CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung für Assistenzsysteme und Medizintechnik

Studiengang
Bachelorstudiengang Informatik

Bachelorarbeit

Bewegungsnachverfolgung in 3D-Punktwolken am Beispiel von verdeckten Händen

vorgelegt von
Meret Björk Lindanis

11. Januar 2021

1. Gutachter: M.Sc. Christian Kowalski
2. Gutachter: Prof. Dr.-Ing. Andreas Hein

Abstract

Robot systems, such as surgical robots, support healthcare professionals in their (specific) activities and can potentially support or take over care activities, especially physically demanding tasks such as patient rearrangements. In order to be able to recreate the movements of the nursing staff with robots, they must first be recorded digitally. Human Motion Capture Systems are used for this purpose. In order to ensure the most authentic execution of the motion sequences, an optical, markerless system was chosen for the collection of the position data in this work and the motion sequences were recorded as a video sequence with the Microsoft® Azure Kinect.

The hands and wrists are an important instrument for interaction in healthcare and are therefore the primary goal in tracking nursing movements. Due to the usually close physical contact between nurses and patients during the interaction, the hands are often not visible for optical tracking in the video sequences. Therefore they cannot be easily recognized and tracked in image processing. To address this problem, the presented work develops software for an annotation tool with Open3D in which the spatial positions of the wrists are manually estimated and annotated in generated point clouds by the users. Due to the sequential annotation the wrists of different persons remain distinguishable, so that any number of wrists can be annotated in the same data set. The data generated from the annotation tool in this way can be made available in form of trajectories for further use in the development and programming of care robots.

An evaluation of the manual annotation method was carried out in form of a comparison with data from two established position estimation methods. For this purpose the Microsoft® Azure Kinect Body Tracking SDK and the Captiv Motion Capture suit from TEA were selected. In both comparisons the annotation tool produced very similar trajectories, for both visible and occluded wrists.

Zusammenfassung

Robotersysteme wie beispielsweise Operationsroboter unterstützen medizinisches Fachpersonal bei ihren (spezifischen) Tätigkeiten und können potenziell auch pflegerische Tätigkeiten, insbesondere körperlich schwere Arbeiten wie Patientenumlagerungen, unterstützen oder gänzlich übernehmen. Um die Bewegungen des Pflegepersonals mit Robotern nachbilden zu können, müssen diese zunächst digital erfasst werden. Hierfür werden Human Motion Capture Systeme eingesetzt. Um eine möglichst authentische Ausführung der Bewegungsabläufe zu gewährleisten, wurde für die Erhebung der Positionsdaten in dieser Arbeit ein optisches, markerloses System gewählt und die Bewegungsabläufe mit der Microsoft® Azure Kinect als Videosequenz aufgenommen.

In der Pflege sind die Hände und Handgelenke ein wichtiges Instrument zur Interaktion und sind daher das vorrangige Ziel in der Nachverfolgung der pflegerischen Bewegungsabläufe. Wegen des meist engen körperlichen Kontakts zwischen Pflegekräften und Patientinnen oder Patienten während der Interaktion sind die Hände für eine optische Nachverfolgung in den Videosequenzen häufig nicht sichtbar, sodass sie in der Bildverarbeitung nicht ohne weiteres erkannt und nachverfolgt werden können. Um diesem Problem zu begegnen, wird in der vorgestellten Arbeit mit Open3D eine Software für ein Annotationstool entwickelt, bei dem in generierten Punktwolken die räumlichen Positionen der Handgelenke manuell durch die Nutzerinnen oder Nutzer geschätzt und annotiert werden. Aufgrund der sequenziell erfolgten Annotation bleiben die Handgelenke verschiedener Personen unterscheidbar, sodass beliebig viele Handgelenke im gleichen Datensatz markiert werden können. Die so erzeugten Daten aus dem Annotationstool können in Form von Trajektorien zur weiteren Nutzung bei der Entwicklung und Programmierung von Pflegerobotern zur Verfügung gestellt werden.

Eine Evaluierung der manuellen Annotationsmethode wurde in Form eines Vergleichs mit Daten zweier etablierter Verfahren zur Positionsschätzung durchgeführt. Hierfür wurden das Microsoft® Azure Kinect Body Tracking SDK sowie der Captiv Motion-Capture-Anzug von TEA ausgewählt. Das Annotationstool erzeugte in beiden Vergleichen weitgehend ähnliche Trajektorien, sowohl für sichtbare als auch für verdeckte Handgelenke.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

List of Algorithms

1 Einleitung

In allen Bereichen der Medizin ist durch den Einsatz unterschiedlichster Technologien ein Fortschritt hinsichtlich der Aspekte Patientenbetreuung zu verzeichnen (vgl. [18]). In Krankenhäusern und Kliniken werden Geräte zur Überwachung der Vitalparameter der Patientinnen und Patienten, Roboter zur Unterstützung während einer Operation oder auch Klinische Entscheidungsunterstützungssysteme[1] eingesetzt, um nur einige Beispiele zu nennen. Dies zeigt, dass medizinische Überwachungssysteme und elekronische Hilfsmittel im klinischen Alltag bereits *State-of-the-Art* sind. Im Pflegebereich gibt es hingegen bisher nur wenige, technisch ausgereifte Hilfsmittel, die das Pflegepersonal bei der Verrichtung ihrer körperlich schweren Tätigkeiten entlasten können (vgl. [17], S. 23).

1.1 Motivation

In der Gesundheits- und Krankenpflege wie auch in der Altenpflege sind Rückenschmerzen eine häufige Folge der körperlich schweren Tätigkeiten, die beispielsweise die Umlagerung von Patientinnen oder Patienten vorsehen. Diese Rückenschmerzen führen zu einer hohen Anzahl an Arbeitsunfähigkeitstagen (AU-Tage) der Beschäftigten in den Pflegeberufen, wie der Grafik in Abbildung 1.1 zu entnehmen ist. In dieser sind die fehlzeitenrelevanten Diagnosen über den AU-Tagen je 100 VJ (Versicherungsjahre) aufgetragen. Mit insgesamt 201 bzw. 279 AU-Tagen für Beschäftige in Kranken- und Altenpflegeberufen sind die Diagnosen *Rückenschmerzen*, *sonstige Bandscheibenschäden* und *Schulterläsionen* die Hauptursachen für AU-Tage und führen somit häufiger zu Fehlzeiten als andere Diagnosen. Bei den sonstigen Berufen sind diese Diagnosen mit 116 AU-Tagen je 100 VJ deutlich geringer.

[1] engl. *Clinical Decision Support Systems* kurz CDSS

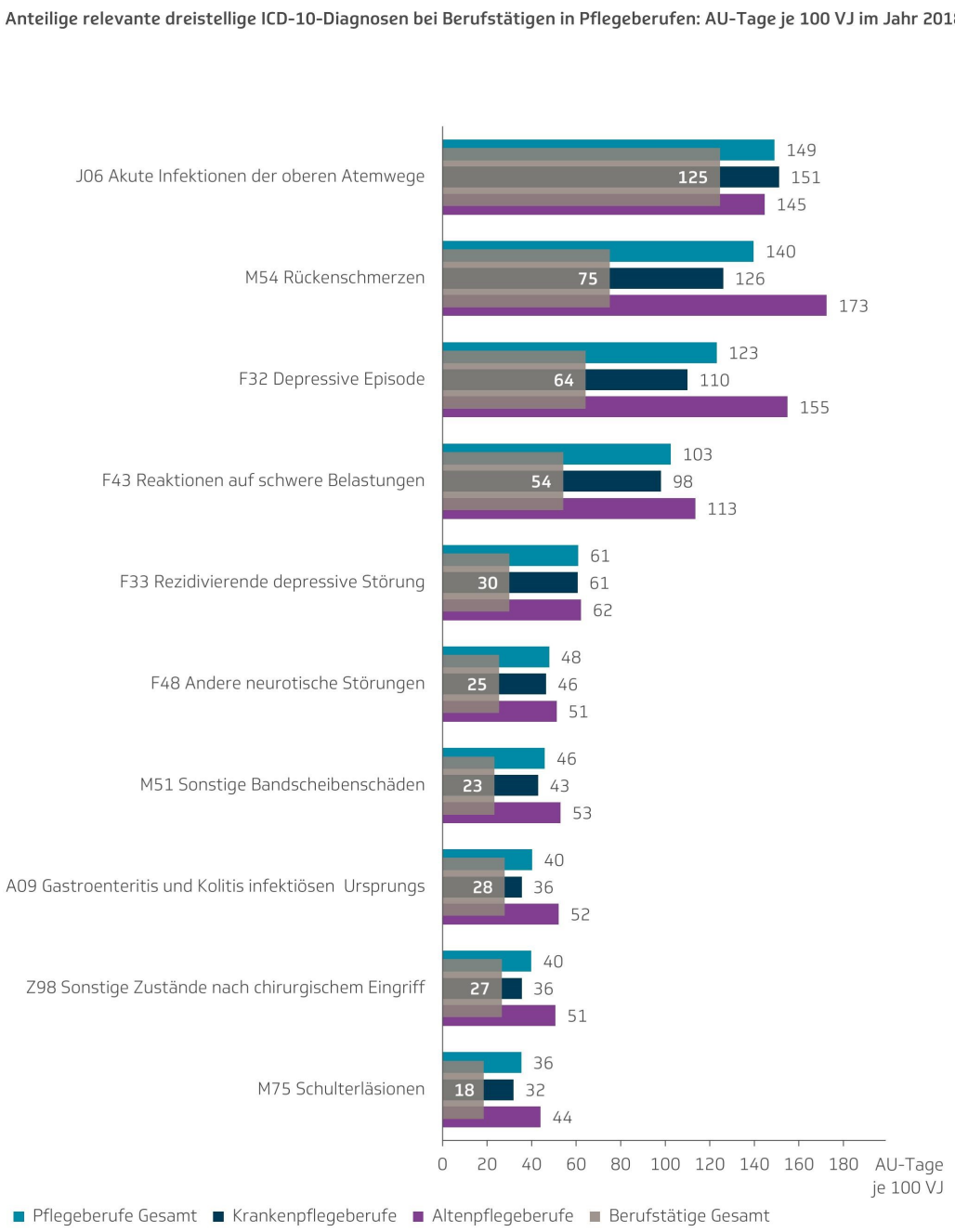

Abb. 1.1: Übersicht der TK über fehlzeitenrelevante Diagnosen in Pflegeberufen [Quelle: [41], S. 38]

Um das Pflegepersonal sowie die Ärztinnen und Ärzte zu unterstützen und zu schützen, sollte für die Arbeit in der Medizin immer bessere Technik eingesetzt werden. Dies geschieht zum einen durch Software, die in medizinischen Einrichtungen benutzt werden kann, und zum anderen durch Roboter, die klinische und pflegerische Tätigkeiten unterstützen oder solche Arbeiten gänzlich übernehmen

können. Bei Robotern ist es essentiell, dass diese bestimmte Arbeitsschritte lernen und anschließend autark ausführen können. Dabei sollen die Roboter die physische Interaktion mit den Patientinnen und Patienten genauso ausführen wie das medizinische Fachpersonal. Ein Ansatz hierfür wäre, dessen spezifische Bewegungsmuster und Handgriffe, beispielsweise bei der Umlagerung einer Patientin, genau zu beobachten und zu analysieren. Das kann z.B. mit einer Tiefenkamera geschehen. Dabei wird die visuell in Form von Bildern aufgenommene Information in digitale Daten, sogenannte Punktwolken, überführt. Stehen komplette Bewegungsabläufe dann in digitaler Form zur Verfügung, können die Daten genutzt werden, um darin beispielsweise die Hände und Handgelenke des Fachpersonals, als wichtige Interaktionswerkzeuge in der Pflege, nachzuverfolgen. Die in diesem Prozess erzeugten Daten bilden den zeitlichen und räumlichen Verlauf der Handposition ab und lassen sich für den Lernprozess von Robotern einsetzen.

1.2 Problemstellung und Zielsetzung

Sollen pflegerische Bewegungsabläufe mit Hilfe visueller Methoden für eine spätere Digitalisierung analysiert werden, stellt sich das Problem, dass saliente Körperteile interagierender Personen, wie die Hände des Pflegepersonals, oftmals im engen körperlichen Kontakt verdeckt und dem Blick der Kamera entzogen sind.

Abb. 1.2: Beispiel verdeckter Hände [Quelle: eigene Aufnahme]

Damit ist eine vollständige visuelle Erfassung des Bewegungsablaufes oftmals nicht möglich. Im Rahmen dieser Arbeit soll daher ein Werkzeug entwickelt werden, mit dem auch verdeckte Körperteile nachverfolgt werden können. Da im

medizinischen Kontext die Hände wichtige pflegerische Instrumente sind, liegt der Schwerpunkt der Softwareentwicklung dieser Arbeit auf der Nachverfolgung verschiedener Positionen und Bewegungsmuster der Hände.

1.3 Aufbau der Arbeit

Im Zuge der zu Beginn durchgeführten Literaturrecherche (Kapitel 2.1) wurden die Grundlagen der Bewegungsnachverfolgung sowie bereits verfügbare technische Lösungen untersucht. Diese werden in den Kapiteln 2.2.1 und 2.2.2 erläutert. Vor diesem Hintergrund wurden die Anforderungen an das zu entwickelnde Annotationstool formuliert.
Bei der Entwicklung des Annotationstools wurden verschiedene Geräte und Bibliotheken genutzt, die in Kapitel 3.1 vorgestellt werden. Das Kapitel 3.2 widmet sich dem methodischen Vorgehen im Rahmen der Implementierung, die in Kapitel 3.3 ausführlich beschrieben wird. Schwierigkeiten und Grenzen bei der Umsetzung werden abschließend in Kapitel 3.4 diskutiert. In einem weiteren Schwerpunkt wird das Annotationstool anhand der generierten Daten ausführlich gegenüber bereits etablierten Verfahren evaluiert (siehe Kapitel 4). Die Arbeit schließt in Kapitel 5 mit einer Reflexion des Entwicklungsprozesses und des Ergebnisses und zeigt Möglichkeiten für Erweiterungen und Verbesserungen auf.

2 Grundlagen

In diesem Kapitel werden zunächst einige im Bezug zur Problemstellung stehende Publikationen vorgestellt, um einen Überblick über den aktuellen Stand der Forschung zu geben. Darauf aufbauend werden in Kapitel 2.2 die theoretischen Grundlagen bezüglich Bewegungsnachverfolgung, Trajektorien und Interpolation erläutert. Die Anforderungen an das Annotationstool werden abschließend in Kapitel 2.3 formuliert.

2.1 Verwandte Arbeiten

In der vorangestellten Literaturrecherche zum Problem der Bewegungsnachverfolgung von verdeckten Händen sind folgende Publikationen als relevant eingestuft worden und werden kurz vorgestellt.

Die Publikation von István Sárándi et al. [36] beschreibt eine Studie, in der sie *State-of-the-Art*-Methoden zur Positionsschätzung hinsichtlich ihrer Robustheit gegenüber Verdeckungen untersuchten. In dieser Studie werden jedoch nur Verdeckungen von bis zu 70% innerhalb einer Bounding-Box um den Menschen betrachtet, was im Nutzungskontext des Annotationstools nicht ausreichend ist, da die Hände oft zu 100% verdeckt sind und keine Bounding-Box verwendet wird. Die Leistung der untersuchten Ansätze verschlechtert sich stark hinsichtlich der Positionsschätzung, wenn sich der Grad der Verdeckung erhöht. In den untersuchten Ansätzen wird versucht, in den gegebenen Bildern die Haltung der Menschen zu schätzen, d.h. ob der Mensch geht oder sitzt, jemanden grüßt oder telefoniert. Das heißt, es wird die Position des gesamten Körpers geschätzt und nicht nur die eines Gelenkes (vgl. [36]).

Mao Ye et al. [49] stellen in ihrer Arbeit eine Methode zur Positionsschätzung des gesamten Körpers am Beispiel von Haltungen im Tennis dar. Hierfür wurde eine Bewegungsdatenbank aufgebaut, in der Informationen über die Positionen von Gelenken und Skelettelementen von Menschen bei verschiedenen Körperhal-

tungen in Punktwolken hinterlegt sind (vgl. [49]).
Punktwolken entstehen durch das laserbasierte Abtasten einer dreidimensionalen Szene, bei der Millionen Einzelmessungen erfolgen. Sie stellen somit eine Art Digitalisierung der Realität dar, bei der die räumliche Position jedes Messpunktes der Einzelmessungen als dreidimensionale Koordinate codiert, gespeichert und visualisiert wird (siehe Abb. 2.1) (vgl. [42]).

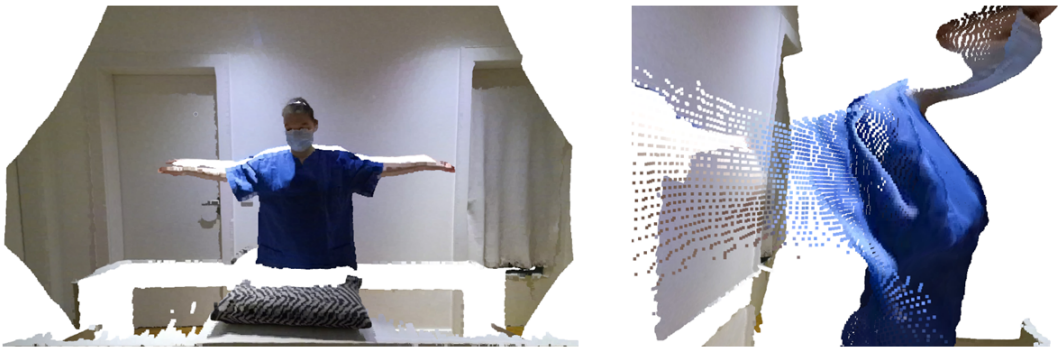

Abb. 2.1: Punktwolken [Quelle: eigene Aufnahme]: a) die Punktwolke in der Frontalansicht, b) die Punktwolke, in der die einzelnen Messpunkte zu sehen sind, in der Seitenansicht

Bei Mao Ye et al. [49] werden die Punktwolken, die Informationen über eine Körperhaltung enthalten, anschließend in die Datenbank eingegeben und mit den darin hinterlegten Punktwolken verglichen. Die Punktwolke, die der eingegebenen Punktwolke am ähnlichsten ist, wird gewählt. Anschließen können die Daten über die Gelenks- und Skelettpositionen in dieser Körperhaltung aus der Punktwolke der Datenbank extrahiert und in die zu markierende Punktwolke übertragen werden, sodass dort die Gelenke und die Körperhaltung ebenfalls markiert sind (vgl. [49]).
Bei diesem Ansatz werden allerdings nur Einzelbilder genutzt, keine Sequenz von Punktwolken, die einen Bewegungsablauf abbilden kann. Inwieweit eine so große Anzahl von Bilder, wie sie bei einer Videosequenz entsteht, bei diesem Verfahren analysiert werden kann, wird nicht erwähnt (vgl. [49]).
Im pflegerischen Kontext sind zunächst die Hände wichtig, die bei der Interaktion häufig vollständig verdeckt sind. Dieses Problem der Verdeckung von Gelenken wird nicht behandelt. Es kann nur die Körperhaltung *einer* Person geschätzt werden (vgl. [49]).

Im medizinischen Kontext ist die Analyse der Gelenkpositionen von zwei oder mehr Personen wichtig, da einige Interaktionen mit der Patientin oder dem Patienten nur mit zwei oder mehr Pflegekräften ausgeführt werden können. Der von Mao Ye et al. [49] vorgestellte Ansatz lässt dies jedoch nicht direkt zu.
Mao Ye et al. [49] implementieren in ihrem Ansatz eine Vorverarbeitung des Bildes, bei der die Punktwolke am Ende ausschließlich den Körper der Person enthält. Da bei verdeckten Gelenken allerdings auch die Umgebung wichtige Informationen enthalten kann, die die Bestimmung der Handgelenksposition erleichtern, ist dieser Ansatz hier nicht sinnvoll (vgl. [49]).

Einen ähnlichen Ansatz verfolgen auch Javier Romero et al. [34], jedoch basiert ihre Datenbank auf Farbbildern mit 10^5 Bildern und nicht auf Punktwolken. Der Fokus der Bilder liegt auf einer einzelnen Hand, d.h. es sind kaum zusätzliche Informationen enthalten und es wird immer nur eine Hand betrachtet. Ziel von Javier Romero et al. ist es, die Handhaltung zu schätzen, wobei die Hand hier auch mit anderen Objekten interagieren kann. Die Handhaltung schätzen sie über die Segementierung der Hand anhand der Hautfarben im HSV-Farbraum[1] und einer *Nearest-Neighbour*-Suche in der Datenbank. Im Rahmen dieser Bachelorarbeit wäre es allerdings zu aufwändig, eine solche Datenbank mit Bildern aus dem pflegerischen Kontext aufzubauen. Ein weiteres Problem ist, dass nur die Hand aufgenommen wird und das Objekt, mit dem interagiert wird. Dabei verdeckt dieses die Hand nie vollständig. Darüberhinaus fehlen die zusätzlichen Informationen durch die Umgebung, die auf die Lage des Handgelenks hindeuten können, wenn dieser Ansatz auf die Problemstellung dieser Arbeit angewendet würde (vgl. [34]).

In ihrer Publikation beschreiben Srinath Sridhar et al. [35] eine Methode, um die Interaktion einer Hand mit einem Objekt nachzuverfolgen. Dies ist im Kontext der Pflege interessant, da dort die Hände der Pflegekraft ebenfalls mit der Patientin oder dem Patienten interagieren. Bei diesem Ansatz werden in jeder Punktwolke einer Punktwolkensequenz die Hand und das Objekt klassifiziert, d.h. die Hand wird als Hand und das Objekt als Objekt erkannt und farblich voneinander abgegrenzt. Anschließend werden sie einem Clustering unterzogen, bei dem jeweils die Punkte der Klassifikation in Gruppen zusammengefasst werden, die den gleichen Teil des Bildes klassifizieren. Mit Optimierungsverfahren wird anschließend in der Punktwolkensequenz die Interaktion zwischen Hand und Objekt verfolgt und die

[1] Im HSV-Farbraum wird die Farbe über den Farbwert (engl. *hue*), die Sättigung der Farbe (engl. *saturation*) und die Dunkelstufe (engl. *value*) definiert (vgl. [38], S. 84).

Haltung der Hand geschätzt und eingezeichnet. Das Problem dieses Ansatzes hinsichtlich der Anwendbarkeit auf das Annotationstool ist zum einen, dass nur eine Hand nachverfolgt werden kann, was im pflegerischen Kontext nicht ausreichend ist, da dort immer mindestens zwei Hände genutzt werden. Zum anderen ist die Hand kaum verdeckt, sodass sie noch gut zu erkennen ist und das Problem der Verdeckung nicht vollumfänglich behandelt und gelöst wird. Weiterhin wird nur die Hand mit dem Objekt aufgenommen, nicht der gesamte Körper gemeinsam mit dem des Patienten oder der Patientin. Dabei würde die Hand dann nur einen kleinen Teil des Bildes ausmachen und wäre nicht mehr gut erkennbar (vgl. [35]).

Weitere Veröffentlichungen, die sich mit der Nachverfolgung von Händen auseinandersetzen sind beispielsweise [22], [10], [25] und [15]. Die letzten beiden Publikationen behandeln auch Verdeckungen der Hand, allerdings ist die Hand niemals ganz verdeckt und sie steht auch wieder im Fokus. Denn die meisten der gerade genannten Veröffentlichungen erarbeiten Ansätze zur Gestenerkennung.

Um die Verdeckungen zu minimieren, vor allem hinsichtlich der Selbstverdeckungen durch die Kameraperspektive, beschreiben Licong Zhong et al. [50] einen Ansatz mit zwei gegenüberliegenden Tiefenkameras. Sie vereinen anschließend die jeweiligen Punktwolken der beiden Kameras zu einer Punkwolke. In der so entstandenen Punktwolke existieren weniger Verdeckungen, allerdings kann dieser Ansatz die Verdeckungen der Hände durch die Interaktion mit einem Patienten oder einer Patientin nicht beheben. Der aufwändigere Aufbau und die aufwändigere Vorverarbeitung dieses Ansatzes steht somit nicht im Verhältnis zum Ziel des Annotationstools (vgl. [50]).

Zwei weitere Publikationen [16] und [19] betrachten die Nachverfolgung oder Erkennung der gesamten Körperhaltung. Frederik Hegger et al. [16] betrachten allerdings die Erkennung von Personen, um Kollisionen von Service-Robotern mit Personen zu vermeiden. Sie behandeln auch Verdeckungen der Personen von bis zu 70 % durch andere Gegenstände, allerdings nur hinsichtlich der Erkennung, ob es sich dabei um einen Menschen handelt oder nicht. Dem gegenüber verfolgen Nicolas Lehment et al. [19] den Ansatz, den Körper zu erkennen, die Körperhaltung zu schätzen und nachzuverfolgen. Mit ihrem Ansatz können sowohl Körperhaltungen des gesamten Körpers als auch nur die des oberen Körpers geschätzt werden. Allerdings behandeln sie bis auf das Schätzen des oberen Körpers keine Verdeckungen (vgl. [19]).

2.2 Theorie

In diesem Kapitel werden zunächst verschiedene Nachverfolgungssysteme vorgestellt, die in dieser Arbeit eingesetzt werden könnten. Anschließend wird das Konzept der Trajektorie erläutert, da dieses in der Evaluierung des Annotationstools in Kapitel 4 verwendet wird.

2.2.1 Nachverfolgung von Bewegungen

Um Bewegungen im Speziellen von Menschen nachverfolgen zu können, gibt es das Verfahren des *Human Motion Capture* (HMC), welches vor allem eingesetzt wird, um in Computerspielen oder auch in Filmen Charaktere mit menschlichen Bewegungen zu animieren. Dabei kann auf verschiedene Systeme zurückgegriffen werden. Diese können aufgrund ihrer verschiedenen Technologien zum Aufzeichnen der Daten in *optische* und *nicht-optische* Systeme unterteilt werden. Zu den nicht-optischen Systemen gehören die elektromagnetischen, die elektromechanischen, die akustischen und die inertialen Systeme. Optische Systeme lassen sich in zwei weitere Systeme unterteilen: markerbasierte und markerlose Systeme (vgl. [43], S. 1) (siehe Abb. 2.2). All diese Systeme werden im Folgenden kurz anhand ihrer Funktionsweise und ihrer Vor- und Nachteile bezüglich ihres Einsatzes im pflegerischen Kontext vorgestellt.

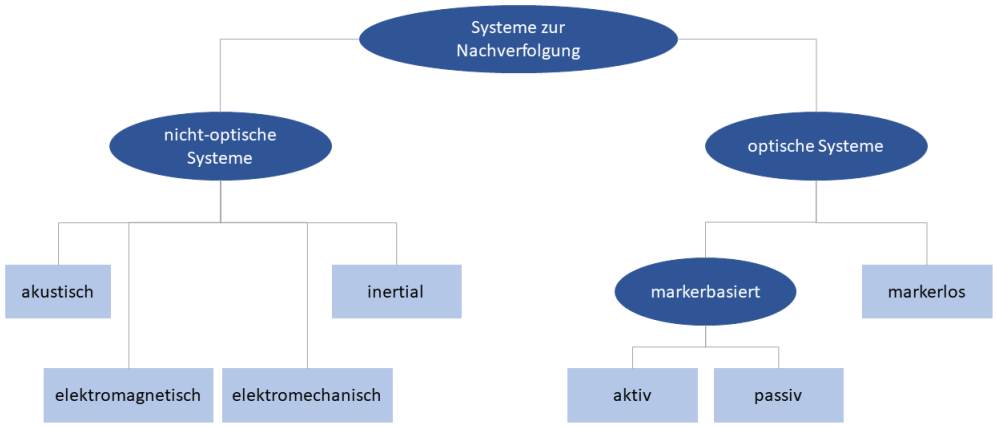

Abb. 2.2: Systeme zur Nachverfolgung

Elektromagnetische Systeme

Hier wird durch einen Sender ein elektromagnetisches Feld niedriger Frequenz erzeugt. Durch die Reichweite, in der diese Frequenz noch gemessen werden kann, wird der Messbereich definiert. Die beobachtete Person trägt Sensoren, die aufgrund der Bewegung der Person relativ zum fest platzierten Sender die Änderungen des Magnetfeldes messen können. Darüber hinaus ist an der Person eine elektronische Kontrolleinheit befestigt, mit der jeder Sensor über ein Kabel verbunden ist. Die Kontrolleinheit filtert und verstärkt die Daten der Sensoren in geeigneter Weise, sodass mit Hilfe einer Software aus diesen Daten die räumlichen Positionen der einzelnen Sensoren rekonstruiert werden können. Da diese Messsysteme auf einem elektromagnetischen Feld basieren, dürfen sich in der unmittelbaren Umgebung keine Metalle oder stromführenden Geräte befinden, da diese durch Induktion andere Magnetfelder erzeugen, die das Magnetfeld des Senders verändern und die Messergebnisse verfälschen können. Daher sollte der Messvorgang in einer metallfreien Umgebung stattfinden. Auch die Distanz der Person zum Sender hat einen Einfluss auf die Genauigkeit der Positionsermittlung, da das Magnetfeld schwächer wird, je weiter sich die Sensoren von dem Sender entfernen. Einen weiteren Nachteil stellen die Kabel dar, die die Bewegungsfreiheit der agierenden Probanden einschränken und in ungünstigen Fällen, beispielsweise beim Umlagern eines Patienten, auch aus den Anschlüssen gerissen werden können. Der Vorteil dieses Messsystems und auch aller anderen nicht-optischen Systeme ist, dass auch verdeckte Gelenke nachverfolgt werden können, da die Messung nicht auf der Sichtbarkeit basiert. Außerdem können beim elektromagnetischen System die Gelenke mehrerer Personen gleichzeitig nachverfolgt werden (vgl. [43], S. 3 ff.).

Elektromechanische Systeme

Die HMC Systeme, die den elektromechanischen Ansatz wählen, nutzen ein Exoskelett zur Messung der Bewegungen. Das heißt, die beobachtete Person zieht eine Art Anzug an, bei dem die Gelenke durch Stäbe, ähnlich den Knochen, miteinander verbunden sind. Darüber können dann die Winkel der einzelnen Glieder zueinander anhand von Potentiometern gemessen werden. Die Bewegungen der Person werden somit über die Winkeländerung über der Zeit repräsentiert. Da bei diesem System nur die Winkel der Glieder zueinander gemessen werden, können die Positionen und die Orientierungen der Gelenke im Raum nicht ohne andere zusätzliche Messsysteme erfasst werden. Ein weiterer offensichtlicher Nachteil

dieses Systems ist das Exoskelett, welches die tragende Person in ihren Bewegungen und in der Interaktion mit anderen Personen wie den Patienten und Patientinnen einschränkt. Bei der Interaktion mit Patientinnen und Patienten kann es zum Verrutschen einzelner Elemente des Exoskeletts kommen, was zu Messfehlern führen kann, da sich dadurch die Winkel zwischen den Gliedern des Exoskeletts ändern. Das Exoskelett muss auch in der Größe und Form an die Personen angepasst werden können, die es tragen sollen. Auch wenn das Exoskelett an die Maße der Person angepasst wurde, besteht ein Abstand zwischen den Gelenken im Exoskelett und den tatsächlichen Gelenken. Dies kann eine weitere Quelle für Ungenauigkeiten in der Positionsbestimmung der Gelenke sein (vgl. [43]), S. 2 f.). Der Vorteil dieses Messsystems ist die Unabhängigkeit von den Lichtverhältnissen während der Messung. Solange keine elektromagnetischen Sensoren zur Messung der globalen Position und Orientierung eingesetzt werden, ist das System auch nicht empfindlich gegenüber weiteren äußeren Einflüssen wie Magnetfeldern. Dadurch dass es keinen Sender oder Empfänger gibt, ist die Messung auch nicht an einen Ort gebunden und es können mehrere Personen gleichzeitig hinsichtlich der Winkel zwischen den Gelenken nachverfolgt werden (vgl. [43], S. 3).

