

FH Aachen

Faculty of Electrical Engineering and Computer Science

Interdisziplinäres Projekt

Markerlose Bewegungserfassung mit Anwendung in OpenSim

Schulte-Glade, Philipp Stock, Maximilian
Tulus, Isabelle Chu, Wenkai
Bahrouni, Alaeddin Mölleken, Adrian

Aachen, 19. Januar 2026

Prüferin: Kolditz, Melanie, Prof. Dr.-Ing

Inhaltsverzeichnis

1 Einleitung	3
1.1 Problemstellung und Motivation	3
1.2 Was ist markerlose Bewegungserfassung?	4
2 Hauptteil	5
2.1 Stand der Technik/Modelle für markerlose Bewegungserfassung	5
2.2 Was ist OpenSim?	6
2.3 Vergleich zweier Ansätze zur markerlosen Pose-Erfassung	11
2.4 Ablauf bei der Verwendung von MediaPipe	12
2.4.1 Softwareversionen und Module	14
2.4.2 Videos	15
2.4.3 MediaPipe Modell	16
2.4.4 CSV aus MediaPipe Daten	17
2.4.5 Skalierung der MediaPipe Daten	18
2.4.6 Marker Datei aus CSV	20
2.4.7 Verwendung von Marker Datei in OpenSim	20
2.4.8 Genauigkeit der 3D-Pose-Schätzung von MediaPipe	21
2.5 Ablauf bei der Verwendung von Pose2Sim	22
2.5.1 Übersicht des Verfahrens	22
2.5.2 Systemvoraussetzungen	22
2.5.3 Installation von Pose2Sim	22
2.5.4 Verwendung des pose2sim_cli.py	23
2.5.5 Erstellen eines OpenSim-Modells mit Pose2Sim	23
2.5.6 Datenverarbeitung in OpenSim	29
2.5.7 Modell-Import und Vorbereitung	29
2.5.8 Modell-Erstellung und Skalierung	29
2.5.9 Inverse Kinematik (IK)	30
2.5.10 Visualisierung der Ergebnisse in OpenSim	30
2.5.11 Ergebnisse von Pose2Sim	31
2.6 Genauigkeitsvergleich der beiden Ansätze	35
3 Schluss	38
3.1 Fazit	38
3.2 Ausblick	41
4 Literaturverzeichnis	43

1 Einleitung

1.1 Problemstellung und Motivation

Die Analyse und Rekonstruktion menschlicher Bewegungen spielt in vielen Bereichen eine wichtige Rolle, darunter Sportwissenschaft, Rehabilitation, Animation und Robotik. Traditionell werden hierfür markerbasierte Bewegungserfassungssysteme eingesetzt, die jedoch mit hohen Kosten, aufwändigen Vorbereitungen und eingeschränkter Flexibilität verbunden sind. In den letzten Jahren haben sich markerlose Methoden zur Pose-Erfassung entwickelt, die diese Herausforderungen adressieren und eine einfachere sowie kostengünstigere Alternative bieten. Die markerlose Erfassung menschlicher Posen gewinnt zunehmend an Bedeutung, da sie eine flexible und alltagstaugliche Alternative zu markerbasierten Bewegungserfassungssystemen darstellt und vielfältige Anwendungen bietet.

Markerlose Bewegungserfassungssysteme liefern in der Regel kinematische Informationen über die räumliche Position und Bewegung einzelner Körpersegmente oder Gelenkpunkte. Diese Verfahren ermöglichen eine effiziente Rekonstruktion der menschlichen Bewegungen, sind jedoch auf die reine Beschreibung der Bewegung beschränkt. Aussagen über Biomechanische Größen wie Gelenkwinkel, Gelenkmomente, Muskelkräfte oder innere Belastungen sind mit solchen Erfassungssystem alleine nicht möglich, wobei für dieses Projekt die Gelenkwinkel im Fokus stehen. Für weitergehende Analysen von menschlichen Bewegungen ist daher eine zusätzliche biomechanische Modellierung erforderlich, welche den menschlichen Bewegungsapparat unter Berücksichtigung seiner anatomischen und physikalischen Zusammenhänge abbildet. Biomechanische Simulationsumgebungen wie z.B. OpenSim, die in diesem Projekt verwendet werden, ermöglichen es, aus den kinematischen Eingangsdaten (Landmarks) dynamische Kenngrößen abzuleiten und Bewegungen hinsichtlich ihrer mechanischen Belastung zu untersuchen. Die Kombination markerloser Bewegungserfassungssysteme mit biomechanischen Modellen erlaubt somit ganzheitliche Analysen menschlicher Bewegungen, bei der die Vorteile einer flexiblen und kostengünstigen Datenerfassung mit der Aussagekraft physikalisch fundierter Simulationen verbunden werden kann.

1.2 Was ist markerlose Bewegungserfassung?

Ziel der Bewegungserfassung ist die realitätsnahe Abbildung des menschlichen Bewegungsapparats, um (menschliche) Bewegungen digital zu erfassen und auszuwerten. Die Bewegungserfassung kann dabei entweder markerbasiert oder markerlos erfolgen.

Bei markerbasierten Verfahren werden retroreflektierende Marker an definierten anatomischen Landmarken einer Person angebracht und von mehreren Kameras erfasst. [1] Alternativ können inertiale Messeinheiten (IMUs) eingesetzt werden, die Bewegungsdaten ohne optische Kamerasyteme erfassen, indem sie mehrere integrierte Sensoren kombinieren. Die gewonnenen Messdaten werden in beiden Fällen mithilfe spezieller Software verarbeitet und auf digitale Modelle übertragen. [2]

Markerlose Bewegungserfassung verzichtet vollständig auf physische Markierungen am Körper und erfolgt ausschließlich bildbasiert. Bewegungen werden mithilfe von Kameras aufgezeichnet und softwaregestützt ausgewertet, wobei menschliche Körpersegmente und Gelenkpunkte durch Bildverarbeitungsverfahren identifiziert werden. Die ermittelten zweidimensionalen Positionsdaten werden im Anschluss mittels Triangulation dreidimensional rekonstruiert. [1]

Markerlose Bewegungserfassung ermöglicht eine größere Flexibilität in der Anwendung, ist weniger zeit- und kostenintensiv und damit für den Alltagsgebrauch besser geeignet. [1] Im Vergleich zu markerbasierten Systemen ist sie jedoch mit einer geringeren Genauigkeit sowie einer höheren Latenz verbunden. [3, S. 275]

Die Einsatzgebiete der markerlosen Bewegungserfassung umfassen sowohl die Sportwissenschaften als auch die Medizin, insbesondere in den Bereichen Rehabilitation, Orthopädie und Ganganalyse. [4, S. 86] Darüber hinaus findet sie Anwendung in der Film- und Spieleentwicklung, beispielsweise in der Animation virtueller Charaktere, sowie in der Robotik und der Mensch-Maschine-Interaktion. [1]

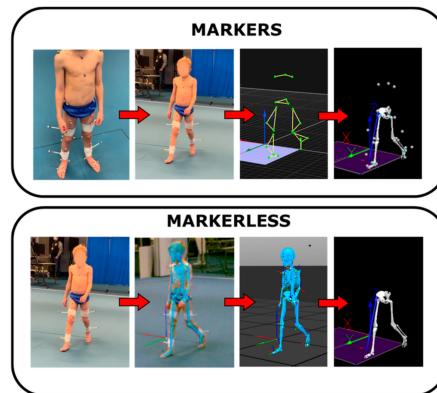


Abbildung 1.1: Vergleich der markerbasierten (oben) und markerlosen (unten) Bewegungserfassung von der Datenerfassung bis zur dreidimensionalen Rekonstruktion. (Quelle: [5])

2 Hauptteil

2.1 Stand der Technik/Modelle für markerlose Bewegungserfassung

Der aktuelle Stand der Technik basiert überwiegend auf Verfahren der sogenannten Human Pose Estimation (HPE). Dabei werden anatomisch definierte Gelenkpunkte, also sogenannte Keypoints, direkt aus Bild- oder Videodaten geschätzt. Moderne Systeme nutzen hierfür überwiegend Deep-Learning-Modelle, insbesondere Convolutional Neural Networks (CNN), die Gelenkpositionen in Form probabilistischer Heatmaps vorhersagen. Diese Verfahren erreichen in der zweidimensionalen Pose-Schätzung inzwischen eine sehr hohe Genauigkeit und Robustheit gegenüber variierenden Hintergründen, Beleuchtungsverhältnissen und Körperperformen.

Die dreidimensionale markerlose Bewegungserfassung stellt weiterhin eine größere technische Herausforderung dar. Ein grundlegendes Problem besteht in der sogenannten Tiefenambiguität, da aus monokularen RGB-Aufnahmen keine eindeutige Tiefeninformation ableitbar ist. Aktuelle Forschungsansätze begegnen diesem Problem durch die Nutzung mehrerer synchronisierter Kameras, durch lernbasierte Rekonstruktion dreidimensionaler Posen aus großen annotierten Datensätzen oder durch zeitliche Modellierung ganzer Bewegungssequenzen. Insbesondere die Kombination räumlicher und zeitlicher Informationen hat sich als wesentlich für robuste 3D-Rekonstruktionen erwiesen.

Neben rein monokularen Ansätzen kommen zunehmend Multi-View-Systeme zum Einsatz, bei denen mehrere Kameras aus unterschiedlichen Blickwinkeln verwendet werden. Durch geometrische Triangulation lassen sich dabei dreidimensionale Gelenkpositionen präziser bestimmen, insbesondere bei Bewegungen mit starker Selbstverdeckung. Diese Systeme erreichen in kontrollierten Umgebungen eine Genauigkeit, die in bestimmten Anwendungen bereits mit markerbasierten Motion-Capture-Systemen vergleichbar ist. Allerdings gehen sie mit höherem Hardware-, Kalibrierungs- und Rechenaufwand einher.

Ein weiterer Entwicklungszweig des Stands der Technik sind hybride Verfahren, die visuelle RGB-Daten mit Tiefeninformationen aus Time-of-Flight- oder Stereo-Kameras kombinieren. Durch die explizite Messung der Entfernung zum Objekt kann die dreidimensionale Rekonstruktion stabilisiert und die Fehleranfälligkeit bei Abdeckung der Sicht reduziert werden. Solche Systeme finden insbesondere in der Rehabilitation, der ergonomischen Analyse und der Mensch-Maschine-Interaktion Anwendung.

In den letzten Jahren hat sich zudem der Einsatz fortgeschrittenen neuronaler Architekturen wie Graph-Neural-Networks und Transformer-Modelle etabliert. Diese ermöglichen eine explizite

Modellierung der kinematischen Abhängigkeiten zwischen einzelnen Gelenken sowie der zeitlichen Dynamik von Bewegungen. Aktuelle Forschungsarbeiten untersuchen darüber hinaus den Einsatz generativer und diffusionsbasierter Modelle, um plausible 3D-Bewegungen auch bei unvollständigen oder verrauschten Eingangsdaten zu rekonstruieren.

Trotz erheblicher Fortschritte bestehen weiterhin offene Herausforderungen. Dazu zählen die robuste Handhabung von Sichtabdeckung, die Übertragbarkeit auf neue Personen und Umgebungen sowie die biomechanische Genauigkeit der rekonstruierten Bewegungen. Insbesondere für Anwendungen, bei denen Gelenkmomente, Muskelkräfte oder Belastungen bestimmt werden sollen, ist die Kopplung markerloser Bewegungserfassung mit biomechanischen Modellen weiterhin Gegenstand aktueller Forschung.

Zur Analyse und Simulation menschlicher Bewegungen existieren verschiedene etablierte Systeme digitaler Menschmodelle. Zu den bekanntesten zählen AnyBody, OpenSim, Santos, LifeMOD und Dynamicus. Ein zentraler Schwerpunkt der Arbeit liegt auf der Erstellung eines reproduzierbaren und praxisnahen Tutorials, das die vollständige Pipeline der markerlosen Bewegungserfassung beschreibt. Dieses Tutorial umfasst die Installation, Konfiguration und Anwendung von MediaPipe sowie von Pose2Sim in Kombination mit OpenSim, beginnend bei der Rohdatenerfassung über die Poseschätzung bis hin zur biomechanischen Auswertung. Durch eine schrittweise und systematische Darstellung soll der Einstieg für neue Anwender erleichtert und typische Fehlerquellen vermieden werden. [6] [7] [8] [9]

2.2 Was ist OpenSim?

OpenSim ist eine kostenlose Open Source Software Plattform für die Modellierung, Simulierung, Kontrollierung und Analyse für das Neuro-Muskel-Skelett-System von Menschen, Tieren und Robotern. Simuliert wird deren Interaktion und Bewegung in der Umgebung, in der sie sich befinden.

Muskel-Skelett-Modelle

OpenSim ermöglicht das Erstellen und Verändern von eigenen 3D-Modellen des menschlichen oder tierischen Bewegungsapparats, inklusive Knochen, Gelenken, Muskeln, Sehnen und anderer Strukturen. Das graphical user interface kurz GUI erlaubt es, die eigenen Modelle oder bereits vorhandene zu laden, Visualisierung und ihre Einstellungen im Detail zu bearbeiten.

