UNIVERSITÄT ZU LÜBECK
INSTITUT FÜR TECHNISCHE INFORMATIK

Echtzeitfähige 3D-Posenbestimmung und Gestenerkennung für die Mensch-Roboter-Interaktion mit dem humanoiden Roboter Pepper

Real-time 3D Pose Estimation and Gesture Recognition for Human-Robot Interaction with the Humanoid Robot Pepper

Masterarbeit

im Rahmen des Studiengangs
Informatik
der Universität zu Lübeck

vorgelegt von
John Paul Jonte

ausgegeben und betreut von
Prof. Dr.-Ing. Erik Maehle

mit Unterstützung von
Kristian Ehlers, M. Sc.

Lübeck, den 18. Oktober 2017

Erklärung

Ich versichere an Eides statt, die vorliegende Arbeit selbstständig und nur unter Benutzung der angegebenen Quellen und Hilfsmittel angefertigt zu haben.

Lübeck, den 18. Oktober 2017

Kurzfassung

Die Mensch-Roboter-Interaktion gewinnt in Anbetracht der stetig wachsenden Verbreitung von autonomen Robotern an Bedeutung. Eine intuitive Möglichkeit der Interaktion bietet die Gestenerkennung, die sich auf Basis der Bestimmung der Pose des menschlichen Körpers realisieren lässt. Im Rahmen dieser Arbeit wurde ein Verfahren zur Körperposenbestimmung optimiert und um eine Gestenerkennung erweitert. Anschließend wurde das Verfahren auf die Soft- und Hardwareplattform des humanoiden Roboters Pepper portiert. Zur Demonstration der Posenbestimmung und Gestenerkennung wurden zwei Anwendungen für Pepper entwickelt. In einer imitiert Pepper die Armposen des Menschen, in der anderen reagiert er auf bestimmte Gesten. Die Evaluation zeigt, dass die Genauigkeit der Posenbestimmung um einige Zentimeter verbessert werden konnte. Die Gestenerkennung erreichte eine sehr hohe Genauigkeit. Nach der Portierung des Verfahrens ist nun mit Pepper eine echtzeitfähige Posenbestimmung und Gestenerkennung erreicht worden.

Abstract

Human-robot interaction is gaining importance considering the steadily increasing prevalence of autonomous robots. An intuitive method of interaction is found in gesture recognition, which can be realized using pose estimation of the human body. For this thesis an approach to human body pose estimation was optimized and expanded with a gesture recognition algorithm. The procedure was subsequently ported to the software and hardware platform of the humanoid robot Pepper. To demonstrate both the pose estimation and gesture recognition, two applications for Pepper were developed. In one application Pepper imitates a human's arm poses, in the other he reacts to gestures. The evaluation shows that the accuracy of the pose estimation was improved by several centimeters. The gesture recognition achieved a very high accuracy. After porting the procedure, real-time capable pose estimation and gesture recognition were achieved with Pepper.

Inhaltsverzeichnis

1. Einleitung **1**

2. Grundlagen **3**
2.1. Der humanoide Roboter Pepper 3
2.2. Tiefenbildkameras und Punktwolken 6
2.3. Posenbestimmung 6
2.4. Support Vector Machines 9

3. Implementierung **13**
3.1. Erweiterung der Posenbestimmung 13
3.2. Gestenerkennung 15
3.3. Portierung auf Pepper 17

4. Evaluation **25**
4.1. Testaufbau 25
4.1.1. Posenbestimmung 25
4.1.2. Gestenerkennung 26
4.1.3. Portierung auf Pepper 27
4.2. Ergebnisse 27
4.2.1. Posenbestimmung 28
4.2.2. Gestenerkennung 31
4.2.3. Portierung auf Pepper 32
4.3. Diskussion 32

5. Anwendungen **39**
5.1. Imitation von Armbewegungen 39
5.2. Reaktion auf Gesten 44

6. Zusammenfassung und Ausblick **47**

A. Fehler nach Abzug des Versatzes **49**

Abbildungsverzeichnis **53**

Tabellenverzeichnis **55**

Literaturverzeichnis **57**

1. Einleitung

Autonome Roboter finden immer mehr Anwendungsbereiche und dringen in unseren Alltag ein. So werden Roboter wie der von der Firma SoftBank Robotics entwickelte „Pepper“ unter anderem als Ansprechpartner in Geschäften eingesetzt [1]. Durch den vermehrten Kontakt von Menschen und Robotern gewinnt die Mensch-Roboter-Interaktion an Bedeutung für unser tägliches Leben. Roboter müssen nicht nur in der Lage sein, ihre Umgebung und die sich darin befindlichen Menschen wahrnehmen zu können, sondern auch die Szenerie zu verstehen. Besitzt ein Roboter die Fähigkeit, die Absichten der Menschen in seiner Umgebung zu interpretieren, kann er seine Aufgabe effizienter und sicherer bewältigen, da er beispielsweise Bewegungen deuten und eine entsprechende Wegplanung vornehmen und folglich Kollisionen vermeiden kann. Ferner sind direkte Interaktionen zwischen Mensch und Roboter denkbar.

Eine Möglichkeit, den Menschen und seine Absichten zu interpretieren, ist die Bestimmung und Analyse der Körperpose und darauf basierend die Erkennung von Körpergesten. Körpergesten lassen sich in statische und dynamische Gesten unterteilen. Statische Gesten bestehen aus bestimmten Posen des Körpers oder Körperteilen wie den Armen. Ein zur Seite herausgestreckter Arm und ein herabhängender Arm sind zwei verschiedene Gesten. Dynamische Gesten sind zeitliche Folgen statischer Gesten. Im Gegensatz zur Sprachsteuerung können Körpergesten auch in sehr lauten Umgebungen wie Fabrikhallen erkannt werden. Darüber hinaus bietet die Bestimmung der Körperpose weitere Möglichkeiten, wie beispielsweise in einem Teach-In-Verfahren den Roboter Bewegungen erlernen zu lassen, anstatt diese manuell einzuprogrammieren.

Ehlers et al. entwickelten ein Verfahren für die Posenbestimmung der Hand, welches auf einem kinematischen Modell basiert [2]. Dieses Verfahren wurde auf die Bestimmung der Körperpose erweitert [3]. Dabei ermöglicht es nicht nur die Bestimmung der Positionen bestimmter Körpermerkmale wie Kopf oder Hände, sondern auch die Ermittlung der Winkel aller berücksichtigten Gelenke. Daher bietet es mehr Informationen über den Menschen im Vergleich zu Verfahren, die nur Merkmalspositionen bestimmen, und mehr Möglichkeiten in der Verwendung im Rahmen der Mensch-Roboter Interaktion. Als Eingabe für das Verfahren dient eine dreidimensionale Punktwolke, die mit Hilfe einer Tiefenbildkamera generiert werden kann.

Ziel dieser Arbeit ist, das Verfahren zur Posenbestimmung für die Interaktion mit dem Roboter Pepper zu verwenden. Für die Steuerung Peppers sollen sowohl die Körperpose als auch darin erkannte Körpergesten verwendet werden. In dieser Arbeit werden nur statische Gesten betrachtet. Zunächst wird das Verfahren aus [3] im Hinblick auf Ro-

bustheit und Genauigkeit optimiert. Es wird eine Erkennung vordefinierter statischer Gesten auf Basis der vorher bestimmten Körperpose entwickelt. Für die Nutzung des Verfahrens mit Pepper wird der vorhandene Programmcode auf die Soft- und Hardwareplattform von Pepper portiert. Die schwache Hardware des Roboters macht eine Optimierung bezüglich der benötigten Ressourcen notwendig. Des Weiteren werden zur Demonstration der Funktionalität zwei auf der Posenbestimmung und Gestenerkennung basierende Anwendungen zur Steuerung Peppers entwickelt.

Die Grundlagen der Arbeit, der Roboter Pepper und das bestehende Verfahren für Posenbestimmung, werden in Kapitel 2 vorgestellt. Kapitel 3 beschreibt die Erweiterung der Posenbestimmung, die Implementierung der Gestenerkennung und die Portierung des Verfahrens auf Pepper. In Kapitel 4 erfolgt die Evaluation der Posenbestimmung, Gestenerkennung und Portierung sowie die Diskussion der Ergebnisse. Anschließend werden in Kapitel 5 die beiden Anwendungen der Posenbestimmung und Gestenerkennung für Pepper vorgestellt. Kapitel 6 schließt die Arbeit mit einer Zusammenfassung und einem Ausblick ab.

2. Grundlagen

In diesem Kapitel werden zunächst der Roboter Pepper, mit dem interagiert werden soll, und die Grundlagen von Punktwolken vorgestellt. Anschließend folgt eine Beschreibung des verwendeten Verfahrens zur Posenbestimmung. Ferner wird die Funktionsweise von SVMs erläutert, mit denen die Gestenerkennung realisiert wird.

2.1. Der humanoide Roboter Pepper

Der humanoide Roboter Pepper wurde von der französischen Firma Aldebaran entwickelt, welche 2015 von der japanischen Firma SoftBank aufgekauft und 2016 in SoftBank Robotics umbenannt wurde [5, 6].

Pepper verfügt über eine Vielzahl an Sensoren und Aktoren [7]. Seine Arme haben nahezu die selben Freiheitsgerade wie die des Menschen. Anstatt Beinen besitzt Pepper eine feste

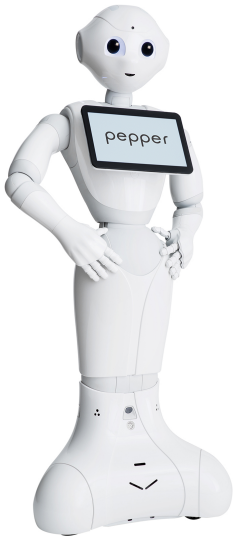


Abbildung 2.1.: Der humanoide Roboter Pepper [4].

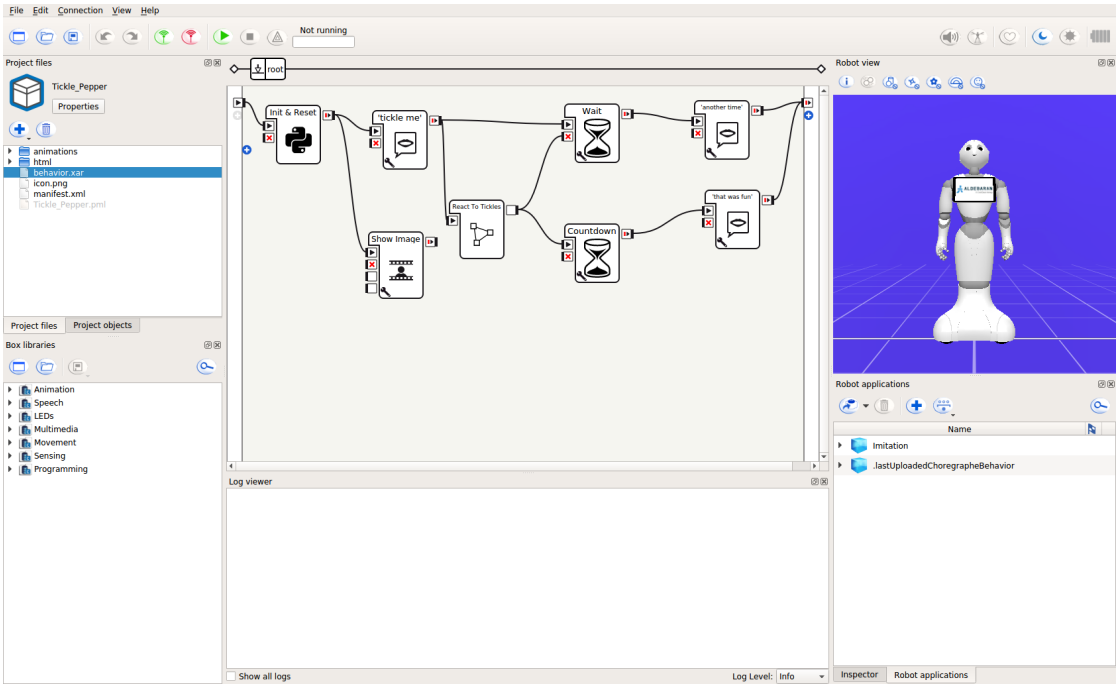


Abbildung 2.2.: Die Benutzeroberfläche des Programmierwerkzeugs Choregraphe. In der Mitte des Bildes sind die Module zu sehen, die miteinander verkettet werden, um das Verhalten von Pepper zu bestimmen. Rechts ist eine simulierte Ansicht des Roboters, um dessen Bewegungen zu visualisieren.

Basis mit drei Rädern, die eine Fortbewegung in alle Richtungen ermöglichen. Im Kopf befinden sich zwei herkömmliche Farbbildkameras und eine Tiefenbildkamera, die ein dreidimensionales Bild aufzeichnet. Für die Navigation besitzt Pepper Sonar-, Infrarot- und Laser-Sensoren.

In Peppers Kopf befinden sich Lautsprecher für die Sprach- und Audiowiedergabe, sowie RGB-LEDs in den Augen und Ohren. Auf der Brust ist ein Android-Tablet montiert, auf dem Bilder, Videos und Web-Seiten dargestellt werden können. Über den Touchscreen des Tablets kann mit Pepper interagiert werden.

Mit der Software Choregraphe lassen sich Programme für Pepper nach dem Baukastenprinzip entwickeln. Abbildung 2.2 zeigt einen Screenshot der Benutzeroberfläche von Choregraphe. Choregraphe ermöglicht es, vorbestimmte Programmodule, die jeweils eine Aktion durchführen, zu einem Programm zu verketten. Die Module beinhalten Python-Skripte, die der Benutzer anpassen kann, um das gewünschte Verhalten zu erreichen. Außerdem ist es möglich, Bewegungsabläufe von Peppers Gelenken zu erstellen und abzuspielen. Mit Choregraphe erstellte Programme können direkt auf Pepper transferiert oder in den „Aldebaran Store“ hochgeladen werden.

Alternativ stehen Software Development Kits (SDK) für verschiedene Programmierspra-

Abbildung 2.3.: Visualisierung einer Punktwolke, die mit einer ASUS XTion Pro-Kamera aufgenommen wurde. Hellblaue Punkte sind nah an der Kamera, dunkelblaue Punkte befinden sich weiter weg. An weißen Stellen sind keine Daten vorhanden.

chen zur Verfügung. Zu den unterstützten Sprachen zählen unter anderem C++ und Python. Programme können sowohl von einem an das Netzwerk angeschlossenen PC als auch auf Pepper selbst ausgeführt werden. Die NaoQI API (Application Programming Interface) bietet Dienste, die die Steuerung von Pepper und Zugriff auf die Sensoren und Kameras erlauben. Darüber hinaus gibt es eine Vielzahl an Diensten, die eine Steuerung Peppers auf einer höheren Ebene ermöglichen. Dazu zählen beispielsweise Dienste, die die Gesichter und Emotionen der Menschen in der Umgebung analysieren, die Pepper einen vorgefertigten Dialog mit dem Benutzer führen lassen, oder Pepper mit Hilfe seiner Sensoren sicher durch die Umgebung navigieren lassen. Für die Sprachkommunikation gibt es Dienste, die Spracherkennung und Text-To-Speech zur Verfügung stellen.

2.2. Tiefenbildkameras und Punktwolken

Mit Hilfe von Tiefenbildkameras ist es möglich, ein dreidimensionales Bild zu erzeugen, aus dem mehr Informationen über die Umgebung gewonnen werden können, als es mit einer herkömmlichen Kamera möglich ist. Dabei wird in Form eines Tiefenbilds jedem Bildpunkt ein Tiefenwert zugewiesen, der die Entfernung dieses Punktes von der Kamera angibt. Die Bilddaten können in eine Punktwolke umgewandelt werden, bei der jeder Bildpunkt als Punkt mit drei Raumkoordinaten dargestellt wird. Ein Beispiel einer Punktwolke ist in Abbildung 2.3 zu sehen. Gleichung 2.1 zeigt die Umrechnung von einem Bildpunkt im Tiefenbild zu einem Punkt in der Punktwolke. Dabei sind (u, v) die Koordinaten des Bildpunkts im Tiefenbild, d der dazugehörige Tiefenwert, (c_x, c_y) das Zentrum des Bildes und (f_x, f_y) der Brennpunkt der Kamera.

$$x = \frac{(u - c_x) \cdot d}{f_x},\ y = \frac{(v - c_y) \cdot d}{f_y},\ z = d \tag{2.1}$$

2.3. Posenbestimmung

Das dieser Arbeit zu Grunde liegende Verfahren zur Posenbestimmung basiert auf einem kinematischen Modell [2]. Das Verfahren wurde ursprünglich für die Bestimmung der Handpose entwickelt und auf die Bestimmung der Körperpose erweitert [3]. Im Gegensatz zu anderen Verfahren werden nicht nur die Positionen von Körpermerkmalen wie Kopf oder Händen bestimmt, sondern auch die Winkel aller berücksichtigten Gelenke. Die Eingabe für das Verfahren ist eine dreidimensionale Punktwolke, die beispielsweise mit einer Tiefenbildkamera aufgenommen werden kann. Der Hintergrund und der Boden der Szene bieten keine für die Pose des Menschen relevante Information. Daher ist vor der Verwendung der Punktwolke eine Entfernung aller Punkte nötig, die nicht zum Körper gehören. Dies geschieht mit einem sogenannten Box-Filter, bei dem für jede Achse ein Minimum und Maximum definiert wird, und alle Punkte außerhalb dieses Bereiches entfernt werden. Als Resultat ergeben sich alle Punkte, die sich innerhalb des Quaders befinden, das durch die Minima und Maxima definiert wird. Abbildung 2.4 zeigt eine Visualisierung des Box-Filters am Beispiel einer Punktwolke. Die Minima und Maxima der x- und y-Achsen sind durch transparente Ebenen dargestellt.

