkennung

Die Erkennung der genanten Gesten auf Basis zuvor erfasster Bewegungsdaten ist ein zentrales Element gestenbasierter Mensch-Computer-Interaktion. Ziel der Gestenerkennung ist es, aus einer Sequenz von Körperhaltungen oder Bewegungen eine semantische Geste zu identifizieren. Dies kann mit Hilfe verschiedener Verfahren erfolgen, die entweder auf maschinellem Lernen oder auf vorab definierten Vorlagen (Templates) basieren. Die Auswahl des Verfahrens hängt stark vom Anwendungskontext, der verfügbaren Datenbasis sowie den Anforderungen an Robustheit und Rechenaufwand ab.

2.3.1 Lernbasierte Verfahren

Lernbasierte Verfahren setzen auf die automatische Generalisierung von Gesten aus Trainingsdaten. Dabei kommen typischerweise Merkmalsvektoren zum Einsatz, die z. B. Positionen oder Winkel zwischen Körperpunkten über die Zeit darstellen.

Hidden Markov Models (HMMs)

HMMs modellieren Gesten als stochastische Prozesse mit latenten Zuständen. Diese Zustände entsprechen typischerweise diskreten Phasen einer Geste. Aufgrund ihrer Fähigkeit, zeitliche Abhängigkeiten und Übergangswahrscheinlichkeiten zu erfassen, waren HMMs lange Zeit ein guter Ansatz in der Gestenerkennung [25].

Convolutional Neural Networks (CNNs)

CNNs eignen sich besonders für bildbasierte Gestenerkennung, bei der aus einer Sequenz von Kamerabildern oder Keypoint-Heatmaps Features extrahiert werden. Durch ihre Fähigkeit, translational invariant zu arbeiten, erfassen CNNs robuste Merkmale in unterschiedlichen Perspektiven [24].

Long Short-Term Memory (LSTM)

LSTMs sind rekurrente neuronale Netze, die speziell für sequenzielle Daten entwickelt wurden. Sie können langfristige Abhängigkeiten erkennen und eignen sich daher hervorragend für die Modellierung dynamischer Gestenverläufe. Ein LSTM kann z. B. eine Abfolge von 3D-Koordinaten einzelner Gelenke analysieren und aus dieser Zeitreihe eine Geste klassifizieren [24].

2.3.2 Template-basierte Verfahren

Im Gegensatz zu lernbasierten Verfahren erfordern Template-basierte Methoden kein aufwändiges Training. Stattdessen werden beispielhafte Bewegungsabläufe (Templates) definiert, mit denen neue Eingaben direkt verglichen werden.

Dynamic Time Warping

Ein populäres Ähnlichkeitsmaß für zeitabhängige Daten ist Dynamic Time Warping. Es erlaubt den Vergleich von Bewegungssequenzen unterschiedlicher Länge und Geschwindigkeit, indem es zeitliche Verzerrungen kompensiert. Dynamic Time Warping berechnet die optimale Zuordnung zweier Zeitreihen unter Berücksichtigung ihrer temporalen

Struktur [3]. Dies ist besonders nützlich, da Nutzer dieselbe Geste mit variierender Geschwindigkeit oder leicht veränderter Ausführung ausführen können.

Zur Anwendung von Dynamic Time Warping auf Gesten werden die Keypoint-Daten (z. B. die 3D-Positionen von Hand- oder Armgelenken) über die Zeit erfasst. Eine neue Eingabe wird dann gegen eine Bibliothek von Referenzgesten verglichen. Diejenige mit der geringsten DTW-Distanz gilt als erkannte Geste. Vorteilhaft ist hierbei, dass auch mit wenigen Beispielgesten eine funktionierende Erkennung aufgebaut werden kann. Die Nachteile bestehen in der Rechenintensität bei großen Template-Mengen sowie in der geringeren Generalisierungsfähigkeit bei stark variierenden Eingaben.

Um die Erkennung zu verbessern, wird Dynamic Time Warping häufig mit Vorverarbeitungsschritten kombiniert, z. B.:

- Normalisierung der Gestenlänge durch Resampling,
- Filterung der Bewegungssignale (z. B. Butterworth-Filter zur Glättung),
- Reduktion auf Merkmale, z. B. nur Handgelenkspositionen oder Winkel zwischen Gelenken.

2.3.3 Vergleich der Verfahren

Tabelle 2.1: Vergleich verschiedener Verfahren zur Gestenerkennung

Verfahren	**Stärken**	**Schwächen**
Hidden Markov Models (HMMs)	- Gut für sequentielle Daten - Robust bei leichtem Rauschen	- Benötigt viele Trainingsdaten - Geringe Flexibilität bei komplexen Bewegungen
Convolutional Neural Networks (CNNs)	- Sehr gute Feature-Extraktion aus Bilddaten - Gute Ergebnisse bei statischen Gesten	- Nicht optimal für zeitabhängige Sequenzen - Hoher Rechenaufwand
Long Short-Term Memory (LSTM)	- Gut für dynamische Gesten (zeitliche Abhängigkeit) - Kann lange Sequenzen verarbeiten	- Braucht große Datenmengen zum Trainieren - Gefahr von Overfitting
Template Matching mit DTW	- Intuitiv und einfach umsetzbar - Keine Trainingsphase nötig - Geringe Latenz	- Sensitiv gegenüber Rauschen und Start-/Endpunkt - Kein Lernverhalten möglich

2.4 Rahmenbedingungen

Für die erfolgreiche Realisierung einer robusten gestenbasierten Interaktion müssen bestimmte Rahmenbedingungen erfüllt sein, die unabhängig von der konkreten technischen Implementierung gelten. Diese betreffen sowohl die räumliche Umgebung als auch die Interaktion des Nutzers mit dem System.

Zunächst wird davon ausgegangen, dass die Kamera den Nutzer in einem festen Winkel und mit freier Sicht erfassen kann. Nur wenn Oberkörper und Hände vollständig sichtbar sind, ist eine zuverlässige Gestenerkennung möglich. Zudem ist ein bestimmter Interaktionsbereich erforderlich: Der Nutzer sollte sich in einem Abstand von etwa ein bis zwei Metern zur Kamera befinden und überwiegend frontal ausgerichtet sein.

Ein weiterer zentraler Faktor sind die Lichtverhältnisse. Die Bildverarbeitung setzt eine gleichmäßige und ausreichend helle Beleuchtung voraus, während starke Schatten oder dynamisch wechselndes Licht die Erkennungsleistung erheblich beeinträchtigen können. Ebenso ist es notwendig, dass sich im Erfassungsbereich keine Objekte oder Personen zwischen Kamera und Nutzer befinden, die Körperteile verdecken oder falsche Signale erzeugen könnten. Auch der Hintergrund sollte möglichst statisch und neutral bleiben, um unnötige Störungen der Algorithmen zu vermeiden.

Das System ist auf die Interaktion mit einer einzelnen Person ausgelegt. Bewegungen mehrerer Personen, stark dynamische Szenen oder Gesten mit Körperteilen wie Beinen oder Gesicht werden in der aktuellen Ausgestaltung nicht berücksichtigt und bilden ein mögliches Feld für zukünftige Erweiterungen.

Diese Rahmenbedingungen definieren somit den notwendigen Kontext, in dem das System verlässlich arbeiten kann. Werden sie missachtet — etwa durch schlechte Beleuchtung oder verdeckte Körperteile — ist mit einer deutlichen Beeinträchtigung der Funktionsweise zu rechnen.

2.5 Fazit der Analyse und technologisches Umsetzungsziel

Die durchgeführte Analyse hat gezeigt, dass eine robuste Gestensteuerung im Wesentlichen auf zwei technologische Kernkomponenten angewiesen ist: zum einen auf die zuverlässige Erfassung der Körperpose, zum anderen auf die algorithmische Verarbeitung und Erkennung der daraus resultierenden Bewegungsdaten. Aus diesen Erkenntnissen lassen

sich die technologischen Schwerpunkte ableiten, die in dieser Arbeit gezielt weiterverfolgt werden.

Die Pose Estimation bildet dabei die unverzichtbare Grundlage jeder gestenbasierten Interaktion. Nur wenn relevante Körperpunkte, insbesondere Schulter, Arm und Hand, präzise im Raum lokalisiert werden, ist eine verlässliche Interpretation von Gesten möglich. Für die Umsetzung dieser Arbeit wird daher eine Stereokamera mit integrierter Pose Estimation eingesetzt, die es erlaubt, neben den zweidimensionalen Bildkoordinaten auch Tiefeninformationen zu erfassen. Im Gegensatz zu monokularen Kameras kann auf diese Weise die räumliche Komponente eines Gestenverlaufs berücksichtigt werden. Gerade für zielgerichtete Selektionsgesten, wie das Zeigen auf Objekte im Raum, ist diese Tiefendimension von zentraler Bedeutung. Sie reduziert perspektivische Verzerrungen und erlaubt eine präzisere Rekonstruktion der Bewegungsbahnen.