OpenSim's Muskelmodelle erfassen die aktiven und passiven generierten Kraft Eigenschaften der Muskeln, die auf gut getesteten Modellen der Muskel Sehnen Dynamik aus der Literatur basieren.

Die Softwareplattform kommt mit einer großen Anzahl an Muskel Skelett Modellen, die beim Herunterladen bereits zur Verfügung gestellt sind, bei denen es sich um Modelle des menschlichen Oberkörpers und der unteren Extremitäten handelt. Zusätzlich zu den Modellen, die in Forschungsqualität angeboten werden, gibt es weitere teilweise stark vereinfachte Modelle für die Erstellung von Prototypen und zum Lehren. Außerdem bietet OpenSim die Möglichkeit von Benutzern selbst erstellte Modelle aus einer Online Bibliothek zu laden.

Zum Analysieren und Simulieren von Modellen und Bewegungen gibt es Tools, um Markerdaten, Gelenkkinematiken und externe Kräfte zu importieren und ebenfalls zu visualisieren. Einige Benutzer haben Toolboxen erstellt und geteilt, die mit verschiedenen gängigen Bewegungserfassungssystemen kompatibel sind.

Tools

Besonders von Bedeutung für dieses Projekt sind zum einen das Scale-Tool und zum anderen das Inverse Kinematics (IK)-Tool.

Das Scale-Tool dient hierbei zur Anpassung der Modellgeometrie an die individuellen Eigenschaften der Körpermaße einer Versuchsperson. Die Skalierung basiert auf einer Kombination aus den gemessenen Abständen zwischen den experimentell erfassten x-y-z-Markerpositionen sowie einigen optional manuell vorgegebenen Skalierungsfaktoren. Die Markerpositionen werden in der Regel mit den in der Einleitung genannten Motion-Capture-Systemen aufgezeichnet. Das unskalierte Modell enthält korrespondierende virtuelle Marker, die an denselben anatomischen Landmarken wie die experimentellen Marker positioniert sind.

Während des Skalierungsprozesses werden die Abmessungen der einzelnen Körpersegmente so angepasst, dass die Abstände zwischen den virtuellen Markern (Modellmarker) und den gemessenen Abständen der experimentellen Marker entsprechen. Ergänzend oder alternativ dazu können zu der markerbasierten Skalierung auch manuell definierte Skalierungsfaktoren für einzelne Körpersegmente verwendet werden, die beispielsweise aus unabhängigen anthropometrischen Messungen stammen. Nach Abschluss der Skalierung ermöglicht das Scale-Tool zudem, ausgewählte oder alle virtuellen Marker gezielt zu verschieben, sodass sie exakt mit den experimentellen Markerpositionen übereinstimmen. Dadurch wird eine verbesserte Übereinstimmung zwischen dem gewählten Modell und den Versuchsdaten aus der Bewegungserkennung erreicht, womit eine Grundlage für nachfolgende Analysen geschaffen wird.

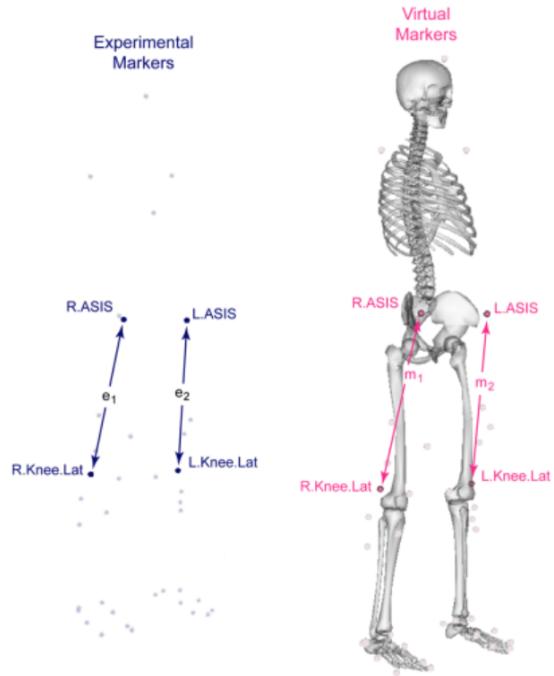


Abbildung 2.1: Experimentelle und virtuelle Marker Quelle: [10]

In der Abbildung 2.1 in Blau sind die experimentellen Marker dargestellt, in Pink die virtuellen Modellmarker nach der Skalierung. [10]

Die Inverse Kinematic (IK) Tool berechnet für jeden Zeitschritt (frame) der einer Bewegung die generalisierten Koordinaten des Modells, sodass dessen Pose bestmöglich mit den experimentell erfassten Marker- und gegebenenfalls Koordinatendaten übereinstimmt. Die beste Übereinstimmung wird mathematisch als ein gewichtetes Least Square Problem (kleinstes Fehlerquadrat) formuliert, bei dem sowohl Marker als auch Koordinatenfehler minimiert werden. (Bild der Rechnung) Die Markerfehler die dabei entstehen sind definiert als der euklidische Abstand zwischen der Position eines experimentell bestimmten Markers und der entsprechenden Markerposition am virtuellen Modell, nachdem dieses durch die berechnete generalisierte Koordinaten positioniert wurde. Wie bereits erwähnt, ist jedem Marker ein Gewicht zugeordnet, das bestimmt, wie stark dessen Abweichung in die Gesamtfehlerfunktion eingeht. Analog dazu werden auch Koordinatenfehler definiert, die die Differenz zwischen einem experimentell vorgegebenen Koordinatenwert, beispielsweise einem Gelenkwinkel, und dem durch die IK berechneten Wert beschreiben. Außerdem können die Experimentellen Koordinaten, die wiederum aus den Motion Capture Systemen, speziellen Auswertealgorithmen oder zusätzlichen Messgeräten wie Goniometern stammen, alternativ auch mit festen Sollwerten vorgegeben werden. Feste Sollwerte sind insbesondere dann sinnvoll, wenn bestimmte Gelenkwinkel konstant bleiben sollen. Bei der IK-Berechnung wird zwischen vorgeschriebenen und nicht vorgeschriebenen Koordinaten unterschieden. Vorgeschriebene Koordinaten besitzen einen bekannten Verlauf und werden während der IK nicht berechnet, sondern direkt auf

ihre vorgegebenen Werte gesetzt (z.B. fixierte Gelenke). Nicht vorgeschriebene Koordinaten werden hingegen durch die IK bestimmt und gehen in das Optimierungsproblem (Least Square) ein. Wie bei den Markern können auch den Koordinaten unterschiedliche Gewichte zugewiesen werden, die deren Einfluss auf den Koordinatenfehler bestimmen.

Weighted Least Squares Equation

The weighted least squares problem solved by IK is

$$\min_{\mathbf{q}} \left[\sum_{i \in \text{markers}} w_i \left\| \mathbf{x}_i^{\text{exp}} - \mathbf{x}_i(\mathbf{q}) \right\|^2 + \sum_{j \in \text{unprescribed coords}} \omega_j (q_j^{\text{exp}} - q_j)^2 \right]$$

$q_j = q_j^{\text{exp}}$ for all prescribed coordinates j

Abbildung 2.2: Least Square Problem der IK (kleinstes Fehlerquadrat) Quelle: [11]

In Abbildung 2.2 bezeichnet \mathbf{q} den Vektor der zu bestimmenden generalisierten Koordinaten, wobei die Abweichung zwischen den experimentell gemessenen Markerpositionen und den modellbasierten Markerpositionen minimiert wird, während vorgegebene Koordinaten während der inversen Kinematik auf feste Werte gesetzt werden.

Die Optimierung ist sensitiv gegenüber der Wahl der Einheiten. In der verwendeten Modellumgebung werden Meter für Längen und Radiant für Winkel verwendet. Dies ist besonders beim Vergleich mit anderen IK-Verfahren relevant, die Winkel in Grad verarbeiten. Um vergleichbare Ergebnisse zu erhalten, müssen die Koordinatengewichte entsprechend an die verwendeten Winkeleinheiten angepasst werden. [11]

Weitere Tools in OpenSim sind das Inverse Dynamics (ID)-Tool, das Static Optimization (SO)-Tool und das Computed Muscle Control (CMC)-Tool, die aber für unser Projekt keine Rolle spielen.

Das Inverse Dynamics (ID)-Tool berechnet die Gelenkmomente, die erforderlich sind, um eine Bewegung auszuführen. Dazu werden die generalisierten Koordinaten, deren zeitliche Ableitungen (Geschwindigkeiten und Beschleunigungen) sowie externe Kräfte wie Bodenreaktionskräfte verwendet. Die berechneten Gelenkmomente können zur Analyse der mechanischen Belastung von Gelenken und zur Validierung von Muskelmodellen genutzt werden.

Mit der statischen Optimierung kann das Muskel-Redundanzproblem gelöst werden, basierend auf etablierten Verfahren aus der wissenschaftlichen Literatur.

Mit Hilfe des Computed Muscle Control (CMC)-Tool lassen sich muskelgetriebene Vorwärtssimulationen erzeugen. Diese Methode wurde erfolgreich für verschiedene Bewegungen wie das Gehen, Laufen, Radfahren, Springen und die Analyse pathologischer Gangmuster eingesetzt.

OpenSim ermöglicht außerdem eine detaillierte Analyse („Probing“) von Modellen und Simulationen. Dabei können Größen wie Gelenkwinkel, Muskelkräfte, Muskelhebelarme, Muskelarbeit oder die Bewegung des Körperschwerpunkts untersucht und grafisch dargestellt werden.

Zur Visualisierung bietet OpenSim eine grafische Benutzeroberfläche, in der nahezu alle Modellkomponenten angezeigt werden können. Externe Geometrien (z. B. STL- oder OBJ-Dateien) lassen sich importieren und mit integrierten Bild- und Video-Tools können anschauliche Darstellungen für Präsentationen und Publikationen erstellt werden.

2.3 Vergleich zweier Ansätze zur markerlosen Pose-Erfassung

In diesem Kapitel werden zwei Ansätze zur markerlosen Pose-Erfassung untersucht und miteinander verglichen: zum einen ein eigener Workflow mit MediaPipe als Ansatz zur 2D- und 3D-Poseschätzung sowie zum anderen ein fertiger Open Source Workflow für OpenSim [12] (mit Verwendung neuerer Modelle wie RTM Pose) zur Rekonstruktion biomechanischer Modelle. Die Motivation für diesen Vergleich liegt darin begründet, dass beide Varianten im Kern denselben algorithmischen Berechnungsschritten folgen, sich jedoch in ihrer systemseitigen Umsetzung unterscheiden. Während Pose2Sim als integrierte Lösung fungiert, in der die gesamte Prozesskette bereits implementiert ist, führt der in 2.4 betrachtete MediaPipe-Workflow die Einzelschritte sequenziell aus und beschränkt sich dabei auf die Nutzung eines monokularen Kameraystems.

Des Weiteren erfolgt ein Genauigkeitsvergleich, bei dem die Pose2Sim-Pipeline als Referenz dient, um die Eignung des MediaPipe-Ansatzes für biomechanische Anwendungen zu bewerten. Ergänzend wird die Robustheit der Pose2Sim-Pipeline gegenüber Störfaktoren wie Kleidung oder unterschiedlichen Beleuchtungsbedingungen analysiert.

Abschließend werden die jeweiligen Vor- und Nachteile der untersuchten Ansätze verglichen und geeignete Einsatzgebiete identifiziert. Aufbauend auf den Ergebnissen dieser Arbeit wird zudem eine potentielle weiterführende Aufgabe formuliert, die es einer nachfolgenden Projektgruppe ermöglicht, die vorgestellten Ansätze weiterzuentwickeln, zu validieren oder auf erweiterte Datensätze und Anwendungsszenarien anzuwenden.

2.4 Ablauf bei der Verwendung von MediaPipe

Zu Beginn brauchen wir aus unseren 2D Daten, in Form eines Videos von unserer Kamera, geschätzte 3D-Koordinaten von markanten Punkten am Körper (Keypoints). Wir haben dabei das BlazePose Modell verwendet welches die in Abbildung 2.3 dargestellten Keypoints liefert.