Kinematisches Modell

Ein kinematisches Modell besteht aus miteinander verketteten Koordinatensystemen, die die Gelenke und Bewegungsachsen eines Objektes repräsentieren. Für jeden Freiheitsgrad gibt es ein Koordinatensystem und einen einstellbaren Winkel. Aufgrund der Verkettung der Gelenke hängt die Position eines Gelenks von den Gelenkwinkeln aller vorigen Gelenke in der Kette ab. Die Koordinatensysteme werden im Bereich der Robotik häufig

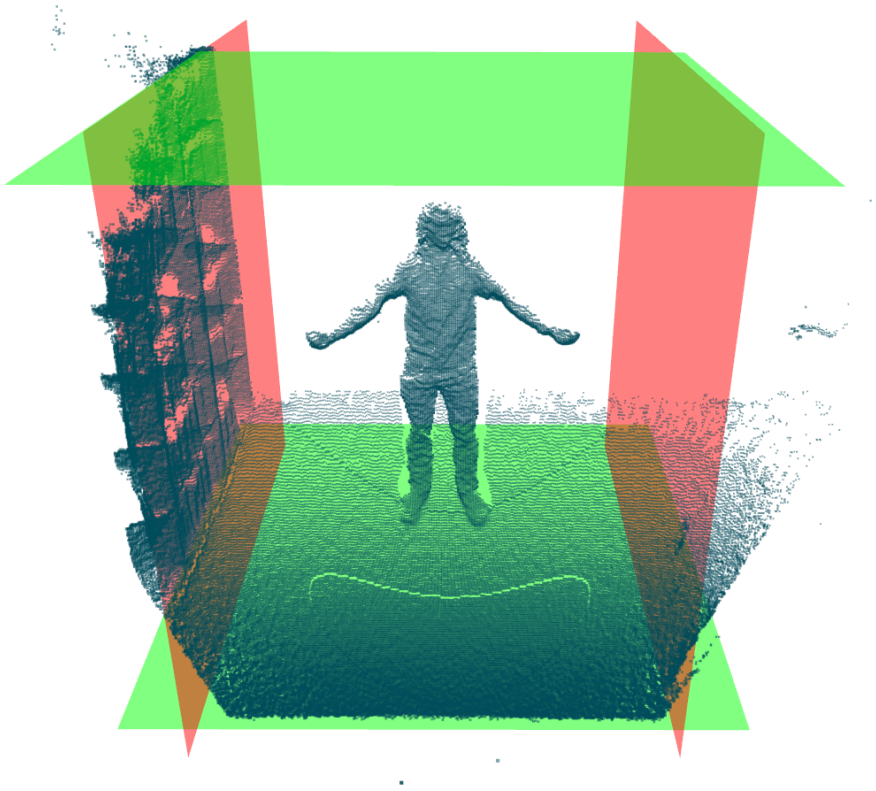


Abbildung 2.4.: Der Box-Filter am Beispiel einer Punktwolke. Die Minima und Maxima der Achsen werden durch transparente Ebenen dargestellt. Nach der Filterung bleiben nur die Punkte des Menschen zwischen den Ebenen übrig. Der Übersicht halber wurden Minimum und Maximum der z-Achse (Blickrichtung der Kamera) nicht visualisiert.

nach der Denavit-Hartenberg-Konvention aufgestellt [8]. Die z-Achse repräsentiert die Rotationsachse des Gelenks. Im Normalfall ergibt sich die x-Achse aus dem Kreuzprodukt der z-Achse und der z-Achse des vorigen Gelenks und ist orthogonal zu beiden Achsen. Die y-Achse wird so gewählt, dass sich ein rechtshändiges Koordinatensystem ergibt.

In dieser Arbeit werden die Gelenke durch Knoten und die Verkettung der Gelenke durch Kanten visualisiert. Das zur Posenbestimmung verwendete kinematische Modell ist dem menschlichen Skelett nachempfunden. Abbildung 2.5 zeigt eine Visualisierung der Topologie des kinematischen Modells und der dazugehörigen Punktwolke. Als initiale Pose des Modells dient die sogenannte T-Pose, bei der davon ausgegangen wird, dass der Mensch gerade steht und die Arme zu den Seiten ausstreckt.

Das Ziel der Posenbestimmung ist, die Gelenkwinkel des kinematischen Modells so einzustellen, dass die Punktwolke des Menschen durch die Positionen der Gelenke möglichst

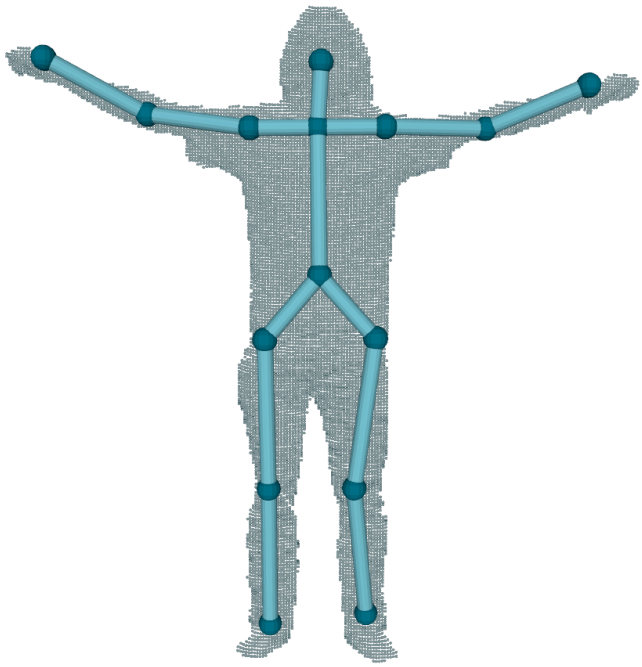


Abbildung 2.5.: Die Topologie des kinematischen Modells. Die Gelenke sind durch Knoten und die Verkettung der Gelenke ist durch Kanten dargestellt. Im Hintergrund ist die Punktwolke eines Menschen zu sehen.

genau approximiert wird. Zunächst wird für jedes Gelenk die Zielposition bestimmt, an der sich das Gelenk aufgrund der aktuellen Punktwolke befinden soll, um diese möglichst gut nachzuempfinden. Dafür werden jedem Gelenk alle Punkte der Punktwolke zugewiesen, die diesem Gelenk am nächsten sind. Der Mittelpunkt aller dem Gelenk zugewiesenen Punkte ist die endgültige Zielposition.

Anschließend werden die Gelenkwinkel mit Hilfe von inverser Kinematik bestimmt, sodass sich die Gelenke an ihren Zielpositionen befinden. Da nicht immer eine Lösung existiert, die alle Zielpositionen erreicht, wird der Levenberg-Marquardt-Algorithmus verwendet, der iterativ die Gelenkwinkel mit der Methode der kleinsten Quadrate approximiert.

Self-Organizing Maps

Eine Self-Organizing Map (**SOM**) ist ein künstliches neuronales Netz, das aus Neuronen und einer Topologie, die die Nachbarschaft der Neuronen beschreibt, besteht [9]. Ziel der SOM ist es, einen Datenraum mit hoher Dimension auf eine Karte niedriger Dimension zu projizieren. Jedem Neuron wird ein Gewichtsvektor zugewiesen, der im hiesigen

Kontext die Position im Raum der Eingabedaten beschreibt. Beim Training der SOM wird iterativ ein Punkt aus dem Datensatz ausgewählt und der Gewichtsvektor, der den geringsten Abstand zum Datenpunkt hat, herausgesucht und in Richtung des Datenpunkts bewegt. Diese Anpassung erfolgt auch mit den Gewichtsvektoren von Neuronen, die in der Topologie benachbart sind, jedoch wird die Anpassung mit größerer Distanz in der Topologie geringer. Mit jeder Iteration verteilen sich die Gewichtsvektoren in den Daten. Dies wird wiederholt, bis eine gewisse Anzahl an Iterationen erreicht ist und die Gewichtsvektoren den Datensatz approximieren. Abbildung 2.6 zeigt das Training einer SOM über mehrere Iterationen an einem Beispiel. Nach der ersten Iteration hat sich das Gewicht, das am nächsten an den Daten liegt, am weitesten in Richtung der Daten verschoben. Nach 50 Iterationen approximiert die SOM den Datensatz gut.

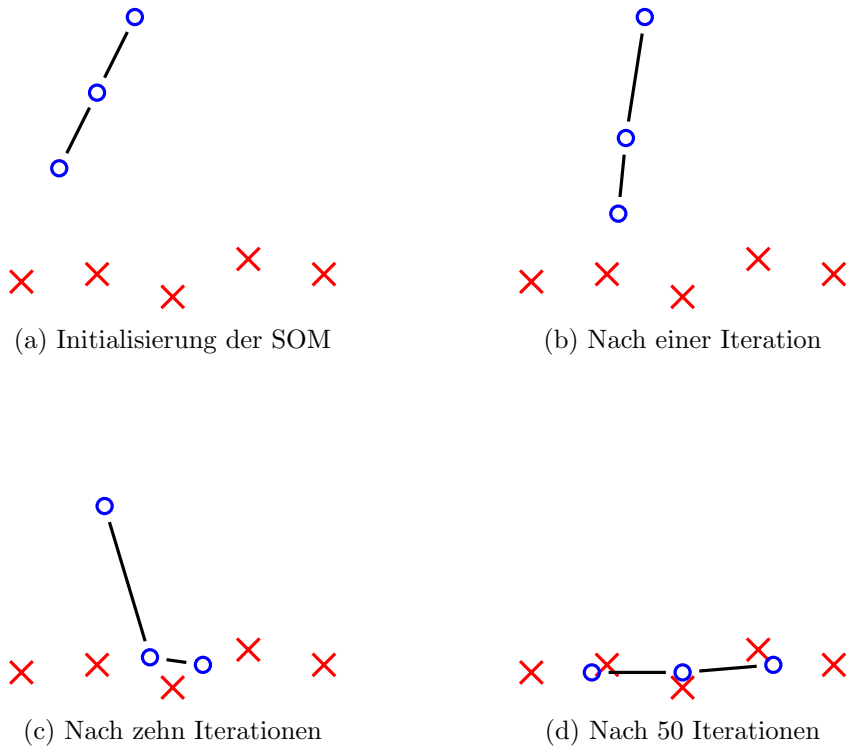
Die Posenbestimmung interpretiert die Gewichtsvektoren der SOMs als dreidimensionale Punkte im Raum, und trainiert die SOM mit der Punktwolke des Menschen. Im Rahmen dieser Arbeit werden SOMs visualisiert, indem die Gewichtsvektoren durch Punkte und die Topologie durch Kanten dargestellt werden.

Das Verfahren verwendet zwei verschiedene Arten von SOMs. Abbildung 2.7a zeigt eine Visualisierung der Topologie der Standard-SOM (**SSOM**), die der Form des menschlichen Körpers ähnelt. Dabei werden größere Flächen, wie der Oberkörper, durch Gitter dargestellt und Gliedmaßen, wie die Arme, durch Ketten dargestellt. Bei der generalisierten SOM (**GSOM**) wird nicht nur der Abstand der Gewichtsvektoren zu den Datenpunkten, sondern auch Abstand der Kanten zwischen den Gewichtsvektoren zu den Daten optimiert. Daher werden Gitter und Ketten wie im Oberkörper oder den Gliedmaßen zu einzelnen Verbindungen vereinfacht, wie in der Visualisierung der Topologie in Abbildung 2.7b zu sehen ist..

In [10] wurde für die Handposenbestimmung eine Unterstützung des kinematischen Modells durch SSOM und GSOM entwickelt. Vor dem Optimierungsschritt werden die SOMs mit der Punktwolke trainiert. Die mit den SOMs bestimmten Positionen der Körpermerkmale werden zur Bestimmung der Zielposition der Gelenke im kinematischen Modell herangezogen. Sind für ein Merkmal die Positionen in SSOM und GSOM weniger als 10 cm voneinander entfernt, wird der Mittelpunkt dieser Positionen berechnet. Ist der Zielpunkt des Gelenks mehr als 10 cm von diesem Punkt entfernt, wird er verworfen und durch den berechneten Mittelpunkt ersetzt.

2.4. Support Vector Machines

Eine Support Vector Machine (**SVM**) ist ein Modell für überwachtes Lernen, das Daten klassifiziert. Sie wird mit einer Menge an Datenpunkten, die jeweils einer Klasse zugeordnet sind, trainiert. Die SVM bestimmt eine sogenannte Hyperebene, die die beiden Klassen im Raum linear voneinander trennt und den größtmöglichen Abstand von beiden Klassen hat. Abbildung 2.8 zeigt einen zweidimensionalen Datensatz, der in zwei



(a) Initialisierung der SOM

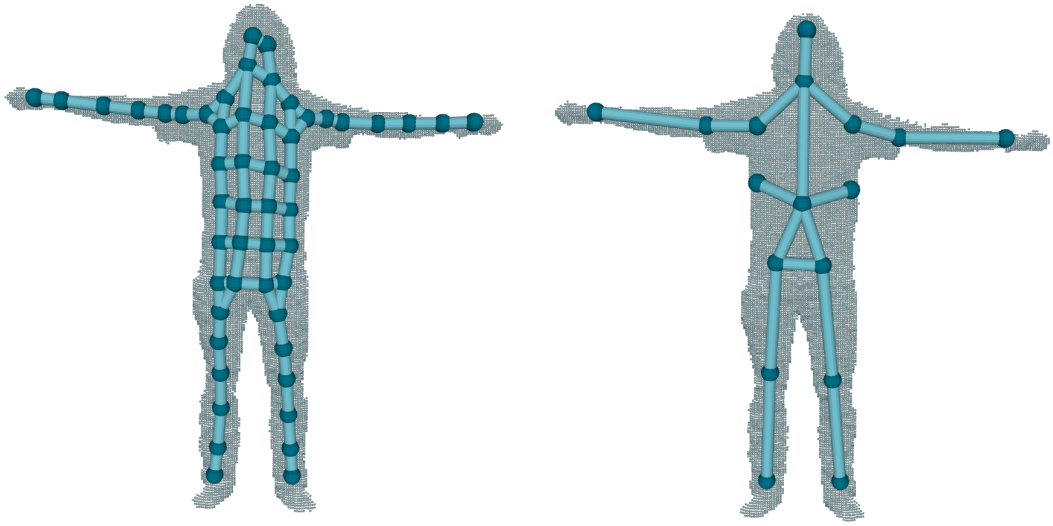
(b) Nach einer Iteration

(c) Nach zehn Iterationen

(d) Nach 50 Iterationen

Abbildung 2.6.: Funktionsweise einer SOM über mehrere Iterationen. Die Datenpunkte werden durch Kreuze, die Gewichtsvektoren der Neuronen durch Kreise dargstellt. Die Kanten stellen die Topologie der SOM dar. Für jeden Datenpunkt wird das Gewicht mit der geringsten Distanz herausgesucht und das Gewicht etwas in Richtung des Datenpunkts bewegt. Die Gewichtsvektoren der benachbarten Neuronen werden mit zunehmendem Abstand in der Topologie schwächer in Richtung des Datenpunkts verschoben. Nach einigen Iterationen werden die Daten durch die SOM approximiert.

Klassen aufgeteilt ist, welche von einer Hyperebene getrennt werden. Ist eine lineare Trennung nicht möglich, können die Datenpunkte mit dem sogenannten *Kernel-Trick* in einen Raum mit einer höheren Dimension transformiert werden, wodurch die Klassen möglicherweise linear trennbar werden. Das Verfahren kann auf eine größere Anzahl an Klassen erweitert werden, indem die Klassifizierung in mehrere Klassifizierungen mit nur zwei Klassen reduziert wird. Bei der *one-versus-all*-Strategie werden viele SVMs erzeugt, die eine Klasse von allen anderen trennen, bei der *one-versus-one*-Strategie wird für jedes Paar aus Klassen eine SVM erzeugt. Im Rahmen dieser Arbeit wurde die SVM-Funktionalität der OpenCV-Bibliothek verwendet, welche die LibSVM-Bibliothek verwendet und auf der *one-versus-one*-Strategie basiert [11].



(a) Topologie der SSOM (b) Topologie der GSOM

Abbildung 2.7.: Topologien der SSOM und GSOM. In der SSOM wird der Oberkörper durch ein Gitter aus Knoten und die Gliedmaßen durch Ketten dargestellt. In der Topologie der GSOM wird das Gitter auf drei Kanten reduziert und die Zwischenpunkte in den Gliedmaßen fallen weg. Dies ist möglich, da bei der GSOM nicht nur die Gewichtsvektoren, sondern auch die Verbindungen zwischen ihnen betrachtet werden.

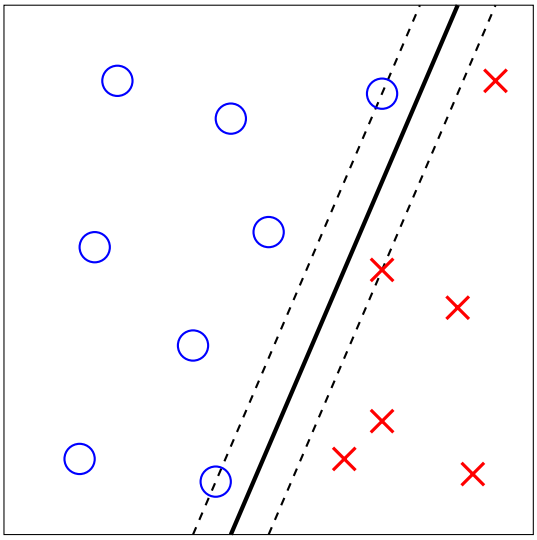


Abbildung 2.8.: Funktionsweise einer SVM. Es gibt eine Menge von Datenpunkten, die jeweils einer von zwei Klassen zugeordnet sind (hier dargestellt durch Kreise und Kreuze). Eine SVM ermittelt eine Hyperebene, die die beiden Klassen mit größtmöglichem Abstand voneinander trennt. Die fettgedruckte Linie stellt die Hyperebene dar, die gestrichelten Linien zeigen den Abstand zu den Datenpunkten.

3. Implementierung

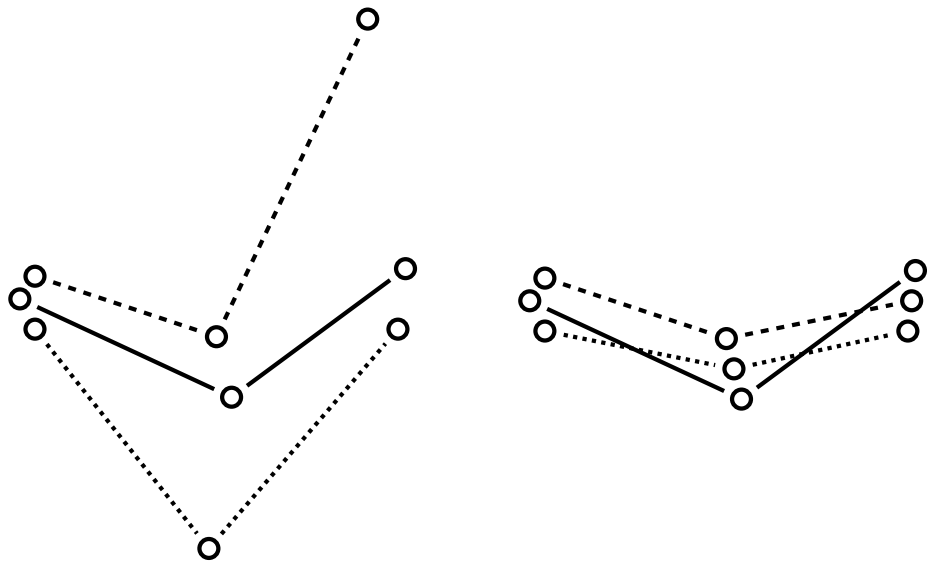
Im Rahmen dieser Arbeit wurde das in Kapitel 2.3 vorgestellte Verfahren zur Posenbestimmung des Körpers zunächst im Hinblick auf Robustheit und Genauigkeit optimiert. Anschließend wurde eine statische Gestenerkennung entwickelt, die der vom Verfahren bestimmten Pose eine Geste zuordnet. Zuletzt erfolgte eine Portierung des Verfahrens auf die Soft- und Hardwareplattform des Roboters Pepper sowie eine Optimierung der Ausführungsgeschwindigkeit.