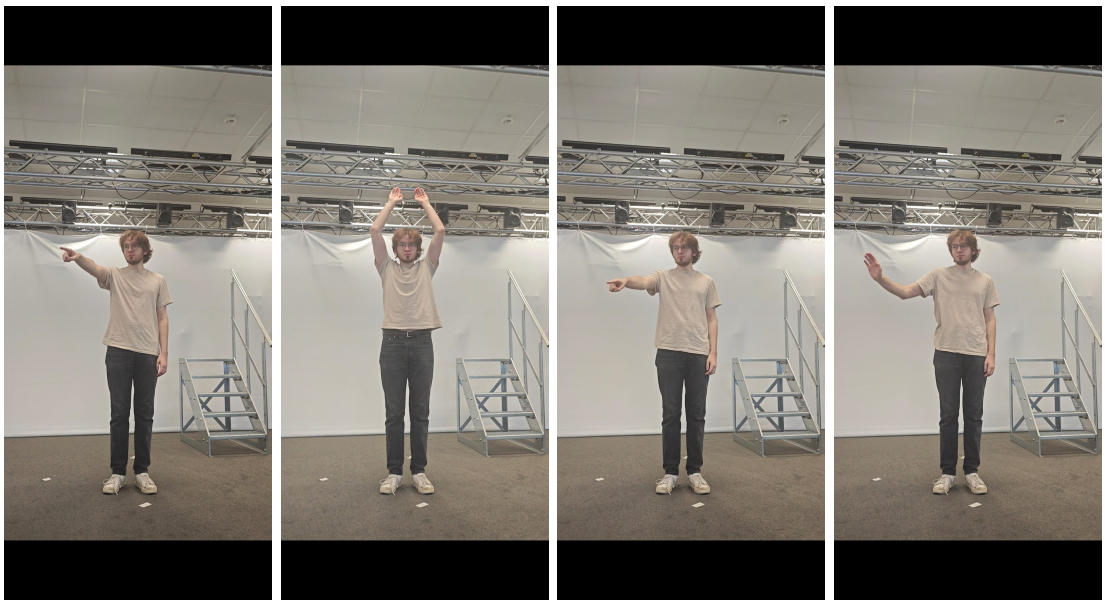
Aufbauend auf den gewonnenen Pose-Daten ist ein geeigneter Klassifikationsmechanismus erforderlich, um wiederkehrende Gesten zuverlässig zu erkennen. Hierbei hat sich insbesondere das Verfahren des Dynamic Time Warping als geeignet herausgestellt. Dynamic Time Warping ermöglicht den Vergleich zeitabhängiger Bewegungssequenzen, selbst wenn diese mit unterschiedlicher Geschwindigkeit oder leicht variierender Ausführung vorgenommen werden. Durch die Fähigkeit, zeitliche Verschiebungen und nichtlineare Dehnungen in den Bewegungsabläufen auszugleichen, bietet Dynamic Time Warping eine robuste Grundlage für die Erkennung konsistenter Gestenmuster.

Das Ergebnis dieser Analyse ist somit ein klar umrissenes technologisches Umsetzungsziel: Die Kombination einer dreidimensionalen Pose-Erfassung mittels Stereokamera mit einer sequenzbasierten Gestenerkennung auf Basis von Dynamic Time Warping. Diese beiden Bausteine bilden das technische Fundament der vorliegenden Arbeit und ermöglichen die Entwicklung eines Systems, das Bewegungen präzise erfasst und erkennt. Damit wird die Grundlage für eine natürliche und alltagstaugliche Gestensteuerung geschaffen.

3 Design

3.1 Gesamtkonzept

Das entwickelte System ermöglicht eine berührungslose Steuerung von Smarthome-Geräten durch Gesteninteraktion. Ein kompletter Ablauf (schematisch dargestellt in Abbildung 3.1) besteht dabei aus einer Selektionsgeste zur Auswahl eines Objektes, gefolgt von einer Manipulationsgeste. Dabei wird die zentrale Logik durch eine modulare Architektur realisiert, die sich in mehrere Verantwortlichkeitsbereiche gliedert. Abbildung 3.2 zeigt die grundlegende Nutzungsarchitektur.



(a) (b) (c) (d)

Abbildung 3.1: Ablauf zwei aufeinanderfolgender Gestenerkennungen. Auswahl eines Objekts (a) mit anschließender „Öffnen“- Geste (b). Danach Auswahl eines niedriger gestapelten Objekts (c) mit anschließender „Wischen“- Geste (d).

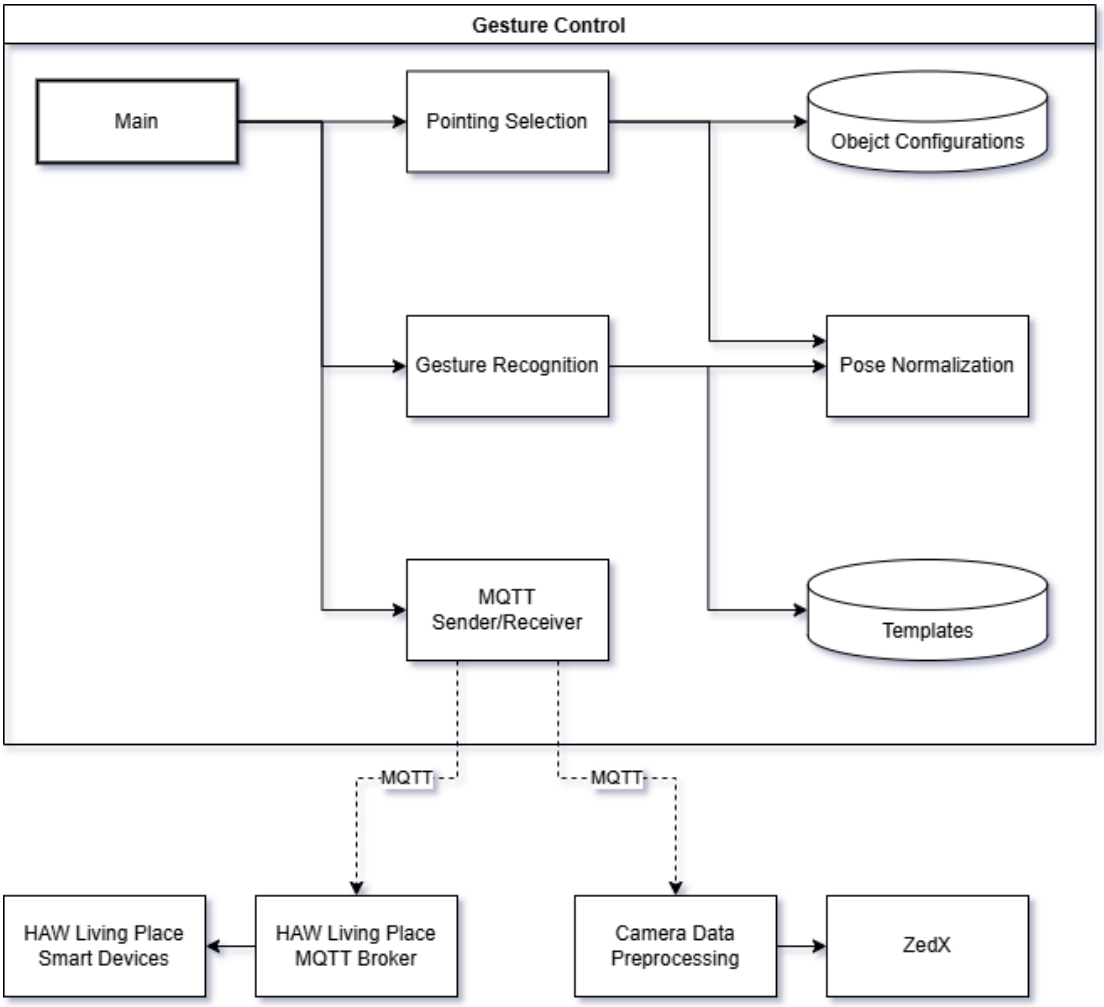


Abbildung 3.2: Schematische Darstellung der Systemarchitektur mit Pfeilen zur Darstellung von Nutzungsbeziehung.

Dies spiegelt sich auch in der Dateistruktur des Projekts wieder:

- `main.py`: Startpunkt der Anwendung, Koordination der Module und Initialisierung aller Komponenten.
- `pointing_selection.py`: Erkennung und Auswahl eines Geräteziels durch Zeigebewegungen anhand übergebener Posendaten.
- `gesture_recognition.py`: Analyse von Körperbewegungen und Erkennung von Manipulationsgesten mittels Dynamic Time Warping.
- `utils/normalization.py`: Funktionen zur Vorverarbeitung und Normalisierung von Gesten-Trajektorien.
- `templates/`: Gespeicherte Gestenvorlagen als Vergleichsdaten für das Matching (Öffnen, Winken, etc.).
- `config/object_config.json`: Definition der im Raum verfügbaren steuerbaren Objekte inklusive ihrer 3D-Position.

Das System folgt einem zweistufigen Interaktionsmodell, bestehend aus einer Selektionsgeste zur Zielbestimmung und einer anschließenden Manipulationsgeste zur Interaktion. Die Architektur ermöglicht eine klare Trennung von Erkennung, Interpretation und Aktorsteuerung. Die Kommunikation mit externen Geräten erfolgt über das MQTT-Protokoll.

Die folgenden Abschnitte beleuchten den Aufbau und die einzelnen Komponenten im Detail.

3.2 Aufbau und Laborbedingungen

Die Umsetzung der zuvor in Abschnitt 2.4 beschriebenen Rahmenbedingungen erfolgte in einer kontrollierten Laborumgebung am Creative Space for Technical Innovation (CSTI) der HAW. Der Aufbau orientierte sich daran, eine möglichst realitätsnahe, aber gleichzeitig reproduzierbare Testumgebung für die Evaluierung des Systems zu schaffen.

Zentraler Bestandteil war die Verwendung einer Stereolabs ZED X Tiefenkamera, die in einem festen Winkel von etwa 30 Grad zur Blickrichtung des Nutzers montiert wurde. Die Positionierung erfolgte in einer Höhe von etwa zwei Metern, sodass Oberkörper und

Hände des Nutzers im gesamten Interaktionsbereich erfasst werden konnten. Die Kamera blieb während aller Tests fixiert, um eine konsistente Datengrundlage zu gewährleisten. Eine Neuausrichtung oder Rekalibrierung war nicht erforderlich.

Für die Lichtverhältnisse wurde auf eine gleichmäßige und diffuse Beleuchtung geachtet. Direktes Gegenlicht, harte Schatten oder wechselnde Lichtquellen wurden vermieden. Der Hintergrund des Laboraufbaus war weitgehend neutral und frei von Bewegungen, um Störungen bei der Gestenerkennung zu reduzieren.

Die Nutzer führten ihre Gesten innerhalb eines klar definierten Interaktionsvolumens aus, das bei der Template-Erstellung berücksichtigt wurde. Der Abstand zur Kamera lag dabei stets zwischen einem und zwei Metern. Störungen durch weitere Personen oder bewegliche Objekte im Vordergrund wurden ausgeschlossen.

Mit diesem in Abbildung 3.3 sichtbaren Aufbau konnte sichergestellt werden, dass die zuvor formulierten Rahmenbedingungen weitgehend erfüllt wurden. Damit war es möglich, die Leistungsfähigkeit des Systems gezielt zu untersuchen, ohne dass äußere Störfaktoren die Ergebnisse verzerrten.