Akustische Systeme

Bei diesen Systemen werden die Positionen und Bewegungen der Person anhand von Ultraschallwellen über Sender und Sensoren gemessen. Das bedeutet, der Sender generiert die Ultraschallwellen beispielsweise mit einem Hochfrequenzlautsprecher und sendet sie aus. Diese werden anschließend vom Sensor z.B. über Mikrophone, die derartige Wellen erkennen können, empfangen. Die Positionen und Bewegungen können anschließend auf zwei Wegen rekonstruiert werden (vgl. [43]): zum einen über das Laufzeitverfahren (engl. *Time-of-Flight* kurz *ToF*) und zum anderen über die Phasendifferenz (Phase Coherence Method). Bei der ersten Methode generiert der Sender Signale, die in regelmäßigen Abständen ausgesendet und von den Sensoren empfangen werden. Über die jeweilige Laufzeit der Signale vom Sender zum Sensor wird der Abstand zwischen Sensor und Sender berechnet. Durch die Änderung der Position ändern sich die Laufzeiten, die die Signale jeweils vom Sender zum Sensor benötigen, womit dann ein anderer Wert für den Abstand berechnet wird. Bei der Phasendifferenz-Methode wird dauerhaft ein Signal in einer bestimmten, vorher bekannten Frequenz vom Sender ausgesendet. Der Sensor empfängt das Signal der gleichen Frequenz, welches um einen Phasenwinkel verschoben ist. Über die Differenz zwischen der Phase beim Sender

und der beim Sensor können relative Abstandsänderungen berechnet werden (vgl. [46], S. 35).
Mit beiden Methoden kann, wenn nur ein Sender und ein Sensor verwendet werden, lediglich der Abstand der Person zum Sender bzw. Sensor gemessen werden. Sollen allerdings die Positionen und Bewegungen mit drei Dimensionen[2] erfasst werden, muss entweder die Anzahl der Sender oder die der Sensoren erhöht werden (vgl. [43], S. 5), d.h. es müssen entweder „drei Sender und ein Empfänger oder ein Sender und drei Empfänger“ ([43], S. 5) verwendet werden.
Beide Messverfahren haben weitere Einschränkungen: So ist es mit dem Laufzeitverfahren nicht möglich, dreidimensionale Momentaufnahmen zu machen, bei denen mehrere Sensoren und Sender verwendet werden, da immer nur die Laufzeit zwischen einem Sensor-Sender-Paar zum selben Zeitpunkt gemessen werden kann und die Laufzeiten der anderen Sensor-Sender-Paare somit sequenziell hintereinander gemessen werden müssen. Bei der Phasendifferenz darf der Anfangs- und Endpunkt der in einer Phase gemessenen Bewegung des Sensors nicht die halbe Wellenlänge des Signals überschreiten, da sonst zwei gültige Werte für die Position des Sensors berechnet werden können und die Position somit nicht eindeutig ist (vgl. [46], S.34).
Bei dem akustischen Messsystem werden die Bewegungen der Person durch die Verwendung kleiner und leichter Mikrophone kaum eingeschränkt. Allerdings können die Messwerte durch Hintergrundgeräusche, wie sie bei pflegerischen Tätigkeiten durch medizinische Geräte oder durch das Sprechen mit dem Patienten oder der Patientin auftreten, verfälscht werden. Auch feste, ultraschall-reflektierende Oberflächen in der Messumgebung beeinflussen das Messergebnis, indem die Reflexion die Laufzeit des Signals durch den Umweg verlängert oder indem die Reflexion die Wellen und damit die Phase verändert. Ein weiterer Nachteil ist die große Menge an Sendern und Sensoren, die für genaue Messungen benötigt werden (vgl. [43], S. 5, [46], S. 34).

Inertiale Systeme

Die Berechnung der Positionen im Bewegungsverlauf basiert bei diesen Systemen auf der Massenträgheit. Dabei trägt die beobachtete Person *inertiale Messeinheiten* (engl. *Inertial Measuring Unit* kurz *IMU*) als Sensoren jeweils fest an den Gelenken. Diese IMUs bestehen aus drei Akzelerometern, die die Beschleunigung des sensortragenden Objekts entlang der x-, y- und z-Achse im Raum messen.

[2] d.h. aufwärts und abwärts, rechts und links sowie nach vorn und nach hinten

Außerdem enthalten die IMUs drei Gyroskope, die die Drehraten dieses Objekts um die x-, y- und z-Achse messen und somit Auskunft über die Richtung geben, in die sich das Objekt bewegt (vgl. [43], S. 5 f., [7]). In einigen IMUs sind darüber hinaus Magnetometer enthalten, die auf Grundlage des Magnetfeldes der Erde die Ausrichtung des Objektes nach Norden bestimmen (vgl. [6], S. 16, 24). Die Sensoren innerhalb der IMUs werden mit einer bestimmten Frequenz ständig ausgelesen. Aus diesen Werten lässt sich anschließend die relative Position in einem zu Beginn der Messung festgelegten Referenzkoordinatensystem berechnen, indem die durch die Akzelerometer gemessenen Beschleunigungen entlang der drei Achsen jeweils zweimal integriert werden, wodurch man über die Geschwindigkeit die Distanz erhält (vgl. [7], [6], S. 11 f.). Das Ergebnis sind dreidimensionale Koordinaten für jede der gemessenen Positionen.
Da bei dieser Messmethode die Bestimmung der neuen Position auf der zuvor berechneten Position beruht, summieren sich Messfehler, sodass es sinnvoll ist, die Sensoren vor der Messung oder erneut nach einigen Messungen zu kalibrieren. Zum Auslesen der IMUs wird ein Empfänger benötigt, welcher den Messbereich einschränkt, sofern er nicht direkt am Körper getragen werden kann. Darüber hinaus benötigt dieses Messsystem im Gegensatz zu einigen anderen nicht-optischen Systemen keine Sender oder Kameras, wodurch die Bewegungsfreiheit der Person sehr groß und nicht auf einen Messbereich beschränkt ist. Außerdem haben Lichtverhältnisse keine Auswirkungen auf die Messungen, und solange die IMUs keine Magnetometer enthalten, sind sie unempfindlich gegenüber Magnetfeldern (vgl. [43], S. 5 f.).
In dieser Arbeit wird ein solches Messsystem von der Marke TEA®zur Evaluation des manuellen Annotationstools verwendet (siehe Kapitel 4).

Optische Systeme

Bei optischen Systemen werden die Bewegungen mit Kameras erfasst, indem die nachzuverfolgenden Objekte, wie z.B. Handgelenke, in jedem einzelnen Bild gesucht werden. Dies kann zum einen mit Hilfe von Markern und zum anderen ohne derartige Unterstützung umgesetzt werden. Da für dieses Messsystem immer mindestens zwei Kameras verwendet werden, müssen alle verwendeten Kameras synchronisiert sein, d.h. es muss gewährleistet sein, dass alle jeweils gleichzeitig die Aufzeichnung beginnen, gleichzeitig die Einzelbilder erfassen und die Aufzeichnung jeweils auch gleichzeitig beenden. Auch die Frequenz, in der aufgezeichnet wird, sollte bei allen Kameras gleich sein und liegt meist zwischen 30 und 1000 Hz.

Da die Kameras, als optisches System, den Messbereich durch ihr jeweiliges Sichtfeld definieren, müssen sie kalibriert werden, um später aus den Bilddaten über die Positionen und Ausrichtung der Kameras die genaue Position des Markers rekonstruieren zu können. Dies geschieht anhand eines Objektes mit bekannten Maßen, welches mit Markern ausgestattet ist. Die Messergebnisse dieser initialen Messung ermöglichen den Rückschluss auf die jeweiligen Kamerapositionen rund um den Messbereich. Da bereits leichte Abweichungen der Kameraposition im Vergleich zu deren Position während der Kalibrierung zu Messfehlern führen, müssen diese Systeme in regelmäßigen Abständen neu kalibriert werden. In kommerziellen Systemen werden meist 20 oder mehr Kameras eingesetzt, da eine höhere Anzahl von Kameras die Genauigkeit erhöht und die Räume für Verdeckungen minimiert (vgl. [43], S. 6, [46], S. 28 ff.).

Markerbasierte Systeme

Markerbasierte Systeme nutzen zur Messung der Bewegungen Sender und Sensoren: Hier sind die Sensoren in Form von Kameras am Rand des Messfeldes aufgestellt und die Sender sind an der beobachteten Person in Form von kugelartigen Markern befestigt. Dies kann auf verschiedene Art und Weise erfolgen: Zum einen werden Marker mit Hilfe von Gurten (meist Klettstreifen) an den jeweiligen Positionen am Körper der Testperson befestigt. So können die Marker an jeder Person jeglicher Größe und Figur angebracht werden. Allerdings können sie nicht direkt auf den Gelenken platziert werden. Zum anderen kommen spezielle Anzüge zum Einsatz, bei denen die Haltepunkte, an die die Marker angebracht werden, genau auf den Gelenken platziert sind. Durch den hautengen Anzug ist ein Verrutschen der Haltepunkte fast unmöglich und darüber hinaus können Verdeckungen durch die Kleidung ausgeschlossen werden. Das ist bei der Anbringung mittels Klettstreifen nicht immer möglich, da diese auch in Verbindung mit relativ weiter Alltagskleidung genutzt werden können. Bei den Anzügen besteht allerdings der Nachteil, dass für verschiedene Personen, die beobachtet werden sollen, verschiedene Anzüge angefertigt werden müssen, um die Marker jeweils optimal auf den Gelenken zu platzieren (vgl. [46], S. 28 f.). Bei beiden Varianten werden meist 20 bis 30 Marker in folgender Verteilung angebracht (vgl. [43], S. 6): „drei Marker am Kopf und je ein Marker am Genick, auf der Wirbelsäule, den Schultern, den Ellenbogen, [...] den Händen, den Hüften, den Knien, den Knöcheln und den Füßen“ ([43], S. 6 f.) sowie je zwei Marker an den Handgelenken, einen oben (dorsal) und einen unten (palmar) (siehe Abb. 2.3).

Abb. 2.3: Motion-Capture-Anzug eines optischen Systems zur Nachverfolgung [Quelle: http://susi.nu/jonni/Chili/Html/photos.htm 2003, zitiert nach [46], S. 30]

Bei den Markern ist zwischen aktiven und passiven optischen Markern zu unterscheiden. Während die aktiven Marker infrarotes Licht aussenden, reflektieren die passiven Marker das infrarote Licht zu der Kamera zurück, deren Strahler das IR-Licht ausgesendet hat. Um die Positionen der passiven Marker möglichst genau berechnen zu können, müssen der Strahler und die Kamera möglichst dicht nebeneinander liegen, weshalb einige Hersteller die Kamera und den Strahler miteinander kombinieren (siehe 2.4) (vgl. [46], S. 30 f.).

Bei den aktiven Markern wird das infrarote Licht ebenfalls von den Kameras erfasst, jedoch besteht hier das Problem, dass die Marker oft nicht im richtigen Abstrahlwinkel zur Kamera liegen. Dies führt dazu, dass sie zwar im Sichtfeld der Kamera liegen, jedoch nicht von der Kamera registriert und somit nicht im Bild der Kamera dargestellt werden (vgl. [46], S. 31).
Nach Abschluss der Messung müssen in einer Nachbereitung die räumlichen Positionen der Marker ermittelt werden. Hierfür werden zunächst die Kamerabilder von eventuellen zusätzlichen Bildinformationen bereinigt, sodass nur noch die Marker sichtbar sind. Anschließend müssen die zweidimensionalen Koordinaten jedes Markers in den Einzelbildern zweier Kameras registriert werden. Hiernach

Abb. 2.4: Kamera-Strahler Kombination eines optischen HMC-Systems von Vicon [Quelle: [44]]

können anhand der *Epipolargeometrie* die dreidimensionalen Positionsdaten der jeweiligen Marker berechnet werden. Dabei muss jeder Marker von mindestens zwei Kameras erfasst worden sein. Eine Erfassung des Markers durch mehr als zwei Kameras erlaubt eine genauere Bestimmung der räumlichen Position der Marker. Um die Bewegungsdaten in Form von Trajektorien oder ähnlichen Darstellungsweisen zu ermitteln, ist es sinnvoll, zunächst die einzelnen Marker den Gelenken und bei mehreren beobachteten Personen auch den Personen zuzuordnen. Dies erleichtert der Software des Systems die Berechnung der Bewegungsdaten des jeweiligen Gelenks oder der jeweiligen Person (vgl. [46], S. 32).
Mit beiden Markerarten ist es möglich, die Bewegungen von mehreren Personen gleichzeitig aufzuzeichnen; die aktiven Marker verwenden hierzu verschiedene Blitzmuster im ausgesendeten IR-Licht (vgl. [46], S. 31). Darüber hinaus liefern markerbasierte optische Systeme „überdurchschnittlich genaue Messwerte“ ([43], S. 7). Gleichzeitig wird zumindest bei den passiven Markern die beobachtete Person nicht durch Kabel oder ein Exoskelett eingeschränkt. Lediglich bei den aktiven Markern sind Kabel vorhanden, da die Marker mit Strom versorgt werden müssen (vgl. [46], S. 31). Ein Nachteil des optischen Systems besteht darin, dass für verdeckte Gelenke eine Rekonstruktion der Positionen nicht möglich ist, vor allem, wenn sie lange verdeckt sind. Dies ist im pflegerischen Kontext jedoch nicht zu vermeiden. Bei der Interaktion mit den Patientinnen und Patienten kann es z.B. beim Umlagern passieren, dass die Marker abgerissen oder die Kabel beschädigt werden. Das Messsystem ist hinsichtlich der Lichtverhältnisse im Messbereich sehr empfindlich, da falsche Beleuchtungen zu ungewollten Reflexionen führen können,

die die Rekonstruktion der Positionen erschweren oder unmöglich machen. Außerdem ist sowohl die Vor- als auch die Nachbearbeitung sowie die Durchführung der Messung an sich sehr aufwändig, da die Kameras in der Vorbereitung und der Durchführung kalibriert werden müssen und in der Nachbearbeitung jedes Bild einzeln aufbereitet und hinsichtlich der Positionsrekonstruktion verarbeitet werden muss (vgl. [43], S. 7 f.).

Markerlose Systeme

Bei den markerlosen optischen Systemen werden die Bewegungen einer Person über eine oder mehrere Kameras aufgezeichnet. Dafür können sowohl Digitalkameras als auch Tiefenkameras verwendet werden. Tiefenkameras zeichnen auch bei Verwendung von nur einer Kamera den Messbereich dreidimensional auf. Bei der Verwendung der Digitalkameras werden dagegen mehrere Kameras benötigt, deren Daten anschließend aufwändig zu einem dreidimensionalen Bild zusammengefügt werden müssen. In den aufgezeichneten Bildern muss dann das beobachtete Objekt gesucht und markiert werden, hier jedoch ohne Marker. Dabei kommen Techniken der Bildverarbeitung zum Einsatz. Allerdings stoßen diese bei Verdeckungen an ihre Grenzen (siehe Body Tracking SDK in 4.1). Dies ist bei pflegerischen Tätigkeiten durch die enge Interaktion mit den zu pflegenden Personen aber häufig und über eine längere Zeit der Fall. Alternativ kann eine Methode angewendet werden, bei der das beobachtete Objekt in jedem Bild manuell annotiert wird. So kann auf Verdeckungen reagiert werden, indem die annotierende Person die Position des verdeckten Gelenks anhand der vorherigen Position und anderer Hinweise im Bild schätzt. Dies ist über die Techniken der Bildverarbeitung in dem Maße nicht möglich. Bei den markerlosen Systemen ist es möglich, mehrere Personen gleichzeitig in ihren Bewegungen nachzuverfolgen, vor allem wenn die manuelle Annotation genutzt wird. Auch bei diesem System müssen die Kameras kalibriert werden, sofern nicht nur eine einzelne Microsoft®Azure Kinect verwendet wird (siehe Kapitel 3.1.1).
Die Nachbearbeitung ist sowohl bei der Verwendung von Bildverarbeitungstechniken als auch bei der Verwendung manueller Annotationen aufwändig. Für die Bildverarbeitung müssen die Bilder entsprechend der gewählten Techniken aufbereitet werden, und das Ergebnis muss analysiert und gegebenenfalls nachgebessert werden. Dagegen muss bei der manuellen Annotation das Objekt in den Bildern jeweils gefunden und manuell annotiert werden. Die Methode des manuellen Annotierens wird in dieser Arbeit entwickelt und vorgestellt.

2.2.2 Trajektorien und Interpolation

Die menschliche Bewegung verläuft kontinuierlich, d.h. es gibt keine „Wissenslücken“ zwischen den Positionen, die innerhalb der ausgeführten Bewegung eingenommen werden. Die derzeit verfügbaren Verfahren, um Bewegungen aufzuzeichnen und digital zu verarbeiten, können die Bewegung lediglich mit einer bestimmten, endlichen Frequenz abtasten, sodass eine Bewegung durch eine Menge von diskreten Punkten beschrieben wird (vgl. [13], S. 15). Um die Bewegung zu analysieren und zu verarbeiten, muss sie anhand eines mathematischen Modells beschrieben werden. Da für die Analyse der Bewegungen von Objekten in diesem Kontext der Auslöser der Bewegung nicht relevant ist, wird hier auf die Kinematik zurückgegriffen. Sie beschreibt die Bewegung von Objekten in Raum und Zeit quantitativ (vgl. [9], S. 2).

Da sich das Pflegepersonal frei bewegen kann und nur durch physikalische Hindernisse, wie beispielsweise Wände, Betten oder andere Gegenstände, eingeschränkt ist, lässt sich die Bewegung im euklidischen Raum modellieren (vgl. [13], S. 17). Dieser wird durch ein rechtshändiges kartesisches Koordinatensystem aufgespannt.

Die Objekte, deren Bewegung modelliert werden sollen, sind in diesem Kontext die Handgelenke der Pflegekräfte. Sie werden jeweils als Punkt $P(x,y,z)$ repräsentiert, dessen Lage im Raum über die drei Koordinaten x, y, z definiert ist. Sie werden zu einem Ortsverktor zusammengefasst, um den Punkt im kartesischen Koordinatensystem zu veranschaulichen (vgl. [11], S. 23):

$$\vec{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{2.1}$$

Da für eine Bewegung nicht nur die Lage des Punktes im Raum relevant ist, sondern auch der Zeitpunkt, an dem das Objekt diese Lage einnimmt, muss der Ortsvektor um eine zeitliche Komponente erweitert werden, d.h. die Koordinaten müssen in Abhängigkeit von der Zeit angegeben werden:

$$\vec{p}(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix} \tag{2.2}$$

Im Rahmen dieser Arbeit wird die Zeit durch den Index des jeweiligen Bildes in der analysierten Videosequenz repräsentiert, der Index ist gleichbedeutend mit einem eindeutigen Zeitpunkt.

Ein annotierter Punkt nimmt im zeitlichen Verlauf verschiedene Orte ein. Werden diese chronologisch geordnet, ergibt sich die Trajektorie dieses Punktes. Eine Trajektorie ist somit eine chronologisch geordnete Folge dieser zeitlich veränderlichen Ortsvektoren (vgl. [13], S.15 und [11], S. 23).
Die Trajektorien sind weiterhin eine Menge von diskreten Punkten, d.h. zwischen diesen Punkten enthält die Trajektorie keine Informationen über die Bewegung der beobachteten Person. Um diesen Informationsverlust zu reduzieren oder diese Lücke angemessen zu schließen, gibt es zwei Verfahren. Das erste Verfahren besteht darin, die Abtastrate entsprechend hoch zu wählen: je höher die Abtastrate, desto mehr Bilder werden pro Sekunde aufgezeichnet und desto mehr Informationen gibt es über die Bewegung. Denn dadurch werden die zeitlichen Abstände zwischen den Bildern geringer, sodass in den kurzen Zeitabständen keine großen Bewegungen gemacht werden können, deren Informationen verloren gehen (vgl. [13], S. 16).
Das zweite Verfahren nutzt die Interpolation, um eine möglichst kontinuierliche Repräsentation der Bewegung zu erhalten. Es gibt verschiedene Interpolationsverfahren, die in Abhängigkeit von der bereits bestehenden Datenmenge und dem Kontext, in dem die Daten eingesetzt werden sollen, ausgewählt werden müssen. Wurde eine sehr hohe Abtastrate gewählt, sodass eine sehr große Datenmenge vorhanden ist, und muss die Präzision der Bewegung in dem Kontext nicht so hoch sein, ist eine lineare Interpolation ausreichend. In diesen Fällen wird eine gleichförmige Bewegung mit unveränderlicher Richtung von einem annotierten Punkt zum nächten angenommen. Ist durch den Kontext, in dem die Daten verwendet werden sollen, eine hohe Präzision gefordert und die bereits bestehende Datenmenge ist nicht so groß, ist ein nichtlineares Interpolationsverfahren sinnvoller. Denn in diesen Fällen können Veränderungen der Bewegungsrichtung und -geschwindigkeit zwischen einzelnen, zeitlich äquidistanten Datenpunkten nicht ausgeschlossen werden, sodass sie räumlich nicht mehr zwingend äquidistant sind. Zum Beispiel können in einem Zeitraum (zwischen zwei annotierten Punkten) sowohl schnelle als auch langsame Bewegungen ausgeführt worden sein, wodurch in den Interpolationsintervallen mit der schnellen Bewegung die räumliche Distanz zwischen zwei Stützstellen größer wäre als bei einer langsamen Bewegung (vgl. [13], S. 16).

2.3 Anforderungen

Aus dem Nutzungskontext des Annotationstools in der Pflege und den Aspekten, die in den aktuellen Tools zur Nachverfolgung und Positionsschätzung nicht behandelt werden, ergeben sich einige Anforderungen an das im Rahmen dieser Arbeit entwickelte Tool.

Um Tätigkeiten in der Pflege durch Roboter ausführen zu können, muss der gesamte Bewegungsablauf der Hände der Pflegekraft aufgezeichnet, beobachtet und analysiert werden. Daher ist es notwendig, Videos, d.h. Sequenzen von Bildern, verarbeiten zu können, da nur hier die Veränderung der Positionen der Hände abgebildet werden. Ein einziges Bild zum Dokumentieren des Bewegungsablaufes ist nicht ausreichend, denn dabei würde sich die Frage stellen, wann das Bild aufgenommen wird: am Anfang, in der Mitte oder am Ende des Bewegungsablaufs. Und welche Bewegungen davor und/oder danach ausgeführt wurden. Ein Video deckt einen großen Teil aller Zeitpunkte einer bestimmten pflegerischen Interaktion ab und ist somit optimal für die Analyse. Diese Videos müssen in einem Vorverarbeitungsschritt in einzelne Bilder zerlegt werden, um die abgebildeten Daten analysieren zu können.

Neben der Notwendigkeit von Bildsequenzen ist es auch sinnvoll, nicht nur mit einer normalen Videokamera den Bewegungsverlauf aufzuzeichnen. Vielmehr ist es erforderlich, die Szene dreidimensional darstellen zu können, damit die Nutzerinnen und Nutzer des Annotationstools sich besser in der Szene orientieren und auch die Handgelenke präziser annotieren können, als es in einem zweidimensionalen Bild möglich wäre. Denn in einem dreidimensionalen Bild kann auch die dritte Dimension für das Ausrichten der Markierung genutzt werden. Außerdem kann in das Bild hineingezoomt werden und so auch die räumliche Anordnung besser erkannt werden oder es kann die Perspektive geändert werden. Denn manchmal kann es hilfreich sein, die Szene nicht nur von vorne zu betrachten, sondern auch von einer anderen Seite. Aus diesem Grund wird das Video mit einer Tiefenkamera aufgezeichnet, denn diese zeichnet neben den normalen Bilddaten auch Informationen über die Tiefe in der Szene auf. In dieser Arbeit wird die Azure Kinect von Microsoft®verwendet (siehe Kap. 3.1.1 auf Seite 22).

Weiterhin muss es möglich sein, mindestens zwei Handgelenke gleichzeitig nachverfolgen zu können. Denn im Gegensatz zur Gestenerkennung, bei der meist nur eine Hand beobachtet und analysiert wird, sind bei pflegerischen Tätigkeiten im-

mer mindestens zwei Hände und damit auch Handgelenke erforderlich. Da sich fast alle Tools, die die Nachverfolgung und Positionsschätzung von Händen behandeln, vorrangig mit der Gestenerkennung und Steuerung über Gesten beschäftigen, sind deren Ansätze in diesem Kontext ungeeignet, wie bereits in Kapitel 2.1 dargelegt. Im manuellen Annotationstool müssen alle Handgelenke annotiert werden können, die die Nutzerinnen und Nutzer für wichtig befinden. Dabei sollen die Annotationen der verschiedenen Handgelenke voneinander unterscheidbar sein.

Bei der Nachverfolgung im pflegerischen Kontext kommt es darauf an, die Trajektorien bestimmter Teile des Körpers, wie beispielsweise der Handgelenke, möglichst vollständig nachzuverfolgen und nicht die Trajektorie des gesamten Körpers. Deshalb ist die vollständige Verdeckung dieser Körperteile durch den Körper der Patientin oder des Patienten bei der engen Interaktion und aufgrund der Kameraperspektive eine Herausforderung, die unbedingt gelöst werden muss. Erst die volltständige Erfassung der Trajektorien beider Handgelenke erlaubt, diese Daten zum Beispiel für die Steuerung eines Pflegeroboters zu nutzen. Da die aktuellen optischen Systeme zur Nachverfolgung und Positionsschätzung dies nicht in ausreichender Präzision leisten können, wird in dieser Arbeit die Software für ein manuelles Annotationstool entwickelt, in welchem die Nutzerinnen und Nutzer selbst die Position der Handgelenke markieren.

3 Implementierung

In diesem Kapitel wird beschrieben, wie das manuelle Annotationstool implementiert wurde. Dazu werden zunächst die verwendeten Geräte und Bibliotheken vorgestellt. Anschließend wird auf die methodische Vorgehensweise eingegangen, die verwendet wurde, um das Annotationstool zu entwickeln. In der Umsetzung werden die verschiedenen Funktionalitäten dargestellt und ihre Entwicklung detailliert erklärt und begründet. Abschließend werden die Grenzen des Annotationstools anhand der technischen Bedingungen und der in Kapitel 2.3 formulierten Anforderungen aufgezeigt und begründet.

3.1 Werkzeuge

Dieses Kapitel beschreibt die Geräte und Tools, die eingesetzt werden, um das Annotationstool zu entwickeln, zu testen und zu evaluieren. Dazu wird zunächst auf das Microsoft Azure Kinect Developement Kit eingegangen, das die Bewegungen in Form von Bildern aufzeichnet. Mit Open3D und PyQt5 werden die Software-Bibliotheken vorgestellt, die zur Implementierung des Annotationstools genutzt wurden. Am Ende des Kapitels wird der Captiv Motion-Capture-Anzug der Firma TEA vorgestellt, welcher zur Validierung dieses Tools verwendet wird.

3.1.1 Microsoft®Azure Kinect DK

Die Bilddaten für die Bewegungsanalyse werden mit der *Microsoft Azure Kinect* (im Folgenden Azure Kinect) des *Azure Kinect Development Kits* (im Folgenden Azure Kinect DK) aufgenommen. Sie ist seit dem Frühjahr 2020 auf dem deutschen Markt erhältlich (vgl. [48]). Mit der Tiefenkamera, der Farbkamera, dem Orientierungssensor und dem räumlichen Mikrofonarray ermöglicht die Azure Kinect die Analyse von Bewegung und Sprache (vgl. [23], S. 3). Da in dieser Arbeit ausschließlich die Bewegung analysiert werden soll, wird im Folgenden nur auf die Tiefen- und die Farbkamera eingegangen.
Neben den Sensoren gehören auch verschiedene Anwendungen und *Software Development Kits* (SDKs) zu dem Azure Kinect DK: Darunter unter anderem die

Microsoft® Azure Kinect Body Tracking SDK (im folgenden Body Tracking SDK). Hiermit können in dreidimensionalen Bilddaten mehrere Körper, die sowohl vollständig als auch nur teilweise sichtbar sein können, segmentiert, über eine ID eindeutig identifiziert und nachverfolgt werden, indem ein anatomisches Gerüst in Form eines kinematischen Skeletts für jeden sichtbaren Körper erstellt wird. Dieses Skelett besteht aus 32 Gelenken, die von der Mitte des Körpers zu den Extremitäten verlaufend hierachisch geordnet sind. Das heißt, dass beispielsweise das Schultergelenk das übergeordnete Gelenk des Ellenbogens ist. In der Beziehung zum Schultergelenk stellt das Ellenbogengelenk daher das untergeordnete Gelenk dar, jedoch fungiert dieses gegenüber dem Handgelenk wiederum als übergeordnetes Gelenk (vgl. [23], S. 4 ff., S. 31 ff.).

Bei der Nachverfolgung werden sowohl die Position der Gelenke als auch deren Orientierung aufgezeichnet. Letzters wird dabei als normalisierte Quaternion repräsentiert, während die Position als Tripel von x-, y-, z-Koordinaten dargestellt wird. Beides wird relativ zum dreidimensionalen Koordinatensystem der Tiefenkamera geschätzt und innerhalb dessen für jedes Gelenk als absolutes Koordinatensystem, auch Gelenkkoordinatensystem genannt, dargestellt (siehe Abbildung 3.1). Diese Nachverfolgung kann im Viewertool der SDK visualisiert werden (vgl. [23], S. 5 und S. 31).