3.1. Erweiterung der Posenbestimmung

Die in Kapitel 2.3 beschriebene Beeinflussung des kinematischen Modells durch SSOM und GSOM wurde von der Handposenbestimmung übernommen und für den Körper implementiert. Zusätzlich dazu wurde eine Beeinflussung der SSOM und GSOM durch das kinematische Modell entwickelt und eine Übernahme der Knochenlängen des kinematischen Modells aus der GSOM hinzugefügt, sodass das Modell nicht an eine bestimmte Statur gebunden ist.

Beeinflussung der SSOM und GSOM durch das kinematische Modell

Nach Bestimmung der Zielpositionen für das kinematische Modell werden die mit der SSOM und GSOM bestimmten Positionen der Merkmale überprüft und gegebenenfalls angepasst. Abbildung 3.1 zeigt die gegenseitige Beeinflussung der Modelle an einem Beispiel. Die Idee der gegenseitigen Beeinflussung ist, dass die Modelle aufgrund ihrer grundlegenden Unterschiede selten zum selben Zeitpunkt einen großen Fehler der Merkmalspositionen aufweisen. Daher wird ein abweichendes Modell angepasst, wenn die Positionen der Merkmale in den anderen beiden Modellen nah beieinander liegen. Sind für ein Merkmal die Positionen in SSOM und GSOM mehr als 10 cm voneinander entfernt, wird zunächst der Abstand von der Zielposition des Merkmals im kinematischen Modell zur Position des Merkmals in der SSOM bestimmt. Ist dieser Abstand weniger als 10 cm, wird die Position des Merkmals in der GSOM auf den Mittelpunkt der Positionen in den anderen beiden Modellen festgelegt. Analog dazu erfolgt die Anpassung der Merkmalsposition in der SSOM, wenn diese von den Positionen im kinematischen Modell und der GSOM abweicht. Da zu diesem Zeitpunkt die SSOM und GSOM bereits berechnet wurden, wirken sich die Änderungen der Merkmalspositionen jedoch erst



(a) Vor Anpassung der Merkmalspositionen (b) Nach Anpassung der Merkmalspositionen

Abbildung 3.1.: Beispiel der gegenseitigen Beeinflussung von SSOM, GSOM und kinematischem Modell. Die durchgezogene Linie stellt das kinematische Modell dar, die gestrichelte Linie zeigt die Stellung der SSOM und die gepunktete Linie repräsentiert die GSOM. Die Positionen des ersten Merkmals in allen Modellen sind nah beieinander; es erfolgt keine Anpassung. Das zweite Merkmal hat in der SSOM und dem kinematischen Modell eine ähnliche Position und die Position in der GSOM weicht ab. Daher wird die GSOM angepasst. Beim dritten Merkmal weicht die Position in der SSOM ab, welche anschließend angeglichen wird. In beiden Fällen verschiebt sich die abweichende Merkmalsposition auf den Mittelpunkt der Positionen in den anderen beiden Modellen.

im nächsten Durchlauf des Verfahrens aus. Die zeitliche Abfolge der Anpassungen zeigt Abbildung 3.2.

Knochenlängenanpassung

Damit das kinematische Modell sich an verschiedene Knochenstrukturen anpassen kann, wurde eine Übernahme der Knochenlängen aus der GSOM implementiert. Nach Berechnung der GSOM wird für jeden Knochen die Länge berechnet, indem der Abstand der Merkmale an den Enden des Knochens bestimmt wird. Diese Knochenlängen übernimmt das kinematische Modell vor der Berechnung der Gelenkwinkel. Abbildung 3.3 zeigt eine Visualisierung des kinematischen Modells vor und nach der Übernahme der Knochenlängen.

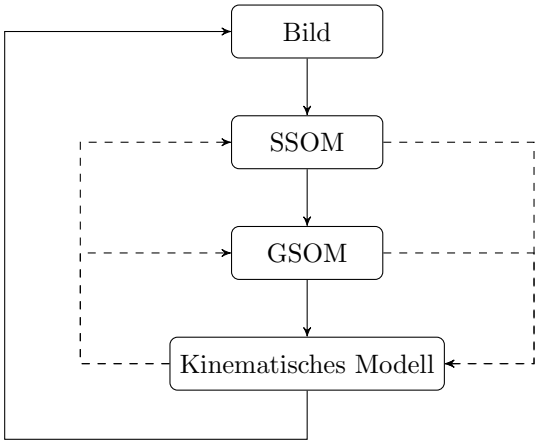
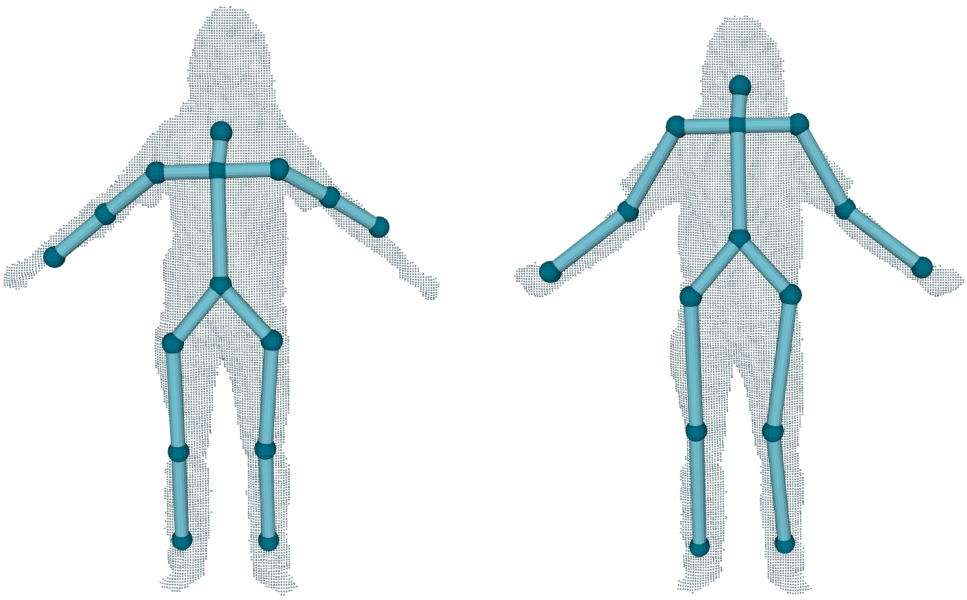


Abbildung 3.2.: Gegenseite Beeinflussung von SSOM, GSOM und kinematischem Modell. Durchgezogene Linien stellen den zeitlichen Ablauf des Verfahrens dar, gestrichelte Linien repräsentieren die Beeinflussung der Modelle. Merkmalspositionen, die in der SSOM oder GSOM ermittelt wurden, werden direkt im selben Durchlauf im kinematischen Modell berücksichtigt. Anpassungen der SSOM und GSOM durch das kinematische Modell wirken sich erst im folgenden Durchlauf bei Ankunft des nächsten Bildes aus.

Zusätzlich dazu erfolgt eine Glättung, damit eventuelle kurzzeitige Fehler oder Ausreißer in der GSOM keine fehlerhaften Knochenlängen im kinematischen Modell hervorrufen. Dabei wird in jedem Durchlauf des Verfahrens die Knochenlänge auf die Summe von 50% des vorherigen Wertes und 50% des Wertes, der anhand der GSOM bestimmt wurde, gesetzt. Außerdem werden Knochenlängen nur während der ersten 100 Durchläufe nach Initialisierung des kinematischen Modells angepasst. Die Initialisierung erfolgt nur, wenn der Mensch die T-Pose eingenommen hat, daher ist in den ersten Durchläufen die Pose bekannt. Durch die simple und eindeutige Pose kann sich die GSOM schnell anpassen und die Knochenlängen zuverlässig bestimmen. Darüber hinaus ist die Wahrscheinlichkeit gering, dass der Mensch innerhalb dieses kurzen Zeitraums komplexe Bewegungen durchführt und die Pose der GSOM nicht mehr korrekt ist. Daher übernimmt das kinematische Modell die Knochenlängen nur kurz nach der Initialisierung.

3.2. Gestenerkennung

Das Verfahren wurde um eine Erkennung statischer Gesten erweitert, die als Grundlage die durch das kinematische Modell bestimmte Körperpose verwendet. Eine statische Geste wird hier als eine bestimmte Pose des Arms definiert, beispielsweise ein zur Seite ausgestreckter oder ein herabhängender Arm. Es werden keine Bewegungen des Körpers oder von Körperteilen betrachet. Für die Gestenerkennung wird jeder Arm einzeln be-



(a) (b)

Abbildung 3.3.: Vergleich der Topologie des kinematischen Modells (a) vor und (b) nach der Übernahme der Knochenlängen aus der GSOM. Anfänglich sind die Knochen nicht lang genug, um die Punktwolke mit dem kinematischen Modell zu approximieren. Nach einigen Iterationen wurden die Knochenlängen übernommen und die Approximation entspricht mehr der Grundwahrheit.

trachtet, damit nicht jede Kombination aus Armposen als eigene Geste definiert werden muss.

Zur Klassifizierung der Gesten wird eine SVM verwendet. Für jeden Arm gibt es für jedes Modell (SSOM, GSOM, kinematisches Modell) eine SVM. Die SVMs werden für jede Geste mit beispielhaften Posen trainiert. Die SVMs erhalten von den SOMs die Gewichtsvektoren und die Gelenkwinkel werden vom kinematischen Modell übergeben.

Für das Training der SVMs wird eine Bewegungssequenz aufgenommen, in der alle Gesten von einem Probanden präsentiert werden. Anschließend wird die Posenbestimmung auf der Sequenz angewendet und manuell für jede Geste eine Menge an Beispielposen herausgesucht. Jeder Datensatz enthält die Gewichte der SOMs und die Gelenkwinkel des kinematischen Modells. Anhand dieser und der zugeordneten Geste werden die SVMs trainiert.

Um eine Unabhängigkeit der SOM-Gewichte von der Orientierung und Position des Körpers und der Körpergröße zu gewährleisten, wird vorher eine Hauptkomponentenanalyse (Principal Component Analysis; PCA) durchgeführt. Die PCA nutzt eine Orthogonaltransformation, um eine Menge von Variablen in linear unkorrelierte Variablen

umzuwandeln. Diese werden *principal components* genannt, deren Anzahl gleich groß oder geringer als die der ursprünglichen Variablen ist. Dabei werden die Achsen ermittelt, entlang derer die Daten die größte Varianz aufweisen. Anschließend erfolgt eine Projektion der Daten in das Koordinatensystem, das durch die principal components aufgespannt wird. Ziel der PCA ist es in diesem Fall, die vertikalen und horizontalen Achsen des Körpers zu finden, um eine einheitliche Basis für die Interpretation der Daten zu schaffen und eine Auswirkung der Orientierung des Körpers auf das Ergebnis der Klassifikation auszuschließen.

Als Berechnungsgrundlage der PCA werden nur die Merkmale des Oberkörpers betrachtet. Dies sorgt dafür, dass eine unterschiedliche Ausrichtung der Arme und Beine nicht zur Folge hat, dass sich das Ergebnis der PCA und somit die Interpretation der Gesten ändert. Die Gewichte der SOMs werden anschließend in das Koordinatensystem der ermittelten Hauptkomponenten projiziert und an die SVMs übergeben. Abbildung 3.4 vergleicht verschiedene Posen der SSOM und das Ergebnis der PCA unter Berücksichtigung aller Punkte des Modells beziehungsweise nur der Punkte des Oberkörpers.

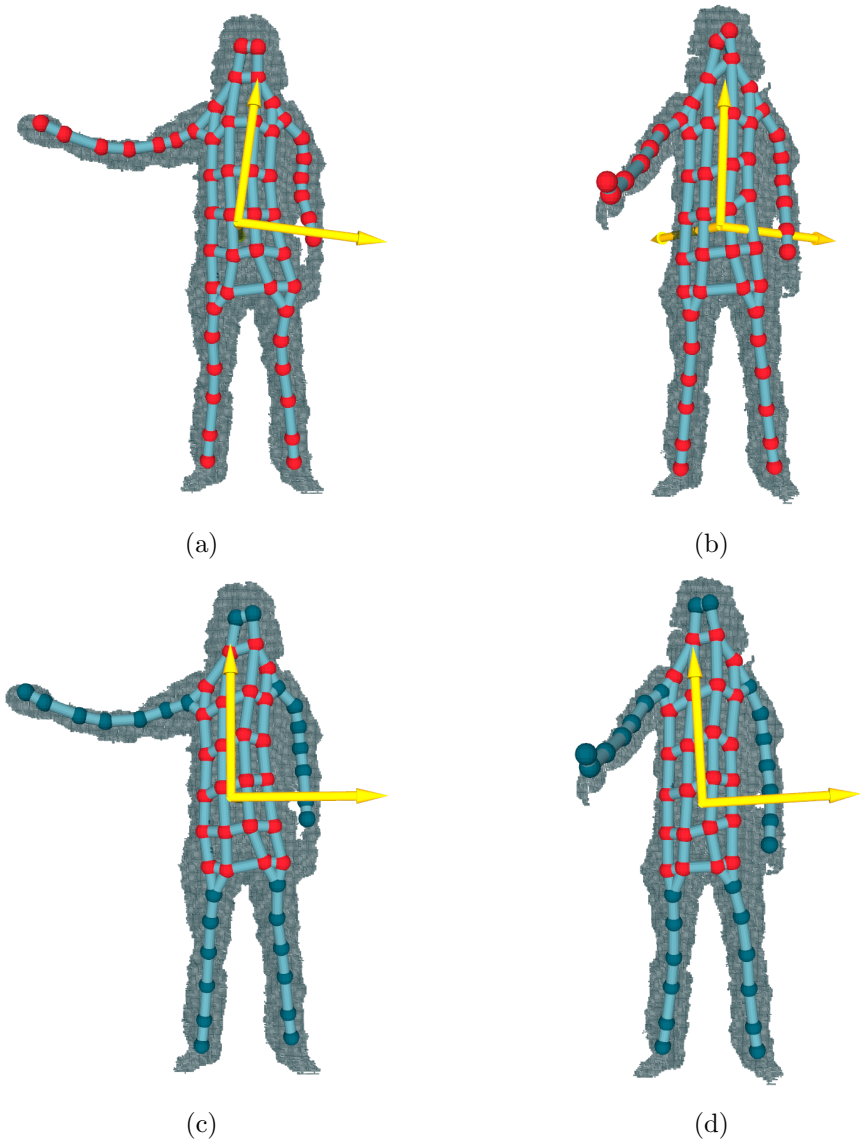
Damit die Pose eines Armes nicht das Ergebnis der Klassifizierung des anderen Armes beeinflusst, erhalten die SVMs nur die Daten des jeweiligen Armes und des Oberkörpers. Dadurch ist für die Geste eines Armes nur die Pose relativ zum Oberkörper relevant.

Anschließend wird mit einem Abstimmungsverfahren eine Geste ausgewählt. Jede SVM stimmt für eine Geste ab und die Geste mit den meisten Stimmen wird als Ergebnis der Gestenerkennung ausgegeben. Kommen alle drei SVMs zu einem anderen Ergebnis, wird die Geste ausgewählt, die bei einer alphabetischen Sortierung der Gestennamen als letztes steht, damit eine Entscheidung getroffen werden kann.

3.3. Portierung auf Pepper

Das Verfahren zur Posenbestimmung wurde in C++ programmiert und verwendet das Robot Operating System (**ROS**) [12] zur Kommunikation mit der Tiefenbildkamera und die Point Cloud Library (**PCL**) [13] zur Verarbeitung der Punktwolke. Für Pepper existiert eine Anbindung an ROS, die es jedoch nicht erlaubt, Programme lokal auf Pepper auszuführen, wodurch die ROS-Anbindung für eine autonome Verwendung der Posenbestimmung nicht geeignet ist. Darüber hinaus ist die Frequenz, in der Bilder von der Kamera über das Netzwerk empfangen werden können, aufgrund der begrenzten Bandbreite unzureichend. Um das langfristige Ziel der lokalen Ausführung zu erreichen, musste daher das Verfahren auf das qi C++ SDK für Pepper portiert werden und von ROS auf die NaoQI API umgestellt werden.

Das qi C++-SDK wird nicht mit einer zu Pepper kompatiblen Version von PCL ausgeliefert. Außerdem ist es nicht möglich, Bibliotheken wie PCL selber für Pepper bereitzustellen. Einzig die Bibliothek OpenCV zur zweidimensionalen Verarbeitung von Bildern ist im qi C++-SDK enthalten. Das Verfahren wurde deshalb von der 3D-Verarbeitung



(a) (b)

(c) (d)

Abbildung 3.4.: Anwendung der PCA. Dargestellt ist jeweils eine Punktwolke mit einer Visualisierung der SSOM und ein Koordinatensystem mit den durch die PCA ermittelten Achsen. Die Punkte, die zur Berechnung der PCA verwendet wurden, sind rot gefärbt. Ziel ist ein Koordinatensystem, das die Orientierung des Körpers widerspiegelt und es ermöglicht, die Posen der Arme relativ zum Oberkörper zu bestimmen. (a) und (b) zeigen das Ergebnis der PCA unter Berücksichtigung aller Punkte. Das Koordinatensystem wird stark durch die Posen der Arme beeinflusst und spiegelt nicht die Ausrichtung des Oberkörpers wider. In (c) und (d) werden nur die Punkte des Oberkörpers berücksichtigt. Die Posen der Arme wirken sich nicht auf die Ausrichtung des Koordinatensystems aus.

der Punktwolke auf die zweidimensionale Verarbeitung des Tiefenbildes mit Hilfe von OpenCV umgestellt.

Optimierung des Verfahrens

Pepper besitzt eine Intel Atom CPU, die auf einen geringen Stromverbrauch ausgelegt ist. Aufgrund der schwachen Hardware erfolgten einige Optimierungen, um eine ausreichende Ausführungsgeschwindigkeit des Verfahrens zu gewährleisten. Peppers Kamera erzeugt Bilder mit einer Frequenz von 30 Hz. Idealerweise sollte das Verfahren daher mindestens 30 Mal pro Sekunde durchlaufen können, damit es Bilder sofort nach Ankunft verarbeiten kann und keine Bilder überspringt.

Zunächst wurde die Anzahl der Punkte, die die SSOM, GSOM und das kinematische Modell erhalten, auf 1000 begrenzt. Die Umrechnung vom Tiefenbild in dreidimensionale Punkte wird nicht mehr in jeder Iteration beim Abruf eines Bildpunkts, sondern einmal im Voraus für alle Bildpunkte durchgeführt. Dadurch wird sichergestellt, dass die Umrechnung pro Punkt nicht mehr als einmal stattfindet und das Verfahren verlangsamt.