3.3 Initialisierung

Vor dem eigentlichen Betrieb des Systems ist eine einmalige Initialisierung erforderlich, um die Selektions- und Manipulationsgesten an die spezifische Umgebung anzupassen. Diese Initialisierungsphase umfasst drei wesentliche Schritte: die Definition des Koordinatensystems, die Aufnahme von Gesten-Templates sowie die Festlegung der im Raum vorhandenen Objekte.

Definition des Koordinatensystems

Zunächst wird ein einheitliches Koordinatensystem festgelegt, das als Bezugsrahmen für alle weiteren Berechnungen dient. In der vorliegenden Implementierung handelt es sich um ein kartesisches, rechtsdrehendes 3D-Koordinatensystem, bei dem die x-Achse horizontal, die y-Achse vertikal und die z-Achse in Richtung der Kamera verläuft. Dieses Koordinatensystem wird hauptsächlich für die räumliche Einordnung der Objekte verwendet. Die Dimensionen des Koordinatensystems sind dabei von der ZedX-Kamera bestimmt und basieren, angegeben durch Konfiguration beim Einrichten der Kamera, auf

Abbildung 3.3: Aufbau der ersten Testumgebung mit Koordinatenmarkierungen. Fotografiert von direkt oberhalb der Stereokamera

dem metrischen System. Für jeden Meter ändert sich der entsprechende Wert der Achse um 0,5. Den Nullpunkt des Systems bildet die Kamera.

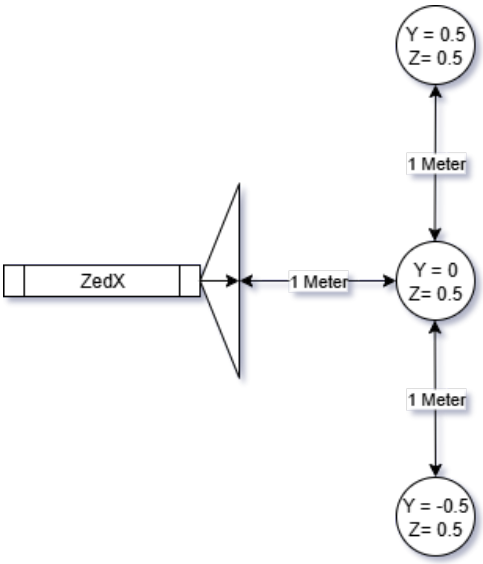


Abbildung 3.4: Schematische Darstellung des Koordinatensystems aus der Seitenperspektive.

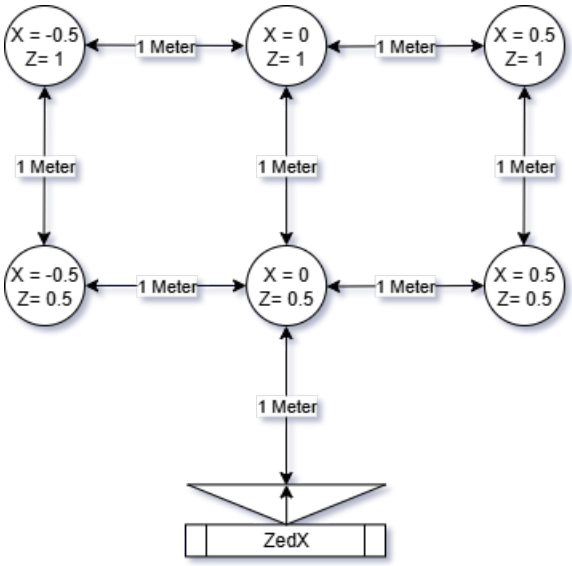


Abbildung 3.5: Schematische Darstellung des Koordinatensystems aus der Vogelperspektive.

Aufnahme von Gesten-Templates

Im zweiten Schritt erfolgt die Aufnahme von Referenzdaten für die zu erkennenden Gesten. Hierzu wird das Programm `recorder.py` eingesetzt, welches als eigenständiger Datenlogger konzipiert ist und über das MQTT-Protokoll fortlaufend Bild- oder Sensordaten von der Kamera entgegennimmt. Die Kommunikation erfolgt dabei über einen konfigurierbaren MQTT-Broker, wobei das Skript beim Start automatisch eine Verbindung aufbaut und das relevante Topic abonniert.

Nach der Initialisierung fragt das Programm den Benutzer interaktiv nach einem Zielordner und einem Basisdateinamen. Existiert der Zielordner nicht, wird er automatisch erstellt. Zusätzlich wird im Setup-Schritt die zu sendende MQTT-Nachricht konfiguriert, die später beim Erkennen einer Geste verwendet wird. Hierbei lassen sich Topic, Payload-Struktur (Text, JSON oder benutzerdefiniert), QoS-Stufe und Retain-Flag festlegen.

Der eigentliche Aufnahmeprozess wird in einem separaten Thread (`recording_loop`) realisiert. Die Aufnahme erfolgt in kurzen, zeitlich begrenzten Sequenzen (standardmäßig drei Sekunden), die durch eine interaktive Eingabe gestartet und gestoppt werden können. Vor Beginn einer Aufnahme wird ein Countdown angezeigt, um dem Benutzer Zeit zur Positionierung zu geben. Innerhalb der Aufnahmedauer werden alle empfangenen MQTT-Datenpakete fortlaufend in eine Textdatei geschrieben. Jede Datei wird mit einem inkrementellen Zähler versehen. Die verwendete ZedX-Kamera liefert je nach Konfiguration Body-Tracking Bilder mit verschiedenen Mengen an Keypoints. In dieser Arbeit wird das in Abbildung 3.6 dargestellte Modell mit 34 verschiedenen Punkten verwendet.

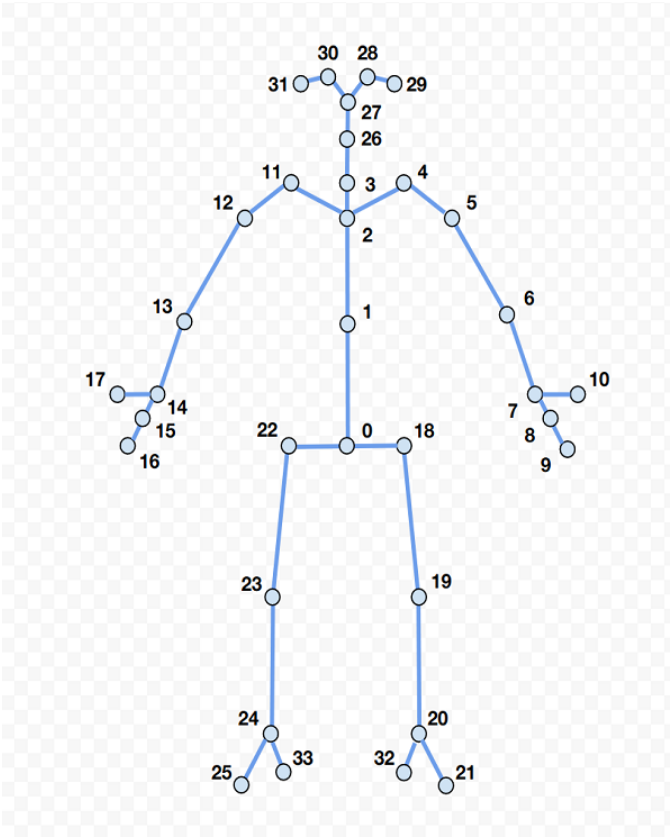


Abbildung 3.6: Index der Pose-Keypoints der ZedX-Kamera. Quelle: [22]

Durch diese Vorgehensweise entsteht für jede Geste eine Sammlung mehrerer leicht variierender Ausführungen, die Unterschiede in Bewegungsgeschwindigkeit, Handposition oder minimaler Orientierung abdecken. Dies erhöht die Robustheit des späteren Erkennungsalgorithmus erheblich, da er auf eine breitere Datenbasis zurückgreifen kann.

Definition der AABB-Objekte

Abschließend erfolgt die Definition aller Objekte, die sich im Erkennungsbereich befinden und später für die Interaktion relevant sind. Jedes Objekt wird dabei durch eine sogenannte *Axis-Aligned Bounding Box* (AABB) beschrieben. Eine AABB ist ein quaderförmiges Volumen im dreidimensionalen Raum, dessen Flächen stets parallel zu den Achsen des zugrunde liegenden Koordinatensystems ausgerichtet sind. Diese Eigenschaft vereinfacht geometrische Berechnungen erheblich, da die Begrenzungen entlang der x-, y- und z-Achse jeweils nur durch ein Minimum und ein Maximum beschrieben werden müssen.

Zur internen Darstellung wird jedes Objekt mit genau acht 3D-Eckpunkten hinterlegt, die den Quader vollständig definieren. Diese Eckpunkte ergeben sich aus allen möglichen Kombinationen der minimalen und maximalen Koordinatenwerte:

$$P(x, y, z) \in \{x_{\min}, x_{\max}\} \times \{y_{\min}, y_{\max}\} \times \{z_{\min}, z_{\max}\}. \tag{3.1}$$

Die Spezifikation der AABBs erfolgt in einer JSON-Konfigurationsdatei, die beim Start des Systems geladen wird. Jedes Objekt erhält dabei einen eindeutigen Bezeichner, die minimalen und maximalen Koordinatenwerte in Metern sowie einen Namen. Dieser Ansatz ermöglicht eine einfache Anpassung des Szenenlayouts, ohne den Quellcode ändern zu müssen, und stellt sicher, dass alle nachfolgenden Algorithmen über ein konsistentes und normiertes Geometriemodell verfügen.

3.4 Vorverarbeitung der Pose-Daten

Ein zentraler Bestandteil der nachfolgend beschriebenen Selektions- und Manipulationsgestenerkennung ist die Vorverarbeitung der von der Kamera erfassten Rohdaten. Die Funktion `normalize_pose_frame(nodes, isPointing = False)` übernimmt dabei die Aufgabe, die im Buffer gespeicherten eingehenden Pose-Nachrichten in eine für den Erkennungsalgorithmus geeignete Form zu überführen. Ziel ist es, aus der Folge von 3D-Koordinaten der Gelenkpunkte eine normalisierte und geglättete Zeitreihe zu erzeugen, die Bewegungsmuster zuverlässig repräsentiert und gleichzeitig robust gegenüber Störeinflüssen wie Rauschen, variierenden Körpergrößen oder leichten Kamera-Verschiebungen ist.