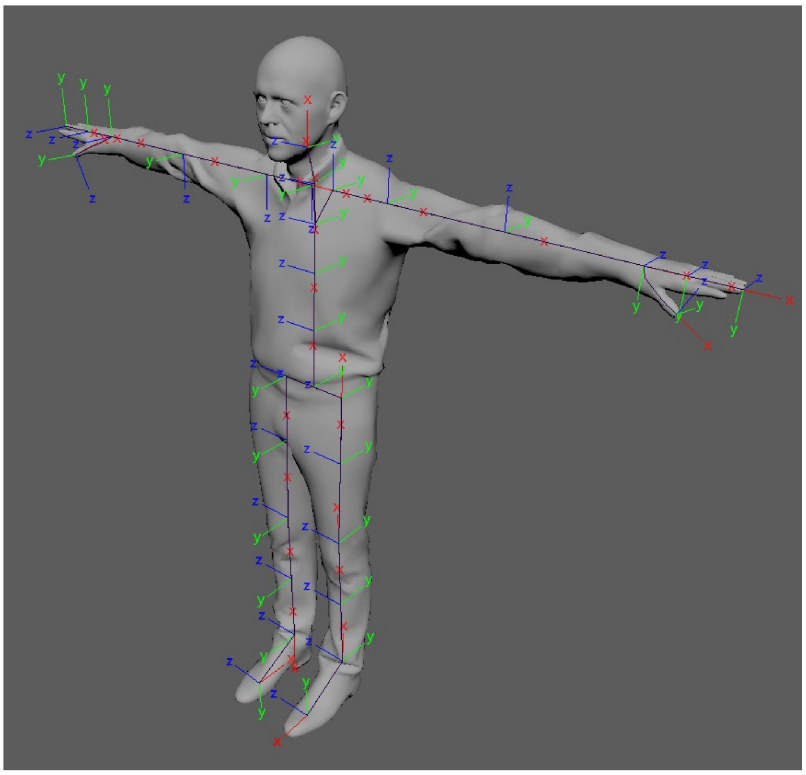

Abb. 3.1: Anordnung der relativen Gelenkkoordinatensysteme [Quelle: (vgl. [23], S. 32)]

Diese Arbeit verwendet das im Rahmen des Projekts TEDIPA entwickelte Body Tracking SDK, welches auch in [2] zum Einsatz kommt. Hiermit werden Ground-Truth-Daten generiert. Als Ergebnis wird eine JSON-Datei ausgegeben, die die Positionskoordinaten sowie die Orientierungsquaternions enthält.

Mit einem weiteren SDK, dem *Sensor SDK*, können sowohl die Tiefen- als auch die Farbkamera gesteuert und synchronisiert werden. Auch auf die Metadaten der Aufzeichnungen kann hierüber zugegriffen werden. Dies ist über ein Viewertool sowie über ein Sensor-Aufzeichungstool realisiert. Im Rahmen dieser Arbeit werden sie jedoch nur zu Kontrollzwecken eingesetzt. Die Daten für die Verarbeitung werden über den *Azure Kinect Recorder* von Open3D aufgenommen (siehe Kap. 3.1.2) (vgl. [23], S. 4).

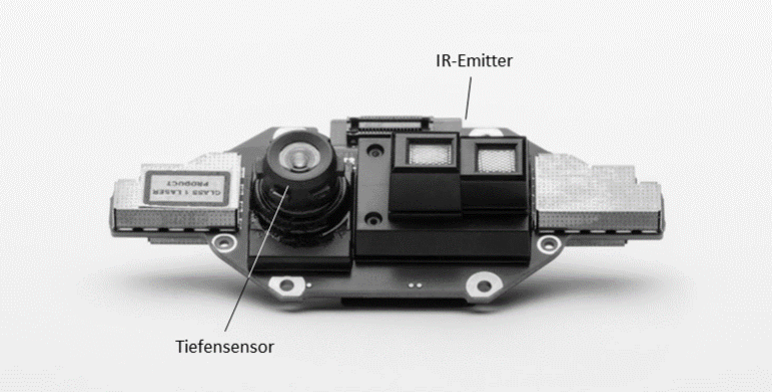

Abb. 3.2: Tiefenkamera bestehend aus Tiefensensor und IR-Sender [Quelle: modifiziert nach [23], S. 23]

Die Tiefenkamera ist eine von der Farbkamera unabhängige Einheit, bestehend aus einem *Infrarot-Sender* (folgend IR-Sender) und einem Tiefensensor. Der IR-Sender, eine nach Continous-Wave-Amplituden modulierte Lichtquelle, sendet Lichtmodule im nicht-sichtbaren Nahinfrarotspektum[1] aus und beleuchtet die Objekte im Sichtfeld der Kamera. Die Objekte reflektieren das Licht, welches dann von dem Tiefensensor registriert wird. Hierbei wird indirekt die sogenannte *Time-of-Flight* (ToF) gemessen, also die Zeit, die das Licht vom IR-Sender zum Objekt und wieder zurück zum Tiefensensor benötigt, d.h. es wird gemessen, wie weit die Phasen der Amplituden des emittierten und des reflektierten Lichts gegeneinander verschoben sind. Diese Daten werden als Rohdatensignale von der Tiefenkamera an den PC weitergeleitet, an den die Kamera angeschlossen ist. Das Sensor-SDK übersetzt im „GPU-beschleunigten Tiefenmodul“ ([23], S. 23) die gemessene Phasendifferenz in Distanzwerte und schreibt sie in die z-Koordinaten der Pixel (siehe Abbildung 3.3). Diese Tiefenzuordnung wird in Millimeter angegeben und ist im Tiefenbild farblich codiert. Zusätzlich zu der Tiefenzuordnung führt die Tiefenkamera noch eine IR-Messung durch. Dabei wird jedem Pixel ein zur jeweils reflektierten Lichtmenge proportionaler Wert zugeordnet (vgl. [5], S. 1 und [23], S. 22 ff.).

[1] das sind Lichtwellen mit einer Wellenlänge von 780 nm bis 3 μm (nach [33])

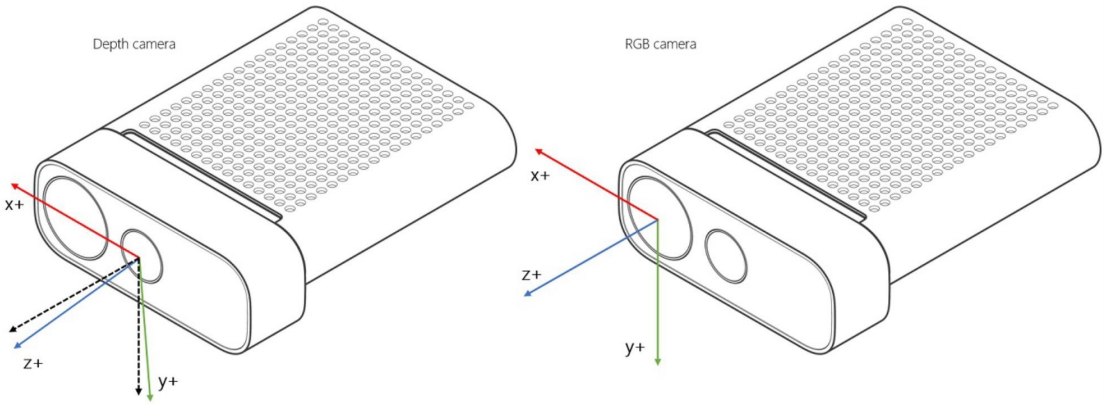

Abb. 3.3: Koordinatensysteme der Kameras der Azure Kinect [Quelle: [23], S. 30]

Die Tiefen- und die Farbkamera, auch RGB-Kamera, besitzen je ein dreidimensionales, rechtshändiges Koordinatensystem, deren Koordinatenursprünge (0, 0, 0) auf dem Fokuspunkt der jeweiligen Kamera liegen (siehe Abbildung 3.3). Aus Sicht des Gerätes zeigt die positive x-Achse jeweils nach rechts, die positive y-Achse jeweils nach unten und die positive z-Achse zeigt jeweils in den Raum. Eine Neigung um 6 ° des Koordinatensystems der Tiefenkamera gegenüber der RGB-Kamera muss hierbei berücksichtigt werden (vgl. [23], S. 30).
Darüberhinaus enthält die Tiefenkamera zwei Belichtungseinheiten: eine für das enge Sichtfeld, auch Narrow Field of View (NFOV) genannnt, und eine weitere für das breite Sichtfeld, auch Wide Field of View (WFOV). Erstere Belichtungseinheit ist am Gehäuse der Kamera ausgerichtet, wohingegen die WFOV-Belichtungseinheit gegenüber der Tiefenkamera zusätzlich um 1,3 ° nach unten geneigt ist. Eine weitere Eigenschaft der Aufnahme mit der NFOV-Belichtung ist, dass sowohl in x- als auch in y-Richtung die Ausdehnung des Sichtfeldes (engl. Field of Interest (FOI)) eingeschränkt ist, jedoch in der z-Richtung größere Distanzen erfasst werden können als in einer Aufnahme mit der WFOV-Belichtungseinheit. Dies kann einen Unterschied von 1,65 m bis zu 2,58 m ausmachen, wie Abbildung 3.4 zu entnehmen ist. Die WFOV-Einheit kann dementsprechend größere Ausdehnungen in der x- und y-Richtung erfassen (je 120 ° zu 75 ° x 65 ° bei NFOV) (vgl. [23], S. 23 und S. 29 ff.).
Weiterhin unterstützen die NFOV- sowie die WFOV-Belichtungseinheit verschiedene Aufnahmemodi. Für beide Einheiten gibt es einen unklassifizierten (engl. *unbinned*) und einen 2x2-klassifizierten (engl. *2x2-binned*) Modus, wobei sich jeweils die erfassbare Distanz zur Kamera vergrößert: Bei NFOV kann die Distanz

im 2x2-klassifizierten Modus um 1,6 m erweitert werden, bei WFOV um 0,67 m (siehe Abbildung 3.4). Jedoch halbiert sich die Auflösung jeweils von unklassifiziert zu 2x2-klassifiziert (siehe Abb. 3.4) (vgl. [23], S. 23).
In allen Modi kann eine Bewegung mit einer Frequenz von 0 Hz bis 30 Hz aufgezeichnet werden. Ausnahme ist der Aufnahmemodus WFOV unklassifiziert, welcher nur mit einer Bildrate von bis zu 15 Hz aufzeichnen kann (vgl. [23], S. 23).

MODE	LÖSUNG	FOI	FPS	BETRIEBSBEREICH*	BELICHTUNGSZEIT
NFOV unklassifiziert	640 × 576	75° × 65°	0, 5, 15, 30	0,5–3,86 m	12,8 ms
NFOV 2 × 2 klassifiziert (SW)	320 × 288	75° × 65°	0, 5, 15, 30	0,5–5,46 m	12,8 ms
WFOV 2 × 2 klassifiziert	512 × 512	120° × 120°	0, 5, 15, 30	0,25–2,88 m	12,8 ms
WFOV unklassifiziert	1.024 × 1.024	120° × 120°	0, 5, 15	0,25–2,21 m	20,3 ms
Passives IR	1.024 × 1.024	–	0, 5, 15, 30	–	1,6 ms

Abb. 3.4: Aufnahmemodi der Tiefenkamera [Quelle: ([23], S. 88)]

Zusätzlich unterstützt die Tiefenkamera noch einen passiven IR-Modus, welcher nur das Umgebungslicht verwendet und nicht aktiv die Szene mit IR-Licht beleuchtet (vgl. [23], S. 22).

Sowohl der Recorder des Sensor SDK als auch der Recorder von Open3D zeichnen mehrere Videospuren, einen für Farb- und einen für Tiefendaten, auf und speichern diese im *Matroska-MKV-Format*. Hierbei werden zusätzlich auch Informationen über die Kamerakalibration und weitere Metadaten gespeichert, wie beispielsweise der verwendete Tiefenmodus, der verwendete Farbmodus und die Bildrate. Der Recorder des Sensor-SDK zeichnet standardmäßig im Modus NFOV unklassifiziert auf, mit einer Frequenz von 30 Hz und im Farbmodus von 1080p, wohingegen der Recorder von Open3D als Standardeinstellung den WFOV 2x2-klassifizierten Modus mit 30 Hz und den Farbmodus 720p definiert hat (vgl. [23], S. 10 und [27]).

Da bei der Beobachtung von Bewegungen von pflegerischen Tätigkeiten keine großen Ausdehnungen in die x- und y-Richtung zu erwarten sind, es jedoch einen größeren Spielraum in der Distanz zur Kamera gibt, wird in dieser Arbeit ebenfalls mit NFOV unklassifiziert aufgezeichnet, wobei sich die beobachtete Person meist in 2 m Entfernung zur Kamera aufhält. Darüberhinaus wird mit einer Frequenz von 30 Hz aufgenommen, damit die Bewegungsrepräsentation die kontinuierlichen, realen Bewegungen möglichst gut approximiert.

3.1.2 Open3D und PyQt5

In diesem Kapitel werden die eingesetzten Bibliotheken Open3D und PyQt5 vorgestellt und es wird erläutert, wie sie in dieser Arbeit eingesetzt werden.

Open3D

Open3D ist eine unter *MIT License* veröffentlichte Open-Source-Bibliothek zur Verarbeitung von dreidimensionalen Daten. Sie ist mit C++ 11 entwickelt und kann durch eine Einbindung von Python über pybind11 in nahezu allen Funktionalitäten sowohl in C++- als auch in Python-Anwendungen eingesetzt werden, sofern diese in Python 3.5 bis 3.8 entwickelt werden (vgl. [27], S. 1, [28]). Darüber hinaus werden die gängigen Plattformen *Ubuntu 18.04+*, *MacOS 10.14+* sowie *Windows 10 in der 64-bit-Version* unterstützt (vgl. [28]). Für die Entwicklung des Annotationstools wurde Python 3.7.4 verwendet.

Open3D ist in Aufbau und Eigenschaften vergleichbar mit OpenCV. Während OpenCV eine Bibliothek zur Verarbeitung von digitalen Bildern ist, stellt Open3D diese Funktionalitäten für dreidimensionale Bilddaten zur Verfügung. Hierbei bildet Open3D relevante 3D-Datenstrukturen wie Punktwolken, Meshes und RGBD-Bilder ab und stellt die dafür jeweils gängigen 3D-Datenverarbeitungsalgo-rithmen bereit, d.h. die Ein- und Ausgabe, das Abtasten (engl. *sampling*) und das Konvertieren der Datenstrukturen. Wie auch in OpenCV sind außerdem die weit verbreiteten Bildverarbeitungsalgorithmen implementiert, die über einfache Funktionsaufrufe eingesetzt werden können. So kann beispielsweise über die folgenden beiden Zeilen aus einem Farb- und einem Tiefenbild ein RGBD-Bild und daraus eine Punktwolke generiert werden:

```
rgbd_image = o3d.geometry.RGBDImage.create_from_color_and_depth(color=color_raw,
                                                                depth=depth_raw,
                                                                depth_scale=1000.0,
                                                                convert_rgb_to_intensity=False)
point_cloud = PointCloud.create_from_rgbd_image(rgbd_image,
                                                o3d.camera.PinholeCameraIntrinsic(o3d.camera.
                                                                                  PinholeCameraIntrinsicParameters.
                                                                                  Kinect2ColorCameraDefault))
```

Abb. 3.5: Generieren einer Punktwolke

Neben der reinen Datenaufbereitung und -verarbeitung können die Datenstrukturen und teilweise auch die Algorithmen dreidimensional visualisiert werden (vgl. [26]). Dies ist über das `visualization`-Modul implementiert. Der in diesem Modul zur Verfügung gestellte Visualizer kann für das reine Visualisieren der Datenstrukturen genutzt werden, wobei es hier möglich ist, über Mausinteraktionen die angezeigte Datenstruktur zu rotieren, zu skalieren und im Raum zu bewegen. Darüber hinaus können über festgelegte Tastenkombinationen vier verschiedene Rendering-Stile eingestellt werden, beispielsweise als *Hot Color Map* (vgl. [32] und [51], S. 2 ff.). Es ist allerdings auch möglich, einen interaktiven Visualizer zu bauen, der über individuell implementierte Python Callback Funktionen gesteuert werden kann. Seit Open3D 0.10.0 kann durch die Nutzung des Dear ImGUI Projektes zusätzlich ein GUI integriert werden (vgl. [29]). Da Open3D zum Visualisieren der 3D-Datenstrukturen OpenGL verwendet, eine API[2] zur Darstellung dreidimensionaler Daten in Echtzeit, werden die anzuzeigenden Datenstrukturen gleichzeitig berechnet und im Fenster angezeigt (vgl. [51], S. 2).
Open3D enthält in seiner API acht weitere Module, die mit einer kurzen Beschreibung ihrer Funktionalität in Tabelle 3.1 dargestellt sind.

[2] Application Programming Interface

Modul	Funktionalität
Camera	enthält verschiedene intrinsische und extrinsische Parameter für das Lochkameramodell, Default Parameter für verschiedene Bildsensoren und Strukturen zum Speichern von Kameratrajektorien
Core	enthält die Kerntypen und -datenstrukturen sowie deren Funktionen
Geometry	bildet die Datenstrukturen Punktwolken, Meshes und RGBD-Bilder ab sowie die zugehörigen grundlegenden Verarbeitungsalgorithmen
IO	Lesen und Schreiben der 3D-Datenstrukturen sowie der Daten von der Azure Kinect
t	enthält die Module *Geometry* und *IO*
ML	stellt die Funktionalitäten von Tensorflow- und Torch-Modulen zur Verfügung
Pipeline	stellt die Algorithmen zur Optimierung der Farbkarten, zur Volumenintegration, zur Nachverfolgung und zur Ausrichtung von RGBD-Bildern sowie zur Bildregistrierung sowohl global als auch lokal zur Verfügung
Utility	stellt Funktionalitäten zum Konvertieren der Datenstrukturen zu Arrays über Numpy zur Verfügung und Zugriffsmöglichkeiten auf die Sichtbarkeit von Konsolenausgaben
Visualization	siehe oben

Tabelle 3.1: Open3D-Module und ihre Funktionalitäten (vgl. [51], S. 4 und alle Module der PythonAPI in [30])

Open3D unterstützt die Azure Kinect über das Azure Kinect Sensor SDK v1.2.0 als Eingabemedium für 3D-Daten, indem Python-Skripte zur Aufzeichnung mit der Azure Kinect, zum Ansehen der Aufnahmen und zum Ansehen und Verifizieren der verschiedenen Kameraeinstellungen zur Verfügung gestellt werden sowie Klassen und Funktionen, mit deren Hilfe auch eigene Anwendungen mit der Azure Kinect entwickelt werden können. Über den *Azure Kinect Viewer* können, ähnlich dem *k4aviewer* des Azure Kinect Sensor SDK, die Sensorströme visualisiert werden, um verschiedene Einstellungen der Kamera auszuprobieren und die Kamera optimal auf die Szene auszurichten. Über den *Azure Kinect Recorder* können Datensätze aufgenommen werden und über eine `config.json`-Datei

können die Kameraeinstellungen wie Farb- und Tiefenmodus und die Einstellung der Frequenz vorgenommen werden, sofern sie von den Standardwerten (siehe Kapitel 3.1.1) abweichen sollen. Allerdings werden im *Azure Kinect Recorder* von Open3D die Daten nicht in der eingestellten Frequenz aufgezeichnet, die einzelnen Bilder der Videosequenz sind in zeitlicher Hinsicht nicht äquidistant. Aus diesem Grund wurde in dieser Arbeit wieder auf die Aufzeichnung der Daten über den *k4arecorder* des Azure Kinect Sensor SDK zurückgegriffen. Der *Azure Kinect MKV Reader* dient der Anzeige der aufgenommenen Daten, da diese nicht unbedingt von den gängigen Mediaplayern unterstützt werden. Hierüber kann das Video auch in die einzelnen Bilder zerteilt werden, indem beim Aufruf des Skripts ein Ausgabeordner angegeben wird (vgl. [27]). Dies wird auch im Rahmen dieser Arbeit eingesetzt, um die Videos für die Annotation vorzubereiten (siehe Kapitel 3.3, S. 37 ff.). Alle drei Skripte nutzen den Open3D-eigenen Visualizer zur Darstellung der Aufnahmedaten in Echtzeit oder zur Wiedergabe der bereits aufgenommenen Daten. Dabei werden immer das Farb- und das Tiefenbild nebeneinander angezeigt.

Während des Einsatzes des Open3D Visualizers hat sich herausgestellt, dass dieser ein linkshändiges Koordinatensystem zum Anzeigen und Verändern der eingefügten Datenstrukturen verwendet, sodass dies bei der Annotation berücksichtigt werden muss. In Kapitel 3.3 wird dies genauer beschrieben.

PyQt5

PyQt5 ist eine API, die Code enthält, der den Einsatz von Qt über Python-Befehle ermöglicht. Qt ist eine von Qt Company bereitgestellte Open-Source-C++-Bibliothek für GUI-Steuerelemente (engl. *widgets*). PyQt5 wird von Riverbank Computing entwickelt und ist unter einer GPL bzw. LGPL Lizenz veröffentlicht (vgl. [14], S. 5).

Mit Qt und damit auch mit PyQt5 können fensterbasierte Anwendungen implementiert werden, die auf verschiedenen Plattformen lauffähig sind, ohne dass für die verschiedenen Betriebssysteme jeweils eigener Code geschrieben werden muss. Zu den Plattformen gehören Windows, Linux, macOS und Android. Neben der reinen GUI-Entwicklung unterstützt Qt auch die Interaktion mit Datenbanken, die Entwicklung von Anwendungen für Multimediainhalte oder Vektorgrafiken und den Einsatz von Model-View-Controller-Schnittstellen (vgl. [14], S. 5).

PyQt5 wird in dieser Arbeit eingesetzt, um GUIs zu entwickeln, die die Prozesse außerhalb von Open3D abbilden und die Daten abfragen, die für die Prozesse von Open3D notwendig sind, wie beispielsweise die Pfade zu Dateien oder Ordnern oder das Anstoßen der Prozesse in Open3D. Dadurch werden die nicht-intuitiven Interaktionsmöglichkeiten wie Konsoleneingaben oder Interaktion über Tastenkombinationen reduziert, was die Interaktion mit dem Annotationstool erleichtert.

Für diese Arbeit wurde PyQt5 den anderen GUI-APIs vorgezogen, da hier über den *QtDesigner* mit Drag-and-drop die GUIs sehr einfach erstellt und designt werden können. Durch die Eingabe von `pyuic5 -x datei.ui -o datei.py` in die Konsole kann aus der Datei im QtDesigner der Quellcode generiert werden. Anschließend kann dann mit Hilfe dieses Quellcodes und der Signal-Strukturen von PyQt5 die entsprechende Anwendung erstellt werden, indem die Steuerelemente der GUI über die Signale mit Python-Methoden verknüpft werden (vgl. [47], S. 4 f.). In dieser Arbeit wird PyQt5 in der Version 5.15.1 verwendet.

3.1.3 Captiv Motion-Capture-Anzug

Um das manuelle Annotationstool evaluieren zu können, werden mit einem inertialen HMC-System die Bewegungsabläufe zusätzlich zur Azure Kinect aufgenommen. Dazu werden der *Captiv Motion Capture Anzug* von TEA®(im Folgenden Captiv) und die dazugehörige Software CAPTIV L7000 verwendet. TEA®stellt verschiedene Sensoren zur Verfügung, die über den Anzug an die beobachtete Person angebracht werden können. In dieser Arbeit werden die TSens Motion Sensoren verwendet, welche die Bewegungen über 15 inertiale Messeinheiten (IMU) aufzeichnen. Diese werden über verstellbare Gurte an den Gelenken der beobachteten Person angebracht, wobei nicht alle Gelenke mit einem Sensor ausgestattet werden müssen. Es können beispielsweise auch nur die Bewegungen des Oberkörpers aufgezeichnet werden. Jedoch müssen immer mindestens zwei Sensoren verwendet werden, um ein Gelenk nachzuverfolgen. In jedem Fall müssen die Sensoren nach der Anleitung in [39], S. 6 ff. platziert werden, um die optimalen Ergebnisse zu erhalten. Die Platzierung der Sensoren ist in Abbildung 3.6 dargestellt. Wie in der Abbildung zu sehen, bestand der Anzug, der in dieser Arbeit verwendet wurde, aus 13 IMUs.

Die von Captiv verwendeten IMUs bestehen aus einem dreiachsigen Akzelerometer, einem dreiachsigen Gyroskope und einem dreiachsigen Magnetometer, die die Bewegungen nach dem in Kapitel 2.2.1 beschriebenen Prinzip aufnehmen. Die

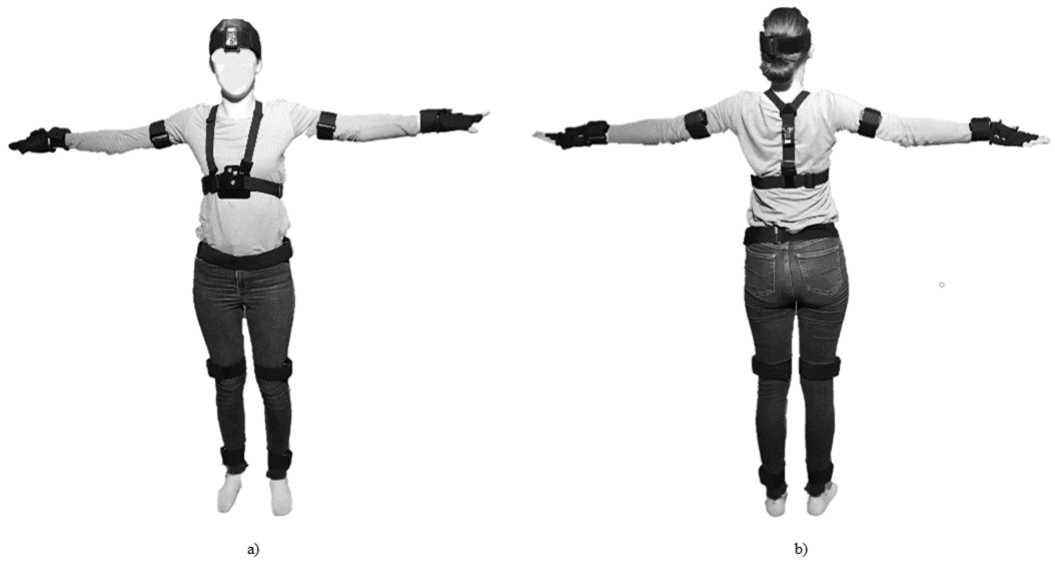

Abb. 3.6: Captiv T-Sens Motion-Capture-Anzug: a) Ansicht von vorn [Quelle: eigene Aufnahme], b) Ansicht von hinten [Quelle: eigene Aufnahme]

IMUs sind akkubasiert, so dass die Bewegungen der Person nicht durch Kabel zur Stromversorgung eingeschränkt werden. Auch die Größe von 60 mm × 35 mm × 19 mm und das Geweicht von 32 g schränkt die Bewegungen nicht ein. Die Sensoren können vier Stunden aufzeichnen. Außerdem kann die Frequenz, in der die Daten aufgezeichnet werden sollen, auf 32 Hz, 64 Hz oder 128 Hz festgelegt werden. Standardmäßig sind 64 Hz eingestellt. Da die Azure Kinect die Daten mit 30 Hz aufzeichnet, wurden hier 32 Hz gewählt (vgl. [4], S. 2).
Um die Daten der IMU zusammenzuführen und auf den Endgeräten nutzbar zu machen, stellt TEA®verschiedene Empfänger zur Verfügung. Der hier verwendete Anzug verwendet den T-Rec, welcher über ein Kabel mit dem Computer verbunden ist, auf dem die Software installiert wurde.
Über die Software CAPTIV L7000 finden die Steuerung und die Vorbereitung der Aufzeichnung statt. Die Sensoren müssen zunächst initialisiert werden, indem sie in einer Umgebung ohne Metall und Magnetfelder in jeweils 10 cm Entfernung voneinander auf einer ebenen Fläche nach Norden ausgerichtet und angeschaltet werden. Dadurch wird das Magnetometer initialisiert. Nach Ablauf einer Minute kann dann über die Software überprüft werden, ob alle Sensoren richtig initialisiert sind. Dies kann über einen Zeiger abgelesen werden, wie in Abbildung 3.7 dargestellt (vgl. [39], S. 3 f.).

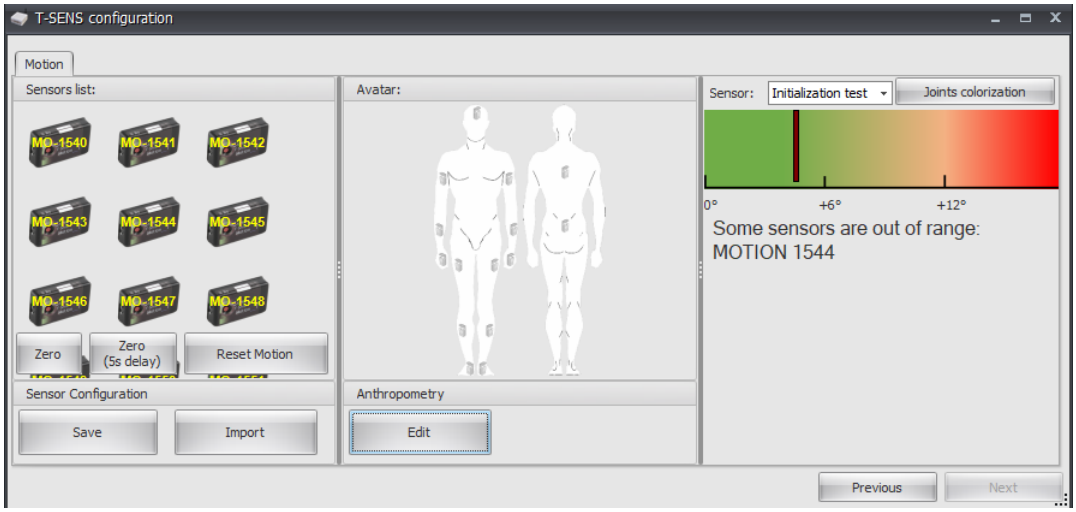

Abb. 3.7: Überprüfung der Initialisierung [Quelle: Screenshot aus [40]]

Anschließend werden die Sensoren jeweils an den Gelenken angebracht und deren jeweilige Position anhand der entsprechenden Nummer in die Software eingegeben sowie die Sensoren tariert. Die Messwerte der Sensoren können in der Software in Echtzeit dargestellt werden (vgl. [4], S. 2). Dies geschieht beispielsweise über einen Avatar. Dieser kann in seinen Proportionen an die Person angepasst werden, die die Sensoren trägt (siehe Abbildung 3.8).

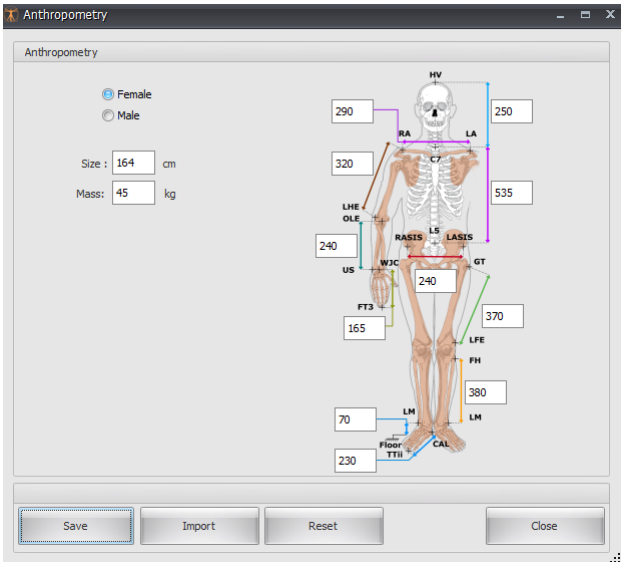

Abb. 3.8: Eingabefenster zum Einstellen der Avatarproportionnen [Quelle: Screenshot aus [40]]

Über die Software können verschiedene Rohdaten in verschiedenen Dateiformaten exportiert werden. Für diese Arbeit sind nur die räumlichen Positionen der Gelenke relevant. Diese Daten können sowohl als relative als auch als absolute Koordinaten exportiert werden. Die absoluten Koordinaten nutzen als Referenzkoordinatensystem ein rechtshändiges Koordinatensystem, dessen Ursprung genau im letzten Lendenwirbels L5 festgelegt wird. Die positive x-Achse zeigt in Richtung des magnetischen Nordens, die y-Achse ist in der waagerechten Ebene des Bodens orthogonal dazu ausgerichtet und die positive z-Achse stellt den Gravitationsvektor der Erde dar. Der Avatar wird ebenfalls anhand dieses Referenzkoordinatensystems dargestellt (siehe Abbildung 3.9) (vgl. [12], S. 11).