An einigen Stellen benötigt das Verfahren die Berechnung von euklidischen Distanzen, welche das rechenintensive ziehen der Wurzel erfordert. Um Rechenzeit zu sparen, wird die Berechnung der Wurzel ausgelassen. Da die Distanzen nur berechnet werden, um die kleinste Distanz zu finden oder zwei Distanzen zu vergleichen, ändert sich das Ergebnis der Berechnungen nicht.

Vorverarbeitung des Tiefenbilds

Um sicherzustellen, dass das Verfahren eine Punktwolke empfängt, die nur den Menschen enthält, der gerade mit Pepper interagiert, wurde ein Modul entwickelt, dass das empfangene Tiefenbild vorverarbeitet und die zum Menschen gehörenden Punkte extrahiert.

Abbildung 3.5 zeigt den Ablauf der Extraktion. Im ersten Schritt wird festgestellt, wo im Bild sich der Mensch befindet, um den Bereich für die Extraktion einzugrenzen. Das Modul empfängt das Tiefenbild von Peppers Kamera und fragt von der NaoQI API Informationen über im Kamerabild gefundene Gesichter ab. Ist kein Gesicht erkannt worden, wird die Verarbeitung abgebrochen und auf das nächste Tiefenbild gewartet.

Falls jedoch ein Gesicht gefunden wurde, werden alle Punkte entfernt, die zu weit vor oder hinter dem Menschen sind. Dazu wird der Tiefenwert an der Position des Gesichts betrachtet. Ist ein Punkt mehr als einen Meter vor oder mehr als 50 cm hinter dem Gesicht, wird er entfernt. Die betrachtete Distanz in Richtung der Kamera ist größer, damit nach vorne ausgestreckte Arme nicht abgeschnitten werden. Abbildung 3.6 zeigt diesen Schritt an einem Beispiel.

Als Nächstes wird aus dem Tiefenbild der Boden entfernt, damit dieser nicht als Teil

des Menschen erkannt wird und diesen nicht mit umliegenden Gegenständen verbindet. Dafür wird der Punkt mit der geringsten y-Koordinate gesucht und alle Punkte entfernt, die bis zu 15 cm höher als dieser Punkt liegen. Dies funktioniert unter der Annahme, dass die Blickrichtung der Kamera parallel zum Boden ist. In Abbildung 3.7 ist ein Beispiel für die Entfernung des Bodens zu sehen. Der Schwellwert wurde zunächst auf 5 cm gesetzt, da von einer geringen Varianz der y-Koordinaten der Bildpunkte im Boden ausgegangen wurde. Erste Versuche ergaben jedoch, dass der tiefste Punkt im Bild deutlich tiefer lag, als vermutet. Um sicherzugehen, dass der Boden komplett entfernt wird, wurde der Schwellwert daher auf 15 cm erhöht.

Anschließend werden alle zusammenhängenden Flächen im Bild gesucht. Dafür wird eine Maske erstellt, die dort weiß ist, wo sich nach den vorherigen Schritten noch Punkte befinden, und ansonsten schwarz ist. In dieser Maske wird jeder Bildpunkt zu einem kleinen Kreis ausgedehnt, damit direkt benachbarte Flächen zusammenwachsen, sodass der Mensch im Bild eine große Fläche darstellt. Mit Hilfe der OpenCV-Bibliothek werden alle zusammenhängenden Flächen extrahiert. Das Ergebnis der Extraktion ist die Fläche, die den Bildpunkt enthält, an dem sich das Gesicht des Menschen befindet. Die Punkte, die zu dieser Fläche gehören, werden an das Verfahren übergeben.

Wurde das Verfahren noch nicht initialisiert, wird zuletzt überprüft, ob der Mensch die T-Pose eingenommen hat. Da dies die initiale Pose der Modelle ist, müssen sie sich nur geringfügig an die Punktwolke anpassen. Entspricht die Höhe der menschlichen Silhouette ungefähr der Breite, hat der Mensch die Arme zur Seite ausgestreckt und steht in der T-Pose. Ist dies nicht der Fall, wird auf das nächste Tiefenbild gewartet.

Das Vorverarbeitungsmodul setzt das Verfahren bei Bedarf auf den Initialzustand zurück. Dies geschieht entweder auf den Befehl eines anderen Programms oder falls der Mensch für einige Sekunden aus dem Bild verschwindet. Letzteres sorgt dafür, dass das Verfahren nicht auf die Statur einer Person festgelegt ist, da die Knochenlängenanpassung nach der Initialisierung erneut geschieht.

Test des Verfahrens über das Netzwerk

Als Hilfe zur Fehlerlokalisierung für die Portierung wurde ein Visualisierungsprogramm entwickelt, das den Ablauf des Verfahrens auf einem PC visualisiert. Das Verfahren sendet das aktuelle Tiefenbild sowie die Topologien der SSOM, der GSOM und des kinematischen Modells und die Namen der erkannten Gesten über das Netzwerk an das Programm, welches die Modelle visualisiert. Dadurch kann im Fall eines Fehlverhaltens bestimmt werden, welcher Teil des Verfahrens nicht korrekt arbeitet. Abbildung 3.8 zeigt die Benutzeroberfläche des Programms.

Abbildung 3.5.: Extraktion des Menschen aus der Szene. (a) zeigt das Tiefenbild, das von der Kamera empfangen wurde. (b) Im nächsten Schritt wird die Position des Gesichts ermittelt. Der rote Kreis zeigt die Position des Gesichts. Alle Punkte, die mehr als einen Meter vor oder mehr als 50 cm hinter dem Gesicht sind, werden entfernt. (c) Anschließend wird der Boden aus dem Bild entfernt. (d) Nach Ausdehnung aller verbleibenden Bildpunkte werden alle zusammenhängenden Flächen im Bild gesucht. Die Fläche, in der sich das Gesicht befindet, ist die Kontur des Menschen. (e) Übrig bleiben nur die Punkte, die zum Menschen gehören.

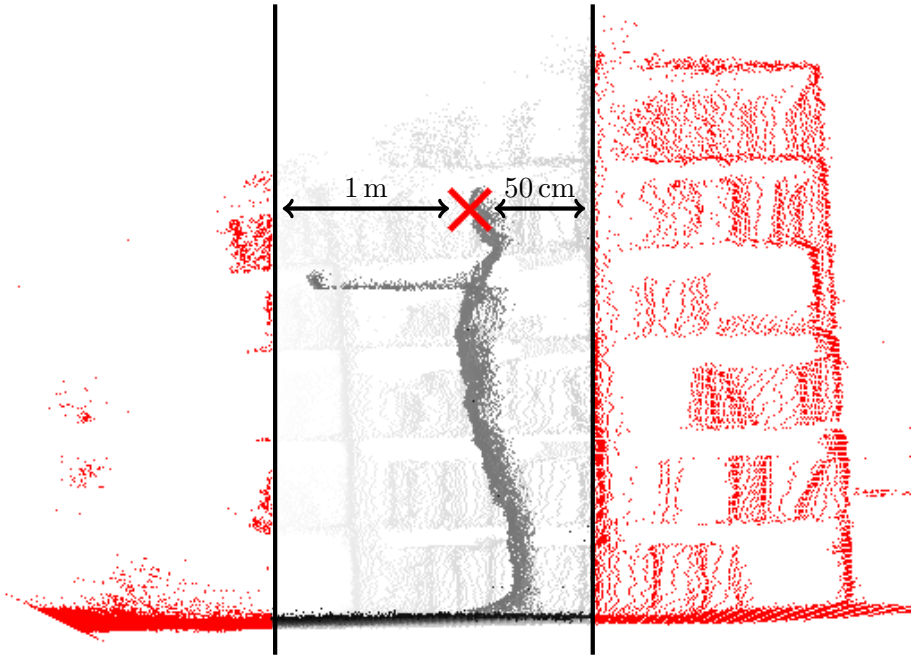


Abbildung 3.6.: Entfernung zu naher oder zu weit entfernter Punkte. Zu sehen ist eine Seitenansicht einer Punktwolke. Die Kamera befindet sich am linken Rand des Bildes. Das Gesicht des Menschen ist durch ein Kreuz markiert. Alle Punkte, die sich mehr als einen Meter vor oder mehr als 50 cm hinter dem Gesicht befinden, werden verworfen. Der Abstand nach vorne ist größer gewählt, sodass die Arme nicht abgeschnitten werden, wenn sie nach vorne ausgestreckt sind.

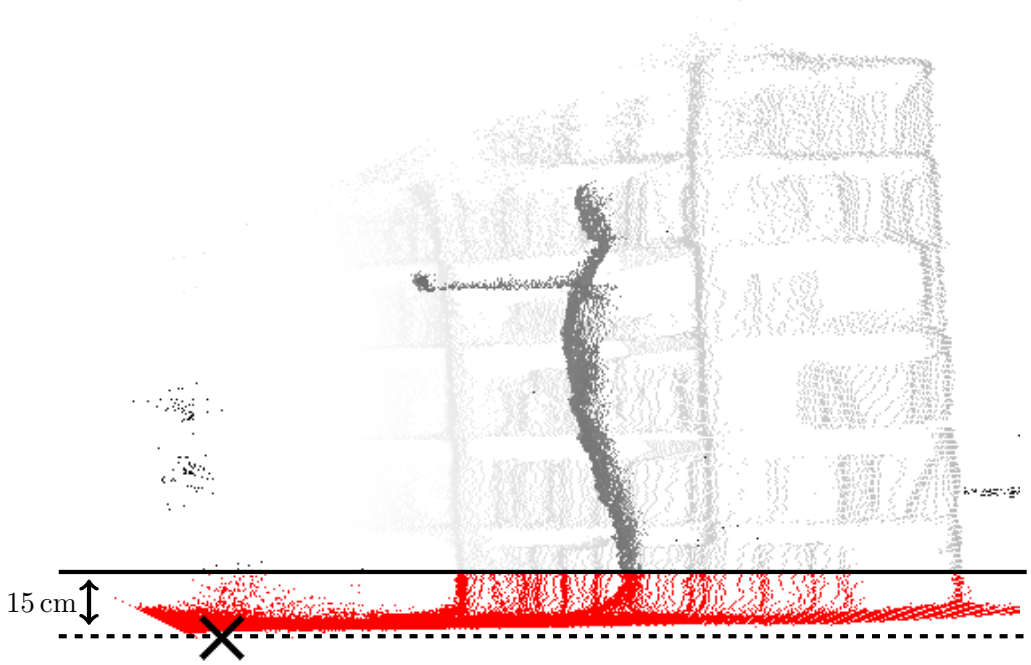


Abbildung 3.7.: Entfernung des Bodens. Zu sehen ist eine Seitenansicht einer Punktwolke. Die Kamera befindet sich am linken Rand des Bildes. Das Kreuz markiert den Punkt mit der geringsten y-Koordinate im Raum. Unter der Annahme, dass dieser Punkt Teil des Bodens ist, werden alle Punkte, die bis zu 15 cm höher liegen, verworfen.

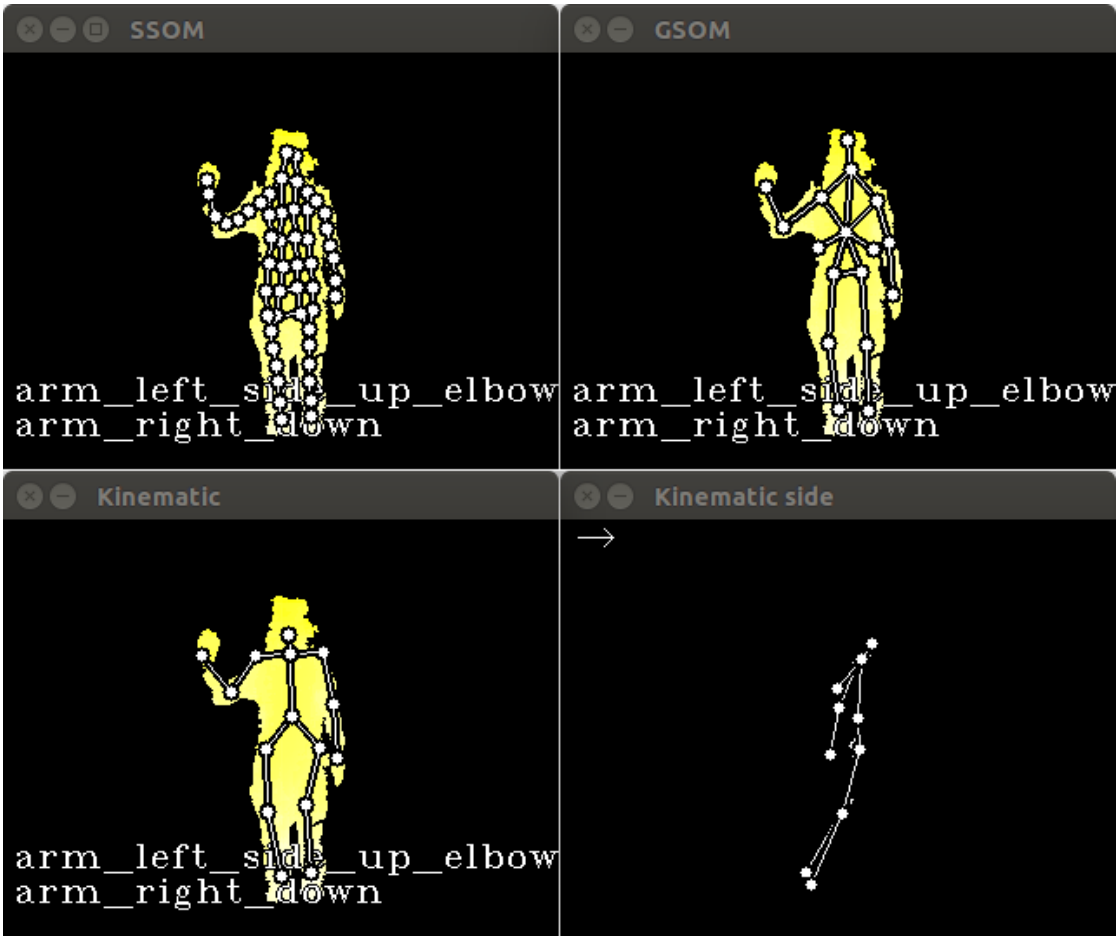


Abbildung 3.8.: Programm zum Testen des Verfahrens über das Netzwerk. Das Programm erhält vom Verfahren das aktuelle Tiefenbild, die Topologien von SSOM, GSOM und dem kinematischen Modell sowie für beide Arme die erkannte Geste. Alle Modelle werden zusammen mit dem Tiefenbild visualisiert, das kinematische Modell wird zusätzlich in einer Seitenansicht gezeigt.

4. Evaluation

In diesem Kapitel erfolgt die Evaluation der Posenbestimmung, der Gestenerkennung und der Portierung des Verfahrens mit anschließender Präsentation und Diskussion der Ergebnisse.

4.1. Testaufbau

Dieser Abschnitt beschreibt den Ablauf der Experimente zur Evaluation der Posenbestimmung, Gestenerkennung und Portierung des Verfahrens.

4.1.1. Posenbestimmung

Die Posenbestimmung wurde mit zwei öffentlichen, von Ganapathi et al. [14, 15] erstellten Datensätzen evaluiert, die Sequenzen von Tiefenbildern und Annotationen mit den Positionen bestimmter Körpermerkmale für jedes Bild der Sequenz enthalten. Im Folgenden werden die Datensätze mit CVPR und ECCV bezeichnet. CVPR besteht aus 28 Sequenzen, ECCV aus 24 Sequenzen. Eine Liste der betrachteten Merkmale und ihrer Abkürzungen findet sich in Tabelle 4.1.

Die Sequenzen wurden abgespielt und die von der Posenbestimmung berechneten Positionen der Merkmale aufgezeichnet. Anschließend wurden die Positionen der Merkmale im kinematischen Modell mit den annotierten Positionen des jeweiligen Datensatzes verglichen. Dazu erfolgte eine Berechnung des euklidischen Abstands zwischen der berechneten Position und der vom jeweiligen Datensatz angegebenen Position. Um einen eventuellen positiven Effekt der gegenseitigen Einflussnahme der Verfahren nachzuweisen, wurde sowohl für das ursprüngliche Verfahren als auch das um die gegenseitige Beeinflussung der Modelle erweiterte Verfahren der Merkmalsabstand bestimmt.

Es sei darauf hingewiesen, dass die in Abbildung 4.1 beispielhaft gezeigten Annotationen der Merkmale nicht in den entsprechenden Zentren, sondern am äußeren Rand der Silhouette liegen, das hier entwickelte Verfahren jedoch die Zentren der Merkmale wie das Zentrum der Handfläche bei frontal zu Kamera gerichteter Hand bestimmt. Dies führt zu einem standardmäßigen Offset als Grundfehler, der sich nicht restlos herausrechnen lässt. Um dennoch zu versuchen, den Fehler auszugleichen, erfolgte eine Berechnung des durchschnittlichen Versatzes zwischen den Annotationen und der tatsächlichen Merk-

Merkmal	Abkürzung
Kopf	H
Linke Schulter	LS
Rechte Schulter	RS
Linker Ellenbogen	LE
Rechter Ellenbogen	RE
Linke Hand	LHa
Rechte Hand	RHa
Linke Hüfte	LHi
Rechte Hüfte	RHi
Linkes Knie	LK
Rechtes Knie	RK
Linker Fuß	LF
Rechter Fuß	RF

Tabelle 4.1.: Liste der betrachteten Merkmale und ihrer Abkürzungen.

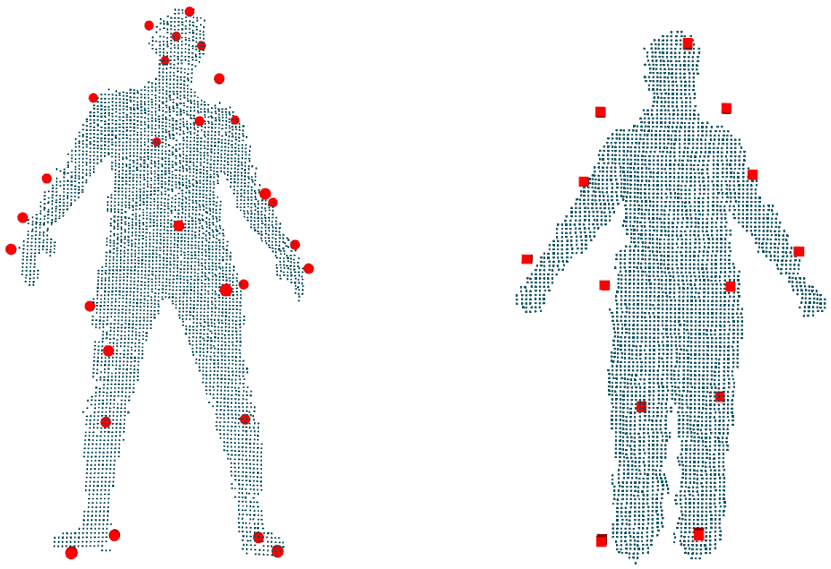
malsposition. Dazu wurden für jeden Datensatz zehn Punktwolken zufällig ausgewählt und manuell neu annotiert. Für jedes Merkmal wurde ein Punkt in der Punktwolke ausgewählt, der in der Mitte des Merkmals liegt, sodass ein Vektor bestimmt werden konnte, der den Unterschied zwischen der ursprünglichen Annotation und der tatsächlichen Position des Merkmals beschreibt. Anschließend wurde der durchschnittliche Merkmalsabstand nach Abzug des Versatzes bestimmt.