Die Verarbeitung erfolgt in mehreren Schritten. Zunächst wird jede Pose-Nachricht in ein numerisches Format überführt, wobei nur eine definierte Auswahl relevanter Gelenkpunkte berücksichtigt wird. Bei den ausgewählten Punkten handelt es sich ausschließlich um die Punkte des Oberkörpers und der Arme. Positionsdaten des Kopfes und des Unterkörpers werden entfernt, wodurch diese als Störfaktor ausgeschlossen werden können.

Um Translationen im Raum bei Manipulationsgesten zu kompensieren, wird ein bestimmter Gelenkpunkt (In der vorliegenden Implementation der Punkt der das Becken repräsentiert) als Referenz gewählt. Sei p_r der Referenzpunkt (Index r). Dann ergeben sich die zentrierten Posen p aus:

$$p'_i = p_i - p_r \quad \forall i. \tag{3.2}$$

Um bei Manipulationsgesten zusätzlich Unterschiede in der Körpergröße auszugleichen, wird der Abstand zwischen dem linken und rechten Schultergelenk als Normgröße herangezogen. Dieses Vorgehen wird zum Beispiel in [8] und [20] genutzt da es sich eignet um Körpergröße zu normieren, unter der Voraussetzung dass Nutzer in ihrer Rotation immer etwa gleich ausgerichtet sind.

Sei p_{ls} die linke und p_{rs} die rechte Schulter. Der Skalierungsfaktor ist dann

$$s = \|p'_{rs} - p'_{ls}\|_2, \tag{3.3}$$

wobei $\|\cdot\|_2$ die euklidische Norm bezeichnet. Die skalierten Punkte ergeben sich durch

$$\tilde{p}_i = \frac{p'_i}{s}. \quad \forall i. \tag{3.4}$$

Da Unterschiede in der Körpergröße für die Erkennung von Selektionsgesten irrelevant und genaue Informationen über die Position im Raum sogar zwingend notwendig sind, werden diese beiden Normalisierungsschritte bei der Vorbereitung dieser Daten durch Übergabe von `isPointing = True` übersprungen.

Die potenziell normalisierte Sequenz $\tilde{p}_i(t)$ enthält noch hochfrequentes Rauschen. Um dieses zu reduzieren, wird ein digitaler Butterworth-Tiefpassfilter angewendet [6]. Der Filter der Ordnung n mit Grenzfrequenz ω_c besitzt die Übertragungsfunktion

$$H(\omega) = \frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_c}\right)^{2n}}}. \tag{3.5}$$

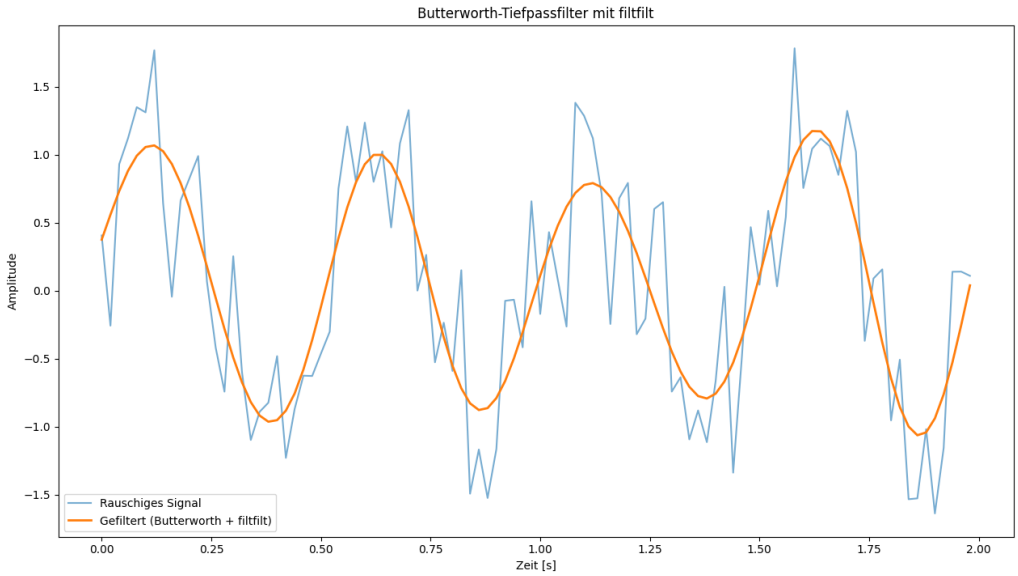


Abbildung 3.7: Effekt des Butterworth-basierten `filtfilt()` Verfahren am Beispiel einer Sinuskurve mit eingebautem Rauschen. Erstellt mit `matplotlib`.

Die Werte für n und ω_c wurden durch systematisches Testen standardmäßig auf $n = 3$ und $\omega_c = 0.175$ festgelegt. Das Design des Filters erfolgt durch die butter(n,ω_c) Funktion. Diese entwirft einen digitalen Butterworth-Tiefpassfilter n-ter Ordnung mit normalisierter Grenzfrequenz ω_c. Die Funktion butter gibt die Zähler- und Nennerkoeffizienten (a und b) der Filterübertragungsfunktion zurück.

Die Filterung der Pose-Sequenz erfolgt über das `filtfilt()`-Verfahren, das das Signal vorwärts und rückwärts filtert, um Phasenverschiebungen zu vermeiden:

$$\hat{p}_i(t) = \texttt{filtfilt}(b, a, \tilde{p}_i(t)). \tag{3.6}$$

wobei a und b das Resultat der `butter()` Funktion mit den zuvor genannten Parametern sind. Der Effekt dieses Filters wurde in Abbildung 3.7 schematisch mit einer Sinuskurve verdeutlicht.

Nach diesen Verarbeitungsschritten liegt eine geglättete Sequenz vor, die die zeitliche Bewegung der relevanten Gelenkpunkte beschreibt. Diese normalisierte und rauschreduzierte Darstellung bildet die Grundlage für die anschließende Gestenerkennung.

3.5 Selektion durch Zeigegesten

Um eine natürliche und berührungslose Interaktion zu ermöglichen, erfolgt die Auswahl von Objekten durch Zeigegesten. Die zugrunde liegende Implementierung befindet sich im Modul `PointingSelector`, das kontinuierlich übergebene Körperposendaten auswertet, um stabile Zeigegesten zu erkennen und mit bekannten Objekten im Raum zu verknüpfen.

3.5.1 Erkennung der Zeigegeste

Die Methode `check_pointing(raw_poses)` ist zentral für die Erkennung von Zeigegesten. Sie nimmt eine Liste von erfassten Posen entgegen, extrahiert aus der letzten Pose die Position von rechtem Schultergelenk (Index 12) und rechter Hand (Index 15) und berechnet daraus einen Richtungsvektor:

$$\vec{d} = \vec{p}_{\text{Hand}} - \vec{p}_{\text{Schulter}}. \tag{3.7}$$

Dieser Vektor $\vec{d}$ wird anschließend normalisiert und als Zeigerichtung interpretiert. Um zufällige oder unabsichtliche Bewegungen herauszufiltern, speichert das System alle Zeigerichtungen und Ursprungspositionen (Schulterpunkte) innerhalb eines konfigurierbaren Zeitfensters (`pointing_history_window`, standardmäßig 2 Sekunden) in einem `deque`-Puffer.

3.5.2 Stabilitätsprüfung

Zur Bestimmung einer stabilen Zeigegeste werden zwei Hauptkriterien herangezogen:

- **Richtungsstabilität:** Die Methode `angle_between(v1, v2)` dient der Berechnung des Winkels zwischen einzelnen Richtungsvektoren und dem gemittelten Richtungsvektor. Die Zeigerrichtung wird nur dann als stabil gewertet und beibehalten, wenn die Standardabweichung dieser Winkel unter dem Schwellwert `stability_stddev_threshold_deg` (z. B. 10°) liegt. Andernfalls erfolgt ein Herausfiltern des Werts. Im Falle einer unzureichenden Anzahl an verfügbaren Richtungsvektoren wird die Funktion unterbrochen.

- **Ursprungsstabilität:** Ebenso wird geprüft, ob sich die Ursprungspunkte (Schulterpositionen) innerhalb eines Radius von `outlier_origin_threshold_m` (z. B. 10 cm) um den Mittelwert befinden. Nur wenn auch hier nach dem Filtern genügend stabile Ursprünge vorhanden sind, wird die Geste weiterverarbeitet.

3.5.3 Objektauswahl durch Strahl-Schnittprüfung

Wird eine stabile Zeigegeste erkannt, erfolgt durch die Methode `check_pointing_intersection(origin, direction)` eine geometrische Überprüfung, ob der Zeigestrahl mit einem der bekannten Objekte im Raum kollidiert. Die Objekte werden beim Start des Systems über `load_objects()` aus einer Konfigurationsdatei geladen.

Die eigentliche Schnittprüfung wird durch die Methode `ray_box_intersect(ray_origin, ray_dir, box_min, box_max)` mittels des sogenannten Slab-Verfahrens durchgeführt [15].