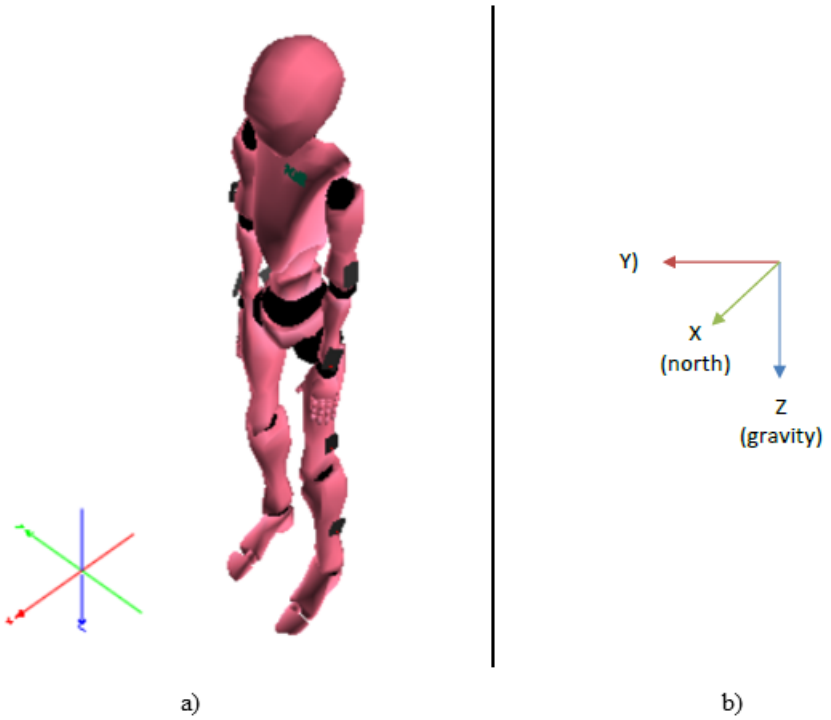

Abb. 3.9: Koordinatensystem von Captiv: a) Ausrichtung des Avatars nach Norden [Quelle: Screenshot aus [40]], b) Referenzkoordinatensystem [Quelle: [12], S. 11]

Die Verwendung von Captiv im Rahmen dieser Arbeit wird in der Evaluation in Kapitel 4.2 detaillierter beschrieben.

3.2 Methodik

Um das manuelle Annotationstool zu entwickeln, wurde nach der in Kapitel 2.1 vorgestellten Literaturrecherche und der Anforderungserhebung (siehe Kapitel 2.3) analysiert, welche Eigenschaften die verwendeten Daten haben und haben müssen. Dazu wurden mit der Azure Kinect mehrere Datensätze in Form von Videodateien aufgezeichnet und miteinander verglichen. Parallel dazu wurde eine einfache Anwendung mit Open3D geschrieben, die es zunächst nur ermöglichte, die Punktwolken aus den Videodateien zu generieren und zu visualisieren.

Nachdem die grundlegende Datenverarbeitung als Basis feststand, d.h. die Punktwolken über Open3D generiert und angezeigt werden konnten, wurde darauf das manuelle Annotationstool aufgebaut. Hierbei wurde meist der Ansatz der testgetriebenen Entwicklung (engl. *Test Driven Development* kurz *TDD*) umgesetzt. Das heißt, es wurden zunächst Testfälle für die zu implementierende Methode erstellt, welche fehlschlugen, da die Methode noch nicht existierte. Anschließend wurde die Methode erstellt und solange entwickelt, bis die entsprechenden Testfälle erfolgreich durchliefen. Neben diesen Tests wurde auch über das Anwenden des Annotationstools überprüft, ob die Funktionalität umgesetzt und erfüllt ist. Traten bei der Verwendung unerwartete oder unerwünschte Effekte auf, die jedoch nicht dazu führten, dass die entsprechenden Testfälle fehlschlugen, wurden sowohl diese Testfälle als auch die betreffenden Methoden derart angepasst, dass das Annotationstool wie erwartet funktionierte und die dazugehörigen Testfälle erfolgreich durchliefen.
Auf diese Weise wurde auch auf Veränderungen reagiert, die sich durch Updates der Versionen der verwendeten Werkzeuge oder durch neue Anforderungen ergaben, die z.B. während der Anwendung des bis dahin entwickelten Annotationstools auftauchten. Beispielsweise fand sowohl die Aufzeichnung der Datensätze in Videodateien als auch das Generieren der Punktwolken über die Konsole statt. Da hierfür die Befehle auswendig gelernt werden mussten und sich vor allem bei der Eingabe von langen Pfaden teilweise Fehler einschlichen, ergab sich eine neue Anforderung an eine intuitivere Interaktion und mehr Nutzerfreundlichkeit. Auf diese Anforderung wurde durch die Implementierung von GUIs eingegangen (siehe Kapitel 3.3 für Details).

Während der Entwicklung wurden die Methoden und Klassen über `doc-strings` dokumentiert, um die Vorgänge in dieser Klasse oder Methode zu beschreiben und darüber eine Weiterentwicklung zu ermöglichen. Zu diesem Zweck wurden zusätz-

lich Inline-Kommentare verwendet, die die nachfolgende Zeile oder den nachfolgenden Code-Abschnitt erklären.

Da das entwickelte Annotationstool in der Art noch nicht existierte, musste es nicht nur durch Tests auf korrekte Funktionalität überprüft werden, sondern auch hinsichtlich der Ergebnisse, die es liefert. Da es sich hierbei um Positionsdaten handelt, die anhand manueller Annotation in Datensätzen aus einem markerlosen optischen HMC-System entstehen, mussten diese Daten anhand von anderen bereits bewerten Systemen überprüft werden. Der Ablauf dieser Überprüfung sowie dessen Ergebnis werden in Kapitel 4 detailliert beschrieben.

Um das Annotationstool auf einfache Weise anderen Nutzerinnen und Nutzern zur Verfügung zu stellen, wurde aus den Python-Skripten über `pyinstaller` eine ausführbare Datei für Windows erstellt. Damit das Annotationstool über den Rahmen dieser Arbeit hinaus erweitert werden kann, wurde es zusätzlich über GitHub/GitLab unter einer MIT License veröffentlicht. Die Inhaberin dieser Lizenz ist Meret B. Lindanis.

3.3 Umsetzung

Für diese Arbeit wird die Interaktion des medizinischen Fachpersonals simuliert und mit Hilfe der Azure Kinect aufgezeichnet. Die aufgezeichneten Daten werden verarbeitet, um über die Annotation von Handgelenken als Ergebnis Trajektorien zu erhalten. Dieser Prozess wird im Folgenden genauer erläutert und ist in Abbildung 3.10 dargestellt.

Das gesamte Annotationstool ist fensterbasiert, d.h. die Nutzerinnen und Nutzer müssen nicht über die Konsole mit dem Tool interagieren, sondern können alle Informationen und Prozesse über die graphischen Benutzeroberflächen (engl. Graphical User Interface, kurz GUI) eingeben und steuern. Dies erhöht die Nutzbarkeit und die Benutzerfreundlichkeit, da keine Konsolenbefehle gelernt werden müssen, sondern der Nutzer oder die Nutzerin intuitiv wie auch in anderen fensterbasierten Tools die erforderlichen Daten suchen, eingeben und die Prozesse über Betätigen der Schaltflächen steuern kann. Um die Nutzbarkeit noch weiter zu erhöhen, sind die GUI und ihre Elemente nach dem Prinzip der Analogie aufgebaut. So sind alle Felder, in die die Pfade zu den erforderlichen Dateien eingetragen werden, mit einem Label versehen, das mithilfe eines eindeutigen

Bezeichnungstextes Aufschluss über die einzufügenden Information gibt. Neben diesem Feld ist auch immer eine *Browse*-Schaltfläche zu finden, über den die Nutzerinnen oder Nutzer im entsprechenden Dialogfenster nach der Datei oder dem Ordner suchen können, die in das Feld eingetragen werden sollen. Sind weitere Informationen notwendig, wie beispielsweise das zulässige Dateiformat oder Informationen zur Art der Speicherung der Daten, werden diese in einem kurzen Text unterhalb des Feldes angegeben. Dies soll gewährleisten, dass die Nutzerinnen und Nutzer alle notwendigen Informationen zur Verfügung haben und das GUI intuitiv nutzen können.

Nach dem Starten der Anwendung gelangt die Benutzerin oder der Benutzer in das zentrale GUI. Von hier aus können alle nachfolgend beschriebenen Prozesse angesteuert werden, in dem auf die entsprechende Schaltfläche geklickt wird. Daraufhin wird immer ein weiteres Fenster geöffnet, in welchem die für diesen Prozess relevanten Informationen abgefragt werden. Erst in diesem Fenster kann dann der entsprechende Prozess gestartet werden. Dieser Aufbau dient zum einen der Übersichtlichkeit der Fenster und zum anderen werden nur die für diesen Prozess wichtigen Informationen erhoben, sodass sich die Nutzerin oder der Nutzer besser orientieren kann.

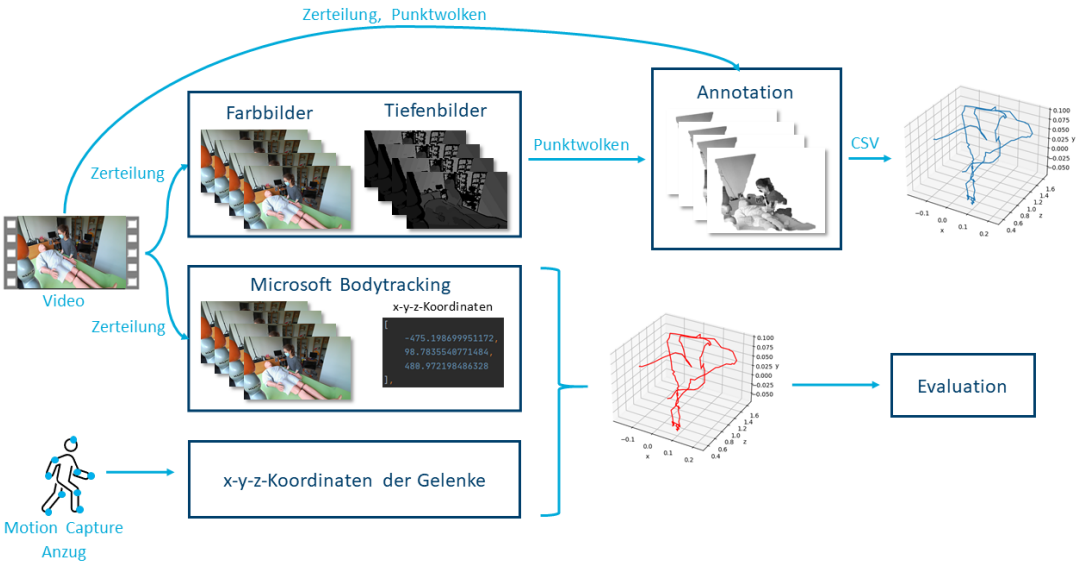

Abb. 3.10: Implementierter Prozess

Um die Bewegungsverläufe der Handgelenke in Punktwolken annotieren zu können, müssen diese Punktwolken vorliegen. Ist dies der Fall, kann der Annotationsvorgang gestartet werden. Ist der Bewegungsverlauf noch nicht aufgezeichnet, muss dies über den *k4arecorder* des Azure Kinect Sensor SDK erfolgen. Hierbei ist der Speicherort des Videos anzugeben. Zusätzlich können optionale Einstellungen vorgenommen werden. Es kann unter anderem eine Konfigurationsdatei ausgewählt werden, in der die verschiedenen Modi der Azure Kinect eingestellt werden. Zum Beispiel werden hier die Aufnahmefrequenz, der Tiefenmodus (siehe Abbildung 3.4) und die Farbauflösung spezifiziert, wie in Abbildung 3.11 dargestellt.

```
{
    "camera_fps" : "K4A_FRAMES_PER_SECOND_30",
    "color_format" : "K4A_IMAGE_FORMAT_COLOR_MJPG",
    "color_resolution" : "K4A_COLOR_RESOLUTION_1080P",
    "depth_delay_off_color_usec" : "0",
    "depth_mode" : "K4A_DEPTH_MODE_NFOV_UNBINNED",
    "disable_streaming_indicator" : "false",
    "subordinate_delay_off_master_usec" : "0",
    "synchronized_images_only" : "false",
    "wired_sync_mode" : "K4A_WIRED_SYNC_MODE_STANDALONE"
}
```

Abb. 3.11: Konfigurationsdatei für die Kameraeinstellungen [Quelle: nach [27]]

Zunächst wird mit der Azure Kinect ein Video von dem Bewegungsablauf einer pflegerischen Tätigkeit über die Konsole aufgezeichnet. Dieses Video wird im *mkv*-Format gespeichert. Das Video besteht, wie in Kapitel 3.1.1 beschrieben, aus einzelnen Bildern (engl. *Frames*), die mit einer definierten Frequenz aufgenommen werden. Standardmäßig sind das 30 Hz. Wurde das Video aufgezeichnet, muss es in einem zweiten Schritt in einzelne Bilder zerlegt werden. Dabei werden zum einen Farbbilder im *.jpg*-Format generiert und zum anderen Tiefenbilder im *.png*-Format. Ihnen werden aufsteigende Nummern zugewiesen. Für diesen Schritt stellt Open3D das Python-Skript `azure_kinect_mkv_reader.py` zur Verfügung. Dieses zerteilt das Video in die einzelnen Bilder und speichert sie direkt. Hierzu muss ein Name für einen neuen Ordner angegeben werden, welcher beim Starten des Python-Skripts angelegt wird. Es darf kein bereits exitierender Ordner angegeben werden, um die Anwenderinnen und Anwender davor zu schützen, unbeabsichtig Daten zu überschreiben. In diesem Ordner werden automatisch auch die Ordner `color` und `depth` angelegt. Die Bilder der beiden Bildarten werden

ebenfalls in aufsteigender Nummerierung in `color` für die Farbbilder beziehungsweise in `depth` für die Tiefenbilder gespeichert, wie in Abbildung 3.12 dargestellt.

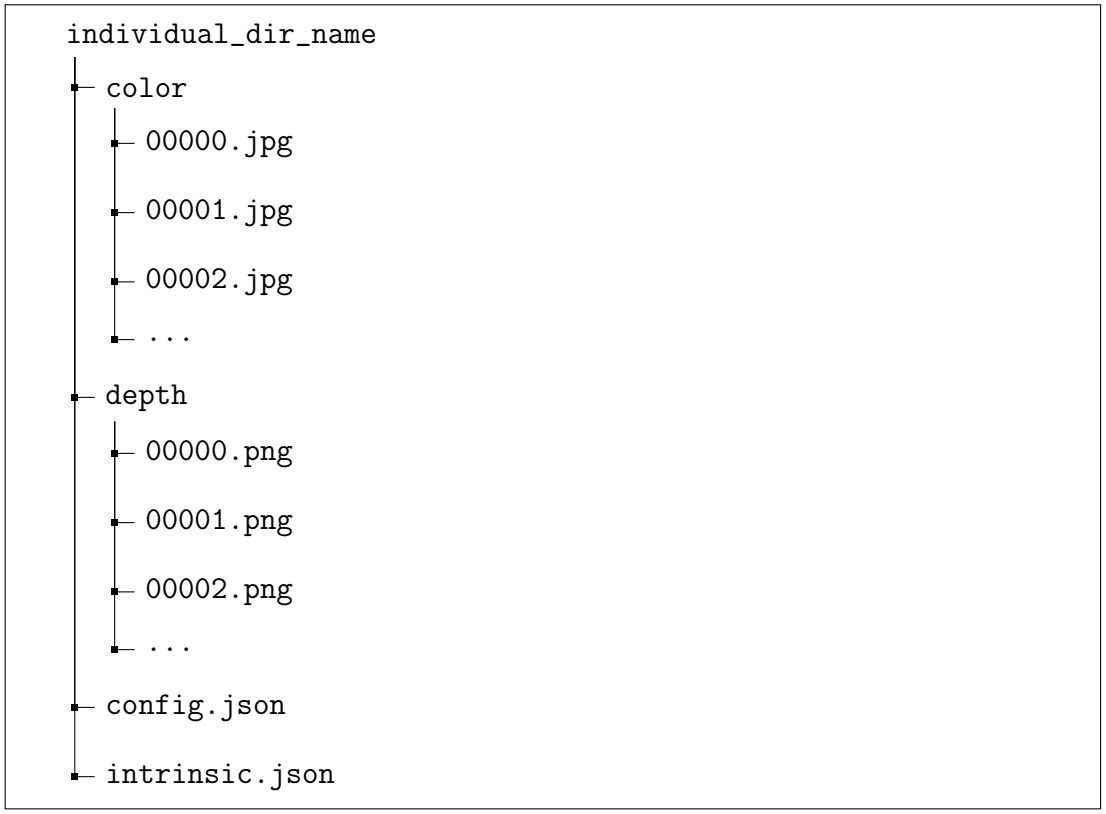

Abb. 3.12: Ordnerstruktur für die Bilder des Videos

Zusätzlich zu den Farb- und Tiefenbildern werden auch eine `config.json`- und eine `intrinsic.json`-Datei erzeugt. Letztere enthält Informationen über den Farbmodus, den Tiefenmodus, die Höhe und Breite der Bilder und die Länge der Aufzeichnung. Außerdem ist eine `intrinsic_matrix` angegeben, welche die intrinsischen Parameter der Kamera enthält, d.h. die Parameter, die die Verschiebungen und Verzerrungen der Bildpunkte ausgleichen, um sie mit dem Modell der Lochkamera (engl. pinhole model) abzubilden. Dieses Modell bildet die Realität optimal in das Bild ab.

Aus den Farb- und Tiefenbildern werden in einem weiteren Schritt die Punktwolken generiert. Dazu werden aus je einem Farb- und einem Tiefenbild mit der gleichen Nummer RGBD-Bilder erzeugt und daraus die Punktwolken generiert (siehe Kapitel 3.1.2). Die generierte Punktwolke muss vor der Speicherung noch transformiert werden, da sie andernfalls kopfüber und von der Rückseite ange-

zeigt wird (siehe Abbildung 3.13). Die Punktwolken werden in einem von den Benutzerinnen oder Benutzern festgelegten Ordner mit gleichermaßen aufsteigender Nummerierung im *.pcd*-Format gespeichert und sind so für die weitere Verarbeitung direkt abrufbar. In diesen Punktwolken findet dann in einem späteren Schritt, unabhängig von der Generierung der Punktwolken, die Annotation statt.

Abb. 3.13: Vergleich der Punktwolken: a) Punktwolke ohne Transformation, b) Punktwolke mit einer Transformation um 180 ° um die x-Achse

Das Video kann außerdem in das Body Tracking SDK, welches auch in [2] verwendet wird, eingegeben werden. Dort werden ebenfalls für jedes einzelne Bild die x-, y- und z-Koordinaten der Gelenkposition im Koordinatensystem der Kamera ermittelt und in einer *JSON*-Datei gespeichert (vgl. [2], S. 4). Im Rahmen dieser Arbeit werden bei einigen Videoaufzeichnungen parallel zusätzlich Aufzeichnungen mit dem Motion-Capture-Anzug von TEA®generiert. Dieser zeichnet die Positionsdaten der Gelenke im Raum auf und gibt ebenfalls die x-, y- und z-Koordinaten der Gelenke zurück. Diesmal allerdings in einer *CSV*-Datei. Die Daten aus dem Microsoft Bodytracking und dem Motion-Capture-Anzug werden für die Evaluation (siehe Kapitel 4) des Annotationstools eingesetzt und müssen später in der Anwendung des Annotationstools nicht generiert und genutzt werden.

Die soeben beschriebenen Schritte sind vor allem Vorverarbeitungsschritte und Schritte zur Überprüfung. Die eigentliche Aufgabe des Tools ist die Annotation der Handgelenke. Hierfür wird der *Visualizer* von Open3D eingesetzt, denn dieser ermöglicht zum einen das Anzeigen von farbigen Punktwolken und zum anderen auch das Zoomen in die Punktwolke hinein oder aus ihr heraus. Dies wäre auch über den *k4aviewer* des Azure Kinect SDK möglich, allerdings bietet dieses nicht die Möglichkeit, die Bilder der Videosequenz einzeln zu betrachten. Es kann ledig-

lich das Video abgespielt werden. Der Open3D-Visualizer ermöglicht das Anzeigen der einzelnen Bilder als Punktwolke (siehe oben). Allerdings müsste so für jede Punktwolke ein neuer Visualizer geöffnet werden. Da dies die Nutzbarkeit beeinträchtigt, wurde die Funktion des Visualizers so erweitert, dass der Visualizer nur einmal geöffnet wird und dann über Tastenbefehle mit *seite hoch* für „vor“ und *seite runter* für „zurück“ durch die einzelnen Punktwolken einer Videosequenz geblättert werden kann. Die jeweilige Punktwolke wird dabei solange angezeigt, bis über die Tastatur ein erneuter Befehl zum Vor- oder Zurückgehen eingegeben wird. Die Anzeige ist also nicht an ein Zeitlimit gebunden. Diese sequenzielle Anzeige der Punktwolken im gleichen Visualizer wird mithilfe einer Liste der Punktwolken und eines Index umgesetzt. Das ist in Abbildung 3.14 anhand des Vorwärtblätterns dargestellt. Die Rückrichtung ist analog umgesetzt. Die Punktwolken stehen anhand ihrer Nummerierung im Dateinamen beginnend bei null in aufsteigender Reihenfolge in der Liste und werden mit dem Index durchlaufen, dessen Nummer mit der Nummer im Dateinamen der Punktwolke übereinstimmt.

```
def skip_forward(self, visualize):
    """Enables the user to skip forward in the list point clouds to see the point clouds in the list of point cloud
    data and to annotate the wrists.
    It raises an IndexError exception if this point cloud is the last one in the list of point cloud data and it is
    visualized at this moment. Because the user shall know the start and the end of the list of point cloud data.

    input:
    * visualizer :: VisualizerWithCallback (the same as self)
    * visualize :: Any: this is the Window of the visualizer where the point clouds are visualized
    """
    try:
        self.idx += 1
        visualize.clear_geometries()
        next_pcd = o3d.io.read_point_cloud(self.pcd_list[self.idx])
        print(self.idx)

        visualize.add_geometry(next_pcd)

        if self.idx in self.list_already_annotated:
            self.preserve_annotation(visualize, self.idx)
    except IndexError:
        print("End of directory is reached.")
        pass
```

Abb. 3.14: Methode `skip_forward()` zum Anzeigen der nächsten Punktwolke

Über Interaktionen mit der Maus können die Nutzerinnen und Nutzer in jeder Punktwolke hinein- und herauszoomen, indem sie das Mausrad drehen, und die Perspektive ändern, indem sie an irgendeiner Position in der Punktwolke die linke Maustaste drücken und anschließend die Maus in die entsprechende Richtung ziehen. Dadurch können sich die Betrachterinnen und Betrachter besser in der

Punktwolke orientieren und eventuell noch andere Elemente in der Punktwolke als Orientierungshilfe nutzen. Diese Funktionalität ist standardmäßig im Visualizer verfügbar.

Ein weiterer Grund für die Verwendung des Open3D Visualizers ist die Möglichkeit, über weitere graphische Elemente wie beispielsweise Kugeln oder Koordinatensysteme Handgelenke in der Punktwolke zu markieren und deren Koordinaten anschließend zu speichern. Das entwickelte Annotationstool verwendet ein Koordinatensystem (siehe Abbildung 3.15), um die Handgelenkspositionen zu markieren, denn es ist besser sichtbar als eine Kugel und kann somit leichter platziert und ausgerichtet werden. Außerdem bietet es Möglichkeiten zur Erweiterung, da mit dem Koordinatensystem nicht nur die Position des Handgelenks im Raum annotiert werden kann, sondern später auch die Orientierung im Raum, indem die y-Achse (in grün dargestellt) immer orthogonal zur Handgelenksoberfläche ausgerichtet wird und damit die Normale des Handgelenks darstellt. Die z-Achse (in blau) sollte dann immer parallel zum Handgelenk ausgerichtet sein und die x-Achse bliebe orthogonal zur y- und z-Achse und würde in Richtung des Daumens zeigen.

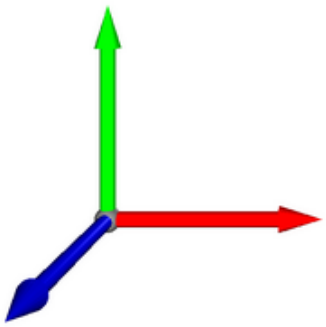

Abb. 3.15: Koordinatensystem zum Annotieren des Handgelenks [Quelle: [31]]

Um den Annotationsprozess zu starten, muss die *Start Annotation*-Schaltfläche im zentralen GUI betätigt werden. Im neuen Fenster ist anschließend der Ordner des Punktwolkendatensatzes anzugeben, der annotiert werden soll, sowie der Pfad zu einer CSV-Datei. Diese dient dazu, die Koordinaten der annotierten Handgelenke zu speichern. Die Nutzenden können hier sowohl eine bereits vorhandene CSV-Datei auswählen als auch einen neuen Dateinamen eingeben. Ist letzteres der Fall, wird nach Betätigen der *Start*-Schaltfläche eine neue CSV-Datei mit dem angegebenen Dateinamen erstellt. Dabei wird auch direkt der *Header* der

CSV-Datei erstellt. In diesem Fall besteht er aus sieben Spaltenüberschriften:

```
Frames, Wrist_Left_X, Wrist_Left_Y, Wrist_Left_Z, Wrist_Right_X,
Wrist_Right_Y, Wrist_Right_Z
```

Das bedeutet, es gibt eine Spalte für die zugehörige Nummer der Punktwolke sowie jeweils drei Spalten für die Koordinaten des rechten und linken Handgelenks. Bevor die *Start*-Schaltfläche angeklickt werden kann, muss ausgewählt werden, welches Gelenk annotiert werden soll. Bisher haben die Nutzenden nur die Auswahl zwischen dem rechten und dem linken Handgelenk. Jedoch wurde dieser Schritt so implementiert, dass weitere Gelenke durch geringe Änderungen am Quellcode hinzugefügt werden können, falls dies erforderlich ist. Es kann dabei immer nur ein Gelenk ausgewählt werden. In diesem Fall heißt das, es kann entweder das rechte Handgelenk in allen Punktwolken annotiert werden oder das linke. Dies erleichtert zum einen den Annotationsprozess, da die Nutzerinnen und Nutzer sich nur auf ein Handgelenk konzentrieren müssen und nicht zwischen den verschiedenen Seiten und später gegebenenfalls auch Gelenken hin und her springen müssen, und zum anderen liegt es in den technischen Beschränkungen durch Open3D begründet. Diese bestehen darin, dass das Koordinatensystem nicht in verschiedenen Farben dargestellt werden kann und auch eine Beschriftung der Koordinatensysteme nicht sinnvoll umzusetzen ist. Dies wäre jedoch notwendig, wenn mehrere Gelenke gleichzeitig annotiert werden sollen, um die Unterscheidbarkeit zu gewährleisten, auch wenn sie nur auf verschiedenen Körperseiten liegen.

Hat die Nutzerin oder der Nutzer auf *Start* geklickt, wird der Open3D-Visualizer geöffnet. Mit dem Visualizer kann ausschließlich über die Tastatur interagiert werden. Die einzige Ausnahme bildet hier die oben beschriebene Interaktion über die Maus für Perspektivwechsel oder Zoom.
Der Visualizer zeigt aus dem ausgewählten Datensatz immer als erstes die Punktwolke mit der Nummer null an. Um in dieser Punktwolke das gewählte Handgelenk zu annotieren, fügt die Nutzerin oder der Nutzer über die *einfg*-Taste ein Koordinatensystem in die Punktwolke im Visualizer ein. Anschließend ist das Koordinatensystem so in der Punktwolke zu verschieben, dass der Ursprung des Koordinatensystems die Mitte des Handgelenks markiert. Das heißt, der Ursprung muss bei einem sichtbaren Handgelenk mittig auf dem *Articulatio radiocarpalis*, dem Gelenkspalt des Handgelenks, und unterhalb des *Os capitatum*, einem Handwurzelknochen mittig über dem Gelenkspalt (siehe 3.16) platziert werden. Ist das Handgelenk teilweise oder vollständig verdeckt, muss die Nutzerin oder der Nutzer versuchen, das Koordinatensystem so nah wie möglich auf diesen Punkt zu

verschieben, indem sie oder er sich an anderen Punkten in der Punktwolke orientiert, wie zum Beispiel wie viel noch vom Unterarm zu sehen ist. Oder sie müssen versuchen, sich an die Anordnung zu erinnern, um die Position möglichst genau markieren zu können. Denn um den Bewegungsverlauf des Handgelenks möglichst genau und vollständig abbilden zu können, muss immer der gleiche Punkt im Handgelenk markiert werden. Ansonsten gäbe es Sprünge in der Trajektorie, die es in der Realität nicht gab.