4.1.2. Gestenerkennung

Für die Evaluation der Gestenerkennung wurde eine Liste an eindeutigen Gesten beider Arme definiert. Einige Beispiele sind in Abbildung 4.2 zu sehen. Die Gesten wurden von einem Probanden präsentiert und die Bewegungen mit einer Kinect for XBox One in Form von Sequenzen aus Punktwolken aufgenommen. Aus diesem Datensatz wurden für jede Geste eine möglichst große Anzahl an Punktwolken ausgewählt, die die jeweilige Geste repräsentieren, ohne zu sehr einer der anderen Gesten zu ähneln. Anschließend wurde die Gestenerkennung mit den ausgewählten Punktwolken trainiert.

Als Testsequenz wurde ein Bewegungsablauf eines Probanden, der möglichst viele Gesten und deren Übergänge präsentieren sollte, mit einer ASUS Xtion Pro aufgenommen. Diese Aufnahme wurde mit einem selbsterstellten Programm annotiert. Dabei wurde in jedem Bild überprüft, ob die Posen der Arme eindeutig einer der Gesten entsprachen und gegebenenfalls der Name der Geste dem Bild zugeordnet. Darüber hinaus zeigt das Programm die Posen der SOMs und des kinematischen Modells und die von den SVMs klassifizierten Gesten, falls die Posenbestimmung bereits mit dem zu annotierenden Datensatz ausgeführt wurde. Abbildung 4.3 zeigt die Benutzeroberfläche des Programms.

Die Testsequenz wurde abgespielt und an die Posenbestimmung und Gestenerkennung



(a) (b)

Abbildung 4.1.: Beispiele der Merkmalspositionen aus den Datensätzen (a) CVPR und (b) ECCV. Die roten Markierungen sind die Positionen der Merkmale im jeweiligen Datensatz. Bei beiden zeigt sich, dass die Positionen der Merkmale nicht immer in der Mitte des zugehörigen Körperteils sind.

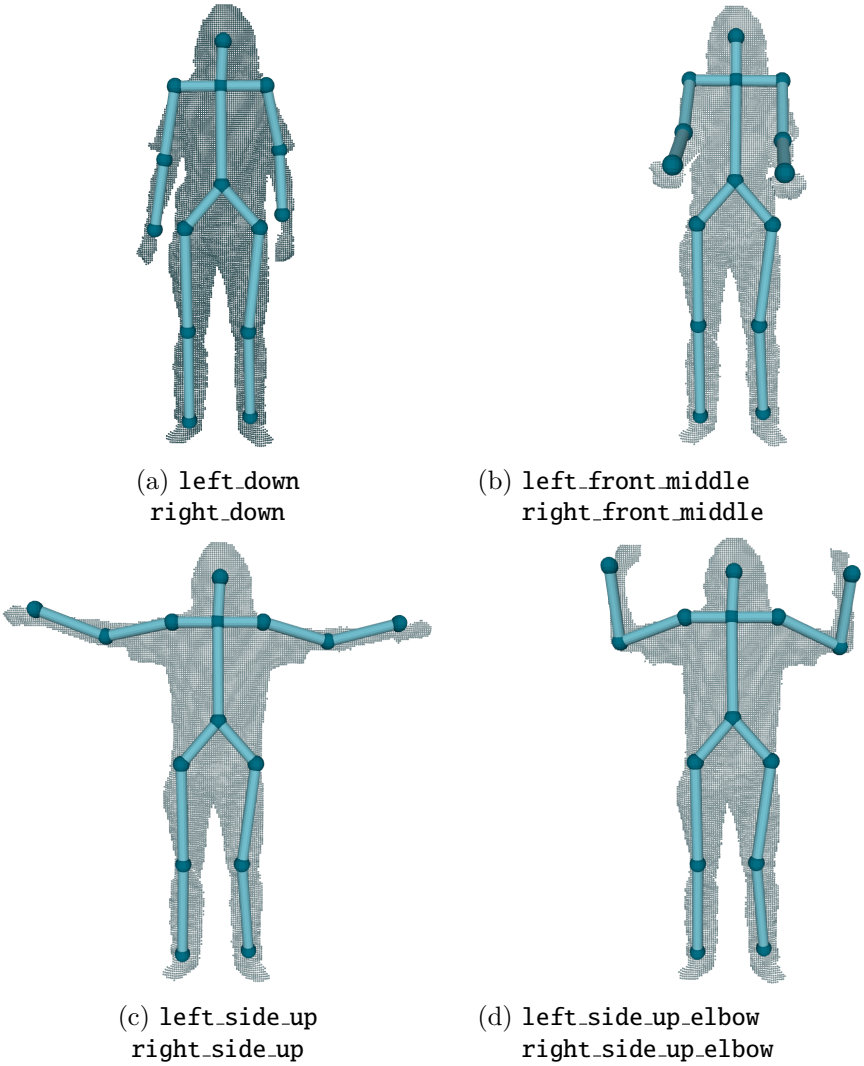
weitergegeben. Für jedes Bild der Sequenz wurden die klassifizierten Gesten aufgezeichnet. Zur Evaluation wurden die klassifizierten Gesten mit den Annotationen verglichen und die Häufigkeit richtig erkannter Gesten bestimmt.

4.1.3. Portierung auf Pepper

Für die Evaluation der Portierung des Verfahrens auf Pepper wurde die erreichte Anzahl an Durchläufen des Verfahrens pro Sekunde gemessen. Diese Messung wurde auf der selben Tiefenbildsequenz für das ursprüngliche ROS-basierte Verfahren, das auf einem PC ausgeführte NaoQI-basierte Verfahren und das direkt auf der Hardware von Pepper ausgeführte Verfahren durchgeführt. Die PC-basierten Evaluationen wurden auf einem PC mit einer Intel Core i7-3770 CPU mit 3.4 GHz durchgeführt. Pepper besitzt eine Intel Atom E3845 CPU mit 1.91 GHz.

4.2. Ergebnisse

In diesem Abschnitt werden die Ergebnisse der Evaluation der Posenbestimmung, der Gestenerkennung und der Portierung präsentiert.



(a) `left_down` `right_down`

(b) `left_front_middle` `right_front_middle`

(c) `left_side_up` `right_side_up`

(d) `left_side_up_elbow` `right_side_up_elbow`

Abbildung 4.2.: Beispiele der Gesten, mit denen evaluiert wurde. Die Bildunterschriften sind die intern vergebenen Namen für die Gesten.

4.2.1. Posenbestimmung

Für die Evaluation der Posenbestimmung wurde der durchschnittliche Fehler je Sequenz und der durchschnittliche Fehler je Merkmal betrachtet.

Fehler je Sequenz

Abbildung 4.4 zeigt den durchschnittlichen Fehler des kinematischen Modells und die Standardabweichung je Sequenz im Datensatz CVPR. Für jede Sequenz ist der Fehler

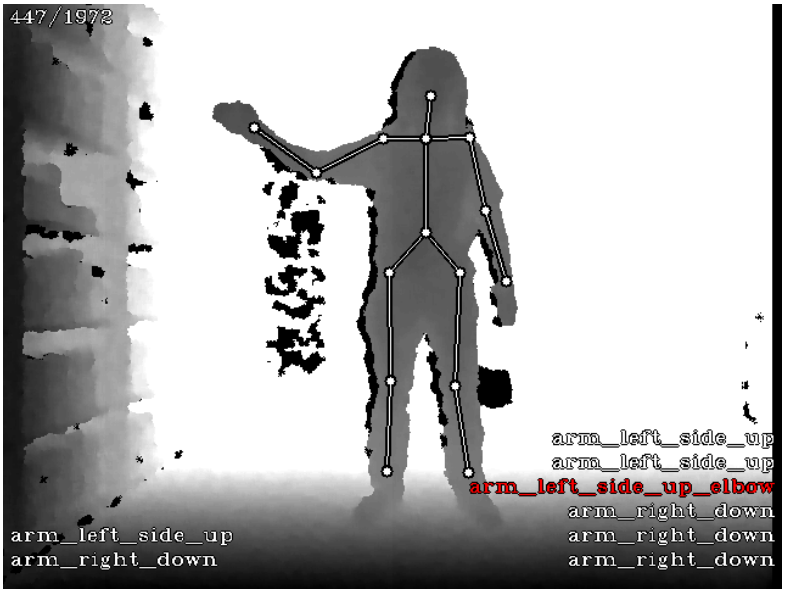


Abbildung 4.3.: Werkzeug zur Annotation von Tiefenbildsequenzen. Unten links werden die vom Benutzer zugewiesenen Gesten angezeigt. Wurde die Sequenz bereits vom Verfahren durchlaufen, wird eine Visualisierung des kinematischen Modells über das Tiefenbild gelegt, es können auch Visualisierungen der SSOM und GSOM angezeigt werden. Unten rechts werden die Klassifizierungsergebnisse aller SVMs angezeigt. Stimmt eine Klassifizierung nicht mit der Annotation des Benutzers überein, wird dies mit roter Schriftfarbe signalisiert.

ohne und mit Beeinflussung durch SSOM und GSOM aufgetragen. Der Fehler verringert sich durch die Beeinflussung im Durchschnitt um 2.2 cm. Der durchschnittliche Fehler für das Verfahren mit Beeinflussung liegt bei 15.5 cm. Nach Abzug des in Abschnitt 4.1.1 erläuterten Versatzes verringert sich der Fehler auf 12.3 cm. Der durchschnittliche Fehler je Sequenz nach Abzug des Versatzes findet sich in Abbildung A.1. Die Standardabweichung des Fehlers liegt bei 5.3 cm ohne Beeinflussung und 4.9 cm mit Beeinflussung. Sequenz 25 hat einen deutlich abweichenden Fehler von 23 cm.

Abbildung 4.5 zeigt den durchschnittlichen Fehler des kinematischen Modells und die Standardabweichung je Sequenz im Datensatz ECCV. Für jede Sequenz ist der Fehler ohne und mit Beeinflussung durch SSOM und GSOM aufgetragen. Durch die Beeinflussung verringert sich der Fehler im Durchschnitt um 3.3 cm. Der durchschnittliche Fehler liegt bei 16.79 cm. Nach Abzug des in Abschnitt 4.1.1 erläuterten Versatzes verringert sich der Fehler auf 13.35 cm. Die Standardabweichung des Fehlers liegt bei 10.5 cm ohne Beeinflussung und 6.7 cm mit Beeinflussung. Der durchschnittliche Fehler je Sequenz nach Abzug des Versatzes findet sich in Abbildung A.2. In den Sequenzen 7, 15 und 23 weicht der Fehler stark ab.

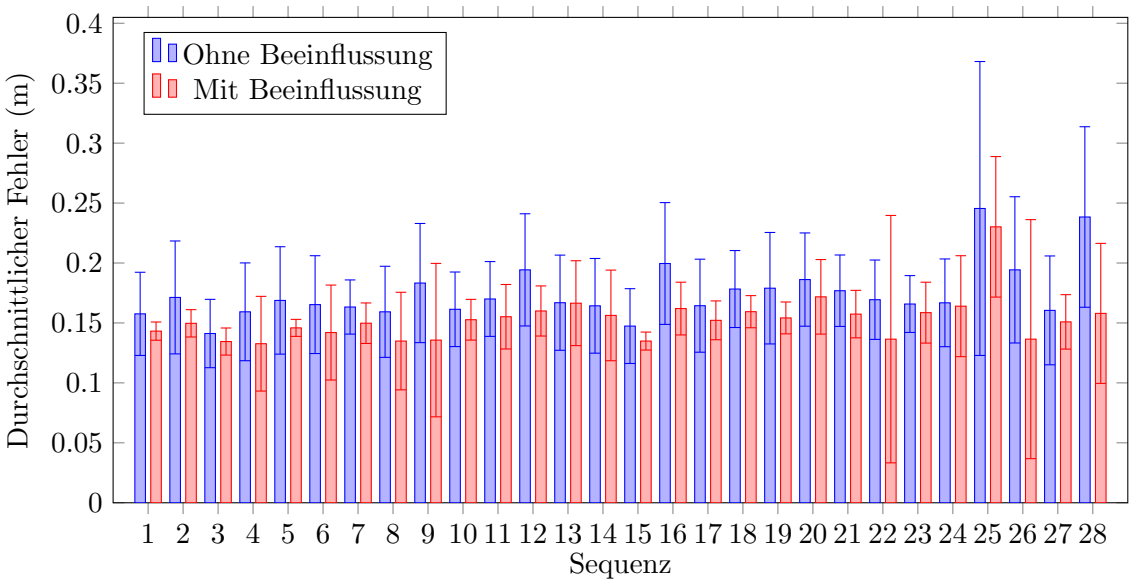


Abbildung 4.4.: Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz CVPR. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt.

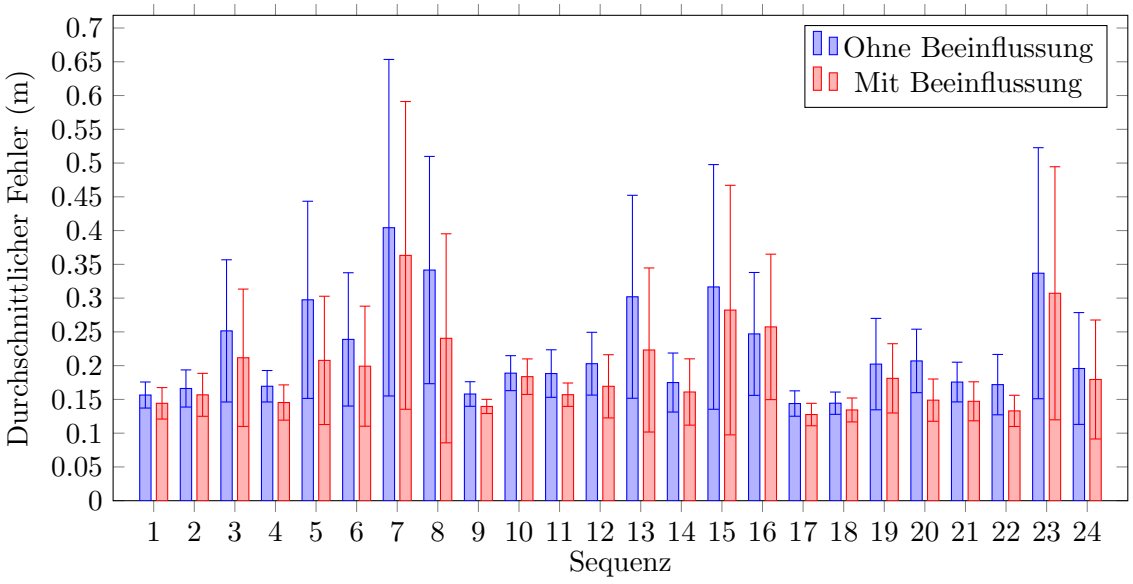


Abbildung 4.5.: Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz ECCV. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt.

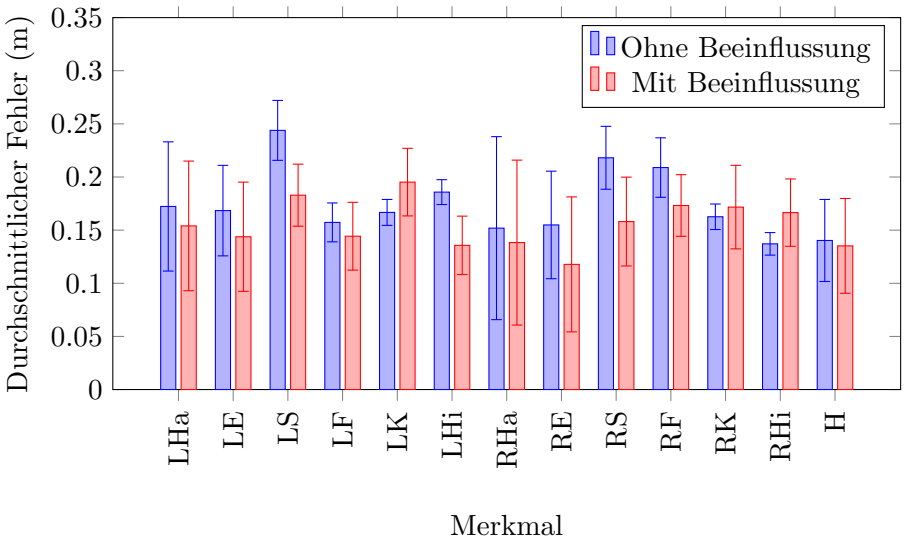


Abbildung 4.6.: Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz CVPR. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt. Die Merkmale und ihre Abkürzungen finden sich in Tabelle 4.1.

Fehler je Merkmal

Abbildung 4.6 zeigt den Fehler des kinematischen Modells und die Standardabweichung je Merkmal im Datensatz CVPR. Für jedes Merkmal ist der Fehler ohne und mit Beeinflussung durch SSOM und GSOM aufgetragen. Durch die Beeinflussung verringert sich der Fehler im Durchschnitt um 2 cm, bei den linken und rechten Knien und der rechten Hüfte zeigt sich jedoch eine Verschlechterung um 2.8 cm, 0.9 cm beziehungsweise 2.9 cm. Der durchschnittliche Fehler je Merkmal nach Abzug des Versatzes findet sich in Abbildung A.3.

Abbildung 4.7 zeigt den Fehler des kinematischen Modells und die Standardabweichung je Merkmal im Datensatz ECCV. Für jedes Merkmal ist der Fehler ohne und mit Beeinflussung durch SSOM und GSOM aufgetragen. Der Fehler verringert sich durch die Beeinflussung im Durchschnitt um 5.6 cm. Der durchschnittliche Fehler je Merkmal nach Abzug des Versatzes findet sich in Abbildung A.4.

4.2.2. Gestenerkennung

Abbildung 4.8 zeigt die Genauigkeit der Gestenerkennung für jede Geste. Insgesamt wurden die Gesten in 97.4% der Fälle korrekt erkannt. Die Genauigkeit liegt bei fast allen Gesten bei über 95%. Lediglich die Gesten `left_front_up` und `right_front_up` wurden in 88.1% beziehungsweise 86.6% der Fälle korrekt erkannt.