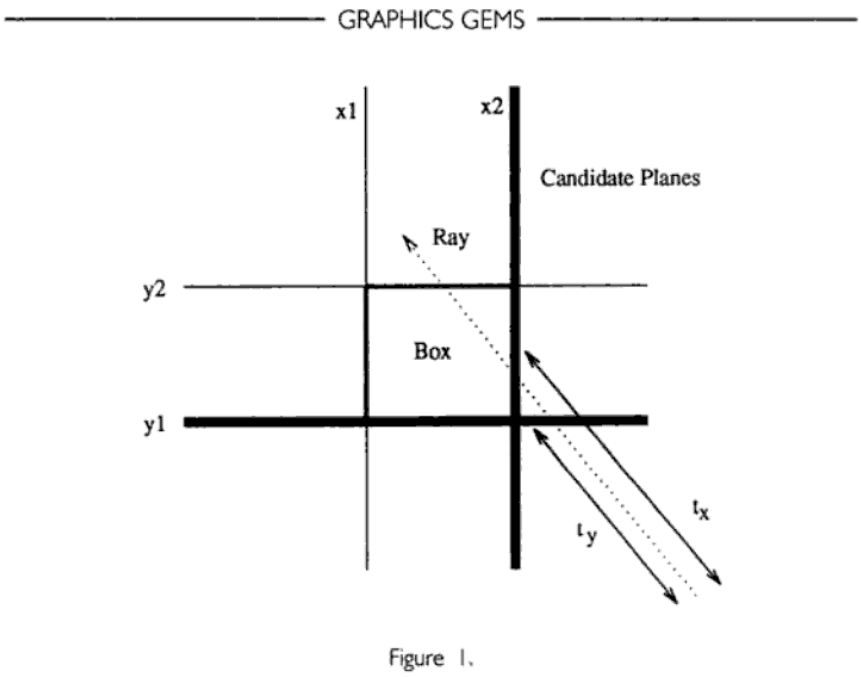


Abbildung 3.8: Algorithmische Illustration für eine 2D-AABB, entnommen aus [10, S. 396].

Das Slab-Vefahren basiert auf dem Prinzip, dass eine AABB als Schnittmenge dreier sogenannter „Slabs“ interpretiert werden kann, also dreier Volumenbereiche, die jeweils zwischen zwei parallelen Ebenen entlang der X-, Y- und Z-Achse definiert sind. Für jede dieser Achsen wird der Eintritts- und Austrittszeitpunkt des Strahls in das jeweilige Intervall berechnet. Diese Zeitpunkte (engl. *t-values*) ergeben sich durch die Lösung einfacher linearer Gleichungen:

$$t_{min} = \frac{(box_{min} - ray_{origin})}{ray_{direction}}, \quad t_{max} = \frac{(box_{max} - ray_{origin})}{ray_{direction}}. \tag{3.8}$$

$$t_{\min}^{\text{global}} = \max(t_{\min,x},\ t_{\min,y},\ t_{\min,z}). \tag{3.9}$$

$$t_{\max}^{\text{global}} = \min(t_{\max,x},\ t_{\max,y},\ t_{\max,z}). \tag{3.10}$$

Die für jede Achse berechneten Eintritts- und Austrittszeitpunkte ($t_{\min}$ und $t_{\max}$) bilden jeweils ein Intervall, in dem sich der Strahl innerhalb der Box entlang der entsprechenden Achse befindet. Um zu bestimmen, ob ein tatsächlicher Schnitt mit dem Volumenkörper der Box vorliegt, werden diese Intervalle über alle drei Achsen hinweg aggregiert: Der späteste Eintrittszeitpunkt über alle Achsen ($t_{\min}^{global}$) und der früheste Austrittszeitpunkt ($t_{\max}^{global}$) werden berechnet. Nur wenn der Eintritt in das Volumen vor dem Austritt erfolgt – also $t_{\min}^{global} \leq t_{\max}^{global}$ – existiert ein Schnittpunkt mit dem Objekt. Dieses Vorgehen stellt sicher, dass der Strahl die Box in allen Raumrichtungen gleichzeitig durchquert und keine Kollision außerhalb des gemeinsamen Raumbereichs fälschlich erkannt wird.

Dieses Verfahren zeichnet sich durch seine Effizienz und Einfachheit aus, da es ohne komplexe Geometrieoperationen auskommt. Es eignet sich besonders gut für Axis-Aligned Bounding Boxs, wie sie in dieser Arbeit zur Darstellung der Zielobjekte verwendet werden.

3.5.4 Verhalten bei erfolgreicher Selektion

Trifft der berechnete Zeigestrahl auf ein Objekt, wird dessen Bezeichner als `pointed_object` zurückgegeben. Das Hauptprogramm (`main.py`) wechselt daraufhin in einen Zustand, in dem eine Manipulationsgeste erwartet wird. Gleichzeitig wird ein Zeitlimit (z. B. 10 Sekunden) gesetzt, nach dessen Ablauf die Auswahl zurückgesetzt wird, falls keine Geste erkannt wurde.

3.5.5 Vorteile des Verfahrens

Dieses Verfahren ermöglicht eine robuste und intuitive Objektauswahl allein durch Körpergesten, ohne zusätzliche Geräte oder Sprachsteuerung. Die Kombination aus historienbasierter Stabilitätsprüfung und geometrischer Schnittprüfung sorgt dabei für hohe Genauigkeit bei gleichzeitig geringer Fehlerrate. Die Trennung zwischen Selektion und Manipulation erlaubt zudem eine klare Interaktionsstruktur.

3.6 Gestenerkennung mittels Dynamic Time Warping

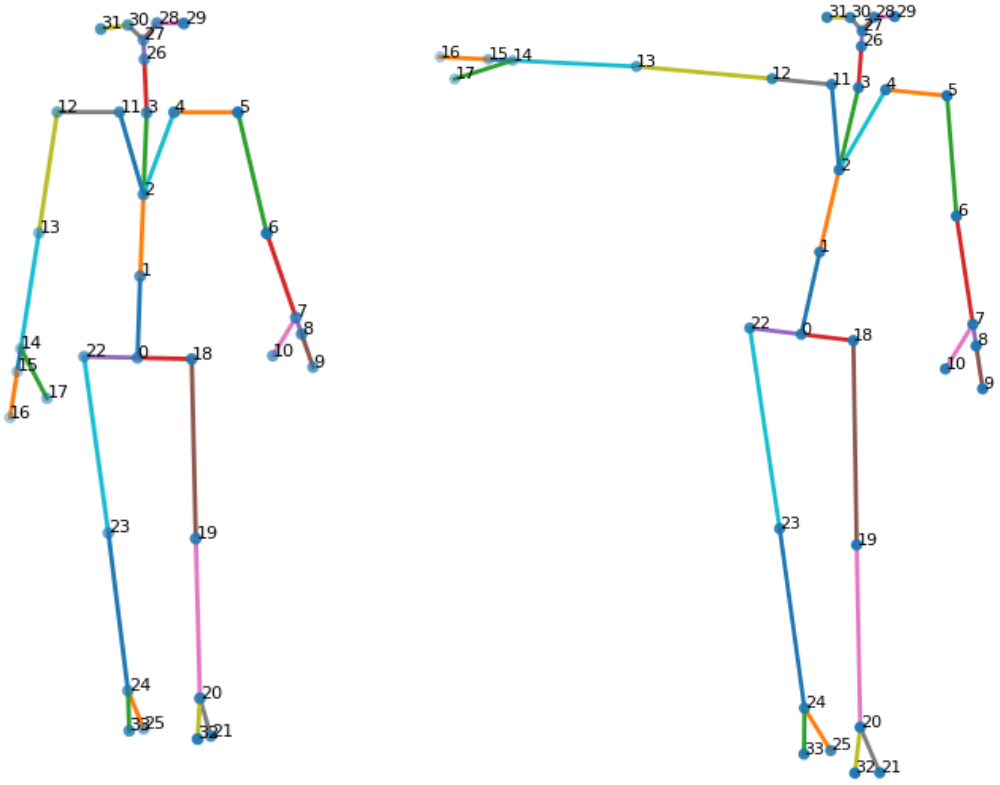
Nach der Objektauswahl über eine stabile Zeigegeste erwartet das System eine nachfolgende Interaktionsgeste. Die Erkennung dieser Gesten erfolgt über das Modul `GestureRecognizer`, welches eine sequenzbasierte Klassifikation unter Verwendung von Dynamic Time Warping durchführt. Die Auswahl der zu analysierenden Sequenzen erfolgt durch ein sogenanntes „Sliding Window“. Dies bedeutet das eine definierte Menge an Momentaufnahmen aus der Kamera zu einer Sequenz zusammengefasst und daraufhin überprüft werden. Sollte in diesem Fenster keine Geste erkannt werden, werden die ersten paar Einträge entfernt und dafür am Ende neue Einträge angehangen, wodurch eine neue Sequenz entsteht, welche zeitlich um einen kurzen Moment verschoben ist.

3.6.1 Gestenvergleich durch Dynamic Time Warping

Die eigentliche Gestenerkennung erfolgt durch die Methode `recognize_gesture(sequence)`, welche die Eingabesequenz mit einer Sammlung vordefinierter Templates vergleicht. Jedes Template entspricht einer gespeicherten Ausführung einer bestimmten Geste, z. B. „Winken“, „Arme heben“, etc.

Zur Vergleichbarkeit unterschiedlich langer oder leicht zeitversetzter Bewegungen wird die DTW-Distanz als Ähnlichkeitsmaß verwendet. Die verwendete Implementierung `fastdtw` berechnet die minimale kumulierte Distanz zwischen zwei Sequenzen:

$$DTW(S_1, S_2) = \min_{p \in \mathcal{P}} \sum_{(i,j) \in p} d(s_{1i}, s_{2j}). \tag{3.11}$$



(a) (b)

Abbildung 3.9: Mittels `matplotlib` wiedergegebene Template Momentaufnahmen der „Wischen“-Geste

Dabei ist $\mathcal{P}$ die Menge möglicher Pfade durch das DTW-Matrixfeld und d die euklidische Distanz zwischen Posen s_{1i} und s_{2j}. Diese grundlegende Formel des Dynamic-Time-Warping wird durch einen mehrstufigen schnellen Approximationsprozess erweitert, wie in der Arbeit bezüglich FastDTW beschrieben [19]. Dadurch wird eine zeitliche Verbesserung von $O(N^2)$ auf $O(N)$ erzielt. Ein Nachteil dieser Implementation besteht darin, dass der Optimalpfad nicht in jedem Fall gefunden wird. Aufgrund der garantierten Findung eines nahezu optimalen Pfads innerhalb eines äußerst kurzen Zeitrahmens erweist sich diese Version des Dynamic Time Warping aber dennoch vorteilhaft für eine interaktive Gestenerkennung.

Der aus diesen Pfad-Ergebnissen entstehende Entscheidungsprozess ist zweistufig aufgebaut:

1. **Bestimmung des besten Treffers:** Für jede verfügbare Template-Sequenz wird der DTW-Abstand zur Eingabesequenz berechnet und durch die Länge des Pfads geteilt. Durch diese Division kann sichergestellt werden das die Länge der Sequenz und die Länge des verglichenen Templates keinen Einfluss auf den schlussendlichen Rückgabe Wert des DTW-Vergleiches haben.