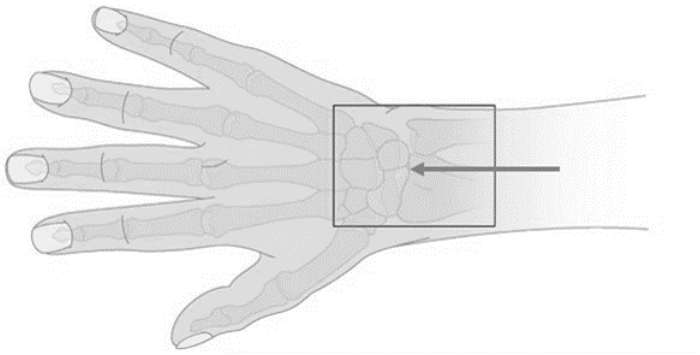

Abb. 3.16: Zu markierender Punkt des Handgelenks [Quelle: modifiziert nach [1]]

Diese Ausrichtung des Koordinatensystems auf das Handgelenk wird über die Tasten *Q, W, E, A, S, D* umgesetzt. Diese Tastenbelegung wurde gewählt, da die *W-A-S-D*-Tasten die Standard-Tastenbelegung zur Interaktion im PC-Gaming ist (vgl.[8]). Dies ist auch bei den Pfeiltasten der Fall, zumal diese außerhalb der Computerspiele etwas intuitiver und die Tasten frei belegbar sind. Doch da bei dem Annotationstool alle drei Richtungen über die Tastenbelegung abgebildet sein müssen und die Tasten *seite hoch* und *seite runter* bereits zum Durchblättern der Punktwolke belegt sind, ist es sinnvoller, die *W-A-S-D*-Tastenbelegung zu nutzen und sie um *Q* und *E* zu erweitern, um auch die dritte Dimension zu ermöglichen.

```
def move_annotation_mesh_upward(self, visualize):
    """Moves the annotation coordinate frame upwards if the user presses the 'w'-key."""
    visualize.remove_geometry(self.annotation_mesh)
    step_upward = float(Decimal(self.annotation_coordinates[1, self.idx]) + Decimal(0.01))
    self.annotation_mesh = self.coordinate_frame.create_coordinate_frame(size=0.1, origin=[
        self.annotation_coordinates[0, self.idx],
        step_upward,
        self.annotation_coordinates[2, self.idx]])
    self.annotation_coordinates[1, self.idx] = step_upward  # y-coordinate

    self.annotation_mesh.transform([[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [0, 0, 0, 1]])
    visualize.add_geometry(self.annotation_mesh)
```

Abb. 3.17: Methode `annotation_upward()` zum Verschieben des Koordinatensystems nach oben

Um das Koordinatensystem nach oben oder unten zu verschieben, d.h. entlang der y-Achse, wird W respektive S gedrückt. Zum Verschieben auf der x-Achse, also nach links oder rechts, stehen die A- respektive D-Taste zur Verfügung. Q und E verschieben das Koordinatensystem entlang der z-Achse, also in den Raum hinein nach hinten oder aus dem Raum nach vorne. In Abbildung 3.17 ist dieser Vorgang exemplarisch für das Verschieben nach oben entlang der y-Achse dargestellt. Um das Koordinatensystem nach jeder Tastatureingabe an der neuen Position anzeigen zu können, muss es zunächst über `remove_geometry()` gelöscht werden. Anschließend wird auf die y-Koordinate 0,01 addiert, da wegen der oben beschriebenen Transformation die positive y-Achse nach oben zeigt. Über das Modul `decimal` wird die Koordinate auf eine Präzision von 12 Nachkommastellen beschränkt, da sonst Rundungsfehler entstehen und das Ergebnis beispielsweise 0.300000000000005e-2 statt 0.03 ist. Die unveränderten x- und z-Koordinatenwerte werden ebenso wie der neu berechnete y-Koordinatenwert als neuer Ursprung für das Koordinatensystem gesetzt. Mit `add_geometry()` wird dieses an der neuen Position wieder in die Punktwolke eingefügt, wobei es zuvor transformiert wird, sodass das Koordinatensystem dem der Tiefenkamera entspricht.

Ist das Koordinatensystem richtig platziert, wird es mit der Eingabetaste bestätigt. Da Open3D in seinem Visualizer ein linkshändiges Koordinatensystem aufspannt, die Azure Kinect Kamera und auch das Bodytracking jedoch mit rechtshändigen Koordinatensystemen arbeiten, müssen die bestätigten Koordinaten jeweils durch die Multiplikation mit -1 gespiegelt und damit vom linkshändigen System in das rechtshändige überführt werden. Anschließend werden die markierten Koordinaten in der zuvor ausgewählten oder angelegten CSV-Datei

gespeichert. Das heißt, für jede Punktwolke wird eine neue Zeile in die Datei geschrieben. Da das rechte und das linke Handgelenk nacheinander annotiert werden und die Koordinaten beider Gelenke in einer Datei gespeichert werden sollen, bedarf es eines bestimmten Schemas zum Speichern der Daten (siehe Algorithmus 1).

Algorithm 1 Speichern in CSV-Datei

```
1: procedure WRITE_POSITION_DATA_TO_CSV

2:     if joint = wrist_right then read csv file
3:         left_position_list ← all lines of csv file

4:         if not left is tracked and idx not in list_already_annotated then
5:             csvfile ← newlinewithframeidx,0,0,0,x_right,y_right,z_right
6:             list_already_annotated ← idx

7:         else if left is tracked or idx in list_already_annotated then
8:             line_to_override ← idx,x_left,y_left,z_left,x_right,y_right,z_right
9:             csvfile ← line_to_overrideatidx + 1
```

Im Algorithmus 1 ist dies exemplarisch für das rechte Handgelenk dargestellt. Wurde das linke Handgelenk noch nicht annotiert, wird die CSV-Datei um eine Zeile erweitert. Hierbei werden die Spalten für das linke Handgelenk mit Nullen gefüllt und nur in die des rechten Handgelenks die bestätigten Werte geschrieben. Um zu ermöglichen, dass bereits bestätigte Markierungen nachträglich korrigiert werden können, wird die Nummer der Punktwolke in eine Liste eingetragen. Soll nun eine korrigierte Annotation gespeichert werden, wird zur Zeile 7 gesprungen. Hier werden die zu überschreibende Zeile und ihr Inhalt festgelegt, indem die Zeile aus der CSV-Datei ausgelesen wird und die Werte an den entsprechenden Stellen ausgetauscht werden. Anschließend wird die Zeile `idx+1` mit dem neuen Inhalt überschrieben. Die Nummer der Punktwolke muss hier um eins erhöht werden, da der Header der CSV-Datei mitzählt.
Wurde das linke Handgelenk bereits in allen Punktwolken annotiert, wird ebenfalls in Zeile 7 gestartet. Die CSV-Datei enthält in diesem Fall die x-, y- und z-Koordinaten des linken Handgelenks und Nullen für die des rechten Handgelenks. Dadurch, dass die CSV-Datei in Zeile 2 Zeile für Zeile ausgelesen wird und jede Zeile in eine Liste eingetragen wird, bleiben die Koordinaten des linken Handgelenks unverändert. Lediglich die Nullen der rechten Hand werden durch die neuen Werte ersetzt. Die Speicherung der Annotationskoordinaten des linken Handgelenks funktioniert analog.

Zusätzlich zur Speicherung in der CSV-Datei wird die Nummer der Punktwolke beim Bestätigen der Annotation in eine Liste der bereits annotierten Punktwolken eingetragen, um diese inklusive der Annotation zu einem späteren Zeitpunkt erneut anzeigen zu können. Dies ist jedoch eine andere Liste als die in Zeile 6, da sie in der Klasse des `Sequential Visualizers` geführt wird, in der die Annotation implementiert ist. Diese beiden Listen müssen getrennt voneinander geführt werden, da andernfalls die `if`-Abfragen in dem Algorithmus 1 nicht funktionieren. Denn würde die Liste aus dem `Sequential Visualizer` in den Algorithmus eingegeben werden und das linke Handgelenk noch nicht annotiert sein, wäre die aktuelle Punktwolke als bereits annotierte Punktwolke vermerkt und die Bedingung in Zeile 4 wäre nicht erfüllt, obwohl die Punktwolke gerade erst annotiert wird.

Durch das Speichern der Koordinaten nach jeder Annotation ist gewährleistet, dass die Koordinaten nicht verloren gehen, sollte das System abstürzen oder die Nutzerin oder der Nutzer den Visualizer schließen. Somit können die Nutzenden die Annotation jederzeit unterbrechen, ohne Gefahr zu laufen, alle bereits annotierten Punktwolken erneut annotieren zu müssen.
Eine andere zuvor angedachte Möglichkeit wäre, alle Koordinaten in eine $(3 \times \mathrm{X})$-Matrix zu schreiben, mit X Anzahl der Punktwolken in dem Datensatz, die beim Öffnen des Visualizers erstellt wird. Hier würden dann nach und nach alle vorhandenen Nullen durch die entsprechenden Koordinaten ersetzt werden. Allerdings müssten hier folgende Fragen geklärt werden:

1. Worin wird die Matrix gespeichert?
2. Wann wird die Matrix am sinnvollsten gespeichert?
3. Soll jede Änderung gespeichert werden oder erst, wenn alle Punktwolken annotiert sind? Oder dann, wenn der Nutzer oder die Nutzerin die Annotation abbricht?

Hinsichtlich der ersten Frage gibt es viele verschiedene Möglichkeiten, allerdings müsste die Matrix für jede der Möglichkeiten vor dem Speichern im gewählten Format so geändert werden, dass sie in diesem Format gespeichert werden kann. Schreibt man die Koordinaten direkt in eine CSV-Datei, fällt dieser Schritt weg und somit auch eine Fehlerquelle, in der die Daten durch die Umformungen beschädigt werden können. Ein weiterer Grund für die Wahl von CSV-Dateien ist,

dass sie ein gängiges Format für diese Art von Daten sind[3] und dass sie mit Excel etc. leicht einzulesen sind. Über Python lässt sich außerdem gut in CSV-Dateien schreiben und gleiche Dateien auch wieder einlesen, was das Speichern erleichtert und auch die Wiederverwendbarkeit dieser Dateien in Python erhöht.
Beim Beantworten der zweiten bzw. dritten Fragen ist festzustellen, dass die zweite Möglichkeit, bei der gespeichert wird, wenn alle Punktwolken annotiert sind, nicht sinnvoll ist. Denn hierbei sind die Nutzenden gezwungen, alle Punktwolken zu annotieren, ohne den Prozess unterbrechen zu können. Auch wenn das System abstürzen sollte, wären alle Daten der bereits durchgeführten Annotationen verloren und die Nutzenden müssten von vorn beginnen. Die dritte Möglichkeit ist nicht umsetzbar, da in diesem Fall abzufragen wäre, wann der Visualizer geschlossen wird. Diese Abfrage stellt Open3D in der Python-API nicht zur Verfügung (vgl. [20]). Aus diesen Gründen werden der Matrix-Ansatz und der Ansatz, dass direkt in die CSV-Datei geschrieben wird, kombiniert. Eine derartige Matrix wird während der Initialisierung des Visualizers erstellt, allerdings dient sie nur zum Zwischenspeichern der Koordinaten: Zum einen um damit die Berechnungen für das Verschieben der Koordinatensysteme zu implementieren, denn zu diesem Zeitpunkt stehen die Koordinaten noch nicht in der CSV-Datei, und zum anderen um bereits eingefügte Koordinaten abzurufen, etwa wenn die Anwenderinnen oder Anwender zurückblättern. Hierzu wird abgefragt, ob diese Punktwolke in der Liste der bereits annotierten Punktwolken eingetragen ist. Ist dies der Fall, wird innerhalb des sequenziellen Visualizers `preserve_annotation()` aufgerufen und der aktuelle Index übergeben (siehe Abbildung 3.18). Innerhalb dieser Methode wird eine weitere Methode aufgerufen, die anhand dieses Indexes in der CSV-Datei die Zeile mit der gleichen Nummer sucht und die Werte der drei Koordinaten in ein Array schreibt, welches zurückgegeben und dann als Ursprung des Koordinatensystems gesetzt wird. Dieses wird transformiert in der entsprechenden Punktwolke wieder eingefügt.

[3] Beispielsweise nutzt Captiv diesen Dateityp unter anderem ebenfalls zur Speicherung der aufgezeichneten Koordinatendaten.

```
def preserve_annotation(self, visualize, idx):
    """Visualizes the already inserted coordinate frames if you skip back to this point cloud."""
    self.annotation_mesh = self.coordinate_frame.create_coordinate_frame(size=0.1,
                                                                          origin=[self.annotation_coordinates[0, self.idx],
                                                                                  self.annotation_coordinates[1, self.idx],
                                                                                  self.annotation_coordinates[2, self.idx]
                                                                                  ])

    self.annotation_mesh.transform([[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0], [0, 0, 0, 1]])
    visualize.add_geometry(self.annotation_mesh)
```

Abb. 3.18: Methode `preserve_annotation()` aus *sequential_visualizer.py*

Die Koordinatensysteme werden nur während des Annotationsprozesses eines Handgelenks in allen annotierten Punktwolken angezeigt, d.h. eine annotierte Punktwolke wird mit dem eingefügten Koordinatensystem nicht neu gespeichert. Es wird lediglich die Position in der CSV-Datei gespeichert. Dies ermöglicht, dass der Annotationsprozess für verschiedene Gelenke nacheinander in denselben Punktwolken durchgeführt werden kann, ohne dass die während einer anderen Annotation markierten Gelenke angezeigt werden. Da die Farben des Koordinatensystems nicht geändert werden können, würde das Mitführen der vorherigen Koordinatensysteme zu Unklarheiten führen, welches der sichtbaren Koordinatensysteme das aktuelle ist.
Da die Methode `preserve_annotation()` (siehe Abbildung 3.18), die die Koordinatensysteme in bereits annotierten Punktwolken anzeigt, nur in den Fällen aufgerufen wird, in denen die Punktwolke in der Liste der annotierten Punktwolken aufgeführt ist und diese Liste zu Beginn eines Annotationsprozesses[4] neu erstellt wird, ist gewährleistet, dass die Punktwolken in einem neu begonnenen Annotationsprozess keine Annotationskoordinatensysteme enthalten.

Ist in einer Punktwolke das Handgelenk annotiert und die Position gespeichert, kann die Nutzerin oder der Nutzer über die *seite hoch*-Taste die nächste Punktwolke anzeigen lassen und dort den Vorgang vom Einfügen des Koordinatensystems über das Verschieben desselben bis hin zur Speicherung der Position wiederholen. Dieses gesamte Verfahren wird solange wiederholt, bis alle Punktwolken eines Datensatzes annotiert sind oder die Nutzerin oder der Nutzer entscheidet, dass alle relevanten Punktwolken annotiert sind.

Ruft die Nutzerin oder der Nutzer die nächste Punktwolke auf, wird das Koordinatensystem in dieser Punktwolke immer an der Position eingefügt, an der es

[4] wenn der `Sequential Visualizer` für ein neues Gelenk geöffnet wird

bei der vorherigen gespeichert wurde. Dies hat den Vorteil, dass sich der Nutzer oder die Nutzerin an der vorherigen Position orientieren kann und dadurch möglicherweise besser die Position des Handgelenks in dieser Punktwolke markieren kann. Besonders wenn das Handgelenk in der Punktwolke zuvor sichtbar war und nun unsichtbar ist, kann dies einen wichtigen Hinweis auf die Position desselben in dieser Punktwolke geben und als weiteres Orientierungsmerkmal genutzt werden. Die Nutzerin oder der Nutzer muss das Koordinatensystem dann in den meisten Fällen auch nicht so weit verschieben. Wurde eine sehr schnelle Bewegung ausgeführt, dann können sich die Positionen in den beiden Punktwolken jedoch stark unterscheiden. In dem Fall wäre die vorherige Position auch kein Hinweis mehr auf die Position des Handgelenks in dieser Punktwolke. Aber bei pflegerischen Tätigkeiten werden die meisten Bewegungen nicht so schnell ausgeführt und die Abtastrate von 30 Hz ist ebenfalls so hoch, dass dieser Effekt verringert wird. Das Bereitstellen dieser Funktion ist ein weiterer Grund für die Kombination von Matrix und CSV-Datei zum Speichern der Koordinaten. Denn die Position des Koordinatensystems in der vorherigen Punktwolke kann nun aus der Spalte mit der Nummer `idx-1` ausgelesen werden und als initiale Position für das Koordinatensystem in dieser Punktwolke (`idx`) verwendet werden (siehe Abbildung 3.19)

```python
def insert_mesh(self, visualize):
    """Inserts and Visualizes the coordinate frame in the same visualizer as the point cloud to annotate
    the wrist."""
    if self.idx == 0:
        self.annotation_coordinates[0, self.idx] = 0     # x-coordinate
        self.annotation_coordinates[1, self.idx] = 0     # y-coordinate
        self.annotation_coordinates[2, self.idx] = 0     # z-coordinate

    # to insert the coordinate frame at the position where it was saved in the previous point cloud
    else:
        self.annotation_coordinates[0, self.idx] = self.annotation_coordinates[0, self.idx - 1]   # x-coordinate
        self.annotation_coordinates[1, self.idx] = self.annotation_coordinates[1, self.idx - 1]   # y-coordinate
        self.annotation_coordinates[2, self.idx] = self.annotation_coordinates[2, self.idx - 1]   # z-coordinate

    visualize.add_geometry(self.annotation_mesh)
```

Abb. 3.19: Methode `insert_mesh()` aus *sequential_visualizer.py*

Da bereits bei einem Video von fünf Sekunden mit einer Frequenz von 30 Hz 150 Punktwolken entstehen, ist der Annotationsprozess aufwändig. Eine Erleichterung ist bereits über das Einfügen des Koordinatensystems an der Stelle, an der es in der vorherigen Punktwolke gespeichert wurde, gegeben. Jedoch müssen auch bei diesem Verfahren alle Punktwolken einzeln annotiert werden. Um das zu vermeiden, wird die lineare Interpolation eingesetzt. Dadurch können Punktwolken

hinsichtlich der Annotation übersprungen werden. Die Nutzerinnen oder Nutzer annotieren dazu in einer Punktwolke das Handgelenk nach dem oben beschriebenen Vorgehen. Anschließend blättern sie einige Punktwolken weiter, ohne in ihnen zu annotieren. Dabei sollte nur so weit vorgeblättert werden, dass es keine großen Bewegungen des Handgelenkes gibt, denn aufgrund der Nutzung einer linearen Interpolation wird von äquidistanten Punkten ausgegangen. Diese äquidistanten Punkte werden hier durch die Zeit repräsentiert und durch Punktwolkennummern realisiert. Die Abstände der Handgelenkspositionen in den verschiedenen Punktwolken sind hingegen nicht unbedingt äquidistant. Wenn in dem Interpolationsintervall sowohl langsame als auch schnelle Bewegungen liegen, ändern sich die Positionen des Handgelenks mit unterschiedlichen Geschwindigkeiten. Das heißt, es werden unterschiedliche Strecken in der gleichen Zeit zurückgelegt. Dadurch weichen die interpolierten Positionen von der Realität ab und die Positionen der Handgelenke werden in den betreffenden Punktwolken nicht richtig annotiert.

Ist eine geeignete Punktwolke für den Endpunkt des Interpolationsintervalls gefunden, wird in dieser das Handgelenk annotiert. Diesmal wird die Position jedoch nicht über die Eingabetaste bestätigt, sondern mit der Taste *i*, welche die Methode `interpolate_position_data()` aufruft. Da nicht vorgegeben ist, wie groß das Interpolationsintervall ist, wird hier zunächst über die Liste der bereits annotierten Punktwolken bestimmt, welche Punktwolke zuletzt annotiert wurde. Die Anzahl der Stützstellen wird anschließend aus der Differenz zwischen der zuletzt annotierten und der aktuellen Punktwolke abgeleitet. Zur Interpolation wird die Funktion `interp1d` von *SciPy* verwendet, welche die äquidistanten Zeitpunkte als x-Werte und die äquidistanten Koordinaten als y-Werte erhält. Da die x-, y- und z-Koordinaten der Positionen voneinander unabhängig sind, wird `interp1d` für jede Koordinate getrennt aufgerufen und es wird jeweils eine Funktion zurückgegeben. Über eine `for`-Schleife werden dann für jede Stützstelle respektive Punktwolke die drei Koordinaten interpoliert und ebenfalls direkt in die CSV-Datei geschrieben. Dadurch ist die CSV-Datei zu jedem Zeitpunkt vollständig und in einem Datensatz können sowohl die Interpolation als auch das manuelle Annotieren angewandt werden.

Blättert die Nutzerin oder der Nutzer nach der Interpolation zurück, sind die Handgelenke in den übersprungenen Punktwolken ebenfalls annotiert und sichtbar. Die (interpolierte) Annotation kann dadurch überprüft und gegebenenfalls korrigiert werden. Für die Korrektur wird das Koordinatensystem wie üblich an

die entsprechende Stelle verschoben und diesmal wieder mit der Eingabetaste bestätigt. In der CSV-Datei wird daraufhin die betreffende Zeile herausgesucht und überschrieben.

Als Ergebnis des Annotationsprozesses erhält die Nutzerin oder der Nutzer die Koordinatendaten in der CSV-Datei, die dann weiteren Nachverarbeitungsschritten unterzogen werden können. Dies ist allerdings nicht Teil des Annotationstools, da es sehr viele verschiedene Anwendungsbereiche für diese Daten gibt und damit auch viele verschiedene Verfahren, die Daten auf geeignete Weise aufzubereiten.

3.4 Grenzen der Umsetzung

Während der Umsetzung traten durch die verwendeten Bibliotheken und Geräte immer wieder Einschränkungen auf. Auch das entwickelte manuelle Annotationstool stößt hinsichtlich der in Kapitel 2.3 ermittelten Anforderungen an Grenzen. Beispielsweise wurde gefordert, dass es möglich sein muss, mehrere Handgelenke gleichzeitig zu annotieren. Da das zum Annotieren verwendete Koordinatensystem nicht verschiedenen gefärbt oder beschriftet werden kann, ist es nicht sinnvoll, gleichzeitig mehrere Koordinatensysteme für verschiedene Handgelenke in eine Punktwolke einzuzeichnen. Eine weitere Schwierigkeit bei diesem Ansatz wäre die Zuordnung von Markierungen zu den jeweiligen Handgelenken, die in dem Fall entweder über weitere Tastatureingaben oder über eine feste Reihenfolge umgesetzt werden müssten, um die verschiedenen Handgelenke annotieren zu können. Darüber hinaus wäre damit die Erweiterbarkeit beeinträchtigt, da für jedes weitere Gelenk, das annotiert werden soll, neue Tastatureingaben und die dazugehörigen Methoden implementiert werden müssten und die Annotation sehr unübersichtlich werden würde. Beide Methoden zum gleichzeitigen Annotieren tragen nicht zur Bedienbarkeit und Nutzbarkeit des Annotationstools bei. Aus diesem Grund wurde der Ansatz gewählt, dass die Gelenke jeweils einzeln in allen Punktwolken annotiert werden, d.h. zu Beginn wird das zu annotierende Handgelenk festgelegt und dann in allen Punktwolken annotiert. Ist dies abgeschlossen, kann das nächste Gelenk gewählt und ebenfalls in allen Punktwolken annotiert werden. Somit ist das gleichzeitige Annotieren von mehreren Handgelenke nicht gegeben, jedoch liefert der Ansatz des sequenziellen, getrennten Annotierens das gleiche Ergebnis, und die Zuordnung der Positionen zu den Gelenken ist einfacher. Außerdem bleiben die Punktwolken übersichtlicher und die Nutzerinnen und Nutzer müssen sich nur auf ein Gelenk konzentrieren.

Die Nutzbarkeit und die Anwendbarkeit dieses Tools kann noch verbessert werden, indem das Koordinatensystem zur Annotation nicht nur über die Tastatureingaben verschoben werden kann, was bei weiten Strecken sehr mühsam ist, sondern sich auch über die Maus mittels Klicken und Drag-and-drop platzieren lässt. Dies ist jedoch über den Visualizer von Open3D derzeit nicht möglich. Auch die auf 0,01 festgelegte Schrittweite beim Verschieben des Koordinatensystems währen der Annotation ist in manchen Fällen nicht passend, was über eine Mausinteraktion ausgeglichen werden könnte.
Eine weitere Einschränkung in der Nutzbarkeit ist, dass beim Öffnen des Visualizers das GUI zum Starten des Annotationsprozesses sichtbar bleibt. Open3D stellt für Python derzeit keine Funktion zur Verfügung, die die Abfrage ermöglicht, ob der Visualizer geschlossen wurde, um das GUI zum Starten des Annotationsprozesses wieder zu öffnen (vgl. [20]).

Die Koordinatensysteme zur Annotation werden nicht in den Punktwolken, sondern ausschließlich in einer CSV-Datei gespeichert, um einen weiteren Annotationsprozess für ein neues Gelenk in den Punktwolken durchführen zu können (siehe Kapitel 3.3). Wurde der Annotationsprozess jedoch unterbrochen, in dem das Annotationstool geschlossen wurde, und zu einem späteren Zeitpunkt für das gleiche Gelenk wieder weitergeführt, werden in den zuvor bereits annotierten Punktwolken ebenfalls keine Koordinatensysteme angezeigt, da auch in diesem Fall die Liste der bereits annotierten Punktwolken neu erstellt wird und zu Beginn leer ist. Dadurch wird `preserve_annotation()` für die vor der Unterbrechung annotierten Punktwolken nicht aufgerufen. Die Nutzerin oder der Nutzer kann in diesem Fall nicht erkennen, welche Punktwolken vor der Unterbrechung bereits annotiert wurden. Dies kann jedoch in der CSV-Datei nachgeschaut werden.

Die Nutzerinnen und Nutzer erhalten derzeit keine Informationen über den Index der angezeigten Punktwolke. Auch eine Rückmeldung, das die Position des Koordinatensystems gespeichert wurde, konnte aus zeitlichen Gründen nicht umgesetzt werden. Die Informationen zur angezeigten Punktwolke und der Speicherung der Koordinatenwerte können nur indirekt über die CSV-Datei herausgefunden werden. Hier können die Nutzerinnen und Nutzer überprüfen, ob die Koordinatenwerte für alle vorherigen Punktwolken gespeichert sind und in welcher Punktwolke sie zuletzt annotiert haben.

Bei der Annotation ist zu beachten, dass in einer CSV-Datei immer gleich viele Einträge für das rechte und das linke Handgelenk vorhanden sind. Das liegt darin begündet, dass, wenn bereits ein Handgelenk annotiert wurde, jedes Mal die gesamte Zeile neu geschrieben wird. Die zuvor für ein Handgelenk gespeicherten Werte werden dabei, wie in Kapitel 3.3 beschrieben, zuvor ausgelesen und anschließend zusammen mit den Werten des neuen Handgelenks in die gleiche Zeile zurückgeschrieben. Dieser Mechanismus wird immer umgesetzt, sobald eines der Handgelenke mit mindestens einem Wert annotiert wurde und für das zweite Handgelenk die Annotation begonnen wurde. Es kann zu einem Indexfehler kommen, sollte man die Anzahl der Annotationen des bereits annotierten Handgelenks überschreiten. Es ist daher sinnvoll, ein Handgelenk in allen Punktwolken zu annotieren, bevor das zweite Handgelenk annotiert wird.

4 Evaluierung

Die Evaluierung des Annotationstools wird mit zwei voneinander unabhängigen Systemen durchgeführt: zum einen mit dem in [2] verwendeten Body Tracking SDK und zum anderen über den Captiv. Dazu wurden zunächst mit der Azure Kinect Videos aufgenommen und entsprechend der in Kapitel 3.3 beschriebenen Vorgehensweise annotiert. Bei zwei Videos wurde gleichzeitig der Captiv getragen und Daten über die Captiv-L7000-Software aufgezeichnet. Für die Daten des Body Tracking SDK wurden die entsprechenden MKV-Dateien in die Bodytracking-Software eingegeben und die Positionsdaten für das rechte und linke Handgelenk extrahiert.

Für die Evaluierung wurden verschiedene Videos für die Systeme verwendet. Die Aufstellung der Kamera war in beiden Videos ähnlich: Die Azure Kinect stand jeweils in zwei Metern Entfernung genau gegenüber der zu beobachtenden Person. Dazu wurde ein Lot über die Kameraoberfläche gelegt und an dessen Auftreffpunkt am Boden ein zwei Meter langer Zollstock angelegt. Die Person stand mit den Zehenspitzen am anderen Ende des Zollstocks. Die Kamera war in 1,145 m Höhe auf einem Stativ befestigt. Für die Aufnahmen, die zusammen mit dem Captiv aufgezeichnet wurden, war die Kamera in Richtung Süden ausgerichtet, für die Aufnahmen für das Body Tracking SDK war die Kamera in 77 ° Ost ausgerichtet[1].

4.1 Evaluierung mit dem Body Tracking SDK

Für die Vergleich mit den Daten des Body Tracking SDK wurden vier bis fünf Sekunden lange Videos aufgezeichnet. In der Dokumentation zum Body Tracking SDK wurde in den Abbildungen zur Darstellung der verwendeten Koordinatensysteme und der Skeletthierarchie eine Person im hüftbreiten Stand mit auf Schulterhöhe zu den Seiten ausgestreckten Armen gezeigt (siehe Abbildung 3.1). Deswegen wurde angenommen, dass dies eine Art Kalibrierungsgeste ist, die die Schätzungen

[1] Die Ausrichtungen wurden mit Hilfe eines Kompasses im Handy vorgenommen.

des Body Tracking SDK verbessert. In der Folge wurden zwei Datensätze annotiert, je einer mit bzw. ohne Kalibrierungsgeste zu Beginn. In beiden Aufzeichnungen wurde der gleiche Bewegungsablauf ausgeführt. Beide Videos zeichnen eine Person auf, die ihr rechtes Handgelenk unter dem einen Ende eines Kissens platziert, das auf einem Tisch liegt. Das Handgelenk ist dadurch vollständig verdeckt. Im Bewegungsablauf richtet die Person dieses Kissenende derart auf, dass es einen Winkel von ca. 80 ° zur Tischplatte einnimmt. Die Hand ist hier wieder sichtbar, sie verdeckt jedoch das Handgelenk. In dieser Position hält die Person kurz inne, bevor sie das Kissen wieder ablegt. Das Ablegen des Kissens ist nicht vollständig in dem Video aufgezeichnet, da es außerhalb der 5 Sekunden Aufzeichnungsdauer liegt. Dieser Bewegungsablauf mit Kalibrierungsgeste ist als Aufzeichnung in der Datei *recording10* gespeichert und in Abbildung A.1 im Anhang A.1 einzusehen. Der Bewegungsablauf ohne Kalibierungsgeste ist als Video in der Datei *recording9* gespeichert und in Abbilung A.5 im Anhang A.1 dargestellt. In beiden Diagrammen wurde x über y (jeweils in Meter) in Abhängigkeit von der Zeit, angegeben durch die Nummern der Punktwolken und daher ohne Einheit, aufgetragen. Die Achsen wurden in beiden Diagrammen neu angeordnet, sodass der Bewegungsverlauf visuell nachvollziehbar dargestellt ist. Da beide Videodateien eine Dauer von fünf Sekunden haben, wurden jeweils 151 Punktwolken annotiert. Das Body Tracking SDK lieferte jedoch einmal 152 und einmal 153 Werte. Um diese Daten mit der manuellen Annotation vergleichen zu können, wurden nur die Werte von 0 bis 150 betrachtet. In allen Punktwolken wurde das rechte Handgelenk annotiert.

Für beide Videos wurden die Koordinatenwerte der manuellen Annotation und des Body Tracking SDK gegenübergestellt, um die Ähnlichkeit des Kurvenverlaufs zu bewerten. Hierzu wurde für alle drei Koordinaten jeweils die absolute Distanz ermittelt, indem jeweils die x-Koordinate der manuellen Annotation und die des Body Tracking SDK des gleichen Zeitpunktes in die Formel

$$d(x_{manu},x_{bt}) = \sqrt{(x_{manu} - x_{bt})^2}$$

eingesetzt wurden. Mit den y- und z-Koordinaten wurde genauso verfahren. Weiterhin wurde jeweils der Mittelwert der absoluten Distanz berechnet, welcher die Annäherung an den konventionell richtigen Wert angibt. Um Aussagen über die Güte dieser Annäherung und damit auch über die Sicherheit der Auswertung treffen zu können, wird die Standardabweichung der absoluten Distanz jeweils für x, y und z ermittelt. Die Werte für die x-, y- und z-Koordinate wurden jeweils in einem eigenen Diagramm über der Zeit aufgetragen. Auf der vertikalen Achse

sind die Werte, die x, y und z jeweils annehmen, in Meter angegeben, während die Zeit auf der horizontalen Achse über die Nummer der zugehörigen Punktwolke angegeben wird. Sie bleibt daher ohne Einheit. Die Werte der Koordinaten des Body Tracking SDK sind als Punkte in Orange dargestellt, die der manuellen Annotation als blaue und die der absoluten Distanz als graue Punkte. Der Mittelwert ist als Linie in Gelb dargestellt.