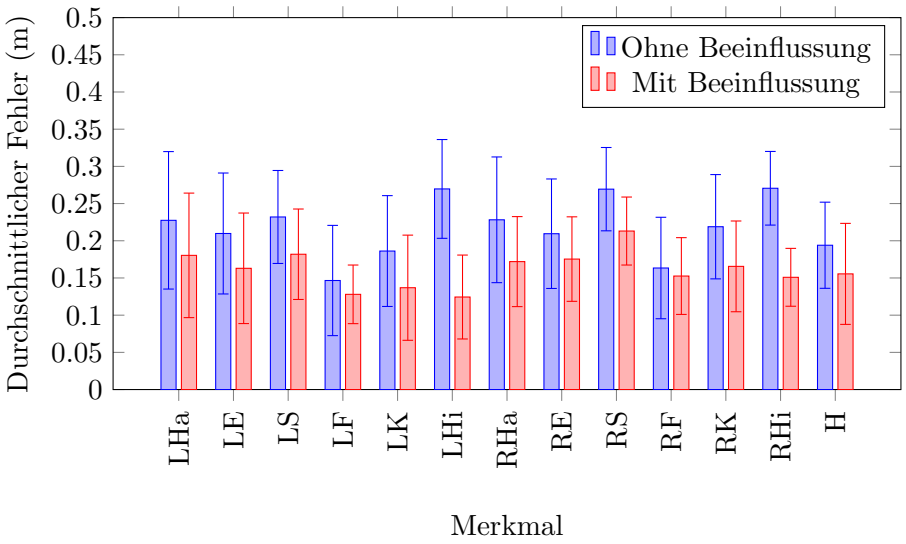


Abbildung 4.7.: Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz ECCV. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt. Die Merkmale und ihre Abkürzungen finden sich in Tabelle 4.1.

4.2.3. Portierung auf Pepper

Abbildung 4.9 zeigt die erreichte Anzahl an Durchläufen pro Sekunde für das ursprüngliche Verfahren, das optimierte Verfahren und das auf Pepper ausgeführte Verfahren. Im Durchschnitt erreichte das ursprüngliche Verfahren 69 Durchläufe pro Sekunde, das optimierte Verfahren konnte durchschnittlich 102 mal pro Sekunde ausgeführt werden. Auf Pepper erreicht das Verfahren eine durchschnittliche Ausführungsrate von 21 Durchläufen pro Sekunde.

4.3. Diskussion

In diesem Abscnitt werden die Ergebnisse der Evaluation der Posenbestimmung, der Gestenerkennung und der Portierung diskutiert.

Posenbestimmung

Bei Betrachtung des Fehlers je Sequenz in den Datensätzen CVPR und ECCV fällt die Verbesserung durch die Beeinflussung des kinematischen Modells durch SSOM und GSOM nur gering aus, der Fehler verringert sich im Durchschnitt um 2.2 cm beziehungsweise 3.3 cm.

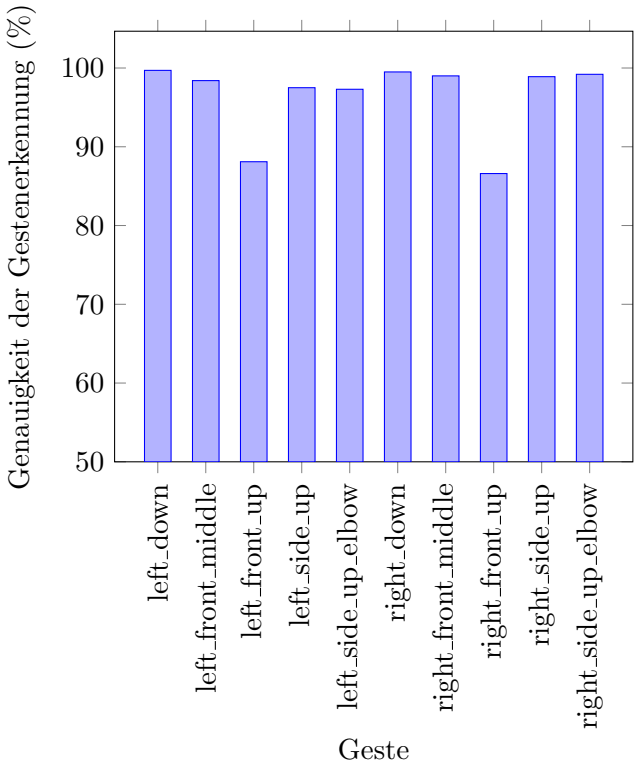


Abbildung 4.8.: Genauigkeit der Gestenerkennung nach Gesten. Bis auf die Gesten `left_front_up` und `right_front_up` wurden alle Gesten in mindestens 95% der Fälle korrekt erkannt.

Für den Datensatz CVPR liegt der durchschnittliche Fehler bei 15.5 cm und damit höher als der Fehler von Ganapathi et al. [14] von ungefähr 7 cm. Ein Grund für diesen Unterschied könnte die in Abbildung 4.1 gezeigte Diskrepanz zwischen der annotierten Position der Körpermerkmale und ihrer eigentlichen Position sein. Dies bestätigt sich auch dadurch, dass nach Abzug des durch manuelle Annotationen ermittelten Versatzes der Fehler sich nochmals um 3.2 cm verringert. Aus der Standardabweichung des Fehlers mit gegenseitiger Beeinflussung von 4.9 cm lässt sich schließen, dass das kinematische Modell trotz des hohen durchschnittlichen Fehlers stabil die Punktwolke approximiert, was sich bei Betrachtung der Visualisierung des Modells bestätigen lässt.

Sequenz 25 hat einen deutlich abweichenden Fehler von 23 cm. Dies liegt daran, dass die Sequenz eine vollständige Drehung des Körpers enthält, wodurch beim Übergang von der Vorderseite zur Rückseite ein Großteil des Körpers verdeckt ist. Außerdem liegen nach der ersten Hälfte der Drehung die Modelle spiegelverkehrt im Körper, wodurch der Fehler stark erhöht wird.

Der durchschnittliche Fehler auf dem Datensatz ECCV liegt bei 16.79 cm. Ganapathi et al. definieren ein Merkmal als korrekt bestimmt, wenn der Fehler weniger als 10 cm

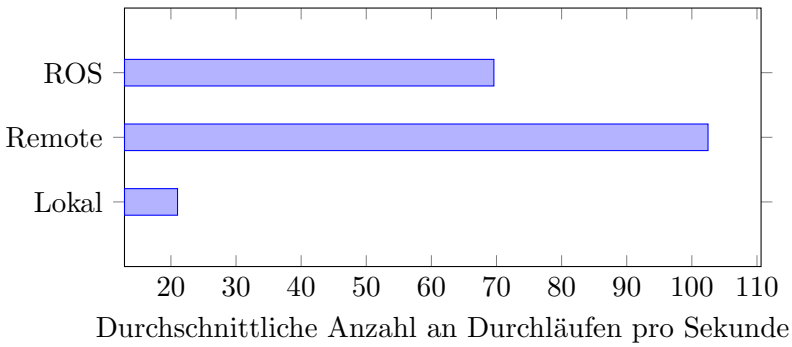


Abbildung 4.9.: Die erreichte Verarbeitungsgeschwindigkeit des ursprünglichen Verfahrens (**ROS**), des auf NAOqi portierten und optimierten Verfahrens (**Remote**) und des lokal auf Pepper ausgeführten Verfahrens (**Lokal**). Das ursprüngliche Verfahren erreicht im Durchschnitt 69 Durchläufe pro Sekunde. Nach der Optimierung ist es mit durchschnittlich 102 Durchläufen pro Sekunde deutlich schneller. Durch die schwache Hardware von Pepper erreicht das Verfahren dort nur durchschnittlich 21 Durchläufe pro Sekunde.

beträgt [15]. Ihr Verfahren erreicht mit dieser Definition eine Genauigkeit von 96 %. Wie bei CVPR lässt sich der deutlich größere Fehler des kinematischen Modells durch die Diskrepanz zwischen den annotierten Merkmalspositionen und den tatsächlichen Positionen erklären, da nach Abzug des durch manuelle Annotationen ermittelten Versatzes der Fehler sich nochmals um 3.4 cm verringert. Auch hier zeigt die Standardabweichung des Fehlers mit gegenseitiger Beeinflussung, die bei 6.7 cm liegt, dass das kinematische Modell die Daten stabil approximiert.

ECCV enthält deutlich komplexere und schnellere Bewegungen als CVPR, wodurch der durchschnittliche Fehler hier höher liegt. In den Sequenzen 7, 15 und 23 wird ein Handstand vorgeführt, die Sequenz 13 zeigt viele schnelle Schläge, hohe Tritte und eine vollständige Drehung des Körpers, wodurch viele Teile des Körpers verdeckt werden. Der Fehler ist in diesen Sequenzen entsprechend höher.

Bei Betrachtung des Fehlers je Merkmal in CVPR und ECCV zeigt sich eine Verbesserung durch die Beeinflussung im Durchschnitt um 2 cm für CVPR und 5.6 cm für ECCV. ECCV weist viele schnellere und komplexere Bewegungen als CVPR auf. Daher scheint die Beeinflussung besonders bei komplexen und schnellen Bewegungen eine Wirkung zu zeigen.

Die Abbildungen 4.10a bis 4.10h zeigen Ausschnitte aus den Sequenzen, in denen komplexe Posen mit verdeckten Körperteilen oder schnelle Bewegungen vorkommen, die das kinematische Modell dennoch gut approximieren konnte. Besonders in Abbildung 4.10c repräsentiert das kinematische Modell die Pose gut, obwohl in diesem Ausschnitt der Mensch einen großen, ausschweifenden Tritt macht, der viele Verdeckungen erzeugt.

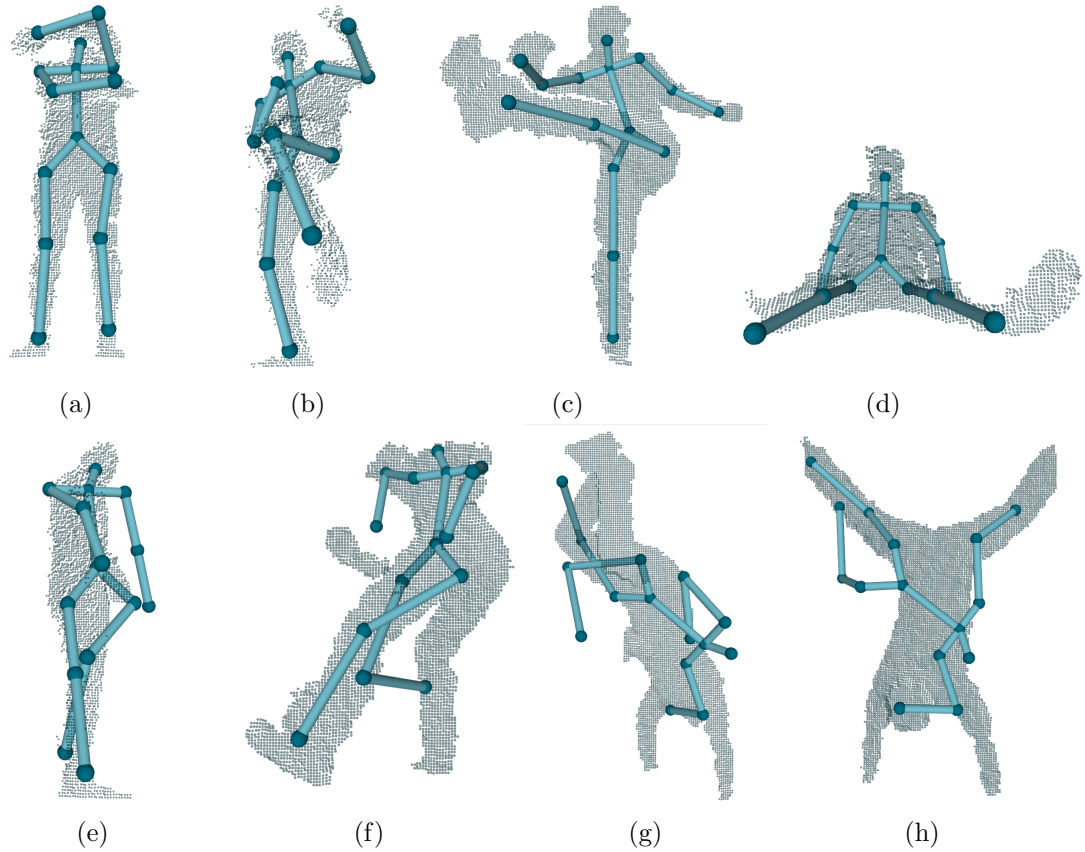


Abbildung 4.10.: Beispiele der Evaluation auf den Datensätzen CVPR und ECCV. (a) bis (d) zeigen komplexe Posen, die das kinematische Modell gut approximiert hat. (e) bis (h) zeigen Fälle, in denen das kinematische Modell große Fehler aufwies.

In den Abbildungen 4.10e bis 4.10h sind Ausschnitte zu sehen, in denen das kinematische Modell nicht korrekt in den Daten liegt. Die Abbildungen 4.10e und 4.10f sind bei Drehungen des Körpers entstanden. Durch den Übergang von einer Körperseite zur anderen, bei dem der Körper nur von der Seite zu sehen ist und die meisten Körpermerkmale verdeckt sind, kann das kinematische Modell die Pose nicht mehr gut bestimmen. In Abbildung 4.10g macht der Mensch einen Handstand, in Abbildung 4.10h schlägt er ein Rad. Auch hier wird der Fehler des kinematischen Modells beim Übergang von einer stehenden zur gezeigten Pose aufgrund vieler Verdeckungen sehr groß.

Gestenerkennung

Die Gestenerkennung hat eine hohe Genauigkeit erzielt. Insgesamt wurden die Gesten in 97.4% der Fälle korrekt erkannt, die Genauigkeit liegt bei fast allen Gesten bei über

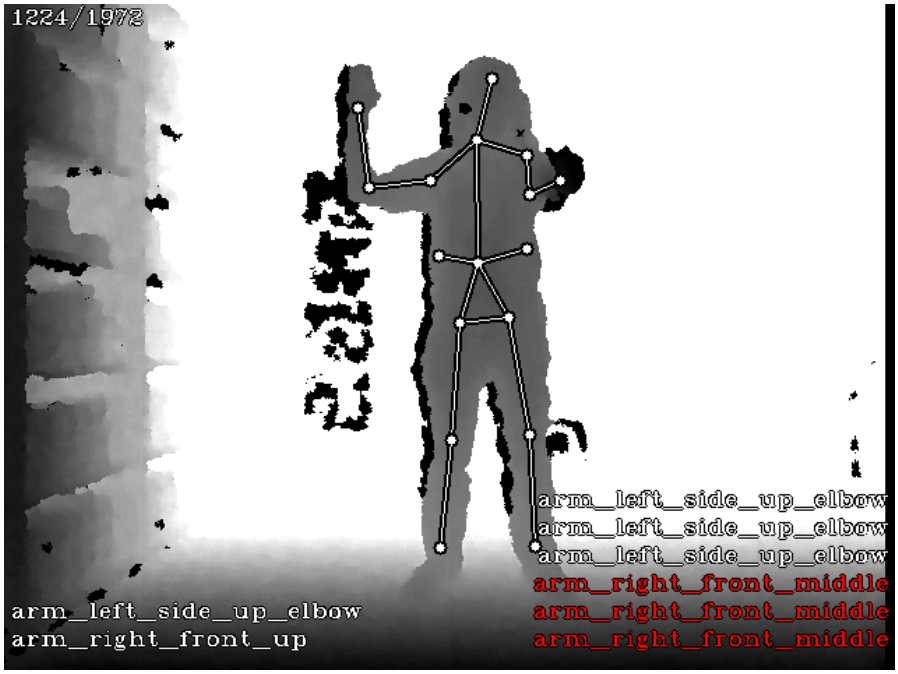


Abbildung 4.11.: Fehlklassifizierung der Geste `right_front_up`. Alle SVMs haben die Geste `right_front_middle` erkannt. Möglicherweise waren die Trainingsdaten für diese Gesten zu ähnlich oder wiesen Überschneidungen auf. Außerdem ist ein Großteil des Arms durch die Hand verdeckt, da der Arm direkt in Richtung der Kamera zeigt. Durch die fehlenden Punkte ist die Pose der Modelle ungenau, was sich auf die Gestenerkennung auswirkt.

95%. Lediglich die Gesten `left_front_up` und `right_front_up` wurden in 88.1% beziehungsweise 86.6% der Fälle korrekt erkannt. Bei beiden könnte dies daran liegen, dass bei einem nach vorne, in Richtung Kamera ausgestreckten Arm die Hand den Großteil des Armes verdeckt, wodurch die Pose der Modelle häufig nicht korrekt ist und dementsprechend die Klassifizierung der Geste fehlschlägt. Darüber hinaus wurden die Gesten häufig als `left_front_middle` beziehungsweise `right_front_middle` erkannt. Dies könnte darauf hinweisen, dass die ausgewählten Punktwolken, mit denen die SVMs für diese Gesten trainiert wurden, zu ähnlich waren, oder es Überschneidungen gab, die die korrekte Bestimmung einer trennenden Hyperebene erschwert haben. Abbildung 4.11 zeigt ein Beispiel, indem die Geste `right_front_up` falsch erkannt wurde.

Portierung auf Pepper

Die Geschwindigkeit des Verfahrens konnte durch die im Zuge der Portierung durchgeführten Optimierungen stark erhöht werden. Im Durchschnitt erreichte das ursprüngliche

Verfahren 69 Durchläufe pro Sekunde, das optimierte Verfahren konnte durchschnittlich 102 mal pro Sekunde ausgeführt werden. Auf Pepper erreicht das Verfahren eine durchschnittliche Ausführungsrate von 21 Durchläufen pro Sekunde. Dieses Ergebnis lässt sich durch die relativ schwache CPU von Pepper erklären, da diese auf einen geringen Stromverbrauch optimiert ist. Da Peppers Tiefenbildkamera Bilder mit einer Rate von 30 Bildern pro Sekunde aufnimmt, ist die erreichte Geschwindigkeit knapp unter dem idealen Wert. Mit einer Ausführungsrate über 30 Durchläufen pro Sekunde könnte jedes Bild, das von der Kamera empfangen wird, genutzt werden und das Verfahren würde kein Bild überspringen. Dennoch ist das Verfahren schnell genug, um eine echtzeitfähige Posenbestimmung auf Pepper zu gewährleisten.