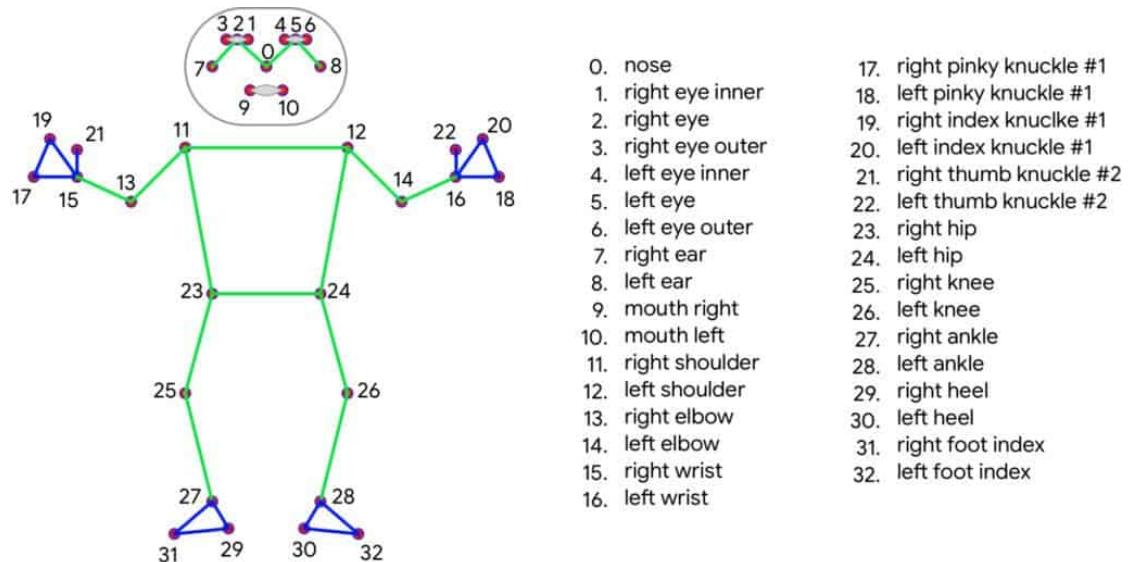


Abbildung 2.3

Für die vorliegende Arbeit wurde bewusst ein vereinfachtes Arm-Modell verwendet, das ausschließlich auf drei Markern basiert:

- 11. rechte Schulter
- 13. rechter Ellbogen
- 15. rechtes Handgelenk

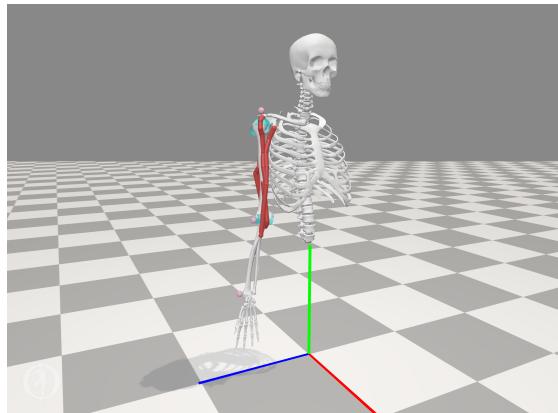


Abbildung 2.4: arm_model

Das in Abbildung 2.4 dargestellte arm_model stellt eine Erweiterung des Standard-OpenSim-Modells namens arm26 dar und wurde um zwei zusätzliche Freiheitsgrade erweitert. Damit hat das Modell die folgenden Gelenkwinkel:

- r_shoulder_elev
- r_shoulder_side
- r_elbow_flex
- r_elbow_rotation

Die Entscheidung für dieses reduzierte Modell ergab sich aus wiederholten, jedoch nicht zufriedenstellenden Versuchen, Ganzkörper-Posen mit der entwickelten Pipeline zu verarbeiten und konsistent darzustellen.

Die Hauptursache liegt in der aktuell verwendeten Skalierungsfunktion, die ausschließlich feste Segmentlängen berücksichtigen kann. Dadurch ist eine korrekte Skalierung dynamischer Verbindungen zwischen Ober- und Unterkörper nicht möglich, was eine realitätsgetreue Abbildung komplexer Ganzkörperbewegungen verhindert.

Unser Ziel ist es, diese drei Keypoints aus MediaPipe BlazePose auf ein Modell in OpenSim zu übersetzen und, mithilfe des Inversen Kinematik Tools von OpenSim [11], Gelenkwinkel von einer Bewegung herauszubekommen. Diese können wir dann mit den Gelenkwinkeln vom Pose2Sim Ansatz vergleichen.

Für dieses Ziel sind vier wesentliche Schritte notwendig:

1. Erhalten der Daten aus einem Video mithilfe von MediaPipe BlazePose
2. Skalierung
3. Erstellen einer Marker Datei für OpenSim
4. Inverse Kinematik

Das Ergebnis des Inversen Kinematik Tools von OpenSim ist dann eine Bewegungs Datei (.mot). In dieser finden wir dann die Gelenkwinkel zu bestimmten Zeitpunkten.

Nun folgt eine Beschreibung des Workflows.

2.4.1 Softwareversionen und Module

Wir haben uns für die Programmiersprache Python entschieden, da es eine einfache und schnelle Möglichkeit bietet, um Bildverarbeitungs Bibliotheken zu verwenden. Außerdem ist Python sehr verbreitet in der Machine Learning Community, was die Verwendung von MediaPipe erleichtert. Es werden folgende Softwareversionen verwendet:

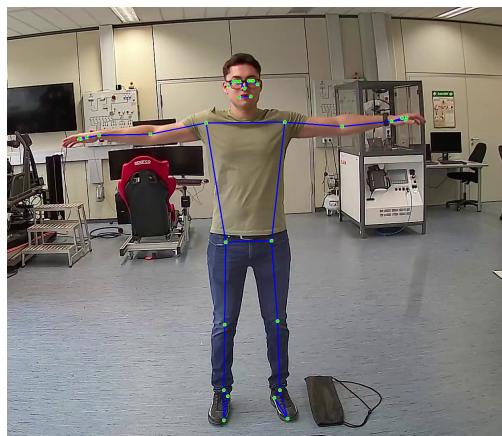
- Python 3.12.12
- csv (kommt mit python) 3.12.12
Standard-Modul zum Schreiben von Textdateien im "Comma Separated Values Format."
- math (kommt mit python) 3.12.12
Standard-Mathematik-Bibliothek von Python.
- cv2 4.12.0.88 (opencv-python)
Bibliothek für Computer Vision. Ist Notwendig zum lesen der Video Dateien und weiteren Bearbeitung.
- MediaPipe 0.10.21
Machine-Learning-Bibliotheken von Google. Wichtig für die Pose Schätzung.
- pandas 2.3.3
Modul für die Datenanalyse und Tabellenkalkulation. Wichtig für die Umwandlung von CSV in TRC Datei.
- numpy 1.26.4
Standardbibliothek für mathematische Berechnungen mit Vektoren und Matrizen. Wichtig für die Skalierung.

2.4.2 Videos

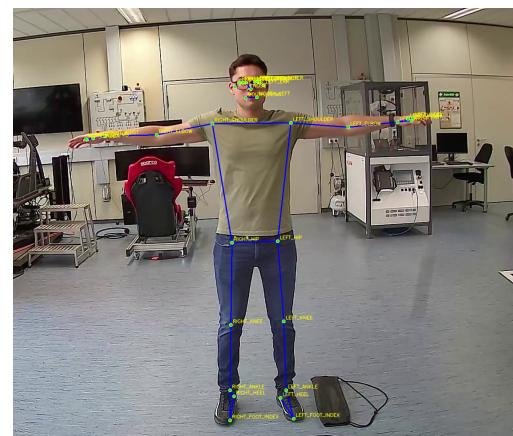
Da das MediaPipe BlazePose Modell in unserem Workflow nur mit Videos arbeitet, brauchen wir als erstes ein Video von **einer** Person. Für eine gute Aufnahme sollte folgendes beachtet werden:

- Die Person sollte komplett im Bild sein.
- Die Person sollte vom Hintergrund unterscheidbar sein.
- Die Beleuchtung sollte ausreichend sein, damit die Kamera die Person gut erkennen kann.
- Die Kamera sollte still stehen, damit keine Bewegungsunschärfe entsteht.
- Die Kamera sollte möglichst horizontal ausgerichtet sein, da ein starker Kamerawinkel dazu führen kann, dass MediaPipe die Körperhaltung fälschlicherweise als geneigt oder lehnend interpretiert.

Unsere erste Funktion gibt uns unsere aufgenommenen Videos mit eingezeichneten Skeletten wieder. Das ist für einen ersten Überblick interessant, um zu sehen ob MediaPipe Blaze Pose die Position der Keypoints richtig erkannt hat. Außerdem lässt sich schonmal feststellen wo Probleme bei der Erkennung auftreten. Beim Aufruf der Anzeige Funktion kann über einen Parameter eingestellt werden, ob die Marker Namen angezeigt werden (Abb. 2.5b) oder nicht (Abb. 2.5a). Zudem verfügt die Funktion über eine Flag-Variable, um das Video bei Bedarf zu drehen, falls es nicht in der korrekten Ausrichtung eingespeichert wurde.



(a) Ohne Marker-Namen



(b) Mit Marker-Namen

Abbildung 2.5: Vergleich der Video-Ausgabe aus dem MediaPipe Workflow

2.4.3 MediaPipe Modell

Das verwendete Pose-Modell wird zu Beginn des Workflows in der Funktion `create_pose` instanziert. Diese Funktion ermöglicht es, zu jedem Start einer Pose-Schätzung ein neues Modell zu erzeugen, das noch nicht auf zuvor verarbeiteten Daten basiert. Dadurch wird sichergestellt, dass beim Wechsel von der statischen zur dynamischen Aufnahme kein zeitlicher Zusammenhang zwischen den beiden Sequenzen besteht. Insbesondere wird verhindert, dass die Marker ausgehend von der Endposition der statischen Aufnahme zur Startposition der dynamischen Bewegung interpoliert werden oder „wandern“, wodurch konsistente und unabhängige Initialbedingungen für beide Aufnahmen gewährleistet werden.

In der folgenden Übersicht sind die für die Modellierung verwendeten Parameter aufgeführt. Diese Parameter bestimmen maßgeblich das Verhalten und die Genauigkeit des Modells. Im Folgenden werden daher die einzelnen Parameter mit den für die Ergebnisse dieser Arbeit verwendeten Werte dargestellt.

- `static_image_mode` (`False`): Gibt an, dass das Modell für Videosequenzen verwendet wird. Dadurch werden erkannte Posen über mehrere Frames hinweg verfolgt, was die Stabilität erhöht und die Rechenlast reduziert.
- `model_complexity` (2): Bestimmt die Komplexität des verwendeten Pose-Modells. Höhere Werte liefern in der Regel genauere Ergebnisse, benötigen jedoch mehr Rechenleistung.
- `enable_segmentation` (`False`): Deaktiviert die Personensegmentierung im Bild. Da nur die Körperlandmarks benötigt werden, wird auf diese zusätzliche Berechnung verzichtet.
- `min_detection_confidence` (0.5): Legt die minimale Konfidenz fest, ab der eine Pose als erfolgreich erkannt gilt.
- `min_tracking_confidence` (0.5): Legt die minimale Konfidenz für das Tracking der Pose zwischen aufeinanderfolgenden Frames fest.
- `smooth_landmarks` (`True`): Aktiviert eine zeitliche Glättung der Landmark-Positionen über mehrere Frames.

2.4.4 CSV aus MediaPipe Daten

Die Funktion `pose_estimation` ruft das MediaPipe BlazePose Modell auf dem Video auf und extrahiert die 3D Positionen der definierten Keypoints. Die meisten Bildverarbeitungs-Bibliotheken, unter anderem auch MediaPipe BlazePose, haben den Ursprung oben links im Bild. Also wird die y-Achse einmal invertiert, sodass das Modell nachher nicht auf dem Kopf steht.

Eine Fehlerquelle ist hier, die Missachtung der Orientierung der Person zur Kamera im Vergleich zum OpenSim Modell. Dadurch kann es passieren, dass die Daten nach import in OpenSim nicht mit dem Modell übereinstimmen. Um dem entgegenzuwirken, gibt es die Möglichkeit das Video um festgelegte Winkel um die y-Achse zu rotieren. Rotiert wird dabei gegen den Uhrzeigersinn um die im Parameter `rotate_around_y_axis` angegebene Gradzahl (0, 90, 180 oder 270). Dabei müssen die statische Aufnahme und die Bewegungsaufnahme gegebenfalls unterschiedlich rotiert werden, je nachdem wie die Person in der jeweiligen Aufnahme zur Kamera steht.

Als Ausgabe erhält man dann eine CSV Datei mit Positions Informationen welche für die Marker Datei später wichtig sind. Die generierte CSV-Datei folgt einer strikten Struktur, die dynamisch anhand der definierten Marker-Liste erstellt wird.

- **Zeile 1:** Metadaten zur Framerate (FPS).
- **Zeile 2 (Header):** Spaltenbeschriftungen. Für jeden Marker n werden automatisch drei Spalten (`Marker n _X`, `Marker n _Y`, `Marker n _Z`) generiert.
- **Zeile 3 ff.:** Die Positionsdaten pro Frame.

```
1 #FPS ,30.0
2 Frame ,Marker1_X ,Marker1_Y ,Marker1_Z ,Marker2_X ,Marker2_Y ,Marker2_Z
3 0 ,0.12 ,-0.55 ,0.80 ,0.45 ,-0.30 ,0.12
4 1 ,0.13 ,-0.54 ,0.81 ,0.46 ,-0.31 ,0.13
5 2 ,0.13 ,-0.54 ,0.81 ,0.46 ,-0.31 ,0.13
6 ...
```

Listing 2.1: Allgemeine Struktur der Ausgabedatei

2.4.5 Skalierung der MediaPipe Daten

Da MediaPipe die erkannten 3D-Koordinaten in einem normierten Raum zurückgibt, müssen diese Koordinaten skaliert werden, um sie in OpenSim verwenden zu können. Dabei geht es hauptsächlich darum, die Längen von Körperteilen korrekt darzustellen und nicht über die frames variieren zu lassen und den Ursprung der MediaPipe Daten auf den Ursprung von OpenSim zu verschieben.