In den Abbildungen A.2 bis A.4 sind die Koordinaten der Handgelenkspositionen des Videos aus *recording10* dargestellt, welches mit der Kalibierungsgeste beginnt. Abbildung A.2 stellt die x-Werte der Positionen des rechten Handgelenks über der Zeit dar, d.h. die Bewegungen des Handgelenks nach rechts und links. Der Nullpunkt entspricht dem Koordinatenursprung des Koordinatensystems der Azure Kinect, welches in der Mitte der Kameralinse liegt und dessen positive x-Achse nach rechts zeigt (siehe Abbildung 3.3). Dieser Punkt liegt durch die Ausrichtung der Kamera mittig auf dem Brustkorb der beobachteten Person. Da die Arme zu Beginn auf Schulterhöhe zu den Seiten ausgebreitet sind, beginnt der Positionsverlauf des rechten Handgelenks im Negativen. Durch das Absenken der rechten Hand aus dieser Kalibierungsgeste bewegt sich die Hand im Koordinatensystem der Kamera nach rechts und nähert sich somit dem Nullpunkt an. Da die Kalibierungsgeste schnell aufgelöst wurde, nähern sich die x-Werte schnell dem Nullpunkt an. Ab Punktwolke 18 flacht der Verlauf ab, da die Hand beim Platzieren unter dem Kissen nicht in der x-Richtung bewegt wird. Erst während des Aufrichtens des Kissens nähern sich die Werte weiter der Null an, jedoch in einer deutlich geringeren Steigung als zu Beginn, da diese Bewegung langsamer ausgeführt wurde. x nimmt nie den Wert 0 an, da das Kissen nicht vollständig aufgerichtet wird und das rechte Handgelenk nicht bis zur Mitte des Brustkorbs geführt wird. Im Diagramm A.2 ähneln sich die Verläufe der x-Werte der Positionen der manuellen Annotation und des Body Tracking SDK. Während die manuelle Annotation einen „glatten“ Verlauf erzeugt, liefert das Body Tracking SDK einen deutlich unruhigeren Verlauf. Ein Verspringen der Werte ist vor allem für die Punktwolken zu erkennen, in denen das Handgelenk durch das Kissen verdeckt ist. Das Body Tracking SDK kann in diesen Fällen die Position des Handgelenks nicht mehr richtig schätzen, da auch das übergeordnete Ellenbogengelenk nicht sichtbar ist. Der Effekt zeigt sich ab Punktwolke 38, ab der das Handgelenk nie vollständig sichtbar ist. Die absolute Distanz, als Maß der Ähnlichkeit der Verläufe, variiert für die x-Werte mit einer Standardabweichung von 2,225 cm um die durchschnittliche Distanz von 2,332 cm.

In der Abbildung A.3 ist der Verlauf der Positionen des rechten Handgelenks entlang der y-Achse zu sehen, d.h. die Aufwärts- und Abwärtsbewegungen. Auch hier beginnt der Verlauf im Negativen und steigt relativ schnell ins Positive, da die Handgelenke aus der Kalibierungsgeste nach unten geführt wurden[2]. Während des Platzierens des rechten Handgelenks unter dem Kissen findet kaum eine Änderung nach oben oder unten statt, sodass der Verlauf der manuellen Annotation zwischen der 41. und der 76. Punktwolke nahezu parallel zur Abszisse verläuft. Das Body Tracking SDK liefert hier einen nach unten abweichenden Verlauf, was wieder auf die vollständige Verdeckung des Handgelenks und des Ellenbogens als übergeordnetem Gelenk zurückzuführen ist.
Während des Aufrichtens des Kissens nähern sich die y-Werte wieder der Abszisse an, da das Handgelenk nach oben, also in die entgegengesetzte Richtung der positiven y-Achse des Kamerakoordinatensystems, bewegt wird. Hier sind die Verläufe der manuellen Annotation und des Body Tracking SDK sehr ähnlich. Dies kann auch an der absoluten Distanz zwischen der 79. und 123. Punktwolke abgelesen werden, die annähernd parallel zum Mittelwert verläuft.
In dem darauffolgenden Abschnitt wird in der aufgerichteten Position innegehalten, sodass sich keine Veränderung entlang der y-Achse ergibt. Das Body Tracking SDK liefert hier, ähnlich dem Verlauf in der Startposition vor dem Aufrichten des Kissens, jedoch einen deutlichen Sprung der Handgelenksposition nach oben (Punktwolke 125 bis 128) und auch wieder nach unten (Punktwolke 142 bis 147). Auch diese Abweichung ist auf die Verdeckung des Handgelenks durch die Hand und die Verdeckung des Ellenbogens durch den Unterarm zurückzuführen.
Insgesamt ist der Verlauf der manuellen Annotation im Vergleich zum Body Tracking SDK „glatter“. Der Mittelwert für die absoulte Distanz ist mit 18,981 cm relativ hoch, die absolute Distanz variiert aber mit einer Standardabweichung von 3,417 cm in einer ähnlichen Größenordnung wie bei der x-Koordinate.
Die Verläufe an sich sind bis auf die Ausreißer sehr ähnlich, jedoch um durchschnittlich 18,981 cm entlang der y-Achse verschoben. Die Ursache für die Verschiebung ist unklar. In [2] ist auf Seite 4 beschrieben, dass die Positionen transformiert wurden, sodass sie in ein gemeinsames Koordinatensystem für alle drei dort verwendeten Kameras passen. Inwieweit das die Ursache für die Verschiebung des Referenzkoordinatensystems des Body Tracking SDK gegenüber der Azure Kinect nach oben ist, kann auf der vorliegenden Datengrundlage nicht beurteilt werden.

[2]Die positive y-Achse des Koordinatensystems der Kamera zeigt nach unten.

Der Verlauf der Positionen des rechten Handgelenks entlang der z-Achse ist in Abbildung A.4 dargestellt, d.h. die Bewegungen des Handgelenks in Richtung der Kamera und von der Kamera weg. Der Verlauf der Werte der manuellen Annotation ist gegenüber dem der Werte des Body Tracking SDK „glatter“ — wie auch in den beiden anderen Abbildungen. Von Punktwolke 0 bis Punktwolke 29 sind die Verläufe nahezu gleich, ab Punktwolke 30 weicht das Body Tracking SDK immer wieder deutlich zwischen 4 cm und 15,8 cm von den Werten der manuellen Annotation nach oben und unten ab. Dies ist ebenfalls auf die Verdeckung von Handgelenk und Ellenbogen in den betreffenden Punktwolken zurückzuführen. Da das rechte Handgelenk bei diesem Bewegungsablauf kaum nach vorne oder nach hinten bewegt wird, verlaufen die Werte der manuellen Annotation und die des Body Tracking SDK annähernd parallel zur Abszisse. Lediglich beim Platzieren des rechten Handgelenks unter dem Kissen in den Punkwolken 40 bis 49 und zu Beginn des Aufrichtens in den Punktwolken 77 bis 84 wird die Hand in Richtung Kamera bewegt, was sich in Abbildung A.4 durch den jeweils leicht abfallenden Verlauf abbildet, denn die positive z-Achse der Kamera zeigt in Richtung der beobachteten Szene. Die Werte der manuellen Annotation weichen gegenüber denen des Body Tracking SDK im Mittel um 5,814 cm voneinander ab. Diese Distanz variiert mit einer Standardabweichung von 4,96 cm.

In Abbildung A.6 sind die Bewegungen des rechten Handgelenks ohne vorherige Kalibrierungsgeste entlang der x-Achse aus dem Video aus *recording9* dargestellt. Da hier das Kissen auch wieder abgelegt wird, ist der Verlauf am Ende fallend. Die Werte des Body Tracking SDK streuen jedoch etwas stärker als die Werte aus dem Video mit der Kalibierungsgeste. Da die absoluten Distanzen zwischen den Werten der manuellen Annotation und den Werten des Body Tracking SDK teilweise mit bis zu 14,23 cm groß sind, ist auch der Mittelwert mit 4,176 cm fast doppelt so groß. Die Standardabweichung beträgt hier 2,781 cm, ist also etwas größer als die des Videos mit der Kalibrierungsgeste.

Der in Abbildung A.7 dargestellte Verlauf der y-Positionen des Handgelenks ähnelt dem in Abbildung A.3 stark, denn in beiden Videos wurde die Bewegung entlang der y-Achse, d.h. auf- und abwärts, durch die Maße des Kissens annähernd gleich ausgeführt. Die Verschiebung des gesamten Kurvenverlaufs der Daten der manuellen Annotation gegenüber denen des Body Tracking SDK entlang der y-Achse ist auch in diesem Diagramm mit durchschnittlich 19,475 cm vorhanden. Die Standardabweichung ist mit 4,313 cm etwas höher als die Standardabweichung

der y-Werte des Videos mit der Kalibrierungsgeste.
Auch die Bewegungen entlang der z-Achse in Abbildung A.8, d.h. auf die Kamera zu und von ihr weg, sind in beiden Videos sehr ähnlich. Jedoch gibt es auch hier Abweichungen zwischen den Werten der manuellen Annotation und denen des Body Tracking SDK von bis zu 18,5 cm. Es gibt aber auch Abschnitte, in denen die manuelle Annotation und das Body Tracking SDK annähernd gleiche Werte liefern. Die absolute Distanz beträgt im Mittel 5,343 cm, die Standardabweichung 4,506 cm.

Das manuelle Annotationstool liefert nach dieser Auswertung zuverlässige Werte für die Positionen der Handgelenke, da das Body Tracking SDK sichtbare Gelenke sehr gut schätzt und in den Bildern, in denen das Handgelenk zu sehen ist, die Werte der beiden Systeme sehr nah beieinander liegen oder in einigen Punkten sogar annähernd gleich sind. In den Bildern, in denen die Handgelenke nicht zu sehen sind, können die Ergebnisse der manuellen Annotation nicht über das Body Tracking SDK evaluiert werden, da das SDK die Positionen von nicht sichtbaren Gelenken nicht gut schätzen kann, vor allem, wenn das übergeordnete Gelenk ebenfalls nicht sichtbar ist. Dies ist in der folgenden Abbildung 4.1 dargestellt.

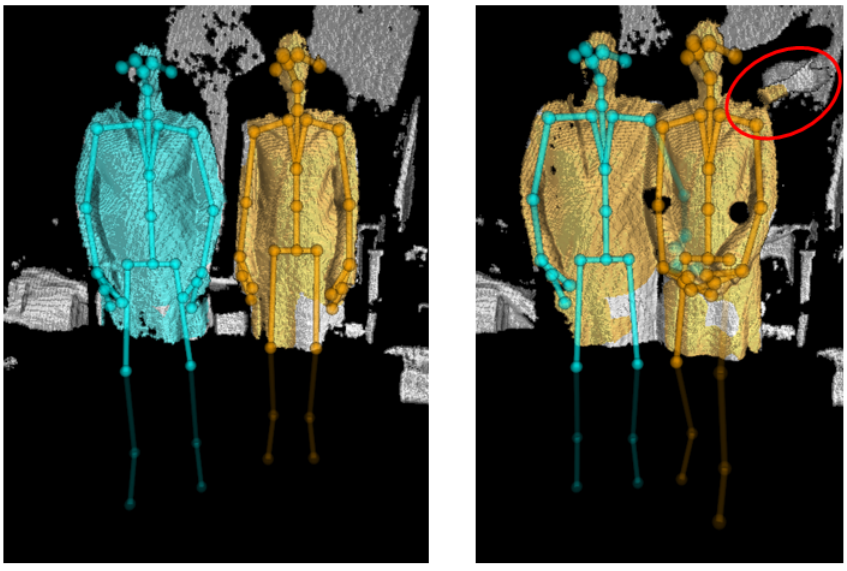

Abb. 4.1: Positionsschätzung des Body Tracking SDK bei verdeckten Gelenken [Quelle: Screenshots aus [24]]: a) alle Gelenke sichtbar, b) Arm der linken Person verdeckt

Hier ist in Abbildung 4.1b) der Arm der linken Person durch die rechte Person verdeckt. Obwohl das Handgelenk der linken Person in dem roten Kreis sichtbar ist, wird die Position des Handgelenks weiterhin parallel zum Körper geschätzt, was durch die sehr hell dargestellte, blaue Linie veranschaulicht ist. Die ähnlichen Verläufe der Handgelenkstrajektorien für die manuelle Annotation und das Body Tracking SDK bei sichtbaren Handgelenken bestätigen die guten Ergebnisse beider Tools. Die sehr „glatten“ Verläufe der manuellen Annotation gegenüber denen des Body Tracking SDK könnten ein Hinweis darauf sein, dass bei Anwendung der manuellen Annotation die Positionen von verdeckten Gelenken durch die Anwenderin oder den Anwender besser geschätzt werden können als durch einen Software-Algorithmus. Wie zuverlässig die Positionen über das manuelle Annotieren tatsächlich sind, hängt allerdings maßgeblich von der annotierenden Person ab.

4.2 Evaluierung mit dem Captiv

Um zu überprüfen, wie gut die manuelle Annotation tatsächlich bei verdeckten Gelenken ist, wurden parallel zu den Videoaufzeichnungen mit der Azure Kinect die Bewegungen mit einem nicht-optischen Motion-Capture-Anzug von Captiv, basierend auf der IMU-Technologie, aufgezeichnet. Bei dieser Technologie haben Verdeckungen keinen Einfluss auf die Positionsmessung und -berechnung (siehe Kapitel 2.2.1 für Details).

Der Captiv wurde bei anderen Videoaufzeichnungen als den oben für die Evaluierung mit dem Body Tracking SDK verwendeten Videos eingesetzt: Für die Videos parallel zur Aufzeichnung mit Captiv wurde die Azure Kinect mit der Linse nach Süden ausgerichtet, damit sie die Person frontal aufnimmt und somit ihr Koordinatensystem einfacher in das Referenzkoordinatensystem des Captiv transformiert werden kann. Im Captiv-System zeigt die positive x-Achse in Richtung des magnetischen Nordens. Die Azure Kinect wurde in 1,145 m Höhe auf einem Stativ befestigt. Die Person stand mit den Hacken auf einem zwei Meter von der Kamera entfernten Punkt (siehe Abbildung A.15 in A.3 im Anhang A.3). Die Captiv-Sensoren wurden vor Beginn der Aufnahme initialisiert und an den Gelenken befestigt. Die Sensoren sind von Captiv mit unterschiedlichen Nummern gekennzeichnet. Anhand dieser Nummern wurden sie in der Captiv-L7000-Software den jeweiligen Gelenken zugeordnet. Die Sensoren wurden über die Software tariert und so konfiguriert, dass die Koordinaten anstelle der Gelenk-

winkel gemessen werden. Danach wurde die Aufnahme über die Azure Kinect und die Captiv-L7000-Software gestartet (der genaue Ablauf wird in Anhang A beschrieben). Das Starten der Aufnahmen erfolgte leicht zeitversetzt.

Das zur Analyse verwendete Video hat eine Dauer von 57 Sekunden und wurde über den Open3D Azure Kinect Recorder aufgezeichnet. Obwohl für diesen eine Frequenz von 30 Hz eingestellt war, konnten nur 394 Punktwolken generiert werden, was einer Frequenz von circa 6,9123 Hz entspricht. Der Captiv zeichnete die Daten mit einer Frequenz von 32 Hz auf. Anschließend wurden die Koordinaten der Captiv-Daten mit Hilfe der Captiv-L7000-Software in absolute Koordinaten in dem Captiv-Referenzkoordinatensystem (siehe Abbildung 3.9) umgerechnet und als CSV-Datei exportiert, um sie mit den Daten der manuellen Annotation vergleichen zu können. Das Video wurde wie in Kapitel 3.3 beschrieben in Punktwolken umgewandelt, in welchen das rechte Handgelenk annotiert wurde. Für die Evaluierung wurden nur die ersten 133 der insgesamt 394 Punktwolken annotiert, was 33,76% entspricht. Um vergleichbare Daten zu generieren, erschien es sinnvoll, auch bei den Captiv-Daten nur die ersten 33,76% zu betrachten. Dies entspricht den Messpunkten 0 bis 612. Auf ein Downsampling der Captiv-Daten wurde im Rahmen dieser Evaluierung verzichtet, da beide Systeme unterschiedliche Zeitstempel verwenden.
Die Daten der manuellen Annotation und des Captiv beziehen sich nicht auf das gleiche Koordinatensystem. In dieser Arbeit gelten die Captiv-Koordinaten als Referenz. Daher wurden die Koordinaten der manuellen Annotation in das Referenzkoordinatensystem von Captiv transformiert. Hierzu wurde das Koordinatensystem der Azure Kinect zunächst um 90 ° um die y-Achse und anschließend ebenfalls um 90 ° um die x-Achse gedreht. Eine Rotation um die z-Achse ist nicht notwendig. Aus der in [21] auf Seite 63 gegebenen Drehmatrix mit φ für die Drehung um die y-Achse, ω für die Drehung um die x-Achse und κ für die Drehung um die z-Achse

$$R_{\varphi\omega\kappa} = \begin{pmatrix} \cos\varphi\cos\kappa + \sin\varphi\sin\omega\sin\kappa & -\cos\varphi\sin\kappa + \sin\varphi\sin\omega\cos\kappa & \sin\varphi\cos\omega \\ \cos\omega\sin\kappa & \cos\omega\cos\kappa & -\sin\omega \\ -\sin\varphi\cos\kappa + \cos\varphi\sin\omega\sin\kappa & \sin\varphi\sin\kappa + \cos\varphi\sin\omega\cos\kappa & \cos\varphi\cos\omega \end{pmatrix}$$

ergibt sich mit $\varphi = 90°$, $\omega = 90°$ und $\kappa = 0°$ folgende Rotationsmatrix:

$$R_{\varphi\omega\kappa} = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -1 \\ -1 & 0 & 0 \end{pmatrix}$$

Das Koordinatensystem der Azure Kinect wird dadurch anhand von trigonometrischen Funktionen in das Referenzkoordinatensystem von Captiv transformiert. Von Captiv liegen in dem betrachteten Zeitintervall deutlich mehr Werte vor als durch die manuelle Annotation. Für die Evaluierung scheint es daher sinnvoll, nicht einzelne Daten mit ähnlichen Zeitstempeln, sondern die Verläufe der Werte miteinander zu vergleichen. Die absoulte Distanz wurde daher hier nicht betrachtet. Da Captiv die Positionen in mm angibt, die manuelle Annotation dagegen Meter als Einheit verwendet, wurden die Werte von Captiv in Meter umgerechnet. Eine Translation des Ursprungs des Koordinatensystems der Azure Kinect zum Ursprung des Referenzkoordinatensystems des Captiv kann nicht durchgeführt werden, da die Position des Ursprungs des Captiv-Referenzkoordinatensystems nicht genau bestimmt werden kann. Dieser liegt, wie in Kapitel 3.1.3 ausgeführt, in L5, d.h. im letzten Lendenwirbel.

In dem analysierten Video wird die Interaktion zweier Personen aufgezeichnet, wobei nur eine Person mit den IMUs von Captiv ausgestattet ist (im Folgenden Person I). Sie stellt die Pflegekraft dar. Die andere Person, welche auf einem Stuhl vor der anderen Person sitzt, übernimmt die Rolle der Patientin (im Folgenden Person II). Da Person II die Aufnahmen startet, beginnt die Interaktion erst nach sieben Sekunden. Es wurden verschiedene Interaktionen ausgeführt: Person I greift mit ihrer linken Hand von unten das Handgelenk und mit der rechten Hand ebenfalls von unten mittig den Oberarm von Person II. Dann führt sie den linken Arm von Person II langsam nach oben und legt ihn wieder ab. Anschließend greift sie zum linken Bein der sitzenden Person II. Hier werden die rechte Hand kurz unterhalb der Kniekehle auf der Wade und die linke Hand am Fußgelenk platziert. Das Bein wird daraufhin um 90 ° nach oben in eine gestreckte Position geführt. Das anschließende Zurückführen des Beins sowie die letzte Interaktion, bei der Person I Person II aus der sitzenden Position in den Stand und wieder zurück in die Ursprungsposition begleitet, wird aus Zeitgründen in der Evaluierung nicht betrachtet. Durch diese Interaktionen sind die Handgelenke von Person I immer wieder vollständig verdeckt.
Die Werte der soeben beschriebenen Interaktionen für die manuellen Annotation und für Captiv werden in den Abbilungen A.9 bis A.14 jeweils getrennt dargestellt, um den Verlauf trotz der verschiedenen Skalierungen der Abszisse besser miteinander vergleichen zu können. Die Werte sind dabei alle im Referenzkoordinatensystem von Captiv angegeben.

In allen Diagrammen ist der Wert der jeweiligen Koordinaten in Meter auf der Ordinate über der Zeit ohne Einheit auf der Abszisse aufgetragen. Die Zeit wird im Fall der manuellen Annotation als Index der Punktwolke und im Fall der Werte von Captiv als Index der Messpunkte repräsentiert, wobei beide eine Schrittweite von eins haben.
In allen Diagrammen ist der Verlauf der Werte zu Beginn, d.h. während der ersten 55 Messpunkte bei der manuellen Annotation und der ersten 270 Messpunkte bei Captiv, annähernd parallel zur Abszisse, da hier Person I auf den Beginn der Interaktion wartet und sich nicht bewegt.

Die Abbildung A.9 bildet die Bewegung des rechten Handgelenks entlang der x-Achse ab, d.h. auf die Kamera zu und von ihr weg. Da sich das rechte Handgelenk während des gesamten betrachteten Bewegungsverlaufs kaum vor oder zurück bewegt, gibt es keine großen Änderungen entlang der Ordinate. Lediglich beim Greifen nach dem Oberarm von Person II bewegt sich das Handgelenk aus der Ausgangsposition nach vorn. Die „Delle“ bei Punktewolke 104 bzw. Messpunkt 504 zeigt eine kleine Bewegung nach hinten, die ausgeführt wird, nachdem der Arm wieder abgelegt wurde und sich Person I wieder aufrichtet, um sich dann zum Bein zu bewegen. Dabei wird das rechte Handgelenk wieder ein wenig nach vorn geführt, was durch den Anstieg des Verlaufs dargestellt wird. Beim Anheben des Beines gibt es dagegen wiederum kaum Bewegung nach vorn oder zurück. Dieser Verlauf kann auch in Abbildung A.10 nachvollzogen werden. Die Verläufe entlang der Ordinate sind um circa 1,9 m gegeneinander verschoben, da der Ursprung des Koordinatensystems der Azure Kinect nicht über eine Translation auf den Ursprung des Referenzkoordinatensystems von Captiv gelegt werden konnte. Die Translationsstrecke vom Ursprung des Koordinatensystems der Azure Kinect zum Referenzkoordinatensystem von Captiv in L5 der instrumentierten Person konnte nicht genau bestimmt werden.

In Abbildung A.11 wird die Bewegung des rechten Handgelenks entlang der y-Achse, d.h. nach rechts und links, im Referenzkoordinatensystem von Captiv dargestellt. Da sich das rechte Handgelenk beim Greifen nach dem Oberarm von der Ausgangsposition nach rechts bewegt, steigen die Werte im Verlauf an, da die positive y-Achse des Referenzkoordinatensystem von Captiv nach rechts zeigt. Beim anschließenden Anheben und Ablegen des Armes der Person II wird das rechte Handgelenk kaum nach links oder rechts bewegt, weshalb die Werte zwischen der 77. und der 104. Punktwolke ebenfalls annähernd parallel zur Abszisse verlaufen.

Der starke Abfall der Werte im anschließenden Verlauf bildet die Bewegung des Handgelenks nach rechts zum Bein von Person II ab. Dieser Verlauf kann auch in Abbildung A.12 beobachtet werden, denn auch hier steigt der Verlauf zuerst und fällt am Ende ebenfalls ab. Hier gibt es ebenfalls aufgrund der nicht durchführbaren Translation eine Verschiebung der Verläufe entlang der Ordinate. Die Abweichung beträgt hier bis zu 20 cm (gemessen in Punktwolke 74 und Messpunkt 355), da Person I nicht auf der z-Achse des Koordinatensystems der Azure Kinect steht. Somit sind die Ursprünge der beiden Koordinatensysteme aus Sicht des Referenzsystems von Captiv entlang der y-Achse nach rechts verschoben.

Die Bewegungen des Handgelenks entlang der z-Achse im Referenzkoordinatensystem von Captiv, also aufwärts und abwärts, werden in den Abbildungen A.13 und A.14 für die manuelle Annotation bzw. die Werte von Captiv dargestellt. Die Verläufe sind auch hier entlang der Ordinate gegeneinander verschoben, denn der Ursprung des Koordinatensystems der Azure Kinect, aus der die Daten der manuellen Annotation kommen, liegt circa 17,5 cm höher. Da das rechte Handgelenk in der Ausgangsposition im transformierten Koordinatensystem der Kamera in etwa 11 cm unterhalb von dessen Ursprung und im Referenzkoordinatensystem des Captiv etwas oberhalb von dessen Ursprung liegt, ist die Differenz zwischen den beiden Verläufen in der Ausgangsposition etwa 6,8 cm. In Abbildung A.13 steigt anschließend der Verlauf der manuell annotierten Werte, dann flacht die Steigung des Verlaufs ab, bevor der Verlauf fällt. Während dieser steigenden Passagen bewegt sich das Handgelenk von Person I aus der Ausgangsposition, den oberhalb der Hüfte eingestützten Händen, nach unten zum Oberarm. Da die positive z-Achse des rotierten Koordinatensystems der Azure Kinect nach unten zeigt, nehmen die Werte zu. Das Fallen der Kurve bildet das Anheben des Arms ab. Anschließend steigt der Verlauf wieder, da der Arm von Person II wieder abgelegt, das Handgelenk also nach unten bewegt wird. Der darauffolgende leichte Abfall spiegelt das Aufrichten von Person I wider, bevor sie sich bückt, um an das Bein von Person II zu kommen. Da diese Bewegung sehr schnell ausgeführt wird, liegen die Punkte in Abbildung A.13 entlang der Ordinate weiter auseinander. Die Werte steigen weit ins Positive, da die Bewegung in Richtung der positiven z-Achse weit nach unten geht. Während das Bein angehoben wird, fallen die Werte nur leicht, da hier hauptsächlich die linke Hand das Bein hebt, während die rechte Hand es nur stützt, sodass keine großen Bewegungen aufwärts oder abwärts auftreten.

Die Abbildung A.14 der Captiv-Werte zeigt bis zum Messpunkt 296 einen sehr ähnlichen Verlauf wie in Abbildung A.13 bis Punktwolke 62. Anschließend gibt es allerdings einen Sprung, nach dem sich die Werte deutlich im negativen Bereich befinden. Da das rechte Handgelenk in dem Fall allerdings oberhalb des Ursprungs des Referenzkoordinatensystems von Captiv wäre und es sich während des gesamten nachfolgenden Bewegungsverlaufs unterhalb des Ursprungs befindet, scheint der Ursprung von Captiv verschoben worden zu sein. Mit Werten von -92 cm direkt nach dem Sprung müsste der Ursprung nun auf dem Boden und nicht mehr im letzten Lendenwirbel L5 liegen. Wie es zu dieser Verschiebung des Ursprungs des Referenzkoordinatensystems des Captiv gekommen ist, ist nicht bekannt.

Die Verläufe der Werte der manuellen Annotation und des Captiv sind sich sehr ähnlich, und die Verschiebungen in den Diagrammen entsprechen ungefähr den Distanzen der beiden Ursprünge zueinander. Dies lässt den Schluss zu, dass die manuelle Annotation auch für verdeckte Handgelenke gute Ergebnisse liefert. Denn bis auf die Abschnitte, in der sich das Handgelenk aus der Ausgangsposition von der Hüfte des Körpers der Person I zum Oberarm der sitzenden Person II bewegt sowie vom Oberarm zum Bein, ist das rechte Handgelenk immer verdeckt. Beim Vergleich der Verläufe ist zwischen den Abschnitten mit sichtbaren Handgelenken und denen mit verdeckten Handgelenken kaum ein Unterschied zu erkennen. Eine Azure-Kinect-Videoaufzeichnung mit 30 Punktwolken pro Sekunde anstelle der intern durch Open3D scheinbar fehlerhaft eingestellten 6,9123 Hz hätte wahrscheinlich besser vergleichbare Verläufe geliefert. Die teilweise nicht abgebildeten Teilverläufe in den Abbildungen A.9, A.11 sowie A.13 sind den fehlenden Daten aufgrund der zu geringen Anzahl der Punktwolken geschuldet. Dies ist z.B. in Abbildung A.11 zu erkennen: Bei Punktwolke 60 ist keine Spitze und kein darauffolgendes „Tal“ dargestellt, wie in Abbildung A.12 vor Messpunkt 300 auftritt.
Dennoch lässt sich mit Hilfe dieses nicht-optischen Systems, das unempfindlich gegenüber Verdeckungen ist, zeigen, dass das manuelle Annotationstool vergleichbar gute Ergebnisse sowohl für verdeckte als auch für sichtbare Gelenke liefert.

5 Fazit und Ausblick

Ziel dieser Arbeit war die Entwicklung eines Tools zur Annotation von sichtbaren bis zu vollständig verdeckten Handgelenken. Die Anforderungen an ein derartiges Tool wurden durch eine Literaturrecherche und die Verarbeitung von Bilddaten konkretisiert. So stellte sich schnell heraus, dass es sinnvoll ist, dreidimensionale Daten eines Bewegungsablaufes zu erfassen, da diese den Bewegungsablauf gut darstellen und das Schätzen der dreidimensionalen Position für die spätere Verwendung erleichtern. Die Daten wurden daher mit einer Azure Kinect aufgezeichnet. Für die anschließende Verarbeitung der Daten wurde die Bibliothek Open3D gewählt, weil sie die gängigen Verarbeitungsalgorithmen für 3D-Daten und Methoden zu deren Visualisierung enthält.

In Kapitel 2.3 wurde unter anderem die Anforderung formuliert, dass das Annotationstool Videosequenzen verarbeiten können muss, um Bewegungsverläufe analysieren zu können. Diese Anforderung konnte mit Hilfe des *Azure Kinect Reader* der Open3D-Bibliothek umgesetzt werden. Der *Azure Kinect MKV Reader* zelegt die Videos in Einzelbilder, diese können anschließend ebenfalls über Open3D in Punktwolken überführt werden. Open3D stellte sich somit als eine sehr umfangreiche API heraus, mit der vieles bereits umgesetzt werden konnte. Lediglich die Eingaben von Daten für Pfade zu Dateien und andere Einstellungen konnten zu Beginn noch nicht über Open3D umgesetzt werden, weshalb für diese Interaktionen auf PyQt5 zurückgegriffen wurde. Hiermit wurden die GUIs erstellt, die die relevanten Informationen auf intuitive Weise von den Nutzerinnen und Nutzern abfragen.