   Nachdem iterativ alle geladenen Templates durchgetestet wurden, besteht die Rückgabe der `recognize_gesture(sequence)` Funktion aus einem Tupel mit

   - `best_match_gesture` - eindeutige Bezeichnung der besten erkannten Geste,
   - `best_score` – bester berechneter DTW-Abstand als Maß für die Ähnlichkeit,
   - `best_match_template` – Kennung der besten verwendeten Referenzsequenz.
   - `best_match_mqtt` – für die Geste hinterlegte MQTT-Nachricht.

2. **Schwellwertprüfung:** Der zurückgegebene Score wird mit einem durch systematisches Testen standardmäßig vordefinierten Schwellwert (`dtw_threshold`) von 1.6 verglichen. Liegt der durch Dynamic Time Warping resultierende Wert unterhalb des Schwellwerts, wird angenommen, dass die aktuelle Bewegung der erkannten Gesten entspricht. Andernfalls wird keine Übereinstimmung festgestellt.

Durch die Verwendung eines einstellbaren Schwellwertes kann die Sensitivität des Systems flexibel angepasst werden. Ein niedriger Schwellwert reduziert falsch-positive Ergebnisse, erfordert jedoch eine sehr präzise Ausführung der Gesten. Ein höherer Schwellwert erhöht die Toleranz gegenüber Variationen, steigert jedoch potenziell die Anzahl falsch-positiver Erkennungen.

Sollte innerhalb des in Abschnitt 3.5.4 erwähnten Zeitfensters kein Rückgabewert den Schwellenwert unterschreiten, wird die Gestenerkennung abgebrochen und das Programm kehrt zur Selektionserkennung zurück.

Wird jedoch eine Geste erkannt, kann durch die Rückgabe der zu dieser Geste konfigurierten MQTT-Nachricht dann das entsprechende Smarthome-Gerät angesteuert werden. Anschließend wird der Buffer der Programms geleert und das System wechselt wieder in den initialen Zustand, bereit für weitere Erkennung von Selektions-/Manipulationsgesten.

4 Evaluation

In diesem Kapitel wird die entwickelte Gestensteuerung experimentell überprüft. Ziel der Evaluation ist es, die Funktionsweise des Systems in realitätsnahen Szenarien zu validieren und Schwächen sowie Optimierungspotenziale aufzuzeigen. Neben quantitativen Messungen zur Erkennungsgenauigkeit und Reaktionszeit fließen dabei auch qualitative Beobachtungen ein.

4.1 Testumgebung

Die Versuche fanden im Creative Space for Technical Innovation (CSTI) Labor an der HAW statt. Dieses bietet eine geräumige und flexibel gestaltbare Umgebung, die es ermöglicht, typische Alltagsszenarien mit technischen Geräten abstrakt nachzustellen und gleichzeitig eine kontrollierte Versuchsdurchführung zu gewährleisten.

Um die in Abschnitt 2.4 definierten Rahmenbedingungen zu erfüllen, wurde der Laboraufbau gezielt angepasst. Zentraler Bestandteil war die Stereolabs ZED X Tiefenkamera, die in einer Höhe von rund zwei Metern in einem leichten Winkel von ca. 30 Grad zur Blickrichtung der Nutzer montiert wurde. Auf diese Weise konnten Oberkörper und Hände zuverlässig erfasst werden, ohne dass die Nutzer direkt in die Kamera blickten. Die Kamera blieb während der Initialisierung und der gesamten Testreihe fixiert, sodass die Datenbasis über alle Versuche hinweg konsistent blieb.

Die Beleuchtung im Labor wurde so eingerichtet, dass gleichmäßige, diffuse Lichtverhältnisse herrschten. Harte Schatten, direkte Sonneneinstrahlung oder wechselnde Lichtquellen wurden vermieden, um die Stabilität der Gestenerkennung nicht zu beeinträchtigen. Der Hintergrund der Testumgebung war neutral und frei von störenden Bewegungen.

Die zu steuernden Geräte wurden durch neutrale Platzhalter-Objekte simuliert, die flexibel im Raum positioniert und auch gestapelt werden konnten. Damit ließ sich zum

Abbildung 4.1: Finale Testumgebung des Systems. Rote Objekte repräsentieren mögliche Selektions-Ziele.

Beispiel die eindeutige Erkennung von kleineren, untereinander angeordneten Zielen testen. Nutzer führten ihre Gesten in einem definierten Interaktionsbereich aus, der bereits bei der Template-Aufnahme berücksichtigt worden war. Der Abstand zur Kamera betrug dabei konstant zwischen ein und zwei Metern.

Durch diesen Aufbau, der in den Abbildungen 4.1 und 4.2 dargestellt ist, konnte eine realitätsnahe, aber zugleich kontrollierte Testumgebung geschaffen werden.

4.2 Testdaten und Durchführung

Für die Evaluation wurden die in Abbildung 4.3 dargestellten drei Gesten untersucht: „Öffnen“ (beide Arme heben), „Wischen“ (einmalige Handbewegung der rechten Hand von links nach rechts) sowie „Winken“(wiederholte Bewegung der rechten Hand von links nach rechts). Jede dieser Gesten wurde in 30 unabhängigen Versuchen ausgeführt, sodass insgesamt $n = 90$ Gesteninstanzen vorlagen. Dabei wurde jeweils ein Objekt selektiert und erfasst, ob die Geste korrekt erkannt, falsch klassifiziert oder nicht erkannt wurde.

Abbildung 4.2: Finale Testumgebung des Systems. Rote Objekte repräsentieren mögliche Selektions-Ziele.

Darüber hinaus wurden pro Geste drei zusätzliche Kontrollversuche durchgeführt, in denen keine definierte Geste, sondern alltägliche Bewegungen (z. B. das Benutzen eines Smartphones) ausgeführt wurden. Diese dienten der Überprüfung der „True Negative“ -Ergebnisse, also der Fähigkeit des Systems, Nicht-Gesten zuverlässig als solche zu ignorieren.

Ergänzend wurde exemplarisch für jede Geste die Zeit bis zur erfolgreichen Erkennung gemessen. Wichtig ist hierbei, dass die in den Tabellen angegebenen Zeiten keine absoluten Reaktionszeiten beschreiben, sondern die durchschnittliche Rechenzeit der Dynamic Time Warping-Analyse pro Sliding Window. Da die kontinuierlichen Pose-Daten segmentiert werden, enthalten die ersten Fenster die Geste häufig noch nicht vollständig und können daher keine Erkennung liefern. Die in Tabelle 4.1 angegebenen Messwerte repräsentieren somit den kumulierten Aufwand an Sliding Windows bis zur erfolgreichen Klassifikation einer Geste.

(a) (b) (c)

Abbildung 4.3: Die drei getesteten Gesten: (a) „Öffnen“, (b) „Wischen“und (c) „Winken“.

4.3 Ergebnisse der Messungen

4.3.1 Ergebnisse der Zeigegesten

Neben der Gestenerkennung wurde auch die Zuverlässigkeit der Objektauswahl mittels Zeigegeste evaluiert. In 87 von 90 Versuchen (96.7%) wurde das intendierte Zielobjekt korrekt selektiert. Dies gilt sowohl für Szenarien mit einzelnen Objekten im Sichtfeld als auch für komplexere Konstellationen, bei denen mehrere Objekte gleichzeitig sichtbar oder sogar gestapelt waren.

Besonders hervorzuheben ist, dass auch niedrige Objekte mit einer Höhe von weniger als einem Meter (ca. 75 cm) zuverlässig erkannt und voneinander unterschieden werden konnten. Damit zeigt sich, dass die räumliche Lokalisierung mittels Stereokamera und geometrischem Ray-Box-Intersection-Ansatz robust funktioniert.

4.3.2 Reaktionszeit

Die durchschnittliche Dynamic Time Warping-Verarbeitungszeit pro Versuch ist in Tabelle 4.1 dargestellt. Die Werte liegen zwischen 660 und 678 Millisekunden. Damit beweist sich die Auswertung einzelner Gestensequenzen als ausreichend performant, könnte jedoch durch den Einsatz stärkerer Hardware verbessert werden. Unterschiede zwischen den Gesten sind nur gering, was darauf hindeutet, dass der Rechenaufwand der Dynamic Time Warping-Analyse von der Art der Geste kaum beeinflusst wird.

Gesture	**Versuche**	**Gesamtzeit (ms)**	**Zeit pro Versuch (ms)**
Öffnen	8	5,285.92	660.74
Wischen	9	6,080.53	675.62
Winken	8	5,419.92	677.49

Tabelle 4.1: Durchschnittliche DTW-Verarbeitungszeit pro Versuch der getesteten Gesten. Die Werte ergeben sich aus der durchschnittlichen Spanne an Sliding Windows bis zur erfolgreichen Erkennung.

4.3.3 Ergebnisse pro Geste

Die Erkennungsleistung variiert je nach Geste. Die Tabellen 4.2 bis 4.4 zeigen die Verteilung der Klassifikationsergebnisse.

- Öffnen: Mit 86% richtigen Erkennungen weist diese Geste die höchste Erfolgsrate auf, bei keinem falsch positiven Ergebnis.
- Wischen: Diese Geste weist 71.9% richtige Erkennungen auf, allerdings traten vergleichsweise viele falsch positive Klassifikationen (18.8%) auf. Hierbei wurde die Geste ausschließlich mit der Geste „Winken“verwechselt.
- Winken: Diese Geste erwies sich als am schwierigsten: Nur 53.1% richtig positive Ergebnisse, begleitet von einer hohen Rate falsch negativer Erkennungen (21.9%). Parallel zu „Wischen“bestanden auch hier die Verwechslungen vollständig aus der ähnlich durchgeführten Geste.

Kategorie	Öffnen	Anteil (%)
Richtig Positiv (TP)	23	76.6%
Falsch Positiv (FP)	0	0%
Falsch Negativ (FN)	4	12.5%
Richtig Negativ (TN)	3	9.4%
Gesamt	30	100%

Tabelle 4.2: Versuchsdatensatz für Geste “Öffnen” (n=30).