Für die Ursprungsverschiebung muss ein Marker als Referenz gewählt und dessen Position in OpenSim als Referenzkoordinate übergeben werden. Dann wird die durchschnittliche Koordinate dieses Markers über die statische Aufnahme berechnet und der Differenzvektor zu der OpenSim Referenzkoordinate bestimmt. Dieser Vektor wird dann auf alle MediaPipe Koordinaten der statischen Aufnahme addiert, um den Ursprung zu verschieben.

Bei der Bewegungsaufnahme wird hingegen ein abweichendes Vorgehen gewählt. Hier wird der Differenzvektor nicht aus der gemittelten statischen Aufnahme bestimmt, sondern aus der Position des Referenzmarkers im ersten Frame der Bewegung. Dieser Vektor dient als initiale Ursprungsverschiebung und wird auf alle nachfolgenden Frames angewendet. Auf diese Weise startet die Bewegung im Ursprung des OpenSim-Modells, während gleichzeitig eine Translation des Körpers im Raum über die Zeit erhalten bleibt.

Die Skalierung der Körperteile kann auf verschiedene Arten erfolgen:

- **Skalierung über die durchschnittliche Länge in der statischen Aufnahme:** Hier werden die Körperteile einfach auf die durchschnittlichen Längen in der statischen Aufnahme skaliert.
- **Skalierung über eine bekannte Referenzlänge:** Hierbei wird eine bekannte Länge (z.B. die Länge eines Armes) verwendet. Dann wird ein Faktor zwischen dieser und der durchschnittlichen Länge des verwendeten Körperteils in der statischen Aufnahme berechnet und anschließend auf alle durchschnittlichen Längen aller Körperteile angewendet, um die skalierten Längen zu erhalten.

Dies ist besonders nützlich, wenn die Person in der Aufnahme eine erhöhte Distanz zur Kamera hat.

Um die eigentliche Skalierung durchführen zu können, müssen zuerst die zu skalierenden Körperteile definiert werden. Für eine saubere Skalierung und um Winkel zwischen den Körperteilen beizubehalten, können die Körperteile in Gruppen zusammengefasst werden. Jede Gruppe kann aus mehreren verbundenen Körperteilen bestehen, die zusammen skaliert werden. Im Code gibt es dafür das Array `limb_groups`, welches die Indizes der zu skalierenden Körperteile im `landmark_map` dictionary als Tupel übergibt. Bei einem einfachen Arme Modell, bestehend aus 6 Markern (jeweils Schulter, Ellbogen, Handgelenk), könnte dies wie folgt aussehen:

```
idx Marker_name
0    r_shoulder
```

```

1 r_elbow
2 r_wrist
3 l_shoulder
4 l_elbow
5 l_wrists

limb_groups = [
    [(0, 1), (1, 2)],
    [(3, 4), (4, 5)]
]

```

Bei der Skalierung wird für jedes Tupel der zweite Marker an den ersten angepasst, daher ist die Reihenfolge hier entscheidend. Nach der Verschiebung eines Markers wird ein Verschiebungsvektor aufaddiert, welcher vor der Skalierung des nächsten Körperteils auch auf den zweiten Marker des Tupels angewendet wird, um die Winkel beizubehalten.

Wenn der entsprechende Parameter in der `scale` Funktion gesetzt ist, werden die zu skalierenden Längen einmal vor (Abb. 2.6a) und nach (Abb. 2.6b) der Skalierung in einem Diagramm dargestellt, um die Skalierung zu überprüfen.

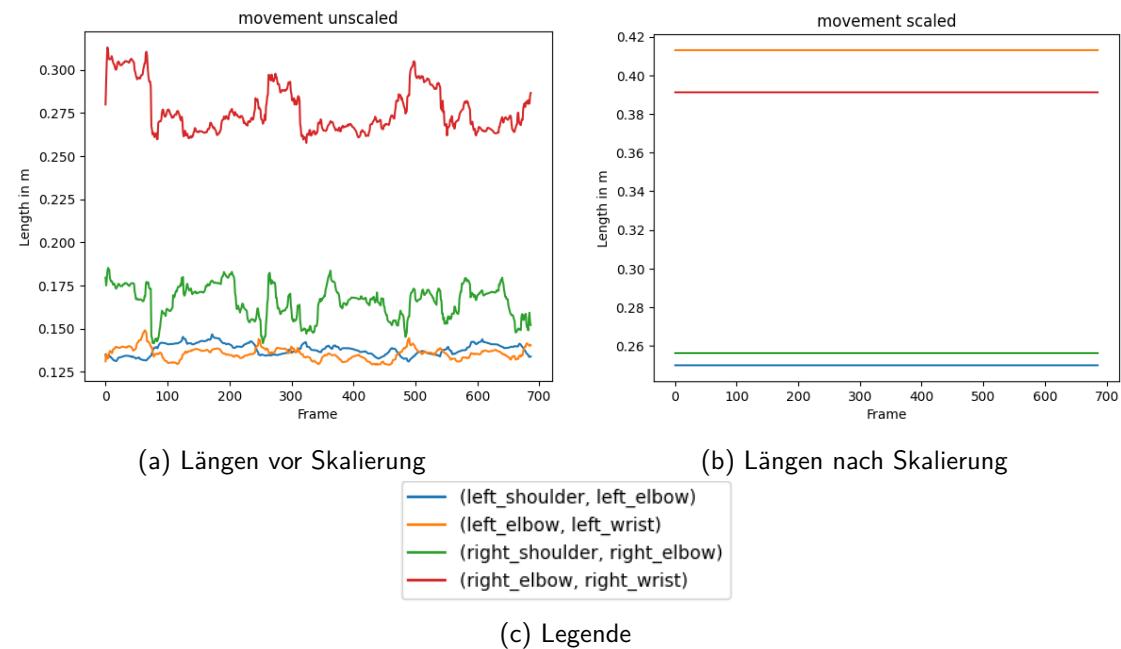


Abbildung 2.6: Skalierungs Diagramme

Sind nach der Skalierung noch Abweichungen in den Längen der Körperteile vorhanden, so liegt wahrscheinlich ein Fehler in der Definition der `limb_groups` vor.

2.4.6 Marker Datei aus CSV

Nachdem wir jetzt die skalierten Positionsdaten in der CSV Datei haben, müssen wir diese noch in eine Marker Datei (.trc) umwandeln, welche OpenSim verwenden kann. Eine weitere Funktion in unserem Workflow erstellt genau diese Marker Datei aus den Positionsdaten in der CSV Datei. Wichtig dabei ist dass unsere drei verwendeten Keypoints aus Mediapipe BlazePose auch den gleichen Namen haben wie die Marker in dem Modell von OpenSim.

Tabelle 2.1: Struktur der .trc Datei (Beispielwerte)

Frame#	Time	r_shoulder			r_elbow		
		X1	Y1	Z1	X2	Y2	Z2
1	0.0000	-0.0032	0.8362	0.1700	-0.0601	0.8085	0.1700
2	0.0167	-0.0016	0.8358	0.1700	-0.0595	0.8103	0.1700
3	0.0333	-0.0009	0.8358	0.1700	-0.0591	0.8109	0.1700
...							

2.4.7 Verwendung von Marker Datei in OpenSim

Anschließend wird diese Marker Datei dann in OpenSim verwendet. Dabei unterscheiden wir die Marker Dateien in zwei Kategorien.

- Statisch: Die Person im Video hat eine Pose eingenommen und diese gehalten.
- Dynamisch: Die Person im Video hat eine Bewegung ausgeführt.

Die Statische Marker Datei ist vor allem für das Scale Tool von OpenSim wichtig [10]. Damit haben wir jedoch wenig Erfolg gehabt, da wir zu wenig Marker hatten um unser Modell korrekt zu skalieren. Das könnte ein weiterer Ansatzpunkt für zukünftige Arbeiten sein.

Jede Art von Marker Datei kann man in OpenSim darstellen unter *File -> Preview experimental Data* um die Marker visualisiert im Raum zu sehen. Als letzten Schritt verwenden wir die Marker Dateien im Inversen Kinematik Tool [11] von OpenSim um eine .mot Bewegungsdatei zu bekommen welche die Gelenkwinkel des Modells, zu unterschiedlichen Zeitpunkten, enthält.

2.4.8 Genauigkeit der 3D-Pose-Schätzung von MediaPipe

Da dies ein monokularer Ansatz ist, sind die 3D Positionen von MediaPipe BlazePose nicht so genau wie bei einem Mehrkamerasytem. Um die Genauigkeit zu überprüfen, haben wir ein Beispielvideo (GitHub-Repository [13] unter [MediaPipe/Vergleich_2D-3D/dynamic.mp4](#)) verwendet. Im Video ist eine nahezu perfekte 2D Bewegung in der sichtbaren x-y-Ebene zu sehen, um die Stabilität der 3D Pose Schätzung zu testen.

Zunächst haben wir das Video mit MediaPipe analysiert, um die 3D Positionen der Keypoints zu extrahieren. Anschließend haben wir den Prozess wiederholt, aber diesmal die Z-Koordinaten aller Keypoints auf Null gesetzt, um eine 2D-Analyse zu simulieren. Danach haben wir die Ergebnisse in OpenSim importiert um sie zu betrachten.

Im Ergebnisvideo (GitHub-Repository [13] unter [MediaPipe/Vergleich_2D-3D/MP_Comp_2D-3D.mp4](#)), so wie in der Abbildung 2.7, ist deutlich zu erkennen, dass die 3D-Analyse von MediaPipe zu erheblichen Schwankungen in der Z-Achse führt, obwohl die Bewegung nahezu ausschließlich in der x-y-Ebene auf der Höhe der Schulter stattfindet.

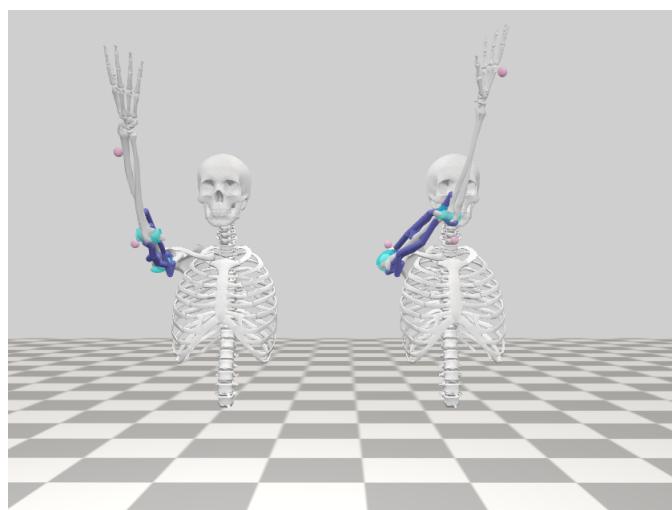


Abbildung 2.7: Vergleich der 2D-(links) und 3D-(rechts) Analyse in OpenSim

Dies zeigt, dass die 3D Pose Schätzung von MediaPipe in diesem Fall nicht stabil genug ist, um präzise Bewegungsdaten zu liefern. Im Gegensatz dazu liefert die 2D-Analyse, bei der die Z-Koordinaten auf Null gesetzt wurden, eine viel stabilere und genauere Darstellung der Bewegung. Dies unterstreicht die Limitationen der 3D Pose Schätzung bei monokularen Systemen und legt nahe, dass für präzise biomechanische Analysen Mehrkamerasyteme oder andere fortschrittlichere Methoden erforderlich sind.

2.5 Ablauf bei der Verwendung von Pose2Sim

2.5.1 Übersicht des Verfahrens

Pose2Sim fungiert als Schnittstelle zwischen Computer Vision und Biomechanik, um aus herkömmlichen Videoaufnahmen quantifizierbare Bewegungsdaten zu generieren. Der Workflow beginnt mit der Erfassung synchronisierter Rohvideos aus mehreren Perspektiven, welche im ersten Schritt durch das KI-Modell RTMPose analysiert werden. Als Zwischenergebnis liefert dieser Algorithmus reine 2D-Pixelkoordinaten sowie Konfidenzwerte für jede Kameraansicht, die anschließend durch Triangulierung in räumliche 3D-Trajektorien überführt werden. Den Abschluss bildet die Integration in die Simulationsumgebung OpenSim: Hier dienen die aufbereiteten 3D-Daten dazu, ein generisches muskuloskelettales Modell individuell an die Versuchsperson zu skalieren und schließlich mittels Inverser Kinematik die biomechanisch relevanten Gelenkwinkelverläufe zu berechnen.