5. Anwendungen

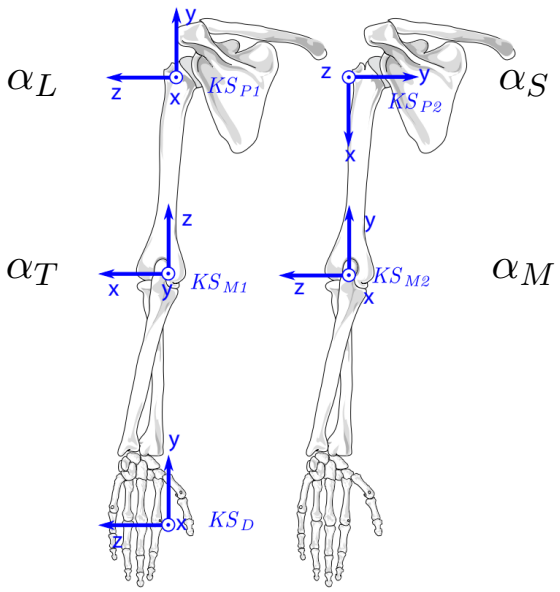
Um die Funktionalität der Posenbestimmung und Gestenerkennung auf Pepper zu demonstrieren, wurden zwei beispielhafte Anwendungen entwickelt. Bei der ersten Anwendung ahmt Pepper die Armbewegungen des Benutzers nach, bei der Zweiten reagiert Pepper auf bestimmte Gesten mit Sprachausgaben und Bewegungen.

5.1. Imitation von Armbewegungen

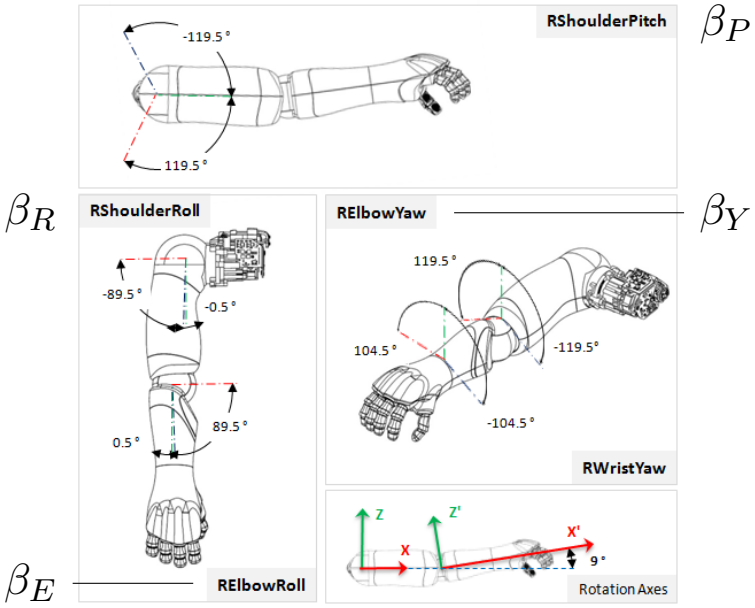
Da das kinematische Modell nicht nur die Positionen von Körpermerkmalen, sondern auch die betrachteten Gelenkwinkel berechnet, liegt es nahe, die Gelenkwinkel in einer Anwendung zu verwenden. Mit dem Programmierungswerkzeug Choregraphe wurde ein Programm erstellt, das mit einem Sprachbefehl gestartet werden kann und Pepper die Armbewegungen eines gegenüberstehenden Menschen nachahmen lässt. Zur Nachahmung führt das Programm ein Python-Skript aus, welches die von der Posenbestimmung berechneten Gelenkwinkel entgegennimmt, umrechnet und die Armgelenke von Pepper entsprechend einstellt.

Das kinematische Modell besitzt in der Schulter und dem Ellenbogen jeweils zwei Gelenke. Abbildung 5.1a zeigt die Gelenke im rechten Arm des kinematischen Modells und ihre Bewegungsachsen. Die Ausgangspose des kinematischen Modells, das heißt die Pose, in der alle Gelenkwinkel null sind, ist die T-Pose, in der die Arme zur Seite ausgestreckt sind. Das erste Schultergelenk dreht den Arm um den Winkel α_L um die Longitudinalachse des Körpers. Das zweite Schultergelenk dreht den Arm um den Winkel α_S um die Sagittalachse des Körpers. Das erste Ellenbogengelenk dreht den Unterarm um den Winkel α_T um die Transversalachse des Körpers. Anatomisch gesehen erfolgt diese Rotation im Schultergelenk, jedoch verändert sich die Position des Ellenbogens durch diese Rotation nicht, weswegen sie der Einfachheit der Modellierung halber im Ellenbogengelenk berechnet wird. Das zweite Ellenbogengelenk beugt den Unterarm um den Winkel α_M.

Pepper besitzt auch jeweils zwei Gelenke in Schulter und Ellenbogen. Abbildung 5.1b zeigt die Gelenke in Peppers rechten Arm und ihre Bewegungsachsen. In der Ausgangspose streckt Pepper beide Arme gerade nach vorne aus. Die beiden Schultergelenke drehen den Arm um die Winkel β_P und β_R um die Transversalachse beziehungsweise die Longitudinalachse des Körpers. Das erste Ellenbogengelenk dreht den Unterarm um den Winkel β_Y um die Sagittalachse des Körpers. Auch hier wurde die Bewegung, die



(a) Die Gelenkkonfiguration im rechten Arm des kinematischen Modells. Jeder Gelenkwinkel ist durch ein Koordinatensystem dargestellt. Die Rotationsachse ist immer die z-Achse [3].



(b) Die Gelenkkonfiguration im rechten Arm von Pepper [16].

Abbildung 5.1.: Die Konfiguration der Gelenke im Rechten Arm (a) des kinematischen Modells und (b) Peppers. Links und rechts der Abbildungen finden sich die in diesem Abschnitt verwendeten Variablennamen für die entsprechenden Gelenkwinkel.

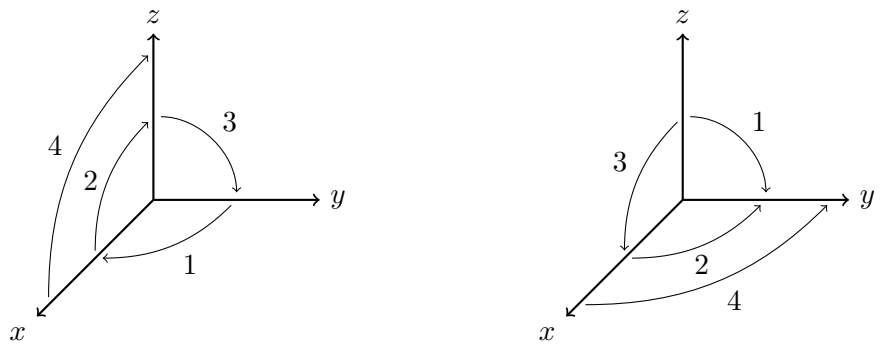


Abbildung 5.2.: Vergleich der Rotationsachsen und deren Reihenfolge zwischen (a) dem kinematischen Modell und (b) Pepper an einem Koordinatensystem. Die Rotationsachsen der Gelenke sind durch nummerierte Pfeile dargestellt. Die Zahlen geben die Reihenfolge an, in der die Gelenke miteinander verkettet sind. Da die Rotationsachsen und deren Reihenfolge sich unterscheiden, ist eine Umrechnung der Gelenkwinkel von einem Modell ins andere notwendig.

anatomisch gesehen in der Schulter geschieht, in den Ellenbogen verlegt. Das zweite Ellenbogengelenk beugt den Unterarm um den Winkel β_E.

Da die Konfiguration der Gelenke bezüglich ihrer Drehrichtung und Verkettung sich zwischen dem kinematischen Modell und Peppers Hardware unterscheidet, muss eine Umrechnung der Gelenkwinkel des kinematischen Modells in die Gelenkwinkel von Pepper stattfinden. Einzig der Winkel der Beugung des Ellenbogens ist beim kinematischen Modell und Pepper, bis auf das Vorzeichen im Fall des linken Arms, gleich. Abbildung 5.2 verdeutlicht die unterschiedliche Reihenfolge und die abweichenden Bewegungsachsen der beiden Modelle anhand eines einheitlichen Koordinatensystems.

Eine Alternative wäre, ein kinematisches Modell zu erstellen, das in den Armen die selbe Gelenkkonfiguration wie Peppers Arme aufweist. Die Erstellung eines solchen kinematischen Modells, insbesondere im Hinblick auf die nötigen Änderungen am vorhandenen Programmcode, wäre jedoch zu aufwendig gewesen, weshalb darauf verzichtet wurde.

Zur Ermittlung der Umrechnungsformeln wurden Transformationsmatrizen und homogene Koordinaten verwendet. Gleichungen 5.1 und 5.2 definieren die hier verwendeten Rotations- und Translationsmatrizen. $R_x(\alpha)$, $R_y(\alpha)$ und $R_z(\alpha)$ definieren eine Rotation um die x-, y- beziehungsweise z-Achse um den Winkel α. $T_x(d)$ und $T_y(d)$ definieren eine Verschiebung entlang der x- beziehungsweise y-Achse um die Länge d. In den initialen Posen der Modelle ist eine Verschiebung entlang der z-Achse nicht relevant. Die Positionen des Ellenbogens relativ zur Schulter und der Hand relativ zum Ellenbogen weisen keinen Versatz in der z-Achse auf. Daher wird eine Verschiebung entlang der z-Achse nicht definiert. In den folgenden Berechnungen hat d einen Wert von 1.

$$R_x(\alpha) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ R_y(\alpha) = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ R_z(\alpha) = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

(5.1)

$$T_x(d) = \begin{pmatrix} 1 & 0 & 0 & d \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},\ T_y(d) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & d \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{5.2}$$

Zunächst wurden die Transformation U_k für die ersten beiden Gelenke und die Ellenbogenposition p_k für das kinematische Modell sowie die Transformation U_p und die Ellenbogenposition p_p für Pepper bestimmt.

$$U_k = R_z(\alpha_L) \cdot R_y(\alpha_S) \cdot T_x(1)$$

$$p_k = U_k \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$$

$$U_p = R_x(\beta_P) \cdot R_z(\beta_R) \cdot T_y(1)$$

$$p_p = U_p \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T$$

Zur Bestimmung der Umrechnungsformeln wurden die Ellenbogenpositionen gleichgesetzt.

$$\begin{aligned} p_k &= p_p \\ R_z(\alpha_L) \cdot R_y(\alpha_S) \cdot T_x(1) \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T &= R_x(\beta_P) \cdot R_z(\beta_R) \cdot T_y(1) \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T \\ R_z(\alpha_L) \cdot R_y(\alpha_S) \cdot \begin{pmatrix} 1 & 0 & 0 & 1 \end{pmatrix}^T &= R_x(\beta_P) \cdot R_z(\beta_R) \cdot \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}^T \\ R_z(\alpha_L) \cdot \begin{pmatrix} \cos(\alpha_S) \\ 0 \\ -\sin(\alpha_S) \\ 1 \end{pmatrix} &= R_x(\beta_P) \cdot \begin{pmatrix} -\sin(\beta_R) \\ \cos(\beta_R) \\ 0 \\ 1 \end{pmatrix} \\ \begin{pmatrix} \cos(\alpha_L) \cdot \cos(\alpha_S) \\ \sin(\alpha_L) \cdot \cos(\alpha_S) \\ -\sin(\alpha_S) \\ 1 \end{pmatrix} &= \begin{pmatrix} -\sin(\beta_R) \\ \cos(\beta_P) \cdot \cos(\beta_R) \\ \sin(\beta_P) \cdot \cos(\beta_R) \\ 1 \end{pmatrix} \end{aligned}$$

Für den Winkel des ersten Schultergelenks ergibt sich

$$\begin{aligned}\tan(\beta_P) &= \frac{\sin(\beta_P)}{\cos(\beta_P)} \\ \beta_P &= \operatorname{atan}\left(\frac{\sin(\beta_P)}{\cos(\beta_P)}\right) \\ &= \operatorname{atan}\left(\frac{\frac{-\sin(\alpha_S)}{\cos(\beta_R)}}{\frac{\sin(\alpha_L)\cdot\cos(\alpha_S)}{\cos(\beta_R)}}\right)\end{aligned}$$

$$\beta_P = \operatorname{atan2}\left(-\sin(\alpha_S), \sin(\alpha_L)\cdot\cos(\alpha_S)\right) \tag{5.3}$$

Für den Winkel des zweiten Schultergelenks ergibt sich

$$\beta_R = \operatorname{asin}\left(-\cos(\alpha_L)\cdot\cos(\alpha_S)\right) \tag{5.4}$$

Daraufhin wurden die Transformation V_k und die Handposition q_k für das kinematische Modell sowie die Transformation V_p und die Handposition q_p für Pepper bestimmt. Zur Ermittlung einer Umrechnungsformel für das erste Ellenbogengelenk wurden die Handpositionen gleichgesetzt.

$$V_k = U_k \cdot R_x(\alpha_T) \cdot R_y(\alpha_M) \cdot T_x(1)$$

$$q_k = V_k \cdot \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix}^T$$

$$V_p = U_p \cdot R_y(\beta_Y) \cdot R_z(\beta_E) \cdot T_y(1)$$

$$q_p = V_p \cdot \begin{pmatrix}0 & 0 & 0 & 1\end{pmatrix}^T$$

$$
\begin{aligned}
q_k &= q_p \\
q_k &= U_p \cdot R_y(\beta_Y) \cdot R_z(\beta_E) \cdot T_y(1) \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T \\
U_p^{-1} \cdot q_k &= R_y(\beta_Y) \cdot R_z(\beta_E) \cdot T_y(1) \cdot \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T \\
U_p^{-1} \cdot q_k &= R_y(\beta_Y) \cdot R_z(\beta_E) \cdot \begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}^T \\
U_p^{-1} \cdot q_k &= R_y(\beta_Y) \cdot \begin{pmatrix} -\sin(\beta_E) \\ \cos(\beta_E) \\ 0 \\ 1 \end{pmatrix} \\
U_p^{-1} \cdot q_k &= \begin{pmatrix} -\sin(\beta_E) \cdot \cos(\beta_Y) \\ \cos(\beta_E) \\ \sin(\beta_E) \cdot \sin(\beta_Y) \\ 1 \end{pmatrix}
\end{aligned}
$$

Für den Winkel des ersten Ellenbogengelenks ergibt sich

$$
\begin{aligned}
\tan(\beta_Y) &= \frac{\sin(\beta_Y)}{\cos(\beta_Y)} \\
\beta_Y &= \operatorname{atan}\left(\frac{\sin(\beta_Y)}{\cos(\beta_Y)}\right) \\
&= \operatorname{atan}\left(\frac{\frac{q_3}{\sin(\beta_E)}}{\frac{-q_1}{\sin(\beta_E)}}\right) \\
\beta_Y &= \operatorname{atan2}(q_3, -q_1) \qquad (5.5)
\end{aligned}
$$

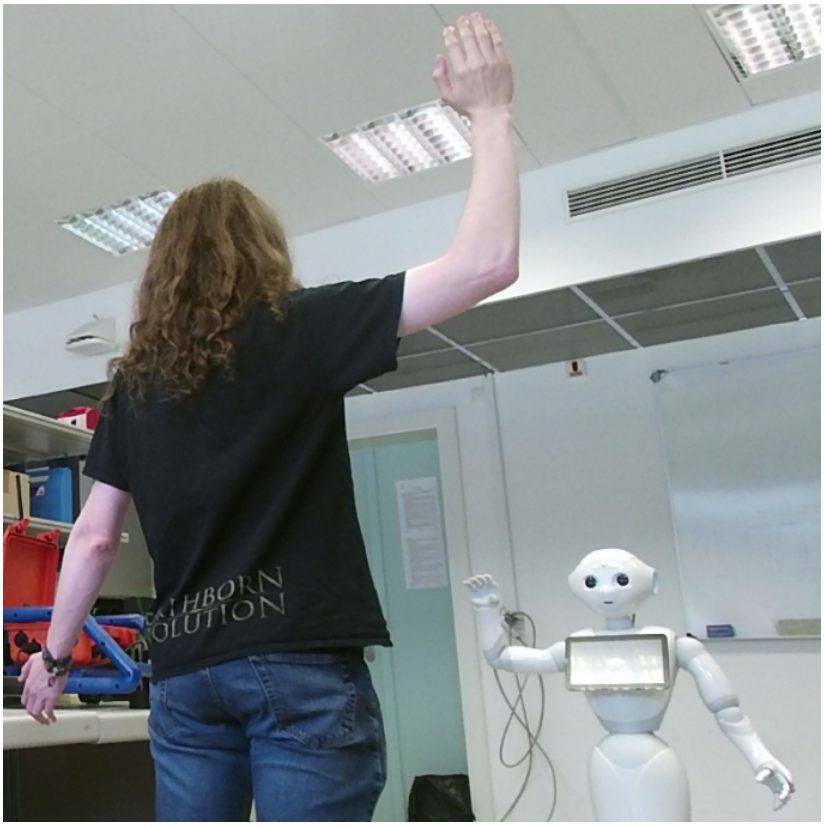
wobei q_n das n-te Element des Vektors $q = U_p^{-1} \cdot q_k$ ist.

Das Programm berechnet mit den Gleichungen 5.3, 5.4 und 5.5 aus den durch die Posenbestimmung ermittelten Gelenkwinkeln die Winkel für Peppers Arme und stellt diese dementsprechend ein.

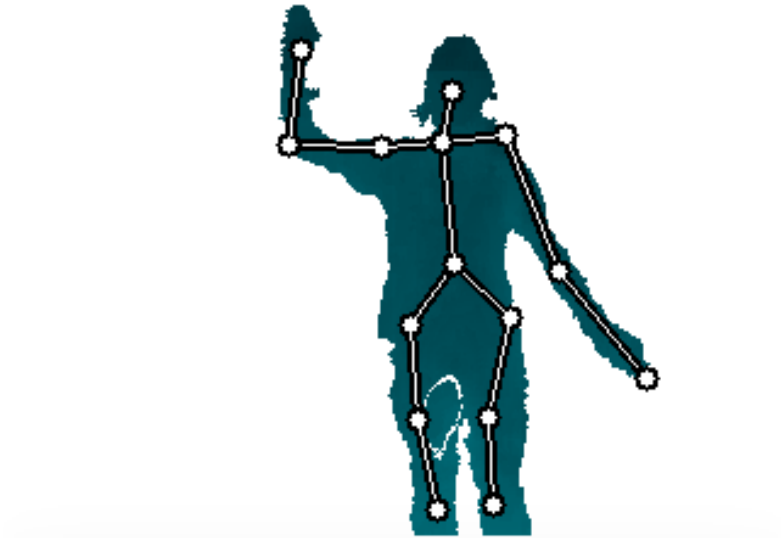
Abbildung 5.3 zeigt die Imitation an einem realen Beispiel. In Abbildung 5.3a ist ein Mensch, der eine Pose eingenommen hat, und Pepper, der diese Pose in Echtzeit nachahmt, zu sehen. Das entsprechende Tiefenbild des Menschen, in dem auch das kinematische Modell visualisiert wird, zeigt Abbildung 5.3b.