Kategorie	Wischen	Anteil (%)
Richtig Positiv (TP)	20	62.5%
Falsch Positiv (FP)	6	18.8%
Falsch Negativ (FN)	1	3.1%
Richtig Negativ (TN)	3	9.4%
Gesamt	30	100%

Tabelle 4.3: Versuchsdatensatz für Geste “Wischen” (n=30).

4.3.4 Gesamtübersicht

Die aggregierte Übersicht über alle 90 Versuche ist in Tabelle 4.5 dargestellt. Daraus ergeben sich die klassischen Metriken in Tabelle 4.6.

4.4 Fehleranalyse und Optimierungspotenzial

Die Analyse der Fehlklassifikationen zeigt, dass diese vor allem bei Gesten mit ähnlichen Bewegungsmustern auftreten. Besonders deutlich wird dies bei der Unterscheidung zwischen „Wischen“ und „Winken“. Während „Wischen“ eine einzelne seitliche Handbewegung darstellt, besteht „Winken“ aus einer Abfolge mehrerer Hin- und Herbewegungen. Dieses komplexere Bewegungsmuster erschwert die eindeutige Abgrenzung und führte zu einer erhöhten Zahl falsch negativer Erkennungen.

Zudem tritt ein strukturelles Problem auf: Viele Gesten enthalten Bewegungsanteile, die auch Teil anderer Gesten sein können. Im konkreten Fall weist der erste Abschnitt der „Winken“-Geste eine hohe Ähnlichkeit mit der „Wischen“-Geste auf. Dadurch kommt es zu Verwechslungen, wenn das System den Beginn der Geste fälschlicherweise als vollständige „Wischen“-Geste interpretiert. Dieses Verhalten verdeutlicht die Herausforderung, Gesten zuverlässig voneinander abzugrenzen, sobald sich ihre Bewegungsphasen partiell

Kategorie	Winken	Anteil (%)
Richtig Positiv (TP)	17	53.1%
Falsch Positiv (FP)	3	9.4%
Falsch Negativ (FN)	7	21.9%
Richtig Negativ (TN)	3	9.4%
Gesamt	30	100%

Tabelle 4.4: Versuchsdatensatz für Geste "Winken" (n=30).

Kategorie	Gesamt	Anteil (%)
Richtig Positiv (TP)	60	66.6%
Falsch Positiv (FP)	9	10%
Falsch Negativ (FN)	12	13.3%
Richtig Negativ (TN)	9	10.0%
Gesamt	90	100%

Tabelle 4.5: Gesamtübersicht der Versuchsdatensätze aller Gesten (n=90).

überlappen. Ähnliche Herausforderungen werden in der Literatur bestätigt. Jiang et al. [12] schlagen in ihrem „multi-layered gesture recognition“ ein Verfahren vor, das Gesten in sequentielle Phasen wie „Motion“, „Location“ und „Shape“ gliedert. Dieses Vorgehen hilft, Gesten mit überlappenden Anfangsabschnitten präziser zu erkennen. Ein anderer Ansatz nutzt ein neuronales Netzwerk mit Clustering-Funktionen, um überlappende Handposen in Videosequenzen zu unterscheiden [17].

Optimierungen im bestehenden Ansatz

Innerhalb des verwendeten Dynamic Time Warping-basierten Verfahrens lassen sich insbesondere folgende Verbesserungen realisieren:

- Erweiterung und Diversifizierung der Gesten-Templates: Für Gesten mit stark variabler Ausführung, insbesondere „Winken“, sollten mehr Referenzsequenzen aufgenommen werden, um unterschiedliche Stile, Geschwindigkeiten und Amplituden abzudecken. Dadurch könnte die Generalisierungsfähigkeit des Systems deutlich verbessert werden.
- Verfeinerung der Normalisierungsschritte: Die bisherige Zentrierung und Skalierung könnte durch weitere Verfahren ergänzt werden um die genutzte Pose Estimation robuster und zuverlässiger zu gestalten.

Metrik	Gesamtwert
Accuracy	76.7%
Precision	87.0%
Recall	83.3%
F1-Score	85.1%

Tabelle 4.6: Metriken der Gestenerkennung über alle Gesten (n=90), berechnet aus Tabelle 4.5.

- Anpassung der Dynamic Time Warping-Metrik: Anstelle eines gleichgewichteten Vergleichs aller Gelenkpunkte könnten bestimmte Körperbereiche (z. B. Hand und Arm) stärker gewichtet werden, während weniger relevante Punkte reduziert berücksichtigt werden. Dies würde die Unterscheidung zwischen ähnlichen Gesten wie „Wischen“ und „Winken“ erleichtern.

- Kontextualisierung der Gestenerkennung durch Objektselektion: Ein vielversprechender Ansatz zur Reduzierung von Fehlklassifikationen besteht darin, die Gestenerkennung nach der Selektion eines Objekts auf die für dieses Objekt relevanten Gesten einzuschränken. Dadurch entfällt die Notwendigkeit, alle bekannten Gesten parallel gegeneinander abzugleichen, was nicht nur die Rechenlast reduziert, sondern vor allem die Wahrscheinlichkeit falsch positiver Erkennungen deutlich senkt.

  Ein Beispiel: Wird ein Heizkörper durch eine Zeigegeste selektiert, sind anschließend nur Gesten wie „Ein-/Ausschalten“ oder „Temperatur erhöhen/verringern“ relevant. Andere Gesten wie „Öffnen“ oder „Schließen“ können in diesem Kontext ausgeschlossen werden. Auf diese Weise könnte die Gestenerkennung kontextsensitivität gewinnen und stärker an den Handlungsmöglichkeiten des jeweiligen Objekts angepasst werden.

  Diese kontextuelle Einschränkung stellt einen pragmatischen Mittelweg dar: Einerseits bleibt die Interaktion für den Nutzer intuitiv und flexibel, andererseits erhöht sich die Robustheit des Systems erheblich, da Fehlklassifikationen durch nicht zutreffende Gesten minimiert werden. Allerdings geht dieser Vorteil mit einer stärkeren Abhängigkeit von der korrekten Objektauswahl sowie mit zusätzlichem Konfigurationsaufwand einher.

Alternative und ergänzende Technologien

Über den bestehenden Ansatz hinaus lassen sich auch andere Verfahren in Betracht ziehen, die die Genauigkeit und Robustheit der Gestenerkennung steigern könnten:

- Tiefe neuronale Netze: Verfahren wie Convolutional Neural Networks (CNNs) oder Long Short-Term Memory (LSTM)-Netze haben sich in der Gestenerkennung etabliert. Sie könnten die Abhängigkeit von expliziten Templates reduzieren und komplexere Gestenmuster automatisch erlernen, erfordern jedoch wesentlich mehr Trainingsdaten.
- Hybridansätze: Eine Kombination von Dynamic Time Warping mit lernbasierten Methoden wäre denkbar. DTW könnte zur schnellen Vorfilterung genutzt werden, während ein trainiertes Modell die finale Klassifikation übernimmt.
- Sensorfusion: Neben der visuellen Pose-Erkennung könnte die Integration zusätzlicher Sensordaten, etwa von Inertial Measurement Units (IMUs) an Handgelenken, die Erkennung robuster machen und Ambiguitäten reduzieren.

Systemische Verbesserungen

Schließlich betreffen Optimierungsmöglichkeiten auch die Gesamtarchitektur des Systems:

- Reduzierung der Latenz: Die Verzögerungen von bis zu zwei Sekunden zwischen Gestenausführung und Systemreaktion sind für eine alltagsnahe Interaktion problematisch. Eine schnellere oder parallele Verarbeitung der Sliding-Window-Analyse durch technische Anpassungen oder stärkere Hardware könnte hier entscheidend sein.
- Adaptive Nutzeranpassung: Das System könnte sich durch kontinuierliches Lernen an den individuellen Gestenstil einzelner Nutzer anpassen, wodurch sich die Erkennungsleistung im praktischen Einsatz verbessert. Dies könnte realisiert werden indem korrekte Erkennungen von Gesten weitere Template - Dateien erstellen.
- Skalierbarkeit auf Mehrpersonen-Szenarien: Derzeit ist das System auf eine einzelne Person ausgelegt. Erweiterungen für gleichzeitige Interaktionen mehrerer Nutzer würden die Anwendbarkeit in realen Smarthome-Umgebungen erheblich steigern.

4.5 Diskussion

Insgesamt erreicht das System bereits eine gute Präzision und eine solide Erkennungsrate, auch in Szenarien mit mehreren Objekten im Sichtfeld. Besonders die zuverlässige Objektauswahl durch Zeigegesten bestätigt die Eignung des Ansatzes für Smarthome-Szenarien. Einschränkungen bestehen jedoch in der Robustheit einzelner Gesten und in der Reaktionsgeschwindigkeit der Gesamtsystemkette.

Für eine alltagsnahe Anwendung erscheint die vorgestellte Lösung vielversprechend, erfordert jedoch weitere Optimierungen. Insbesondere eine stärkere Diversifizierung der Templates sowie eine Reduzierung der Systemlatenz stellen zentrale Ansatzpunkte dar, um das System von einem Laborprototyp hin zu einer praxistauglichen Lösung weiterzuentwickeln.

5 Zusammenfassung und Ausblick

In dieser Arbeit wurde ein System zur gestenbasierten Interaktion in Smarthomes entwickelt und evaluiert. Der Ansatz kombiniert die präzise 3D-Pose-Erfassung mittels Stereokamera mit einer robusten Template-basierten Gestenerkennung auf Basis von Dynamic Time Warping. Die Evaluation zeigte, dass insbesondere Zeigegesten zur Objektauswahl zuverlässig und intuitiv funktionieren, während einzelne Manipulationsgesten noch Optimierungspotenzial in Bezug auf Robustheit und Verwechslungsfreiheit aufweisen. Insgesamt bestätigt sich die Eignung des Ansatzes für den Einsatz im Bereich der Human-Computer Interaction (HCI), da er eine natürliche, berührungslose und kontextsensitive Interaktionsform ermöglicht.