2.5.2 Systemvoraussetzungen

Die Implementierung erfolgte in einer Python 3.11-Umgebung unter Verwendung von Pose2Sim (v0.10.39). Essenzielle Abhängigkeiten umfassen SciPy (v1.15.3) sowie die Anbindung an OpenSim. Die Installation und Verwaltung der virtuellen Umgebung wurde mittels Miniconda realisiert, um Versionskonflikte zu vermeiden.

2.5.3 Installation von Pose2Sim

Die Installation von Pose2Sim erfolgt über Miniconda.

1. Erstellen einer neuen Umgebung mit der passenden Python-Version:
`conda create -n pose2sim python=3.11 -y`
2. Aktivierung der Umgebung:
`conda activate pose2sim`
3. **Installation von OpenSim:**
Vor der eigentlichen Pose2Sim-Installation muss die OpenSim-Umgebung eingerichtet werden:
`conda install -c opensim-org opensim -y`
4. **Installation von Pose2Sim:**
Abschließend werden Pose2Sim und die Abhängigkeiten in den spezifizierten Versionen installiert:
`pip install pose2sim==0.10.39 scipy==1.15.3`

2.5.4 Verwendung des pose2sim_cli.py

Zur Steuerung des Workflows wurde das Skript pose2sim_cli.py eingesetzt. Es bündelt die Prozessschritte (Kalibrierung, Triangulierung, Filterung) und führt eine für OpenSim kritische Rotationsmatrix-Transformation durch, um Orientierungsfehler (z. B. invertierte Y-Achse) zu korrigieren. Das Skript ermöglicht sowohl die sequenzielle Ausführung der gesamten Pipeline als auch die modulare Ansteuerung einzelner Teilschritte.

Bedienung und Ausführung:

Das Skript wird direkt über die Kommandozeile gesteuert. Standardmäßig führt der Befehl `python pose2sim_cli.py` den gesamten Workflow sequenziell aus. Alternativ ermöglicht das Tool eine modulare Steuerung über spezifische Flags (z. B. `-c` für Kalibrierung oder `-t` für Triangulierung), wodurch einzelne Prozessschritte isoliert berechnet oder wiederholt werden können.

Verfügbarkeit des Codes:

Der vollständige Quellcode, inklusive des `pose2sim_cli.py`-Skripts sowie detaillierter Installationsanleitungen, ist im zugehörigen GitHub-Repository verfügbar:

<https://github.com/PhilippSG/Markerlose-Bewegungserfassung>

2.5.5 Erstellen eines OpenSim-Modells mit Pose2Sim

Ordnerstruktur

Für eine fehlerfreie Datenverarbeitung ist die Einhaltung einer spezifischen Ordnerstruktur zwingend erforderlich. Dies bildet die Grundlage dafür, dass sowohl die Standard-Funktionen von **Pose2Sim** als auch die Automatisierung durch das **CLI-Skript** korrekt funktionieren können.

Das Projektverzeichnis muss wie folgt organisiert sein:

```
Projekt_Verzeichnis/
|-- calibration/
|   |-- intrinsics/
|   |   |-- cam01/
|   |   '-- cam02/
|   '-- extrinsics/
|       |-- cam01/
|       '-- cam02/
|-- videos/           (Enthält die Rohvideos der Bewegung)
|-- Config.toml        (Zentrale Konfigurationsdatei)
`-- pose2sim_cli.py  (Skript für Rotation und Automatisierung)
```

Diese Struktur stellt sicher, dass Pose2Sim die Kalibrierungsdaten (in den Unterordnern `intrinsics` und `extrinsics`) eindeutig den Videodaten zuordnen kann. Gleichzeitig ermöglicht sie dem CLI-Tool, alle Schritte im Batch-Verfahren auszuführen.

Kamera Setup

Das Kamera-Setup bildet die Basis für die gesamte Datenerfassung. Ziel ist es, die Bewegungen der Versuchsperson aus möglichst vielen Perspektiven gleichzeitig aufzuzeichnen, um Verdeckungen (Okklusionen) zu minimieren.

Für eine erfolgreiche Aufnahme und zur Vermeidung typischer Fehlerquellen sind folgende Punkte entscheidend:

- **Positionierung:** Es werden 2 Kameras verwendet, die um das Aufnahmeverumherum platziert sind. Sie müssen so ausgerichtet sein, dass jedes Körperteil der Person zu jedem Zeitpunkt von mindestens zwei Kameras gesehen wird.
- **Zeitliche Synchronisation:** Dies ist die kritischste Fehlerquelle. Alle Kameras müssen exakt zum gleichen Zeitpunkt aufnehmen. Bereits minimale zeitliche Abweichungen führen bei schnellen Bewegungen zu erheblichen Fehlern in der 3D-Rekonstruktion.
- **Stabilität:** Die Kameras dürfen sich nach der Kalibrierung keinesfalls bewegen. Jede noch so kleine Positionsänderung macht die Kalibrierungsdaten unbrauchbar.

Kalibrierung der Kameras

Damit das System die 2D-Pixelkoordinaten aus den Videos in metrische 3D-Koordinaten umrechnen kann, ist eine präzise Kalibrierung zwingend erforderlich. Dieser Prozess nutzt das im *Kamera Setup* gezeigte Schachbrettmuster, um die geometrischen Eigenschaften jeder Kamera zu bestimmen.

Die Kalibrierung unterteilt sich in zwei wesentliche Komponenten:

- **Intrinsische Kalibrierung:**
Hierbei werden die internen physikalischen Eigenschaften der Kamera berechnet. Die Ergebnisse werden für jede Kamera im Ordner `calibration/intrinsics` gespeichert.
- **Extrinsische Kalibrierung:**
Hierbei wird die Position und Orientierung der Kameras im 3D-Raum relativ zueinander bestimmt. Pose2Sim berechnet eine Rotationsmatrix und einen Translationsvektor. Diese Daten landen im Ordner `calibration/extrinsics`.

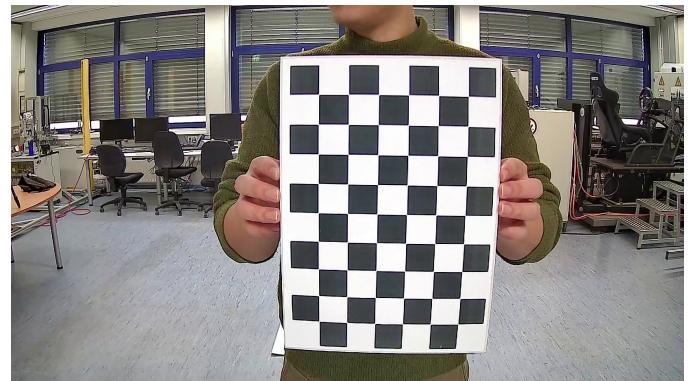
Durchführung:

Die Berechnung erfolgt automatisiert durch Pose2Sim (bzw. das CLI-Tool mit dem Flag `-c`). Dabei analysiert der Algorithmus Bilder des Schachbrettmusters aus verschiedenen Winkeln und nutzt die bekannten Kantenlängen der Quadrate als Referenzmaßstab.

Wichtig: Damit der Algorithmus den korrekten metrischen Maßstab (Skalierung) berechnen kann, ist es entscheidend, die **Kantenlänge der Schachbrett-Quadrate** in der `Config.toml` einzutragen.



(a) Kamera-Setup mit Schachbrett
(Extrinsische Kalibrierung)



(b) Intrinsische Kalibrierung

Abbildung 2.8: Visualisierung der Kalibrierung.

Konfigurationsdatei

Die Datei `Config.toml` fungiert als zentrale Steuerzentrale. Neben den physikalischen Parametern (wie der Schachbrettgröße) wurden basierend auf empirischen Tests spezifische Einstellungen gewählt, um die Stabilität der Pipeline zu optimieren.

Die wichtigsten Anpassungen, gegliedert nach den Sektionen der Konfigurationsdatei, umfassen:

[project]

- `multi_person`: Dieser Parameter wurde dauerhaft auf `true` gesetzt. Dies deckt sowohl Einzel- als auch Mehrpersonenaufnahmen ab und vermeidet Konfigurationsfehler.
- `frame_rate`: Kann auf `'auto'` belassen werden, da Pose2Sim die Bildwiederholrate der Videos zuverlässig aus den Metadaten ausliest.
- `frame_range`: Es wird empfohlen, den Bereich manuell zu definieren oder alle Frames zu verarbeiten. Die Einstellung `'auto'` (automatische Erkennung, wann eine Person das Bild betritt) erwies sich in Tests als fehleranfällig.

[pose]

- `pose_model`: Als Modell wurde RTMPose (in der Variante `body_with_feet`) gewählt.
Hinweis zur Modellwahl: In Voruntersuchungen wurde auch das Framework **Media-Pipe** evaluiert. Dieses lieferte jedoch signifikant ungenauere Ergebnisse bei der 3D-Triangulierung, da die Konsistenz der Keypoints zwischen den Kameraperspektiven unzureichend war. Daher wurde es zugunsten von RTMPose verworfen.
- `mode`: Die Einstellung 'performance' bietet den besten Kompromiss aus Genauigkeit und Rechengeschwindigkeit.
- `tracking_mode`: Für das Verfolgen der Personen über die Frames hinweg lieferte der Modus 'sport2d' deutlich robustere Ergebnisse als der 'deepsort'-Algorithmus.

[synchronization]

- `synchronization_gui`: Da die Videos bereits durch OBS Studio synchron aufgezeichnet wurden, wird dieser Wert auf `false` gesetzt. *Hinweis:* Bei manuellen, unsynchronisierten Aufnahmen müsste dieser Wert aktiviert werden, um die Videos nachträglich anhand eines visuellen Signals (z. B. Klatschen) anzulegen.

[calibration]

- `type`: Standardmäßig auf 'calculate' gesetzt.
- `method`: Es wird die Methode 'board' (Schachbrett) verwendet. Die Alternative 'scene' (Nutzung statischer Hintergrundpunkte) ist mathematisch komplexer und lieferte unzuverlässigere Ergebnisse.
- **Physikalische Parameter:** Entscheidend sind die korrekte Angabe der `checkerboard_square_size` (z. B. 0.03 m) und der `corners_nb` (Anzahl innerer Ecken), da diese als metrische Referenz für die 3D-Umrechnung dienen.
- **Effizienz:** Die Berechnung der intrinsischen Parameter muss theoretisch nur einmalig erfolgen, solange die Objektiveinstellungen (Fokus/Zoom) der Kameras unverändert bleiben.

Software und Plugins

Zur Realisierung der synchronisierten Aufnahme kommt die Software **OBS Studio** in Kombination mit dem Plugin **Source Record** zum Einsatz. Dies ermöglicht die gleichzeitige Aufzeichnung beider Kameras.

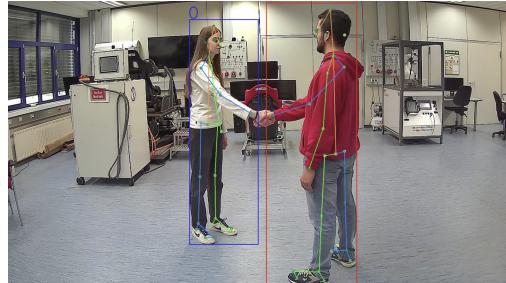
2D Pose Erkennung

Nach der Datenerfassung folgt der erste Berechnungsschritt: die Extraktion anatomischer Landmarken aus den Rohvideos. Hierbei kommt das Deep-Learning-Modell **RTMPose** zum Einsatz, welches in der Konfigurationsdatei definiert wurde.

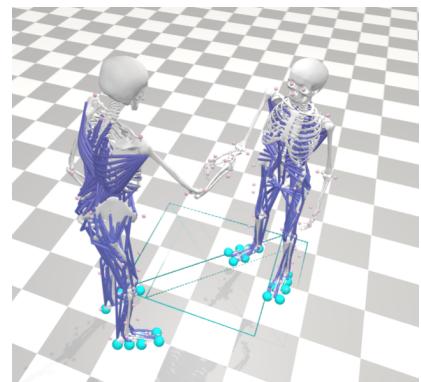
Der Prozess läuft wie folgt ab:

1. **Dektection:** Der Algorithmus analysiert bildweise (frame-by-frame) die Aufnahmen aller Kameras separat. Dabei werden die Positionen relevanter Körpergelenke (z. B. Schulter, Ellbogen, Handgelenk) im zweidimensionalen Bildraum identifiziert.
2. **Datenstruktur:** Für jeden Keypoint generiert das Modell einen Datensatz bestehend aus:
 - Den Pixel-Koordinaten (x, y) .
 - Einem Konfidenzwert (Likelihood), der angibt, mit welcher Wahrscheinlichkeit der Punkt korrekt erkannt wurde.
3. **Speicherung:** Die Ergebnisse werden als JSON-Dateien im Ordner `pose` abgelegt.