Das Annotationstool wurde mit dem Schwerpunkt auf der Nachverfolgung von Handgelenken in (pflegerischen) Interaktionen von mindestens zwei Personen entwickelt. Da an einer Interaktion, besonders im pflegerischen Kontext, meistens beide Hände beteiligt sind, war es notwendig, dass mehrere Handgelenke in einem Datensatz annotiert werden können. Dies wurde umgesetzt, indem das zu annotierende Gelenk vor Beginn des Annotationsprozesses abgefragt und alle an-

notierten Positionen in eine CSV-Datei geschrieben werden. Die Positionen, die bei der Annotation eines weiteren Gelenks im Anschluss an die bereits abgeschlossene Annotation entstehen, werden in dieselbe Datei geschrieben, sodass die Positionen eines Gelenks sowohl für die rechte als auch die linke Seite in einer Datei stehen und mehrere Gelenke in einem Datensatz annotiert werden können. Durch diesen Ansatz ist es auch möglich, die Gelenke von mehreren Personen nachzuverfolgen. Denn für jede Person lässt sich eine eigene CSV-Datei anlegen, in der die Gelenkpositionen gespeichert werden. Bei eindeutiger Benennung der Dateien kann somit auch gut der Überblick über die Daten der verschiedenen Personen behalten werden.

Die in der untersuchten Literatur zur Nachverfolgung von Objekten vorgestellten Verfahren behandeln die Nachverfolgung von Handgelenken außerhalb der Gestenerkennung und -steuerung kaum. Um das Problem der Verdeckung zu reduzieren wurden meist mehrere Kameras eingesetzt (vgl. [50]). Im pflegerischen Kontext findet die Interaktion mit dem Patienten oder der Patientin meist am Bett statt. Dabei können sich die Handgelenke der Pflegekraft zeitweise auch unter dem Patienten oder der Patientin befinden, wie z.B. bei einer Umlagerung. In diesem Fall würden auch mehrere Kameras die Verdeckungen nicht reduzieren. Auch die *State-of-the-Art*-Methoden zur Nachverfolgung von Objekten in Bilddaten können die Positionen bei Verdeckungen nicht präzise schätzen (vgl. [35]). Somit ergab sich der Ansatz, eine Software für ein manuelles Annotationstool zu entwickeln, bei dem der Nutzer oder die Nutzerin die Bilddaten analysiert und interpretiert und auf dieser Grundlage das Handgelenk annotiert.

Die Evaluierung des manuellen Annotationstools fand über einen Vergleich zu zwei unabhängigen Systemen statt: zum einen im Vergleich zu einem ebenfalls optischen, markerlosen System in Form des Body Tracking SDK und zum anderen im Vergleich zu einem nicht-optischen, inertialen System über den Captiv Motion-Capture-Anzug von TEA. Das manuelle Annotationstool wurde hierbei jeweils über die Ähnlichkeit der mittels Annotationstool und Referenzsystemen erzeugten Trajektorien evaluiert. In beiden Fällen zeigte sich, dass die manuelle Annotation plausible und im Vergleich zu den Referenzsystemen ähnliche Verläufe erzeugt und die Positionen daher gut schätzt. Wie präzise die Positionsschätzung des Annotationstools ist, konnte in dieser Arbeit nicht ermittelt werden, da Daten von den Vergleichssystemen bezüglich ihrer Präzision hinsichtlich der Positionsschätzung fehlen: Bei Captiv ist die Position des Ursprungs des

Referenzkoordinatensystems bekannt, jedoch lässt sich diese durch die Lage im letzten Lendenwirbel L5 nicht genau genug bestimmen. Da L5 im Körper liegt und kein sichtbarer Punkt ist, konnte der Abstand vom Boden und von der Azure Kinect in dieser Arbeit nicht gemessen werden. Hierzu wäre eine Schablone hilfreich, auf der gekennzeichnet ist, wo die Füße platziert werden müssen, damit L5 genau über dem auf dieser Schablone markierten Punkt liegt. Die Präzision des Captiv wurde bereits in der Publikation von Tim Steinebach et al. [37] evaluiert, jedoch betrachtet sie die Genauigkeit der Gelenkwinkel und nicht die der absoluten Koordinaten. Da der verwendete Captiv in erster Linie für die Messung der Gelenkwinkel entwickelt wurde, ist er für die Evaluierung des Annotationstools nur bedingt geeignet (vgl. [45]). Jedoch war dieser eine der verfügbaren Möglichkeiten, die Daten des Annotationstools bei verdeckten Gelenken zu validieren.
Für das Body Tracking SDK lagen genaue Informationen zur Position des Ursprungs des Koordinatensystems nicht vor, sodass auch hier die Daten des manuellen Annotationstools nicht im Detail mit denen des Body Tracking SDK verglichen werden konnten. Denn durch die fehlenden Informationen konnten die Ursprünge der beiden Koordinatensysteme nicht genau aufeinander gelegt werden. Daher kann auch hier keine Aussage über die Präzision der manuell annotierten Daten gemacht werden.

Der Annotationsansatz könnte automatisiert werden, indem beispielsweise die Positionsdaten aus dem Body Tracking SDK für das jeweilige Gelenk in die Punktwolken gelegt werden. Anschließend müssten die Nutzerinnen und Nutzer die Punktwolken nur noch durchgehen und gegebenenfalls in einigen Punktwolken nachbessern. Sofern auch das übergeordnete Gelenk verdeckt ist, lieferte das Body Tracking SDK für verdeckte Handgelenke keine guten Werte (siehe Kapitel 4.1). Da auch in den Punktwolken mit sichtbaren Gelenke die Werte nicht gut passten, wurde dieser Ansatz im Rahmen dieser Arbeit nicht verfolgt. Um den Aufwand der manuellen Annotation dennoch zu reduzieren, wurde eine Interpolation implementiert, mit der über die Punktwolken interpoliert werden kann, die eine lineare Bewegung des Handgelenks abbilden.
Ein weiterer Ansatz für die Automatisierung der Positionsschätzung von verdeckten Gelenken wäre, den Ellenbogen als übergeordnetes, sichtbares Gelenk nachzuverfolgen und aufgrund seiner jeweiligen Position und dem Abstand zwischen Handgelenk und Ellenbogen die Position des Handgelenks zu schätzen. Sollte der Ellenbogen ebenfalls vollständig verdeckt sein, könnte auch über die Nachverfolgung der Schultern gearbeitet werden, um über den Winkel zum Unterarm und

dem Abstand vom Ellenbogen zum Handgelenk auf die Position des Handgelenks zu schließen. Dies ließe sich mit Techniken der Künstlichen Intelligenz umsetzen, konnte aber im Rahmen dieser Arbeit aufgrund des hohen Aufwandes nicht realisiert werden.

Insgesamt erfüllt das Annotationstool alle formulierten Anforderungen und ermöglicht somit die Nachverfolgung von vollständig verdeckten Händen. Durch die Art der Umsetzung kann das Annotationstool auf einfache Weise um weitere Gelenke erweitert werden, sodass es auch auf weitere Einsatzgebiete angewendet werden kann.

A Anhang

A.1 Diagramme zur Evaluierung mit dem Body Tracking SDK

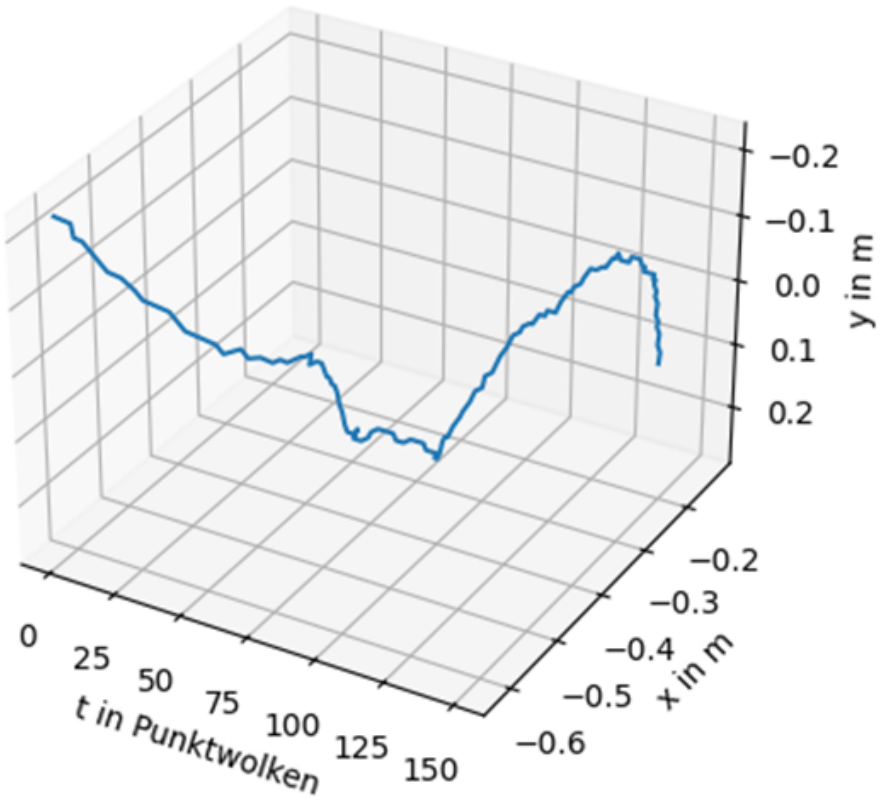

Abb. A.1: Bewegungsverlauf entlang der x- und y-Achse unabhängig von der Höhe für das Video aus *recording10*

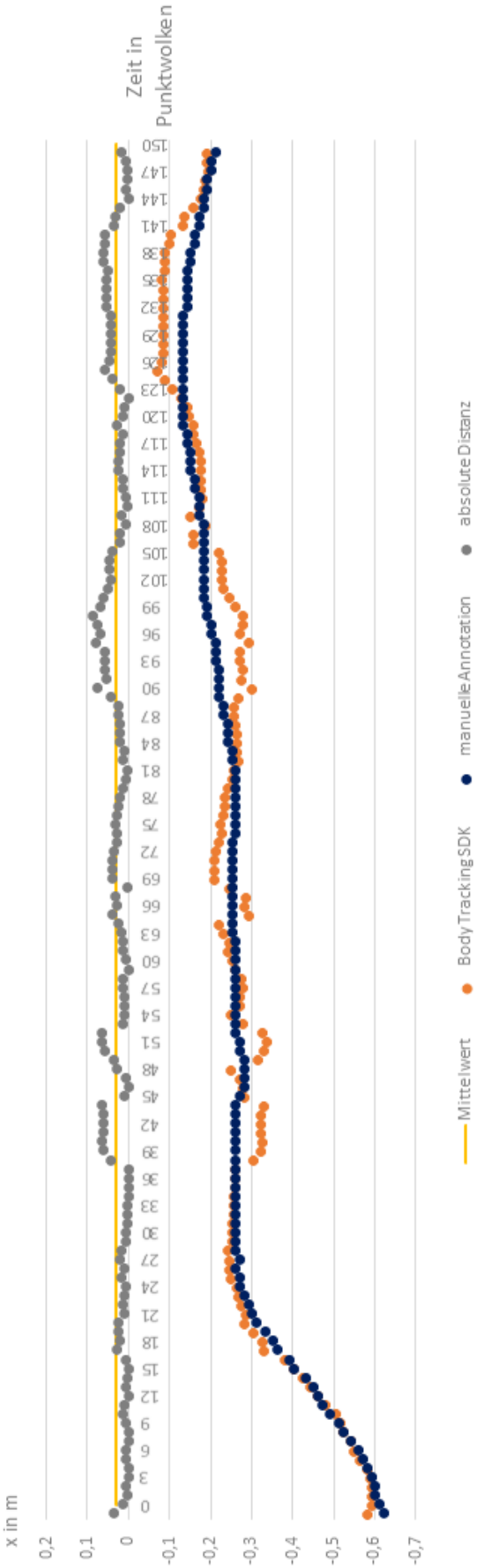

Abb. A.2: Bewegungsverlauf entlang der x-Achse für das Video aus *recording10*

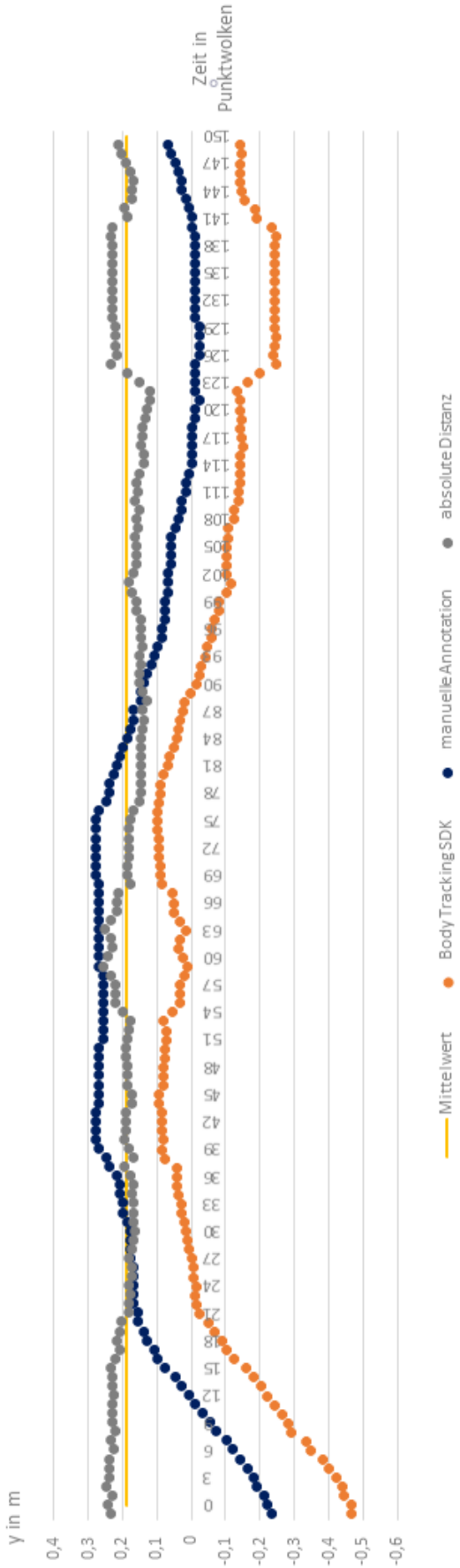

Abb. A.3: Bewegungsverlauf entlang der y-Achse für das Video aus *recording10*

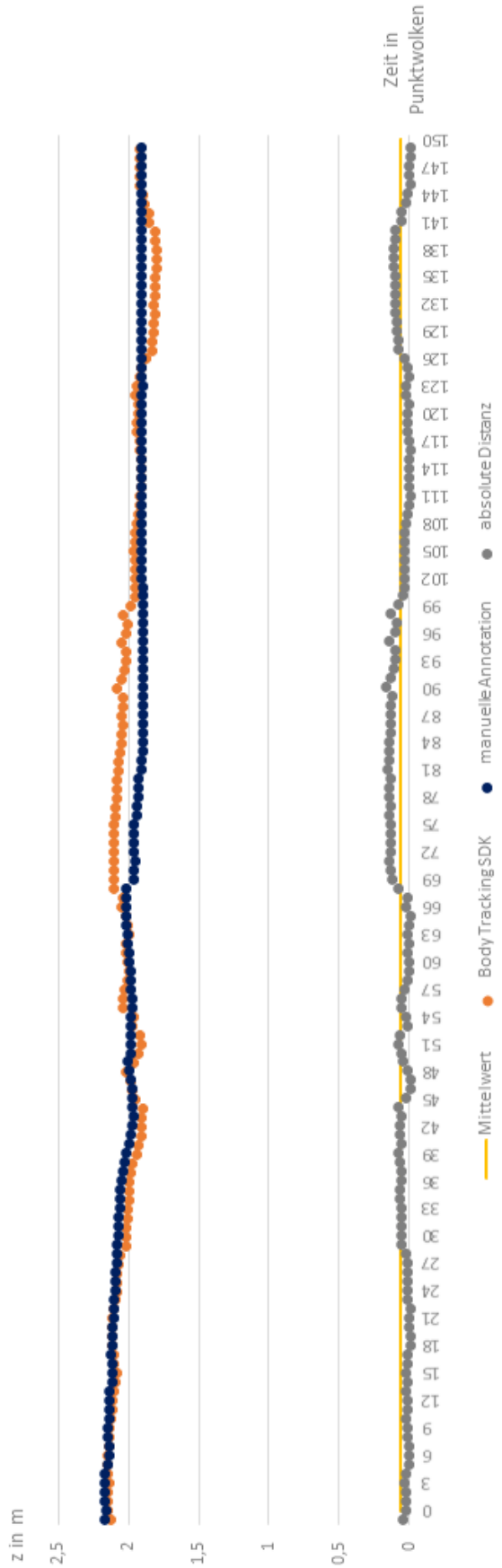

Abb. A.4: Bewegungsverlauf entlang der z-Achse für das Video aus *recording10*

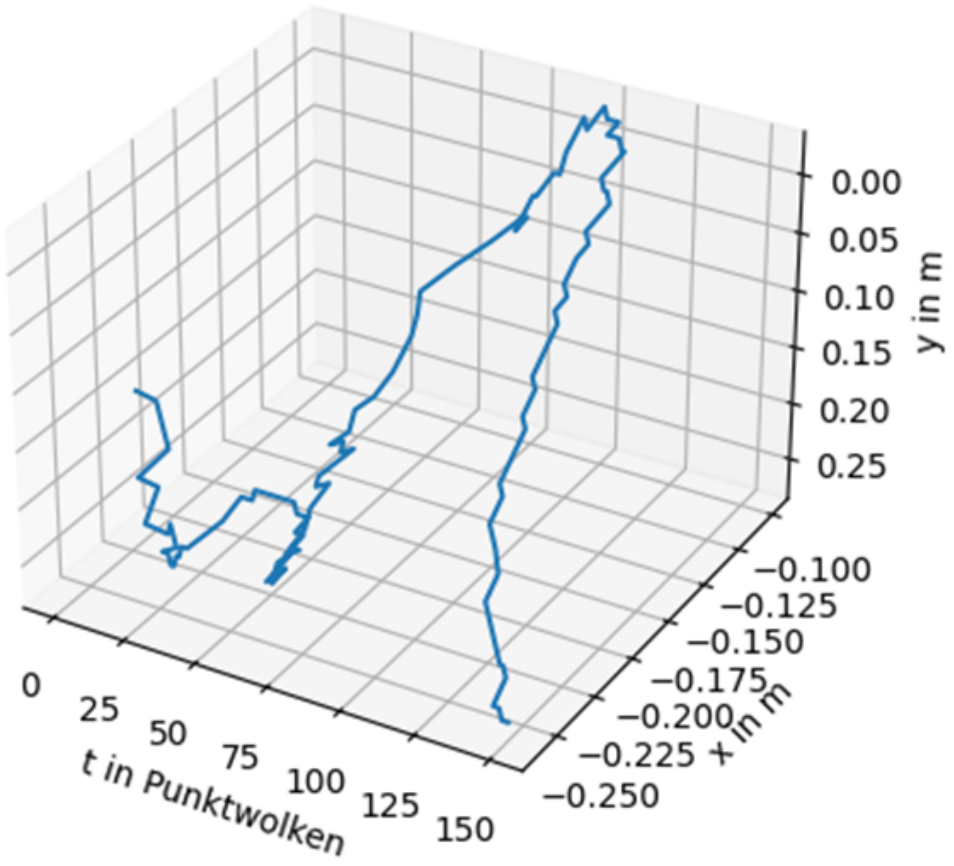

Abb. A.5: Bewegungsverlauf entlang der x- und y-Achse unabhängig von der Höhe für das Video aus *recording9*