Über den Smarthome-Kontext hinaus besitzt das entwickelte System auch Potenzial für weitere Anwendungsfelder:

- Lager- und Logistiksysteme: In modernen Lagerumgebungen ließe sich eine Gestensteuerung nutzen, um den Zugriff auf Objekte effizienter zu gestalten. Ein Mitarbeiter könnte beispielsweise durch einfaches Zeigen eine bestimmte Box oder Palette selektieren und durch eine anschließende Manipulationsgeste deren automatisierte Entnahme oder Weitertransport anstoßen. Dies würde den Einsatz zusätzlicher Geräte wie Handscanner oder Terminals überflüssig machen, den Arbeitsablauf beschleunigen und Fehler durch manuelle Eingaben reduzieren.
- Industrielle Fertigung: In Produktionshallen arbeiten Menschen oft mit Werkzeugen oder tragen Handschuhe, sodass herkömmliche Eingabegeräte schwer nutzbar sind. Hier bietet die Gestensteuerung eine berührungslose und intuitive Alternative. Maschinenparameter könnten durch einfache Handbewegungen angepasst oder Assistenzsysteme aktiviert werden, ohne den Arbeitsprozess zu unterbrechen. Dadurch wird nicht nur die Benutzerfreundlichkeit verbessert, sondern auch die Effizienz gesteigert, da unnötige Unterbrechungen entfallen.

- Automobil- und Mobilitätssektor: Auch in Fahrzeugen eröffnen Gestensteuerungen neue Möglichkeiten der Mensch-Maschine-Interaktion. Fahrer oder Fahrgäste könnten beispielsweise Klimaanlage, Musik oder Navigation durch einfache Gesten bedienen, ohne Tasten oder Touchscreens betätigen zu müssen. Dies reduziert potenzielle Ablenkungen während der Fahrt und trägt so zur Sicherheit bei.

Gleichzeitig gibt es Einsatzbereiche, die sich weniger für eine gestenbasierte Steuerung eignen. Dazu zählen hochsicherheitsrelevante Systeme (z. B. in der Luftfahrt, bei medizinischen Notfallgeräten oder in kritischer Infrastruktur), in denen bereits kleine Fehlinterpretationen schwerwiegende Folgen haben könnten. Auch in stark frequentierten, unübersichtlichen oder schlecht beleuchteten Umgebungen stößt das Verfahren an Grenzen, da Störungen durch andere Personen, wechselnde Lichtverhältnisse oder verdeckte Körperteile die Erkennungsgenauigkeit beeinträchtigen können.

Insgesamt zeigt die Arbeit, dass Template-basierte Gestenerkennung in Verbindung mit moderner Pose Estimation eine vielversprechende Interaktionsform darstellt. Während die Praxistauglichkeit im Smarthome bereits greifbar ist, eröffnen sich spannende Perspektiven für die zukünftige Übertragung in industrielle, logistische und medizinische Szenarien. Für eine breite Anwendbarkeit sind jedoch weitere Optimierungen hinsichtlich Robustheit, Mehrbenutzerfähigkeit und Reaktionsgeschwindigkeit notwendig.

Literaturverzeichnis

[1] 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

[2] Bernat, Karolina ; Ghose, Sobin ; Luck, Kai von ; Vogt, Florian: A method for an agile, user centered development of natural user interfaces. In: *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2017, S. 1–8

[3] Berndt, Donald J. ; Clifford, James: Using Dynamic Time Warping to Find Patterns in Time Series. In: *Proceedings of the AAAI-94 Workshop on Knowledge Discovery in Databases (KDD-94)*. Seattle, Washington, USA, 1994

[4] Bukschat, Yannick ; Vetter, Marcus: *EfficientPose: An efficient, accurate and scalable end-to-end 6D multi object pose estimation approach*. 2020

[5] Butterworth, George ; Morissette, Paul: Onset of Pointing and the Acquisition of Language in Infancy. In: *Journal of Reproductive and Infant Psychology* 14 (1996), Nr. 3, S. 219–231

[6] Butterworth, S.: On the Theory of Filter Amplifiers. In: *Experimental Wireless and the Wireless Engineer* 7 (1930), October, Nr. 10, S. 536–541

[7] Cao, Z. ; Hidalgo Martinez, G. ; Simon, T. ; Wei, S. ; Sheikh, Y. A.: OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019)

[8] Celebi, S. ; Aydin, A.S. ; Temiz, T.T. ; Arici, T.: Gesture recognition using skeleton data with weighted dynamic time warping. In: *VISAPP 2013 - Proceedings of the International Conference on Computer Vision Theory and Applications* 1 (2013), 01, S. 620–625

[9] Developers, Google: *MediaPipe Pose Landmarker*. 2025. – URL https://ai.google.dev/edge/mediapipe/solutions/vision/pose_landmarker. – Zugriff am 20. Juli 2025

[10] Glassner, Andrew S. (Hrsg.): *Graphics Gems*. Boston, MA, USA : Academic Press, 1995. – ISBN 0-12-059756-X

[11] Hélène Cochet, Jacques V.: Pointing gestures produced by toddlers from 15 to 30 months: different functions, hand shapes and laterality patterns. In: *infant behavior and development* 33 (2010), S. 431–441

[12] Jiang, Y. ; Dai, Q. ; Han, F. ; Xie, X. ; Wu, Y.: Multi-Layered Gesture Recognition with Kinect. In: *Journal of Machine Learning Research* 16 (2015), Nr. 18, S. 227–254. – URL http://jmlr.org/papers/v16/jiang15a.html

[13] Jiefeng Li, Hao Zhu et a.: Crowdpose: efficient crowded scenes pose estimation and a new benchmark. (2019)

[14] Juniper Research: *Number of Voice Assistant Devices in Use to Overtake World Population by 2024, Reaching 8.4bn, Led by Smartphones*. April 2020. – URL https://www.businesswire.com/news/home/20200427005609/en/Juniper-Research-Number-of-Voice-Assistant-Devices-in-Use-to-Overtake-World-Population-by-2024-Reaching-8.4bn-Led-by-Smartphones. – Pressemitteilung, Zugriff am 13. August 2025

[15] Kay, Timothy L. ; Kajiya, James T.: Ray tracing complex scenes. In: *Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA : Association for Computing Machinery, 1986 (SIGGRAPH '86), S. 269–278. – URL https://doi.org/10.1145/15922.15916. – ISBN 0897911962

[16] Kurtenbach, Gordon ; Hulteen, Elizabeth A.: Gestures in Human-Computer Communication. In: Laurel, Brenda (Hrsg.): *The Art of Human-Computer Interface Design*. Reading, Mass. : Addison-Wesley, 1990, S. 309–317. – S. Joy Mountford, Manager of the Human Interface Group, Apple Computer Inc., conceived of and technically supported the development of this book.. – ISBN 0201517973

[17] Mira, Anwar: *Developed Deep Learning in Hand Gesture Recognition*, Technische Universität Berlin, Fakultät IV Elektrotechnik und Informatik, Fachgebiet

Computer Vision & Remote Sensing, Doctoral Thesis, 2025. – URL https://doi.org/10.14279/depositonce-23873

[18] Müller-Tomfelde, Christian: Dwell-based pointing in applications of human computer interaction. In: *Proceedings of the 11th IFIP TC 13 International Conference on Human-Computer Interaction*. Berlin, Heidelberg : Springer-Verlag, 2007 (INTERACT'07), S. 560–573. – ISBN 354074794X

[19] 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

[20] Schneider, Pascal ; Memmesheimer, Raphael ; Kramer, Ivanna ; Paulus, Dietrich: Gesture Recognition in RGB Videos Using Human Body Keypoints and Dynamic Time Warping. In: *RoboCup 2019: Robot World Cup XXIII*. Berlin, Heidelberg : Springer-Verlag, 2019, S. 281–293. – URL https://doi.org/10.1007/978-3-030-35699-6_22. – ISBN 978-3-030-35698-9

[21] Song–Hai Zhang, Xin Dong et a.: Poseg: pose-aware refinement network for human instance segmentation. In: *ieee access* 8 (2020), S. 15007–15016

[22] Stereolabs Inc.: *Body Tracking Overview*. 2025. – URL https://www.stereolabs.com/docs/body-tracking/. – Zugriff am 13. August 2025

[23] Sun, Ke ; Xiao, Bin ; Liu, Dong ; Wang, Jingdong: *Deep High-Resolution Representation Learning for Human Pose Estimation*. 2019. – URL https://arxiv.org/abs/1902.09212

[24] Xiong, Yonghua: Research on Gesture Recognition Technology Based on Machine Learning. In: *Applied and Computational Engineering* (2024)

[25] Yang, Tie ; Xu, Yangsheng: Hidden Markov Model for Gesture Recognition / The Robotics Institute, Carnegie Mellon University. Pittsburgh, PA, USA, Mai 1994 (CMU-RI-TR-94-10). – Forschungsbericht. Carnegie Mellon University Technical Report

[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

[27] Zimmerman, Thomas ; Lanier, Jaron ; Blanchard, Chuck ; Bryson, Steve ; Harvill, Young: A hand gesture interface device, 05 1986, S. 189–192

A Anhang

A.1 Verwendete Hilfsmittel

In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Bachelorarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	Verwendung
LaTeX	Textsatz- und Layout-Werkzeug verwendet zur Erstellung dieses Dokuments
Overleaf	Online LaTeX- Editor verwendet zur Erstellung dieses Dokuments
ChatGPT	Hilfe bei Stiloptimierung und LaTeX- Formatierungen
DeepL-Write	Hilfe bei Stiloptimierung

Glossar

Axis-Aligned Bounding Box Quaderförmiges Volumen im dreidimensionalen Raum, dessen Flächen stets parallel zu den Achsen des zugrunde liegenden Koordinatensystems ausgerichtet sind.

Dynamic Time Warping Algorithmisches Verfahren zum Vergleich zeitabhängiger Sequenzen, das in dieser Arbeit zur Gestenerkennung genutzt wird.

Human-Computer Interaction Forschungsgebiet, das sich mit der Gestaltung, Implementierung und Bewertung interaktiver Computersysteme zur Nutzung durch Menschen beschäftigt.

Pose Estimation Verfahren zur Erkennung und räumlichen Lokalisierung signifikanter Körperpunkte (Keypoints) einer Person aus Bild- oder Videodaten.

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