Die Ausführung erfolgt automatisiert über das CLI-Tool mit dem Flag `-p`. Das Resultat ist eine Sammlung von unabhängigen 2D-Skelettmodellen aus verschiedenen Perspektiven, die noch keine räumliche Tiefe besitzen.



(a) 2D-Pose-Erkennung (RTMPose)



(b) OpenSim-Modell (Inverse Kinematik)

Abbildung 2.9: **Von 2D zu 3D:** Visualisierung einer komplexen Interaktion (Händeschütteln). Links ist die robuste Detektion der Keypoints durch RTMPose zu sehen. Rechts das daraus generierte, muskuloskelettale OpenSim-Modell, das die Bewegung räumlich korrekt wiedergibt.

Triangulierung

Der zentrale Schritt der Pipeline ist die Überführung der planaren 2D-Daten in ein räumliches 3D-Modell. Dies geschieht durch das mathematische Verfahren der **Triangulierung**.

Dabei werden die 2D-Informationen aus den verschiedenen Kameraperspektiven unter Einbeziehung der Kalibrierungsdaten (Position und Orientierung der Kameras) fusioniert. Für jeden Gelenkpunkt wird der Schnittpunkt der Sichtstrahlen im 3D-Raum berechnet.

Qualitätssicherung:

Ein entscheidender Vorteil von Pose2Sim ist die gewichtete Triangulierung. Der Algorithmus berücksichtigt den in der vorherigen Stufe ermittelten **Konfidenzwert** (Likelihood).

- Punkte mit hoher Erkennungswahrscheinlichkeit gehen stärker in die Berechnung ein.
- Punkte unterhalb des in der `Config.toml` definierten Schwellenwerts (`likelihood_threshold`) werden ignoriert, um Ausreißer zu minimieren.

Dieser Schritt wird über das CLI-Tool mit dem Flag `-t` ausgeführt. Das Ergebnis ist eine Punktwolke (Point Cloud) der Gelenke im metrischen Raum (x, y, z in Metern).

TRC Datei Erstellung

Der letzte Schritt der Pipeline ist die Generierung der finalen Bewegungsdaten. Das primäre Zielformat ist das **.trc-Format** (Track Row Column), welches die Trajektorien aller virtuellen Marker über die Zeit speichert.

Bevor die Dateien exportiert werden, durchlaufen die Daten zwei wesentliche Nachbearbeitungsschritte:

▪ **Filterung (Filtering):**

Da die rohen triangulierten Daten oft ein hochfrequentes Rauschen (Jitter) aufweisen, wird ein **Butterworth-Tiefpassfilter** angewendet. Die in der Konfiguration definierte Grenzfrequenz (z. B. 6 Hz) sorgt dafür, dass das unnatürliche Zittern der Keypoints entfernt wird, während die eigentliche menschliche Bewegung erhalten bleibt.

▪ **Rotation und Koordinatentransformation:**

Dies ist ein kritischer Punkt für die Kompatibilität. Ohne Korrektur würde das Modell in OpenSim falsch orientiert erscheinen (z. B. **Kopf nach unten**). Das CLI-Tool führt automatisch eine Rotationsmatrix-Transformation durch, um die Daten an das Welt-Koordinatensystem von OpenSim anzupassen.

Ergebnis und generierte Dateien:

Nach Abschluss des Prozesses liegen im Ordner `pose-3d` alle für die Simulation notwendigen Dateien bereit. Neben der `.trc`-Datei generiert Pose2Sim automatisch:

- **.osim (OpenSim Modell):** Eine Modelldatei, die bereits das korrekte, virtuelle Marker-Set (basierend auf `body_with_feet`) enthält. Dies erspart das manuelle Hinzufügen von Markern in der OpenSim-GUI.
- **.mot (Motion File):** Eine ergänzende Bewegungsdatei, die Referenzdaten oder Koordinaten für die weitere Verarbeitung enthält.

Damit ist der Datensatz vollständig und kann ohne weitere Konvertierung direkt in OpenSim geladen werden.

2.5.6 Datenverarbeitung in OpenSim

Nachdem die videobasierten Daten in Pose2Sim aufbereitet wurden, erfolgt die biomechanische Analyse in der Simulationsumgebung **OpenSim**. Dieser Prozessschritt transformiert die reinen Positionsdaten (Trajektorien) in physiologisch bedeutsame Gelenkwinkel.

2.5.7 Modell-Import und Vorbereitung

Ein wesentlicher Vorteil des verwendeten Workflows ist die nahtlose Integration der Daten. Anstatt ein Standard-Modell manuell mit Markern zu versehen, wird das von Pose2Sim im Ordner `pose-3d` generierte **.osim-Modell** geladen.

Dieses Modell enthält bereits ein **virtuelles Markerset**, das exakt den anatomischen Landmarken des verwendeten Pose-Modells (`body_with_feet`) entspricht. Parallel dazu wird die erstellte `.trc`-Datei als Bewegungsdatenquelle importiert.

2.5.8 Modell-Erstellung und Skalierung

Ein besonderer Vorteil der verwendeten Pipeline ist die automatisierte Erstellung eines personen-spezifischen Modells. Im klassischen Workflow müsste ein Standard-Modell in OpenSim manuell skaliert werden.

Hier erfolgt der Prozess datengesteuert durch Pose2Sim:

1. **Segment-Anpassung:** Der Algorithmus berechnet aus den triangulierten 3D-Koordinaten die tatsächlichen Segmentlängen der Versuchsperson (z. B. Distanz zwischen Hüft- und Kniegelenk).
2. **Modell-Generierung:** Basierend auf diesen Maßen wird das generische OpenSim-Modell geometrisch angepasst (skaliert).
3. **Marker-Platzierung:** Gleichzeitig werden die virtuellen Marker exakt an den Positionen der erkannten Gelenke platziert.

Das Resultat ist die im Ergebnisordner vorliegende .osim-Datei. Dieses Modell repräsentiert bereits die Statur der Versuchsperson und kann direkt für die weitere Analyse verwendet werden, ohne dass eine manuelle Skalierung notwendig ist.

2.5.9 Inverse Kinematik (IK)

Der finale Schritt der Datenverarbeitung ist die Berechnung der tatsächlichen Gelenkwinkel. Hierbei kommt das Verfahren der **Inversen Kinematik (IK)** zum Einsatz.

Funktionsweise:

Bei der Inversen Kinematik wird das (automatisch skalierte) Modell durch die Bewegungsdaten der .trc-Datei „angetrieben“. Für jeden einzelnen Zeitschritt (Frame) löst OpenSim ein mathematisches Optimierungsproblem. Das Ziel ist es, die Gelenkwinkel des Modells so einzustellen, dass der Abstand zwischen den *experimentellen Markern* (aus der Pose2Sim-Berechnung) und den *virtuellen Markern* auf dem Modell minimiert wird.

Ergebnis (Endpunkt):

Das Resultat dieses Prozesses ist eine **.mot-Datei** (Motion File). Diese Datei enthält die finalen, zeitabhängigen **Gelenkwinkel** (in Grad) für alle Freiheitsgrade des Modells.

Damit ist der gesamte Workflow abgeschlossen: Aus den ursprünglichen 2D-Rohvideos wurden quantifizierbare, biomechanische Kinematikdaten gewonnen, die nun für die weitere grafische Auswertung oder statistische Analyse bereitstehen.

2.5.10 Visualisierung der Ergebnisse in OpenSim

Nach der Berechnung dient die grafische Benutzeroberfläche (GUI) von OpenSim zur visuellen Validierung der Bewegungsdaten.

Visualisierung eines einzelnen Modells Um die berechnete Kinematik einer einzelnen Aufnahme zu betrachten:

1. **Modell laden:** Über *File → Open Model* wird die generierte .osim-Datei geöffnet.
2. **Bewegungsdaten laden:** Über *File → Load Motion* wird die zugehörige .mot-Datei (aus der IK-Berechnung) importiert.
3. **Wiedergabe:** Die Bewegung kann nun über die Zeitleiste (Seek Bar) abgespielt oder bildweise analysiert werden.

Synchronisieren mehrerer Modelle (Vergleich) Um beispielsweise zwei verschiedene Versuche oder Perspektiven gleichzeitig abzuspielen (z. B. für einen Vorher-Nachher-Vergleich):

1. Beide Modelle und deren zugehörige Motion-Dateien (.mot) in OpenSim laden.
2. Im *Navigator*-Fenster (links) die Option *Coordinates* beider Modelle gleichzeitig markieren (mit Strg + Mausklick).
3. Rechtsklick auf die markierten Elemente und die Option *Sync. Motions* wählen.
4. Beim Abspielen laufen nun beide Modelle exakt synchron.

2.5.11 Ergebnisse von Pose2Sim

Die durchgeführte Pipeline ermöglichte die erfolgreiche Extraktion kinematischer 3D-Daten aus den synchronisierten Videoaufnahmen. Bevor die physiologischen Gelenkwinkel betrachtet werden, erfolgt zunächst eine Analyse der Signalqualität.

Qualität der Datenaufbereitung und Filterung

Ein wesentlicher Schritt der Datenverarbeitung war die Reduktion des hochfrequenten Rauschens (Jitter), das bei markerlosen Tracking-Verfahren systembedingt auftritt.

Abbildung 2.6 demonstriert die Effektivität des angewendeten Butterworth-Tiefpassfilters am Beispiel der 3D-Koordinaten des Hüftgelenks (Hip):

- **Ungefiltertes Signal (Blau):** Die rohen Triangulierungsdaten zeigen deutliche, hochfrequente Ausschläge. Diese Instabilität würde in der Inversen Kinematik zu fehlerhaften Gelenkwinkelberechnungen führen.
- **Gefiltertes Signal (Orange):** Der Filter (Cut-off: 6 Hz) glättet den Signalverlauf effektiv. Entscheidend ist hierbei, dass die grundlegende Form der Bewegungskurve erhalten bleibt (kein „Over-Smoothing“), während die Störsignale eliminiert werden.

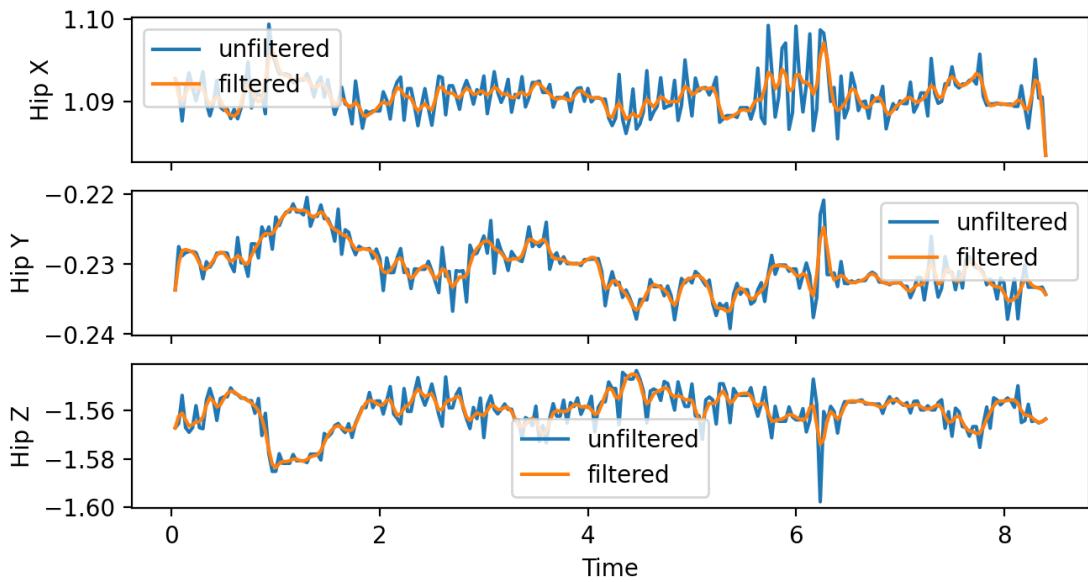


Abbildung 2.10: **Einfluss der Filterung auf die 3D-Trajektorien:** Darstellung der X-, Y- und Z-Koordinaten des Hüftgelenks über die Zeit. Der Vergleich zwischen rohen (blau) und gefilterten (orange) Daten belegt die notwendige Stabilisierung des Signals für die weitere Analyse in OpenSim.

Die hier gezeigte Signalglättung bildet die notwendige Basis für die darauf folgende Berechnung der Gelenkwinkel.

Generierung der Simulationsdaten (.mot)

Aufbauend auf den bereinigten Trajektorien konnten im nächsten Schritt die Motion-Files (.mot) generiert werden. Diese Dateien stellen das finale Ergebnis der kinematischen Berechnung dar und beinhalten die zeitlichen Verläufe aller Gelenkwinkel (in Grad) für die Freiheitsgrade des Modells.

Eine Überprüfung der Daten zeigt physiologisch plausible Wertebereiche (z. B. für Knie- und Hüftflexion), die frei von den in den Rohdaten beobachteten hochfrequenten Störungen sind. Damit stehen validierte Datensätze bereit, die direkt in der OpenSim-GUI visualisiert oder für weiterführende biomechanische Analysen herangezogen werden können.