5.2. Reaktion auf Gesten

Außerdem wurde ein Python-Skript erstellt, das die erkannten Gesten entgegennimmt und bei bestimmten Gesten eine Reaktion abspielt. Hebt der Benutzer einen Arm und



(a)



(b)

Abbildung 5.3.: Reale Imitation eines Menschen durch Pepper. (a) Zeigt den Menschen, der eine Pose eingenommen hat, und Pepper, der diese Pose nachahmt. (b) zeigt die Punktwolke des Menschen mit einer Visualisierung des kinematischen Modells.

hält ihn, als würde er Pepper zuwinken, spricht Pepper ein Grußwort aus. Streckt der Benutzer den einem Arm zur Seite aus und lässt den anderen Arm hängen, dreht sich Pepper in der angezeigten Richtung einmal im Kreis.

Das Skript startet das Modul für Posenbestimmung und Gestenerkennung und fordert Benachrichtigungen über erkannte Gesten an. Wurde eine Geste erkannt, wird zunächst überprüft, ob diese einem gehobenen, angewinkelten Arm wie beim Winken entspricht. Trifft dies zu, spricht Pepper einen Gruß aus. Als nächstes wird überprüft, ob die Geste einem ausgestreckten Arm entspricht. In dem Fall wird, sofern die letzte Geste des anderen Arms einem lose herabhängenden Arm entspricht, die Posenbestimmung angehalten, der Befehl zum Drehen in die entsprechende Richtung gegeben, und abschließend die Posenbestimmung wieder aktiviert.

6. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Verfahren für die Mensch-Roboter-Interaktion für den humanoiden Roboter Pepper entwickelt, das auf der von Ehlers et al. entwickelten Posenbestimmung basiert [2]. Ziel war es, Pepper mit Körpergesten zu steuern. Zunächst wurde die Posenbestimmung im Hinblick auf Robustheit und Genauigkeit optimiert, indem eine gegenseitige Beeinflussung der zugrunde liegenden Modelle implementiert wurde. Des Weiteren wurde eine Erkennung statischer Gesten auf Basis der bestimmten Pose entwickelt. Zuletzt erfolgte einer Portierung des Verfahrens auf die Soft- und Hardwareplattform des Roboters Pepper. Aufgrund der begrenzten Leistung von Peppers Hardware wurde die Geschwindigkeit des Verfahrens optimiert.

Die Genauigkeit und Robustheit der Posenbestimmung konnten durch die gegenseitige Beeinflussung der zugrunde liegenden Modelle verbessert werden. Insbesondere bei schnellen und komplexen Bewegungen verringerte sich dadurch der Fehler der Posenbestimmung. Der Vergleich mit den Ergebnissen anderer Verfahren zeigt einen größeren Fehler, jedoch erschweren die Annotationen, die von den tatsächlichen Mittelpunkten der Körpermerkmale abweichen, den Vergleich. Bei Betrachtung der Standardabweichung des Fehlers fällt jedoch auf, dass das kinematische Modell trotz des hohen durchschnittlichen Fehlers stabil bleibt und die Punktwolke gut approximiert. Darüber hinaus verringert sich der Fehler deutlich nach Abzug eines manuell berechneten Versatzes, der aus dem Unterschied der annotierten Merkmalspositionen und der tatsächlichen Positionen hervorgeht.

Die Gestenerkennung erzielte eine hohe Genauigkeit von durchschnittlich 97.4 %. Zwei Gesten wichen jedoch in der Genauigkeit der Erkennung von den anderen Gesten ab. Bei beiden ist ein großer Teil des Arms durch die Hände verdeckt, wodurch die Auswahl von Beispielposen für das Training und die Klassifizierung erschwert werden.

Die Optimierung der Geschwindigkeit der Posenbestimmung im Rahmen der Portierung auf den Roboter Pepper war erfolgreich; sie konnte um fast 50 % verbessert werden. Trotz der schwachen Hardware von Pepper konnte das Verfahren eine Geschwindigkeit von 21 Durchläufen pro Sekunde erreichen.

Zur Demonstration der Posenbestimmung und Gestenerkennung mit Pepper wurden zwei Anwendungen entwickelt. Die erste Anwendung ist eine Imitation der Armposen des Benutzers durch Pepper auf Basis der Gelenkwinkel des kinematischen Modells. Aufgrund der unterschiedlichen Gelenkkonfigurationen des kinematischen Modells und Peppers war eine Herleitung von Umrechnungsformeln für die Gelenkwinkel notwendig. In der

zweiten Anwendung reagiert Pepper auf präsentierte Gesten mit einer Sprachausgabe oder einer Bewegung.

In Zukunft ist eine Erweiterung der Gestenerkennung auf dynamische Gesten denkbar. Dies könnte durch die Betrachtung von Folgen statischer Gesten geschehen. Eine andere Möglichkeit wäre die Untersuchung des Pfades, den ein Körpermerkmal im Raum beschreibt. So könnte ein Benutzer beispielsweise mit dem Finger eine Form in die Luft „zeichnen“, um einen Befehl zu geben. Darüber hinaus wäre eine Erweiterung auf mehr Gliedmaßen oder den ganzen Körper möglich.

A. Fehler nach Abzug des Versatzes

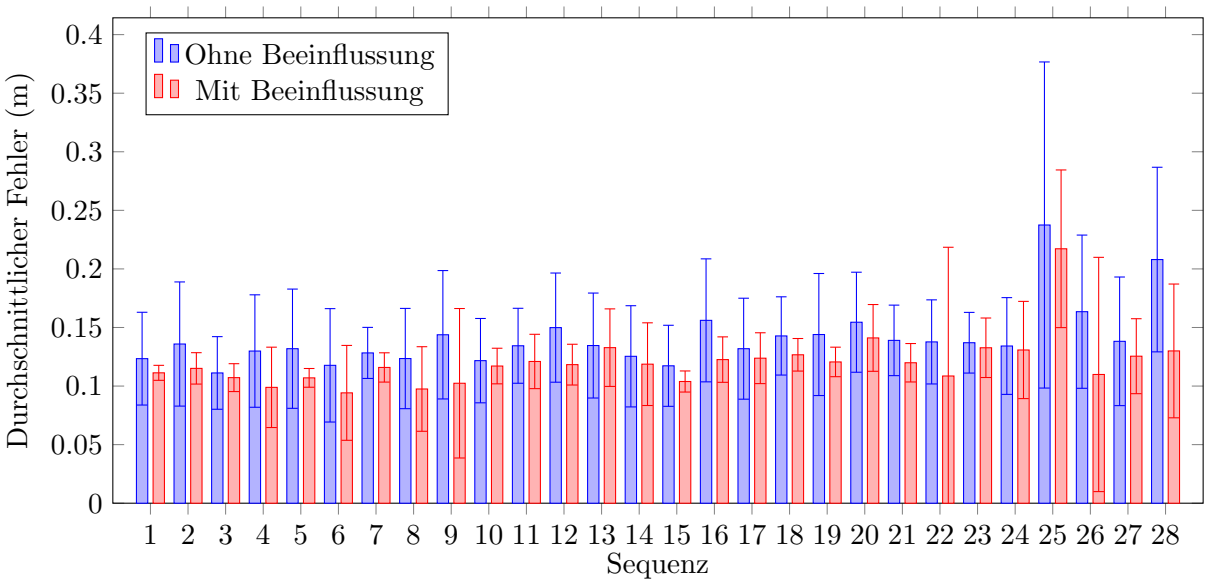


Abbildung A.1.: Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz CVPR nach Abzug des Versatzes. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt.

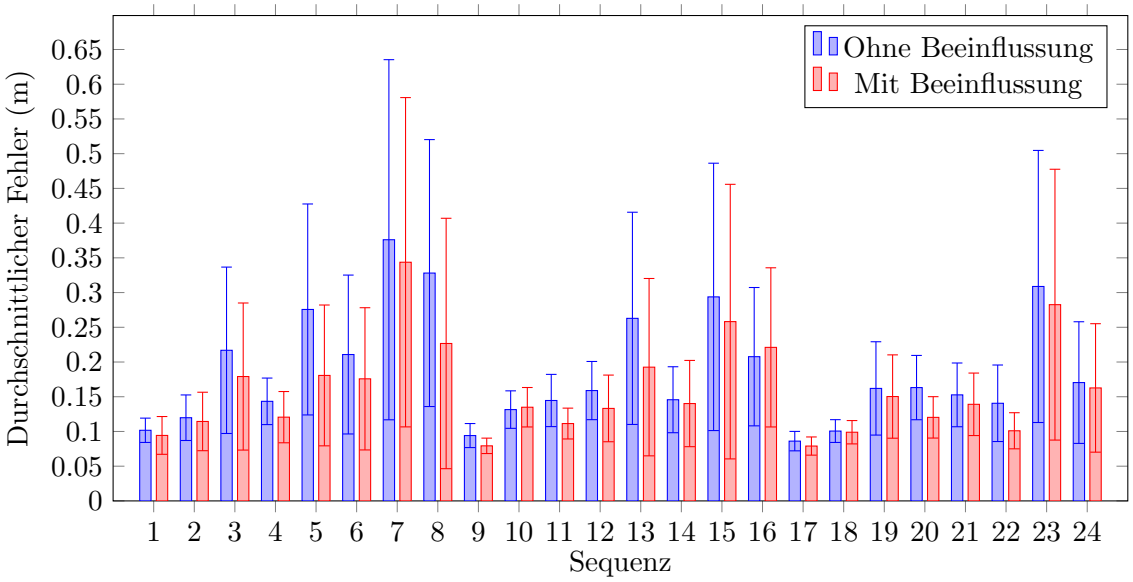


Abbildung A.2.: Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz ECCV nach Abzug des Versatzes. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt.

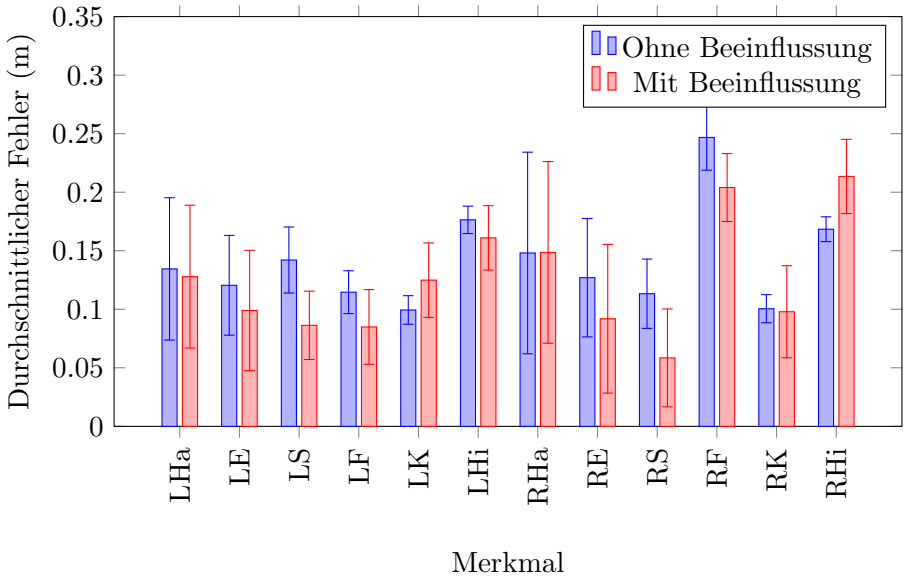


Abbildung A.3.: Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz CVPR nach Abzug des Versatzes. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt. Die Merkmale und ihre Abkürzungen finden sich in Tabelle 4.1.

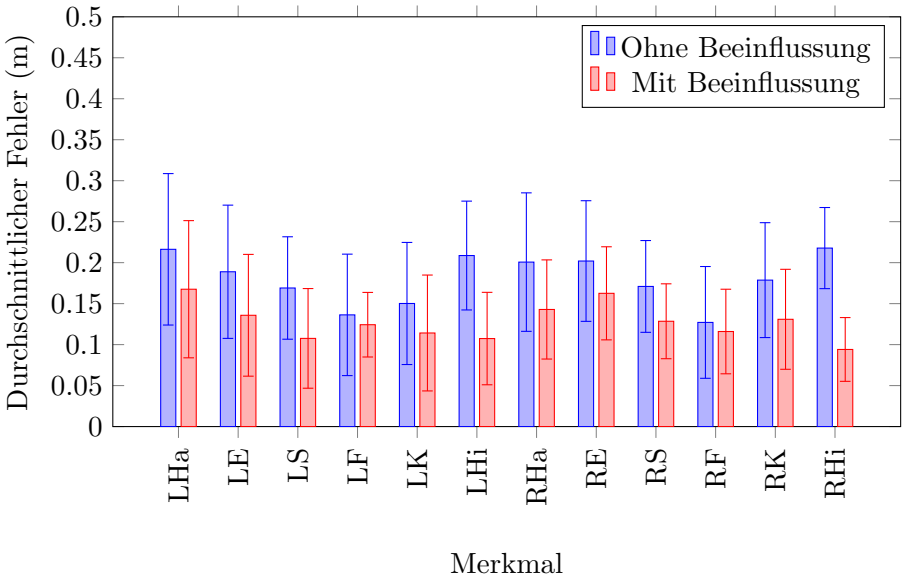


Abbildung A.4.: Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz ECCV nach Abzug des Versatzes. Es wird jeweils der Fehler ohne und mit Beeinflussung durch SSOM und GSOM sowie die Standardabweichung gezeigt. Die Merkmale und ihre Abkürzungen finden sich in Tabelle 4.1.

Abbildungsverzeichnis

2.1. Der humanoide Roboter Pepper . . . 3
2.2. Die Benutzeroberfläche von Choregraphe . . . 4
2.3. Visualisierung einer Punktwolke . . . 5
2.4. Der Box-Filter . . . 7
2.5. Topologie des kinematischen Modells . . . 8
2.6. Funktionsweise einer SOM . . . 10
2.7. Topologien der SSOM und GSOM . . . 11
2.8. Funktionsweise einer SVM . . . 12

3.1. Gegenseite Beeinflussung von SSOM, GSOM und kinematischem Modell an einem Beispiel . . . 14
3.2. Gegenseite Beeinflussung von SSOM, GSOM und kinematischem Modell . 15
3.3. Anpassung der Knochenlängen . . . 16
3.4. Anwendung der PCA . . . 18
3.5. Extraktion des Menschen aus der Szene . . . 21
3.6. Entfernung zu naher oder zu weit entfernter Punkte . . . 22
3.7. Entfernung des Bodens . . . 23
3.8. Programm zum Testen des Verfahrens . . . 24

4.1. Beispiele der Merkmalspositionen aus den Datensätzen CVPR und ECCV 27
4.2. Beispielgesten . . . 28
4.3. Werkzeug zur Annotation von Tiefenbildsequenzen . . . 29
4.4. Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz CVPR . . . 30
4.5. Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz ECCV . . . 30
4.6. Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz CVPR . . . 31
4.7. Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz ECCV . . . 32
4.8. Genauigkeit der Gestenerkennung nach Gesten . . . 33
4.9. Verarbeitungsgeschwindigkeit des optimierten Verfahrens . . . 34
4.10. Beispiele der Evaluation auf den Datensätzen CVPR und ECCV . . . 35
4.11. Fehlklassifizierung der Geste `right_front_up` . . . 36

5.1. Gelenkkonfigurationen von kinematischem Modell und Pepper . . . 40

5.2. Vergleich der Rotationsachsen und deren Reihenfolge zwischen Pepper und dem kinematischen Modell 41
5.3. Reale Imitation eines Menschen durch Pepper 45

A.1. Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz CVPR nach Abzug des Versatzes 49
A.2. Durchschnittlicher Fehler des kinematischen Modells je Sequenz im Datensatz ECCV nach Abzug des Versatzes 50
A.3. Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz CVPR nach Abzug des Versatzes 50
A.4. Durchschnittlicher Fehler des kinematischen Modells je Merkmal im Datensatz ECCV nach Abzug des Versatzes 51

Tabellenverzeichnis

4.1. Liste der betrachteten Merkmale und ihrer Abkürzungen 26

Literaturverzeichnis

[1] "SoftBank to staff cellphone store with 'pepper' robots." https://blogs.wsj.com/japanrealtime/2016/01/28/softbank-to-staff-mobile-phone-store-with-pepper-robots/. Zuletzt aufgerufen am 04.10.2017.

[2] K. Ehlers and J. H. Klüssendorff, "Self-scaling kinematic hand skeleton for real-time 3d hand-finger pose estimation," in *Proceedings of the 10th International Conference on Computer Vision Theory and Applications (VISIGRAPP 2015)*, pp. 185–196, 2015.

[3] J. Krause, "Kinematisches Modell für die echtzeitfähige 3D Posenbestimmung des menschlichen Körpers," Masterarbeit, Universität zu Lübeck, 2016. Betreut von Prof. Dr.-Ing. Erik Maehle, Unterstützt von Kristian Ehlers.

[4] "Softbank robotics: Pepper." https://www.ald.softbankrobotics.com/en/press/gallery/pepper. Zuletzt aufgerufen am 19.09.2017.

[5] "Softbank increases its interest in aldebaran to 95%." https://www.ald.softbankrobotics.com/en/press/press-releases/softbank-increases-its-interest. Zuletzt aufgerufen am 26.09.2017.

[6] "Aldebaran becomes softbank robotics." https://www.ald.softbankrobotics.com/en/press/press-releases/aldebaran-becomes-softbank-robotics. Zuletzt aufgerufen am 26.09.2017.

[7] "Find out more about pepper." https://www.ald.softbankrobotics.com/en/robots/pepper/find-out-more-about-pepper. Zuletzt aufgerufen am 26.09.2017.

[8] J. Denavit and R. S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Trans. ASME E, Journal of Applied Mechanics*, vol. 22, pp. 215–221, Jun 1955.

[9] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, pp. 1464–1480, Sep 1990.

[10] J. Petzold, "Kombination zweier Verfahren für die echtzeitfähige 3D Posenbestimmung der Hand," Bachelorarbeit, Universität zu Lübeck, 2016. Betreut von Prof. Dr.-Ing. Erik Maehle, Unterstützt von Kristian Ehlers.

[11] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, pp. 27:1–27:27, May 2011.

[12] "ROS.org — Powering the world's robots." `http://www.ros.org/`. Zuletzt aufgerufen am 17.10.2017.

[13] "PCL - Point Cloud Library (PCL)." `http://pointclouds.org/`. Zuletzt aufgerufen am 17.10.2017.

[14] V. Ganapathi, C. Plagemann, S. Thrun, and D. Koller, "Real time motion capture using a single time-of-flight camera," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 755–762, Jun 2010.

[15] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun, "Real-time human pose tracking from range data," in *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pp. 738–751, Springer-Verlag, 2012.

[16] "Joints – Aldebaran 2.5.7.1 documentation." `http://doc.aldebaran.com/2-5/family/pepper_technical/joints_pep.html#right-arm-joints-and-actuators`. Zuletzt aufgerufen am 13.10.2017.