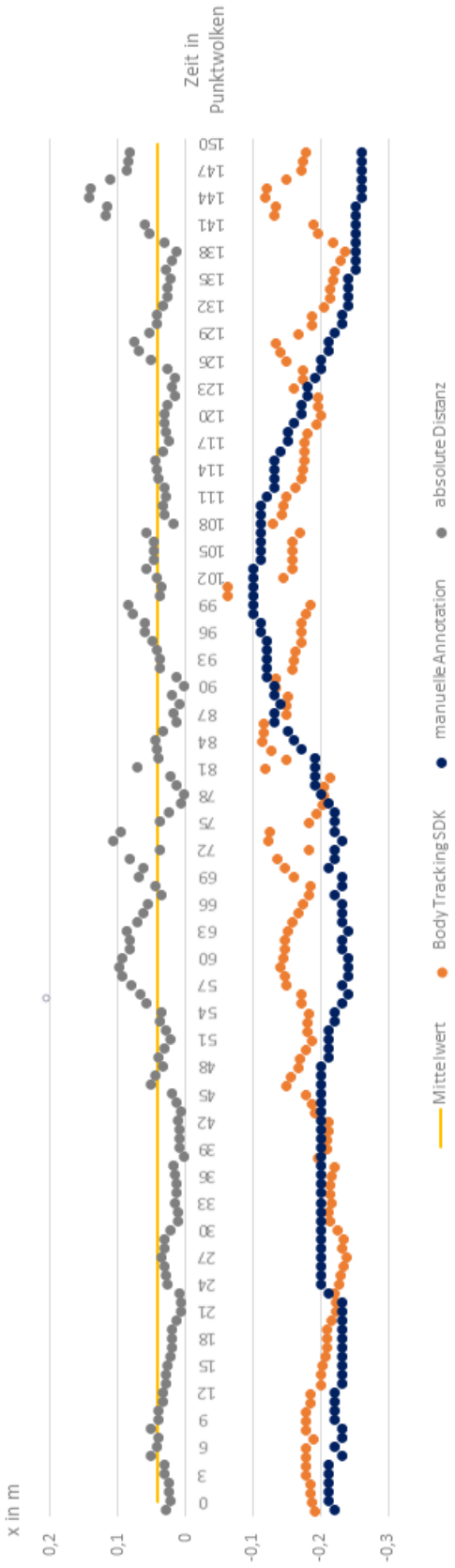

Abb. A.6: Bewegungsverlauf entlang der x-Achse für das Video aus *recording9*

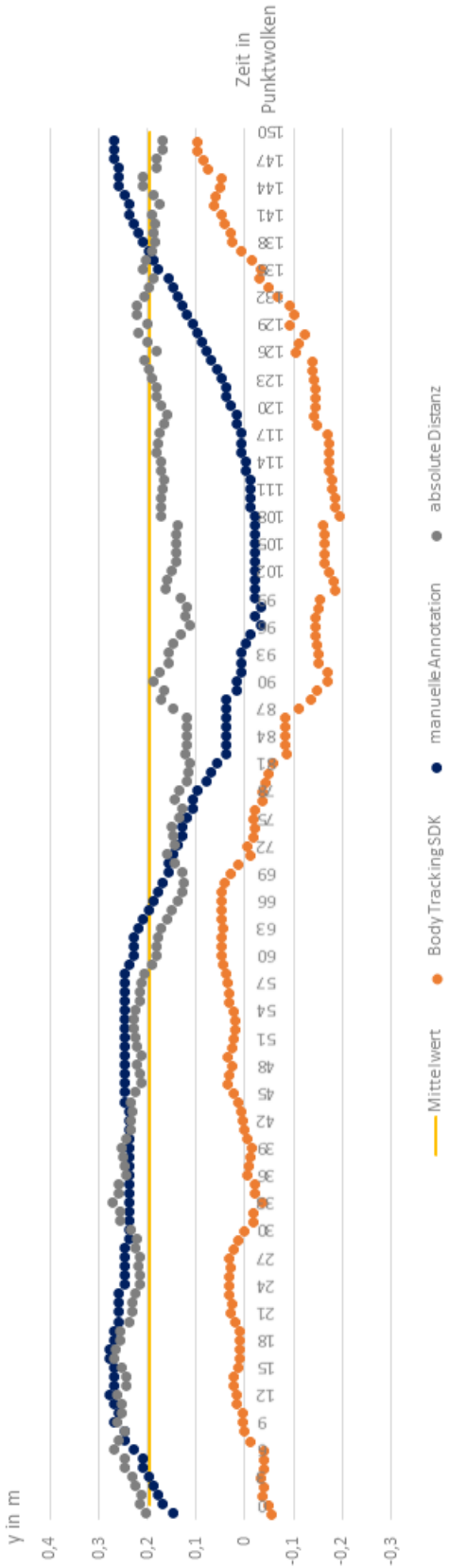

Abb. A.7: Bewegungsverlauf entlang der y-Achse für das Video aus *recording9*

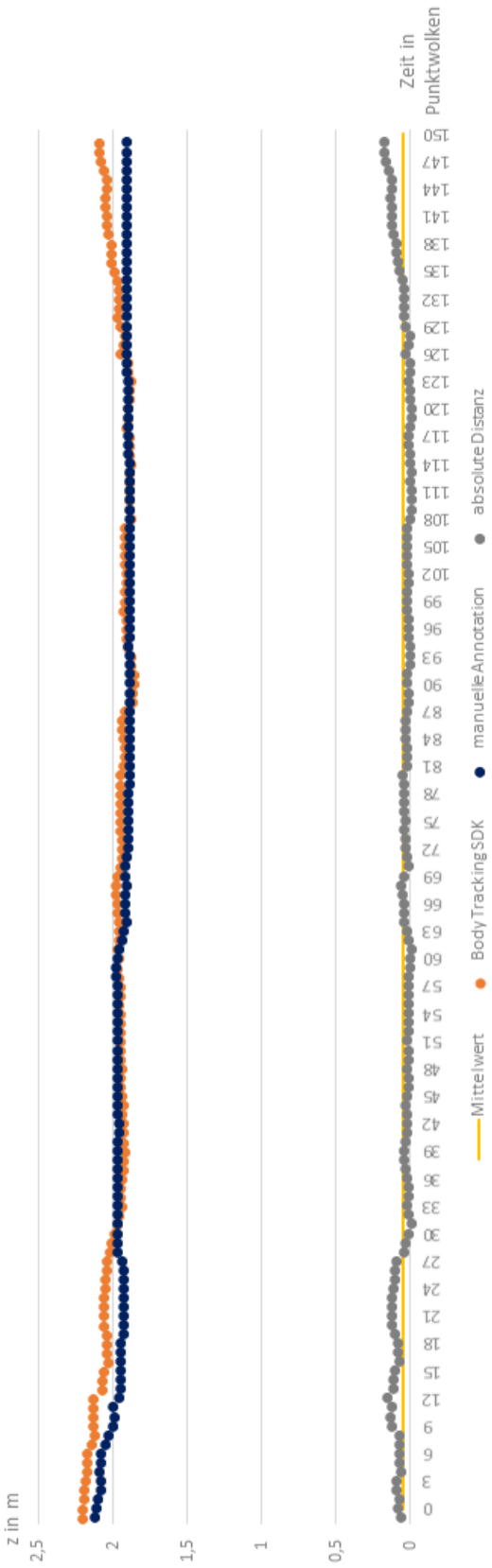

Abb. A.8: Bewegungsverlauf entlang der z-Achse für das Video aus *recording9*

A.2 Diagramme zur Evaluierung mit dem Captiv

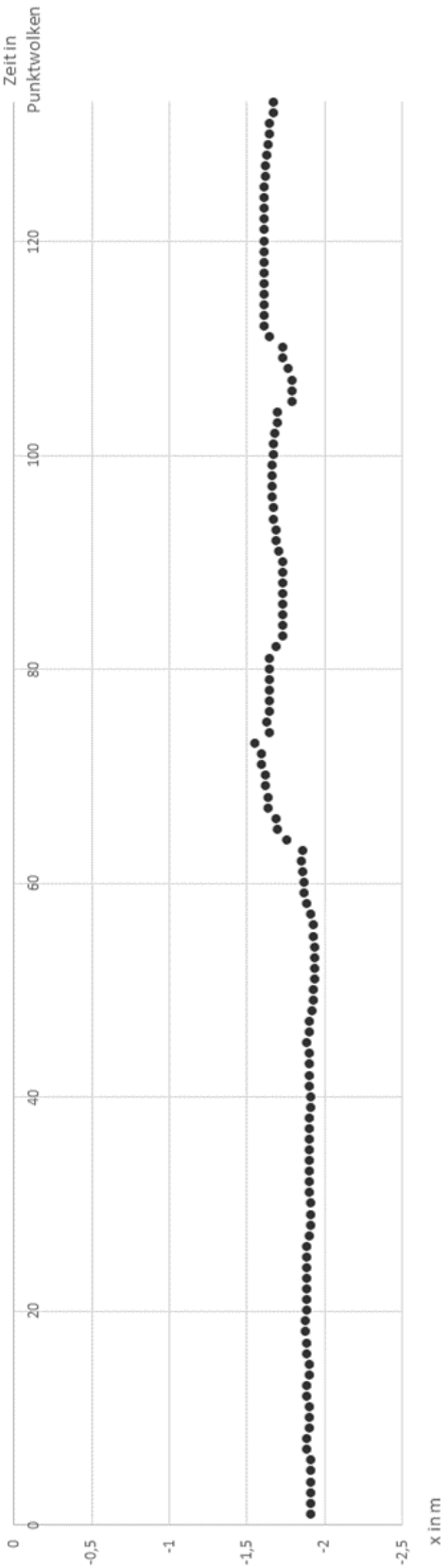

Abb. A.9: Bewegungsverlauf entlang der x-Achse für die manuelle Annotation

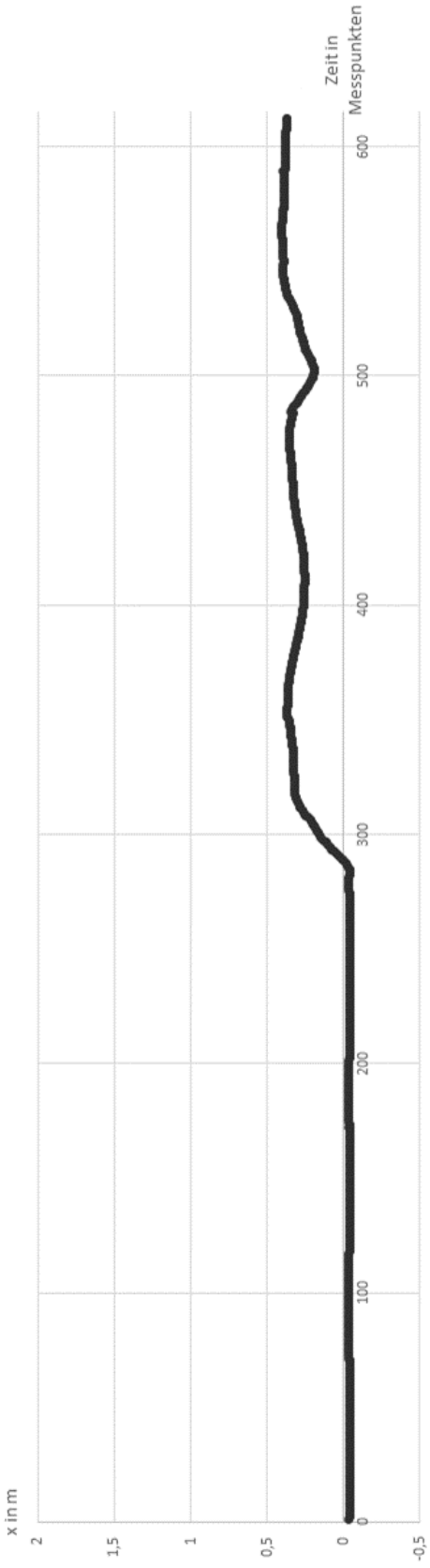

Abb. A.10: Bewegungsverlauf entlang der x-Achse für den Captiv

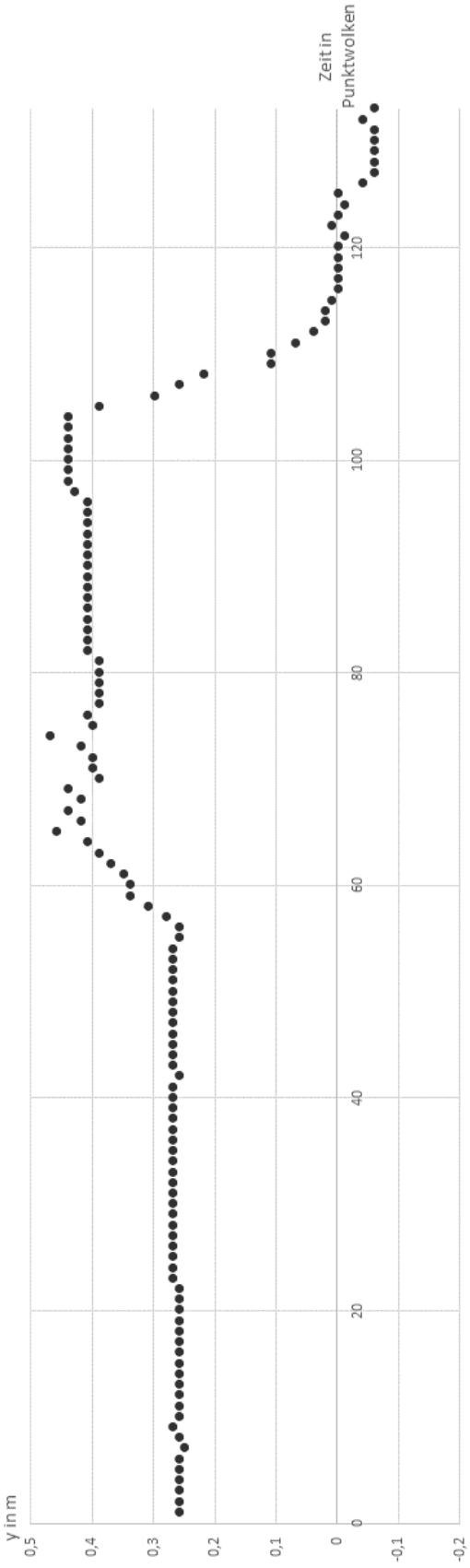

Abb. A.11: Bewegungsverlauf entlang der y-Achse für die manuelle Annotation

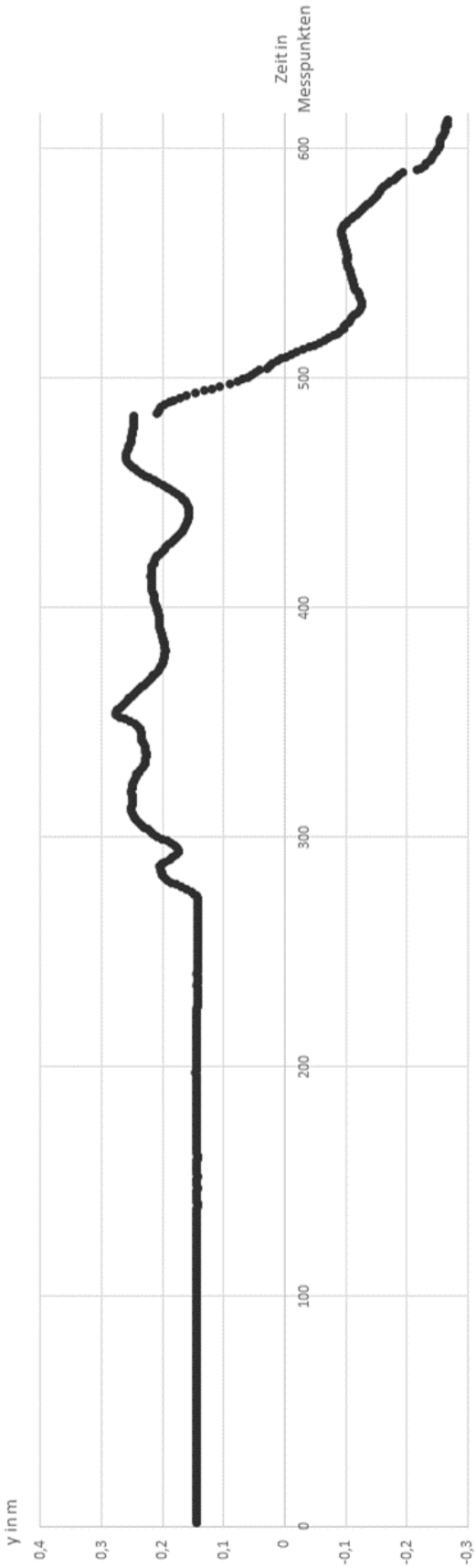

Abb. A.12: Bewegungsverlauf entlang der y-Achse für den Captiv

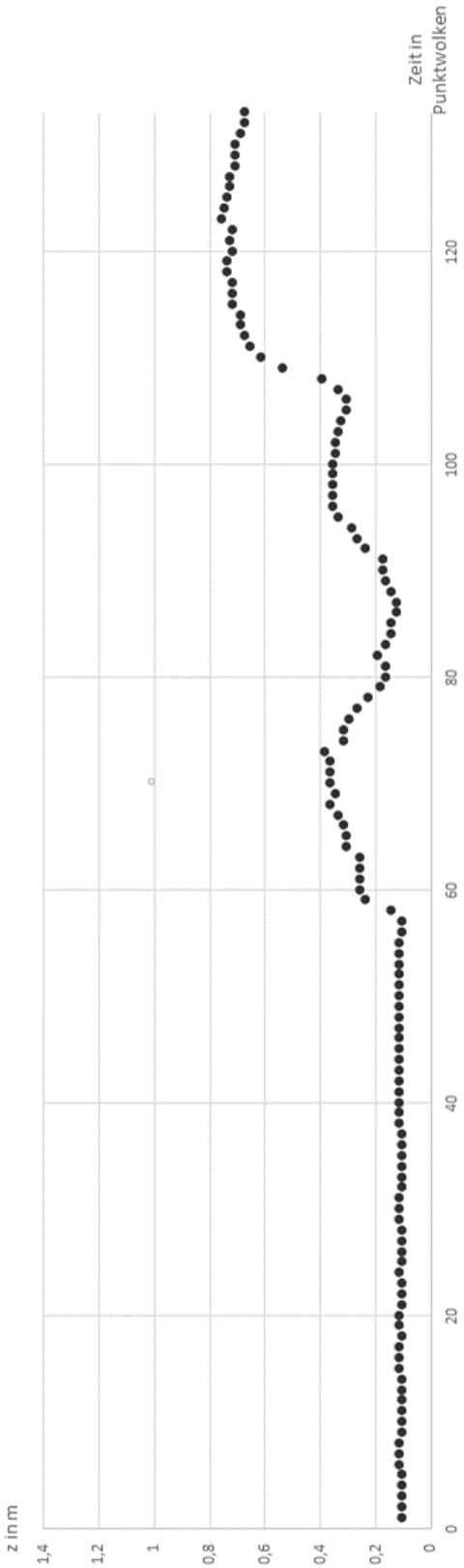

Abb. A.13: Bewegungsverlauf entlang der z-Achse für die manuelle Annotation

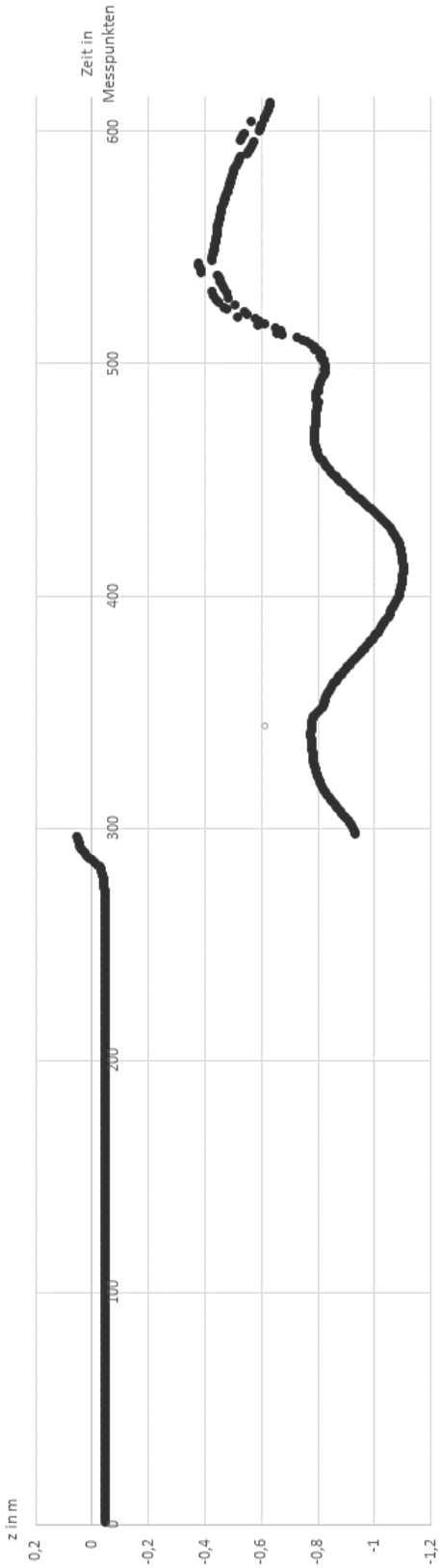

Abb. A.14: Bewegungsverlauf entlang der z-Achse für den Captiv

A.3 Vorgehen bei der Aufnahme mit Captiv

Die folgende Beschreibung des Vorgehens basiert auf den Anleitungen des Herstellers TEA zum Captiv (vgl. [12], [39] und [3]), wobei einige Schritte nicht in dieser Form in den Anleitungen von TEA beschrieben, sondern durch Ausprobieren ermittelt wurden:

1. Die Sensorunterlage in Richtung des magnetischen Nordens ausrichten, ohne das die Sensoren darauf liegen, da diese durch ihr Magnetometer den Kompass beeinflussen.

2. Die Sensoren auf der Unterlage platzieren, ohne diese zu verschieben.

3. „Die Sensoren anschalten bis die grüne LED leuchtet, ohne sie oder die Unterlage zu bewegen. “ ([39], S. 3)

4. Die Sensoren eine Minute in Ruhe lassen, damit sich die Magnetometer initialisieren können (vgl. [39], S. 3).
   Hinweis: Während des gesamten Prozesses unnötige magnetische, metallische oder stromführende Objekte und Geräte weit wegräumen (vgl. [39], S. 3).

5. Die Captiv-L7000-Software starten und den T-Rec aus dem Recording-Reiter auswählen.

6. Nun den T-Rec an den PC anschließen und warten, bis die Verbindung aufgebaut wurde (vgl. [3], S. 85).

7. Den *Carrier* für die Sensoren wählen (in diesem Fall Carrier = 37) und warten, bis alle Sensoren in der Liste aufgeführt sind (vgl. [3], S. 85).

8. Nun kann für jeden Sensor die *sampling rate* festgelegt werden (in diesem Fall wurde für alle Sensoren 32 Hz festgelegt) (vgl. [3], S. 89).

9. Auf *Next* klicken. Der *initialization test* wird gestartet.

10. Beim Initialization Test sollte der Zeiger im grünen Bereich sein (vgl. [39], S. 4).

11. Ist der Initialization Test erfolgreich abgeschlossen, können die Proportionen des Avatars angepasst werden. Hierzu auf Anthropometry klicken und die entsprechenden Maße eintragen. Ansonsten, d.h. wenn der Zeiger im

orangen oder roten Bereich ist, wieder bei Schritt 1 beginnen (vgl. [39], S.4).

12. Im nächsten Schritt die Sensoren an den Körper anbringen und in der Captiv-L7000-Software die Sensornummern über Drag-and-drop den entsprechenden Gelenken zuordnen (vgl. [39], S. 6 ff.).

13. Gerade, hüftbreit hinstellen, Kopf geradeaus richten und die Arme parallel zum Körper halten. In dieser Position *zeroing* durchführen (tarieren) (vgl. [39], S. 10 f.).

14. Die Azure Kinect aufstellen und nach Süden ausrichten

15. 2 m von der Kamera entfernt einen Punkt markieren. Dort steht die Person zu Beginn der Aufnahme. Dieser Punkt wurde mit Hilfe eines Lots und eines Zollstocks bestimmt.

16. Die Aufnahme der Azure Kinect vorbereiten: Den Ordner, in dem das Video gespeichert werden soll und die Ausgabedatei festlegen.

17. Die Aufnahme mit dem Captiv vorbereiten:
    - *T-Server* auswählen (vgl. [12], S. 8)
    - *Text* auswählen (vgl. [12], S. 20)
    - *Enable* auswählen und *Refreshrate* auf 32 festlegen (vgl. [12], S. 11)
    - *Coordinate* als Ausgabetyp auswählen (vgl. [12], S. 11)
    - Alle Sensoren aktivieren, in dem ein Haken an jedem Sensor in der Software gesetzt wird. (vgl. [12], S. 8)
    - Der Aufnahme einen Namen geben.
    - In den *recording Mode* wechseln. (vgl. [12], S. 8)

18. Die Aufnahme der Azure Kinect starten.

19. Die Aufnahme des Captiv starten.

20 Eine Kalibierungsgeste (z.B. schnelle Bewegung) machen.

21. Die Daten aufnehmen.

22. Die Captiv-Aufnahme stoppen.

23. Die Azure-Kinect-Aufnahme stoppen.

24. Über den Reiter *Tools* und den Reiter *Avatar* im Reiter *Absoulte Coordinates* die absoluten Koordinaten berechnen lassen.

25. Im Reiter *Export* im Avatar-Menü die absoluten Koordinaten im CSV-Format exportieren.

26. Eventuell auch die 3D-Animationsdaten im BVH-Format exportieren.

27. Auswerten und neue Aufnahmen machen.

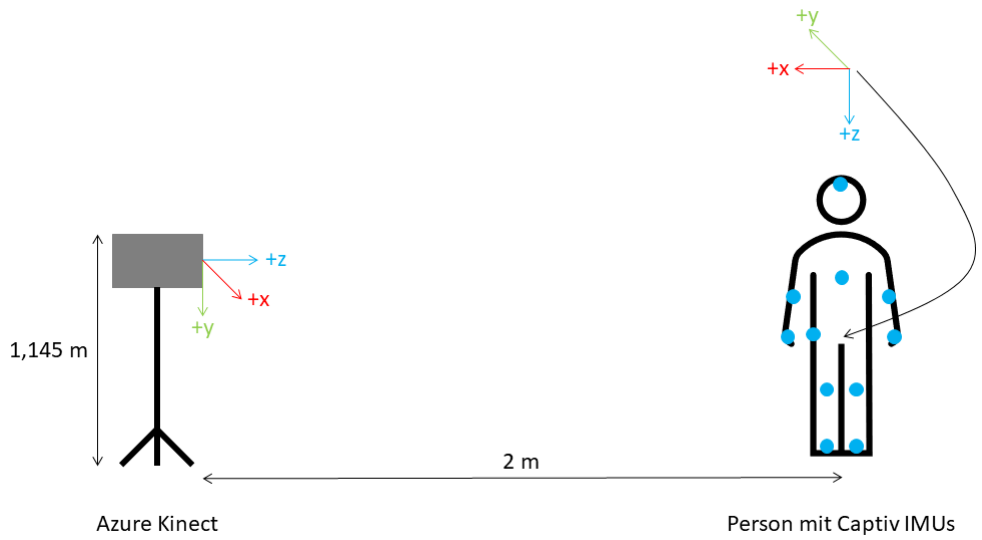

Abb. A.15: Der schematische Versuchsaufbau für die Aufnahme mit dem Captiv

B Inhalt der DVD

Auf der DVD befinden sich folgende Dateien:

- *Bachelorarbeit_Lindanis.pdf*: PDF-Datei der Bachelorarbeit
- *followIT.exe*: dies ist die installierbare, ausführbare Anwendung des Annotationstools
- Ordner *followIT*: enthält den Python-Source-Code des Annotationstools sowie die Tests
- *README.txt*
- *recording10.mkv*: Videoaufzeichnung des beschriebenen Bewegungsverlaufs zu Testzwecken
- Ordner *recording10*: enthält die Farb- und Tiefenbilder des Videos, die daraus generierten Punktwolken sowie die *config.json*- und *intrinsic.json*-Datei

Literaturverzeichnis

[1] Amboss: *Hand.* https://www.amboss.com/de/wissen/Hand. Version: 2021. – letzter Zugriff: 06.01.2021

[2] Arizpe-Gómez, Pedro ; Harms, Kirsten ; Fudickar, Sebastian ; Janitzky, Kathrin ; Witt, Karsten ; Hein, Andreas: Preliminary Viability Test of a 3-D-Consumer-Camera-Based System for Automatic Gait Feature Detection in People with and without Parkinson's Disease. In: *8th IEEE International Conference on Healthcare Informatics*, 2020, S. 7

[3] Balmont, Julien: *Captiv software V1.5: User manual.* 2014

[4] Balmont, Julien: *Captiv Motion Wireless Sensors and Measurements : Wireless Measurement of 3D Motion Real time via long-range wireless Receiver Datalogger for tetherless Recording with no distance limitation.* https://wearablesensing.com/wp-content/uploads/2018/08/CAPTIV-Motion-WSv10.pdf. Version: 2018. – letzter Zugriff: 31.12.2020

[5] Bamji, Cyrus: *1Mpixel 65nm BSI 320MHz Demodulated TOF Image Sensor with 3.5U+03BCm Global Shutter Pixels and Analog Binning.* https://opdhsblobprod03.blob.core.windows.net/contents/503db294612a42b3b95420aaabac44cc/e6dc19e3bcf53fed8fcb1291e68adf20?sv=2018-03-28&sr=b&si=ReadPolicy&sig=%2Frx6cArXr8MRvdqDFkVHvmbEMfR2syEmew7IYGU754Y%3D&st=2020-11-17T13%3A33%3A39Z&se=2020-11-18T13%3A43%3A39Z. Version: 2018. – letzter Zugriff: 17.11.2020

[6] Bauer, Eva: *Indoor-Positionierung mit Inertialsensoren.* https://diglib.tugraz.at/download.php?id=576a763df3bb8&location=browse. Version: 2014. – letzter Zugriff: 29.12.2020

[7] Boettcher, Jörg: *8. Sensoren für Drehzahl, Geschwindigkeit, Beschleunigung und Position im Raum – Messtechnik und Sensorik.* https://messtechnik-und-sensorik.org/8-sensoren-fuer-

drehzahl-geschwindigkeit-beschleunigung-und-position-im-raum/. Version: 2020. – letzter Zugriff: 29.12.2020

[8] DerStandard: *Wie „WASD“ zum Standard im PC-Gaming wurde*. https://www.derstandard.de/story/2000077219661/wie-wasd-zum-standard-im-pc-gaming-wurde. Version: 2018. – letzter Zugriff: 04.01.2021

[9] Dlaboha, Meike: *vorles2*. https://www.physik.uni-muenchen.de/lehre/vorlesungen/wise_08_09/EP/vorlesung/vorlesung2.pdf. Version: 2012. – letzter Zugriff: 13.12.2020

[10] Ehlers, Kristian ; Klüssendorff, Jan: Self-scaling Kinematic Hand Skeleton for Real-time 3D Hand-finger Pose Estimation. In: *VISAPP 2015 - 10th International Conference on Computer Vision Theory and Applications; VISIGRAPP, Proceedings* 2 (2015), 01, S. 185–196. http://dx.doi.org/10.5220/0005257501850196. – DOI 10.5220/0005257501850196

[11] Eller, Conrad: *Holzmann/Meyer/Schumpich Technische Mechanik Kinematik und Kinetik*. 13. überarb. u. erw. Aufl. Wiesbaden : Springer Fachmedien Wiesbaden, 2019. – 23 S. http://dx.doi.org/10.1007/978-3-658-25587-9. http://dx.doi.org/10.1007/978-3-658-25587-9. – ISBN 978–3–658–25586–2

[12] Ferveur, Nicolas: *TEA - T-Server Manual*. 2018

[13] Feuerhake, Udo: *Erfassung von Trajektorien und Erkennung von Bewegungsmustern*. Version: 2018. http://publikationen.badw.de/de/044982151/pdf/CC%20BY letzter Zugriff: 19.09.2020

[14] Fitzpatrick, Martin: *Create GUI Applications with Python & Qt5: The hands-on guide to making apps with Python*. https://paddle.s3.amazonaws.com/fulfillment_downloads/16090/561130/D6eHc7VCSfGrmsulClgK_create-gui-applications-pyqt5.pdf. Version: 2020. – letzter Zugriff: 29.12.2020

[15] Hamer, Henning ; Koller-Meier, Esther ; Van Gool, Luc: Tracking a Hand Manipulating an Object. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2009, S. 1475–1482

[16] Hegger, Frederik ; Hochgeschwender, Nico ; Kraetzschmar, Gerhard ; Plöger, Paul: People Detection in 3D Point Clouds Using Local Surface Normals. In: *RoboCup* Bd. 7500, 2012, S. 154–165

[17] Helfrich, Rudolf ; DAA-Stiftung Bildung und Beruf (Hrsg.): *Digitalisierung und Technisierung der Pflege in Deutschland ohne Umschlag.* https://www.daa-stiftung.de/fileadmin/user_upload/digitalisierung_und_technisierung_der_pflege_2.pdf. Version: 2017. – letzter Zugriff: 05.01.2021

[18] Krüger-Brand, Heike E.: Technisierung der Medizin: "Die Technik ist uns auf den Leib gerückt". In: *Deutsches Ärzteblatt International* 111 (2014), Nr. 50. https://www.aerzteblatt.de/int/article.asp?id=165615. – letzter Zugriff: 22.09.2020

[19] Lehment, Nicolas ; Arsic, Dejan ; Kaiser, Moritz ; Rigoll, Gerhard: Automated pose estimation in 3D point clouds applying annealing particle filters and inverse kinematics on a GPU. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops, CVPRW 2010*, 2010, S. 87–92

[20] Lindanis, Meret ; Github (Hrsg.): *Question: Ask If Visualizer Is Closed: Issue #2787 - intel-isl/Open3D.* Forumsbeitrag. https://github.com/intel-isl/Open3D/issues/2787. Version: 2020. – letzter Zugriff: 23.12.2020

[21] Luhmann, Thomas: *Nahbereichsphotogrammetrie : Grundlagen — Methoden — Beispiele.* 4., neu bearb. u. erw. Aufl. 2018

[22] Melax, Stan ; Keselman, Leonid ; Orsten, Sterling: *Dynamics Based 3D Skeletal Hand Tracking.* https://www.researchgate.net/publication/317061847_Dynamics_Based_3D_Skeletal_Hand_Tracking. Version: 05 2017. – letzter Zugriff: 10.08.2020

[23] Microsoft®: *Informationen zu Azure Kinect DK.* https://opdhsblobprod01.blob.core.windows.net/contents/4a6d75bb3af747de838e6ccc97c5d978/6386e729e25c7fe595ebe5f2cb162455?sv=2018-03-28&sr=b&si=ReadPolicy&sig=unSi4TQk%2F1qoYw8mOMCft0tqTZ8lwvFp7NL6RECSwFQ%3D&st=2020-10-13T13%3A35%3A53Z&se=2020-10-14T13%3A45%3A53Z. Version: 2020. – letzter Zugriff: 13.10.2020

[24] MICROSOFT®: *k4abt_simple_3d_viewer von Azure Kinect Body Tracking SDK [Software]*. https://docs.microsoft.com/en-us/azure/kinect-dk/body-sdk-download. Version: 2020. – letzter Zugriff: 20.11.2020

[25] MUELLER, Franziska ; MEHTA, Dushyant ; SOTNYCHENKO, Oleksandr ; SRIDHAR, Srinath ; CASAS, Dan ; THEOBALT, Christian: Real-Time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In: *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017, S. 1163–1172

[26] OPEN3D: *About Open3D — Open3D 0.10.0 documentation*. http://www.open3d.org/docs/0.10.0/introduction.html. Version: 2020. – letzter Zugriff: 20.12.2020

[27] OPEN3D: *Azure Kinect with Open3D — Open3D 0.10.0 documentation*. http://www.open3d.org/docs/0.10.0/tutorial/Basic/azure_kinect.html. Version: 2020. – letzter Zugriff: 20.12.2020

[28] OPEN3D: *Getting Started — Open3D 0.10.0 documentation*. http://www.open3d.org/docs/0.10.0/getting_started.html. Version: 2020. – letzter Zugriff: 20.12.2020

[29] OPEN3D: *Open3D 0.10.0 is out!* http://www.open3d.org/2020/05/19/open3d-0-10/. Version: 2020. – letzter Zugriff: 20.12.2020

[30] OPEN3D: *Open3D: A Modern Library for 3D Data Processing — Open3D 0.12.0 documentation*. http://www.open3d.org/docs/release/index.html. Version: 2020. – letzter Zugriff: 28.12.2020

[31] OPEN3D: *Transformation — Open3D 0.10.0 documentation*. http://www.open3d.org/docs/0.10.0/tutorial/Basic/transformation.html. Version: 2020. – letzter Zugriff: 06.01.2021

[32] OPEN3D: *Visualization — Open3D 0.10.0 documentation*. http://www.open3d.org/docs/0.10.0/tutorial/Basic/visualization.html. Version: 2020. – letzter Zugriff: 03.01.2021

[33] REIDENBACH, Hans-Dieter: *Leitfaden „Sichtbare und infrarote Strahlung“*. https://www.fs-ev.org/fileadmin/user_upload/04_Arbeitsgruppen/08_Nichtionisierende_Strahlung/02_Dokumente/Leitfaeden/Leitfaden-SB-IR-AKNIR-15122011_b.pdf. Version: 2011. – letzter Zugriff: 09.11.2020

[34] Romero, Javier ; Kjellström, Hedvig ; Kragic, Danica: Hands in Action: Real-Time 3D Reconstruction of Hands in Interaction with Objects. In: *Proceedings - IEEE International Conference on Robotics and Automation*, 2010, S. 458–463

[35] Sridhar, Srinath ; Mueller, Franziska ; Zollhöfer, Michael ; Casas, Dan ; Oulasvirta, Antti ; Theobalt, Christian: Real-time Joint Tracking of a Hand Manipulating an Object from RGB-D Input. In: *European Conference on Computer Vision* Bd. 9906, 2016. – ISBN 978–3–319–46474–9, S. 294–310

[36] Sárándi, István ; Linder, Timm ; Arras, Kai O. ; Leibe, Bastian: *How Robust is 3D Human Pose Estimation to Occlusion?* https://www.researchgate.net/publication/327280543_How_Robust_is_3D_Human_Pose_Estimation_to_Occlusion. Version: 2018. – letzter Zugriff: 10.08.2020

[37] Steinebach, Tim ; Grosse, Eric H. ; Glock, Christoph H. ; Wakula, Jurij ; Lunin, Alexander: Accuracy evaluation of two markerless motion capture systems for measurement of upper extremities: Kinect V2 and Captiv. In: *Human Factors and Ergonomics in Manufacturing* 30 (2020), Nr. 4, 291–302. http://dx.doi.org/10.1002/hfm.20840. – DOI 10.1002/hfm.20840. – ISSN 1520–6564. – letzter Zugriff: 14.10.2020

[38] Stricker, Didier: *Computer-Vision-basierte Tracking- und Kalibrierungsverfahren für Augmented Reality.* https://tuprints.ulb.tu-darmstadt.de/288/1/stricker_diss.pdf. Version: 2002. – letzter Zugriff: 07.01.2021

[39] TEA: *Quick setup guide T-SENS Motion.* 2017

[40] TEA: *Captiv L7000 [Software].* 2020

[41] Techniker Krankenkasse: *Gesundheitsreport 2019: Pflegefall Pflegebranche? So geht's Deutschlands Pflegekräften.* https://www.tk.de/resource/blob/2059766/2ee52f34b8d545eb81ef1f3d87278e0e/gesundheitsreport-2019-data.pdf. Version: 2019. – letzter Zugriff: 22.09.2020

[42] Thiele, Eike ; Laserscanning Europe (Hrsg.): *Punktwolken Auswertung.* https://www.laserscanning-europe.com/de/content/punktwolken-auswertung. Version: 2013. – letzter Zugriff: 05.01.2021

[43] Ulbricht, Christiane: *Human Motion Capture*. `https://silo.tips/download/human-motion-capture`. Version: 2020. – letzter Zugriff: 28.12.2020

[44] Vicon: *Motion Capture Cameras | The Full Range from Vicon*. `https://www.vicon.com/hardware/cameras/`. Version: 2021. – letzter Zugriff: 01.01.2021

[45] Wearable Sensing: *Captiv Motion*. `https://wearablesensing.com/products/captiv-motion/`. Version: 2020. – letzter Zugriff: 05.01.2021

[46] Weisel, Christian: *Entwicklung eines markerbasierten Motion Capturing-Systems fur Echtzeitanwendungen*. `http://digdok.bib.thm.de/volltexte/2005/3321/pdf/diplomarbeit.pdf`. Version: 2004. – letzter Zugriff: 28.12.2020

[47] Willman, Joshua M.: *Beginning PyQt : A Hands-on Approach to GUI Programming*. 2020 (Springer eBook Collection). – ISBN 9781484258576

[48] XboxDev: *Azure Kinect nun auch in Deutschland, Japan und Großbritannien erhältlich*. `https://xboxdev.com/azure-kinect-nun-auch-in-deutschland-japan-und-grossbritannien-erhaeltlich/`. Version: 2020. – letzter Zugriff: 09.11.2020

[49] Ye, Mao ; Wang, Xianwang ; Yang, Ruigang ; Ren, Liu ; Pollefeys, Marc: Accurate 3D Pose Estimation From a Single Depth Image. In: *Proceedings of the IEEE International Conference on Computer Vision*, 2011, S. 731–738

[50] Zhang, Licong ; Sturm, Jurgen ; Cremers, Daniel ; Lee, Dongheui: Real-time human motion tracking using multiple depth cameras. In: *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2012. – ISBN 978–1–4673–1737–5, S. 2389–2395

[51] Zhou, Qian-Yi ; Park, Jaesik ; Koltun, Vladlen: Open3D : A Modern Library for 3D Data Processing. In: *arXiv:1801.09847* (2018). `http://arxiv.org/pdf/1801.09847v1`. – letzter Zugriff: 03.01.2021

Erklärung

Hiermit erkläre ich an Eides statt, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Außerdem versichere ich, dass ich die allgemeinen Prinzipien wissenschaftlicher Arbeit und Veröffentlichung, wie sie in den Leitlinien guter wissenschaftlicher Praxis der Carl von Ossietzky Universität Oldenburg festgelegt sind, befolgt habe.

Oldenburg, den 11. Januar 2021

Meret Björk Lindanis
Matrikelnummer 4539617