Kritische Einflussfaktoren

Die Evaluation der Pipeline identifizierte folgende Parameter als qualitätsbestimmend:

- **Hardwareseitige Restriktionen (Kamera-Anzahl):** Der Versuchsaufbau beschränkte sich auf ein Dual-Kamera-Setup. Voruntersuchungen zeigten, dass die Einbindung weiterer Perspektiven auf der verwendeten mobilen Workstation (Intel Core i5-11400H, 16GB RAM, NVIDIA RTX 3060) zu einer instabilen Bildrate (Frame Drops) führte. Dieser Leistungsengpass ist primär auf die begrenzte **Video-Encodierungs-Bandbreite** der GPU zurückzuführen, welche eine simultane Echtzeitverarbeitung von mehr als zwei HD-Streams verhinderte.
- **Beleuchtung & Kontrast:** Eine ausreichende Ausleuchtung ist zwingend. Dunkle Kleidung in dunkler Umgebung führt zu massivem Signalverlust („Teleportation“).
- **Kleidung:** Eng anliegende Kleidung (Tight-Fit) ist essenziell, da weite Stoffe die Lokalisierung der anatomischen Gelenkzentren verfälschen.
- **Okklusion:** Verdeckungen durch externe Objekte (z. B. Rollstuhl) oder Selbstverdeckung (z. B. verschränkte Arme) führen zu Tracking-Aussetzern und Artefakten.
- **Foot Sliding:** Da keine Bodenreaktionskräfte gemessen werden, kann es im Modell zu minimalem Rutschen der Füße kommen.

Systemgrenzen und Beobachtungen

Stärken (Positiv):

- **Komplexe Interaktionen:** Das System erfasst nicht nur zirkuläres Gehen, sondern auch komplexe, nicht-zirkuläre Bewegungen wie **Händeschütteln** oder Bewegungen in **sitzender Position** (im Stuhl).
- **Robuste Identifizierung (ID-Tracking):** Eine bemerkenswerte Stärke ist die Trennschärfe bei mehreren Personen. Der Algorithmus unterscheidet Individuen zuverlässig, selbst wenn diese eine **ähnliche Statur**, **identische Kleidung** und gleiche Bewegungsmuster aufweisen.
- **Anthropometrische Skalierung:** Die generierten OpenSim-Modelle bilden individuelle **Körpergrößen und Proportionen** der Teilnehmer maßstabsgetreu ab, sodass Größenunterschiede in der Simulation visuell und rechnerisch exakt wiedergegeben werden.
- **Tiefenstabilität:** Das System unterscheidet robust zwischen aktiven Personen im Vordergrund und störenden Bewegungen im Hintergrund.

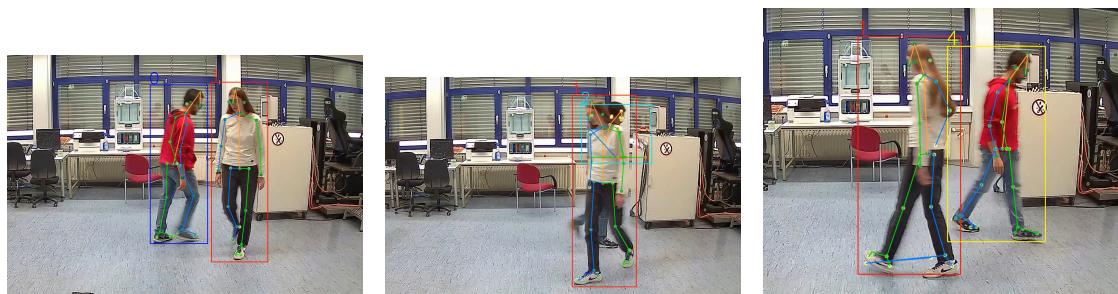
Schwächen (Limitierungen):

- **Beleuchtung:** Kombinationen aus dunkler Kleidung und schwachem Licht führen zu massivem Signalverlust („Teleportation“).

Person 0: Less than 10 valid frames in a row. Deleting person.

Abbildung 2.11: **Tracking-Abbruch bei Dunkelheit:** Auszug aus dem System-Log. Aufgrund mangelnden Kontrasts, schlechter Kalibrierung sowie unvollständiger Abbildung der Person im Frame konnte keine kontinuierliche Erkennung gewährleistet werden.

- **Okklusion (Verdeckung):** Neben statischen Objekten (z. B. Rollstuhl) stellen insbesondere *dynamische Überlappungen* eine Fehlerquelle dar. Bei Multi-Personen-Aufnahmen ist eine strikte räumliche Trennung der Akteure zwingend erforderlich. Kreuzen sich die Laufwege der Personen (Crossing), kann der Algorithmus die Gelenke nicht mehr eindeutig zuordnen (Identity Switch), was zum Zusammenbruch der Simulation führt.



(a) Vor der Kreuzung: Korrekte IDs (0 und 1). (b) Überlappung: Verlust der Skelettstruktur. (c) Nach der Kreuzung: Identity Switch (ID 0 → 4).

Abbildung 2.12: **Fehleranalyse bei dynamischer Okklusion (Crossing):** Die Bildsequenz demonstriert das Versagen des Trackings bei sich kreuzenden Laufwegen. (a) Die Personen werden korrekt getrackt (Blau: ID 0, Rot: ID 1). (b) Während der Überlappung kann der Algorithmus die Gliedmaßen nicht mehr zuordnen. (c) Nach der Trennung erhält die männliche Person fälschlicherweise eine neue ID (Gelb: ID 4 statt Blau: ID 0), was die Kontinuität der Simulation unterbricht.

- **Foot Sliding:** Da keine Kraftmessplatten verwendet werden, neigen die Füße im Modell zu leichtem Rutschen auf dem Boden.
- **Bewegungsunschärfe:** Sehr schnelle Bewegungen führen zu ungenauen Keypoints und erfordern stärkere Filterung.

2.6 Genauigkeitsvergleich der beiden Ansätze

Bei diesem Vergleich geht es darum, die eigenentwickelte MediaPipe-Pipeline und das zugrunde liegende MediaPipe-Modell zu bewerten, mit besonderem Blick auf die unterschiedlichen Möglichkeiten und Grenzen eines monokularen Ansatzes. Zur Einordnung der Ergebnisse wird die Pose2Sim-Pipeline als Referenz herangezogen, da dort keine eigentliche 3D-Poseschätzung erfolgt, sondern die Positionen der Marker durch das Multikamera-Setup mittels Triangulierung präziser bestimmt werden können.

Um einen angemessenen Vergleich der beiden Ansätze durchführen zu können, haben wir uns entschieden, ein identisches OpenSim-Modell zu verwenden. Verwendet wurde hier das bereits erwähnte arm_model.osim, befindlich im GitHub-Repository [13]. Dafür mussten wir jedoch frühzeitig in die Pose2Sim-Pipeline eingreifen und die bis dahin erstellte .trc-Datei extrahieren, da im nächsten Schritt das pipelineeigene Modell erstellt worden wäre.

Um dennoch eine faire Skalierung der Modelle zu ermöglichen, haben wir von beiden Pipelines eine statische Aufnahme verarbeiten lassen und die daraus resultierenden .trc-Dateien verwendet, um das jeweilige OpenSim-Modell zu skalieren.

Anschließend wurde die dynamische Bewegungsdatei im OpenSim-Tool für inverse Kinematik verarbeitet und die daraus resultierenden Markerfehler extrahiert. In den verwendeten Videos wird dabei die Bewegung des rechten Arms von einer nach vorne ausgestreckten Position bis zu einem Ellbogenwinkel von über 90 Grad ausgeführt, wobei die Hand in einer Bewegung zum Gesicht und wieder zurückgeführt wird. Das Video, welches für MediaPipe verwendet wurde, zeigt die Person in Seitenansicht, sodass die Bewegung in der sichtbaren x-y-Ebene stattfindet. Die Tiefeninformation (z-Richtung) wird dabei ausschließlich durch die 3D-Schätzung des MediaPipe-Modells bestimmt. Um die Markerfehler in einen Kontext setzen zu können, wurden zudem noch die von OpenSim erstellten .mot-Dateien extrahiert, die die resultierenden Gelenkwinkel beinhalten.

Die verwendeten Dateien, die Originalvideos sowie ein Ergebnisvideo, welches die Eingabevideos gemeinsam mit dem entsprechenden OpenSim-Ergebnis darstellt, sind im GitHub-Repository [13] unter Vergleich_MP-P2S/ wiederzufinden.

Zuerst betrachten wir die aus der inversen Kinematik resultierenden Markerfehler. Diese geben an, wie nah die virtuellen Marker an die Marker der extrahierten Daten angepasst werden konnten.

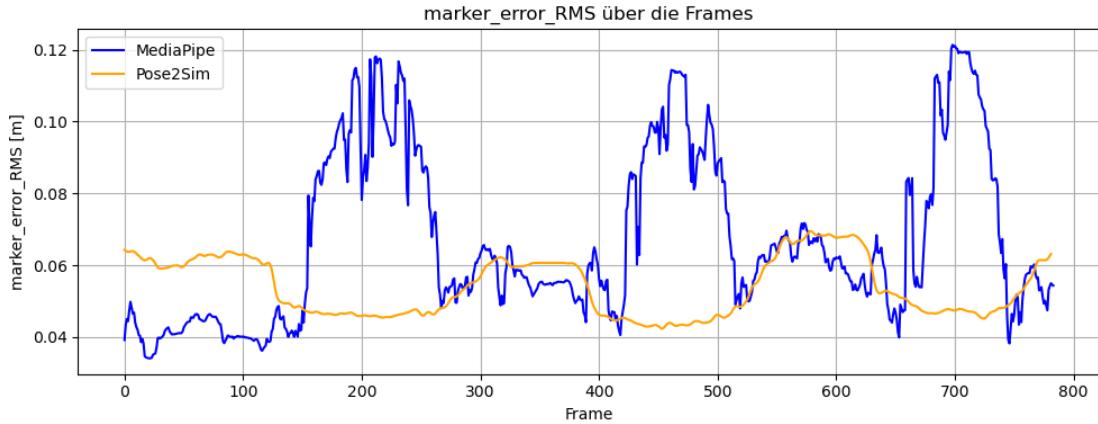


Abbildung 2.13: RMS-Fehler der Marker

In Abbildung 2.13 ist gut zu erkennen, dass die Markerfehler aus den MediaPipe Daten teils sehr nah an denen von Pose2Sim liegen. Interessant ist, dass die MediaPipe Markerfehler immer dann ansteigen, wenn die Hand im Video in die obere Endlage gelangt. Dieses Verhalten lässt sich darauf zurückführen, dass MediaPipe in dieser Phase durch die stärkere Selbstüberdeckung sowie die geringe Bewegungsdynamik der Marker eine erhöhte Instabilität der 3D-Rekonstruktion aufweist [14]. Pose2Sim hingegen liefert hier leicht geringere Markerfehler, da das Multikamera-Setup eine robustere 3D-Triangulierung ermöglicht und Selbstüberdeckungen einzelner Marker besser kompensiert werden können.

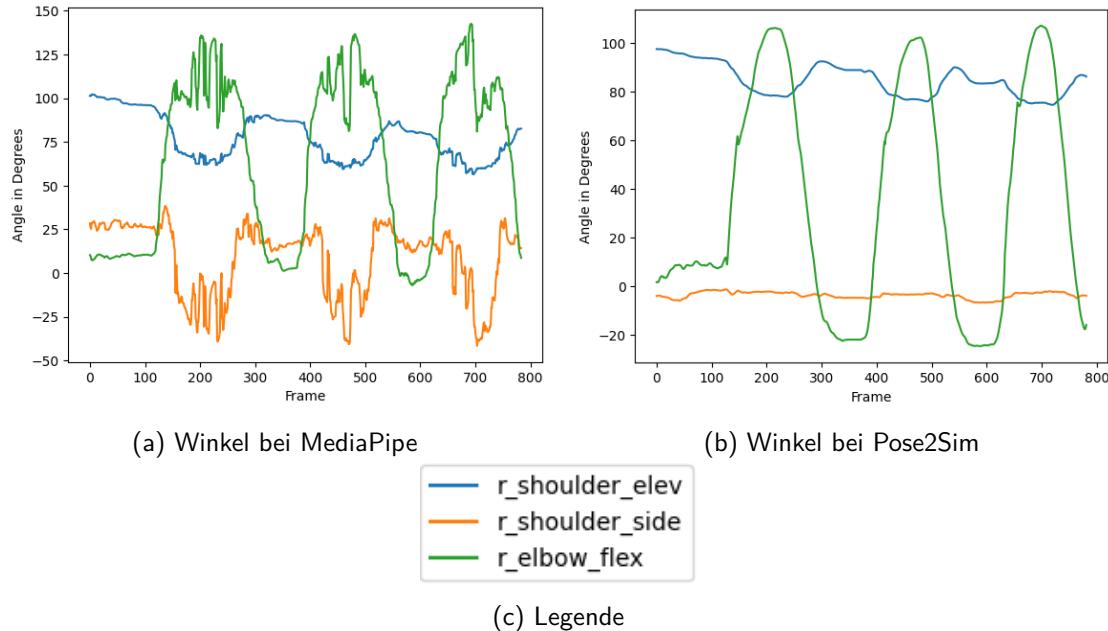


Abbildung 2.14: Winkel Diagramme

Die Abbildung 2.14 zeigt die aus der inversen Kinematik resultierenden Gelenkwinkel beider Ansätze über die Frames. Der vierte Modellwinkel (`r_elbow_rotation`) wurde dabei bewusst nicht dargestellt, da dieser bei der ausgeführten Bewegung keine relevante Aussage liefert. Die dort entstandenen Rotationswerte ergeben sich primär aus den relativen Abständen und Orientierungen der verwendeten Marker und erlauben daher keinen sinnvollen Vergleich der beiden Pipelines.

Die Winkelergebnisse zeigen sehr deutlich, dass MediaPipe deutliche Fluktuationen besonders in den verwendeten Endlagen liefert. Dies lässt vermuten, dass die Erkennung besser in einer langsamen Bewegung funktioniert, da hier vermutlich die Bewegungsrichtung und -geschwindigkeit berücksichtigt werden können.

Die größten Abweichungen zeigen sich jedoch, wie zu erwarten, in der 3D Schätzung. Besonders deutlich wird dies anhand des Winkels `r_shoulder_side`, der maßgeblich für die Bewegung in z-Richtung ist. Während MediaPipe hier starke Schwankungen zeigt – insbesondere erneut in der oberen Endlage – bleibt der entsprechende Winkel bei Pose2Sim über das gesamte Video hinweg nahezu konstant.

3 Schluss

3.1 Fazit

Im Rahmen dieses Projekts wurde untersucht, inwieweit markerlose Bewegungserfassungssysteme zur Rekonstruktion menschlicher Bewegungen und zur Bestimmung von Gelenkwinkeln in der biomechanischen Simulationsumgebung OpenSim eingesetzt werden können. Dabei wurden zwei unterschiedliche Ansätze betrachtet und miteinander verglichen: ein eigenentwickelter Workflow auf Basis eines monokularen Pose-Schätzverfahrens sowie eine etablierte Multi-Kamera-Pipeline, die als Referenz diente.

Die Ergebnisse zeigen, dass markerlose Bewegungserfassung grundsätzlich geeignet ist, kinematische Bewegungsdaten zu erfassen und diese erfolgreich in OpenSim zu integrieren. Insbesondere konnte demonstriert werden, dass aus videobasierten Poseschätzungen Markerdateien erzeugt werden können, die eine inverse Kinematik und damit die Berechnung von Gelenkwinkeln ermöglichen. Der entwickelte Workflow stellt dabei eine kostengünstige und flexibel einsetzbare Alternative zu klassischen markerbasierten Bewegungserfassungssystemen dar.

Gegenüberstellung der Vor- und Nachteile

Genauigkeit und biomechanische Qualität

Im Zuge der Untersuchung konnten deutliche Unterschiede in der Rekonstruktionsqualität der beiden Ansätze festgestellt werden. Der monokulare MediaPipe-Ansatz weist insbesondere in der 3D-Pose-Schätzung Limitationen auf, die sich vor allem in Instabilitäten der Tiefenrichtung äußern. Diese führen zu erhöhten Markerfehlern und starken Schwankungen einzelner Gelenkwinkel, insbesondere in den Endlagen der Bewegung, bei Selbstüberdeckungen sowie bei geneigter Kameraperspektive. Da MediaPipe ohne explizite geometrische Kamerakalibrierung arbeitet, ist die räumliche Rekonstruktion empfindlich und fehleranfälliger gegenüber Perspektivänderungen.

Der Vergleich mit der Pose2Sim-OpenSim-Pipeline zeigt hingegen, dass die multikamera-basierte Triangulation mit geometrisch kalibrierten Kameras stabilere und biomechanisch konsistenter 3D-Rekonstruktionen ermöglicht. Pose2Sim erweist sich damit als deutlich präziser für biomechanische Analysen.

Zusätzlich stellte sich die Skalierung der OpenSim-Modelle bei MediaPipe als kritischer Faktor heraus. Die Genauigkeit der Gelenkwinkelberechnung hängt stark von der Anzahl und Qualität der verwendeten Marker ab. Insbesondere bei Ganzkörpermodellen führten unzureichend präzise

Landmarken und Probleme in der Tiefenschätzung zu größeren Abweichungen, wodurch die biomechanische Aussagekraft weiter eingeschränkt wird.

Technischer Aufwand/Komplexität und Benutzerfreundlichkeit

Im Hinblick auf den technischen Aufwand und Benutzerfreundlichkeit zeigen sich folgende Vor-und Nachteile beider Ansätze.

Die Pose2Sim-OpenSim-Pipeline erfordert ein komplexes Setup mit zwei (oder mehr) Kameras, präziser Kalibrierung und einer mehrstufigen Verarbeitungskette. Initial ist die Einrichtung aufwendiger, jedoch steht anschließend eine weitgehend standardisierte und reproduzierbare Verarbeitungskette zur Verfügung.

Bei MediaPipe ist die initiale Erstellung einer biomechanisch verwertbaren Pipeline mit erheblichem Entwicklungsaufwand verbunden, insbesondere für die Skalierung der Daten, die Transformation der Koordinatensysteme und die Integration in OpenSim. Ist diese Pipeline jedoch einmal implementiert, gestaltet sich die anschließende Anwendung vergleichsweise benutzerfreundlich und unkompliziert, da vor allem kein aufwendiges Kamerasetup mit geometrischer Kalibrierung erforderlich ist.

Alltagstauglichkeit

MediaPipe erweist sich als alltagstauglicher, insbesondere für mobile Einsatzszenarien, da es mit minimaler Hardware ohne aufwendiges Kamerasetup flexibel in unterschiedlichen Umgebungen eingesetzt werden kann. Der Pose2Sim-Ansatz erfordert eine präzise Kamerakalibrierung und ist dadurch in seiner Einsatzfähigkeit eher unflexibel.

Kosten und Skalierbarkeit

Auch hinsichtlich der Kosten unterscheiden sich die beiden Ansätze. Der MediaPipe-Workflow kann mit handelsüblicher Hardware betrieben werden und erfordert in der Regel lediglich eine Standardkamera sowie einen Rechner. Dadurch ist der Ansatz kostengünstig skalierbar und auch für den Einsatz außerhalb speziälisierter Labore geeignet.

Die Pose2Sim-OpenSim-Pipeline ist demgegenüber mit höheren Investitionskosten verbunden. Für den Betrieb sind zwei oder mehr Kameras, sowie eine (leistungsstarke) Grafikkarte erforderlich, um die umfangreichen Bilddaten verarbeiten zu können.

Geeignete Anwendungsbereiche von MediaPipe und Pose2Sim

Aus den Ergebnissen dieser Arbeit lassen sich Anwendungsfelder für die beiden untersuchten Ansätze ableiten. MediaPipe eignet sich insbesondere für mobile, flexible und alltagsnahe Anwendungen, bei denen eine schnelle und unkomplizierte Bewegungserfassung im Vordergrund steht und keine hochpräzise biomechanische Analyse zwingend erforderlich ist. Mögliche Einsatzgebiete wären Fitness- und Trainings-Apps, Bewegungstracking im Alltag, Gaming, einfache Haltungsanalysen sowie Voranalysen.

Die Pose2Sim-OpenSim-Pipeline ist hingegen vor allem für wissenschaftliche und klinische Anwendungen geeignet, bei denen eine hohe Genauigkeit und biomechanische Konsistenz der Bewegungsdaten erforderlich sind. Besonders vorteilhaft ist der Einsatz in Umgebungen, in denen das Kamerasetup über längere Zeit unverändert bleibt, beispielsweise in Laboren, beispielsweise bei Entwicklung humanoider Roboter oder Kliniken, z.B. für präzise Ganganalysen,

Rehabilitation nach Verletzungen, etc. In solchen Szenarien kann die aufwendige Kalibrierung langfristig genutzt werden und die hohe Rekonstruktionsqualität optimal genutzt werden.

Einordnung der Ergebnisse

Zusammenfassend zeigt das Projekt, dass markerlose Bewegungserfassung in Kombination mit biomechanischen Modellen ein hohes Potenzial für zukünftige Anwendungen besitzt, insbesondere dort, wo flexible, kostengünstige und alltagstaugliche Erfassungssysteme gefragt sind.

Für präzise biomechanische Analysen sind jedoch weiterhin robuste 3D-Rekonstruktionsverfahren, eine sorgfältige Modellskalierung sowie eine Validierung der Ergebnisse erforderlich. Außerdem wird man niemals eine genauso hohe Genauigkeit bei monokularen Systemen, wie im Vergleich zu Mehrkamerasystemen erreichen können, da die Tiefeninformation bei Mehrkamerasystemen durch Triangulierung wesentlich besser bestimmt werden kann.

3.2 Ausblick

Das vorliegende Projekt stellt eine funktionale Pipeline zur markerlosen Bewegungserfassung und biomechanischen Auswertung in OpenSim bereit. Die erzielten Ergebnisse zeigen jedoch zugleich mehrere Ansatzpunkte für weiterführende Arbeiten, die von einer nachfolgenden Projektgruppe aufgegriffen und vertieft werden können.

Ein zentraler Schwerpunkt zukünftiger Arbeiten liegt in der weiteren Verbesserung und Erweiterung der bestehenden Pipeline. Insbesondere bietet sich ein systematischer Vergleich unterschiedlicher Pose-Estimation-Modelle an. Neben weiteren Modellen innerhalb von MediaPipe könnten auch alternative Frameworks wie OpenPose oder vergleichbare Open-Source-Ansätze untersucht werden, um deren Robustheit, Genauigkeit und Eignung für biomechanische Anwendungen zu bewerten.

Darüber hinaus kann die Genauigkeit der 3D-Rekonstruktion insbesondere im Pose2Sim-Ansatz weiter erhöht werden. Während im aktuellen Projekt ein Setup mit zwei Kameras betrachtet wurde, könnte der Einsatz zusätzlicher Kameras die Tiefenrekonstruktion stabilisieren und die Auswirkungen von Selbstüberdeckungen reduzieren. Eine Analyse des Einflusses der Kamerazahl, Positionierung und Kalibrierungsqualität auf die Rekonstruktionsergebnisse stellt dabei einen vielversprechenden Ansatz dar.

Ein weiterer möglicher Entwicklungsschritt betrifft die inverse Kinematik. Neben dem in OpenSim integrierten IK-Tool könnten alternative IK-Verfahren untersucht werden, beispielsweise durch eine eigene Implementierung oder durch den Vergleich mit anderen Optimierungsansätzen. Ziel wäre es, die Sensitivität gegenüber Markerfehlern zu reduzieren und stabilere Gelenkwinkelverläufe zu erzielen.

Auch die Skalierung der biomechanischen Modelle bietet weiteres Optimierungspotenzial. Insbesondere Körpersegmente mit variabler Länge, wie beispielsweise die Verbindung zwischen Schulter und Hüfte in markerlosen Pose-Schätzungen, stellen derzeit eine Herausforderung dar. Zukünftige Arbeiten könnten neue Skalierungsstrategien entwickeln, etwa durch zusätzliche geometrische Constraints, anthropometrische Modelle oder zeitliche Mittelung über geeignete Bewegungssequenzen.

Darüber hinaus könnte der Pose2Sim-Ansatz weiter verbessert und systematisch mit markerbasierten Bewegungserfassungssystemen verglichen werden. Da markerbasierte Systeme derzeit als Referenzstandard gelten, würde ein solcher Vergleich eine fundierte Validierung der markerlosen Pipeline ermöglichen und deren biomechanische Genauigkeit besser einordnen.

Ein weiterer vielversprechender Ansatz besteht in der Erweiterung der Pose-Estimation durch zusätzliche Sensorik. Beispielsweise könnte die Kombination von Smartphone-Kameras mit integrierten LiDAR-Sensoren genutzt werden, um zusätzliche Tiefeninformationen zu erfassen und die 3D-Rekonstruktion insbesondere bei monokularen Ansätzen zu verbessern. Die Fusion visueller und distanzbasierter Daten stellt dabei ein interessantes Forschungsfeld dar.

Insgesamt bietet das Projekt eine solide Grundlage für weiterführende Untersuchungen, die sowohl die methodische Genauigkeit als auch die praktische Anwendbarkeit markerloser Bewegungserfassungssysteme in biomechanischen Analysen weiter verbessern können.

4 Literaturverzeichnis

- [1] B. Rosenhahn, "Markerfreies Motion Capture: Neue Wege zur Analyse menschlicher Bewegungen." <https://www.mpg.de/417219/forschungsSchwerpunkt1>, 2007. Zugriff am: 15.01.2026.
- [2] Mimic Productions, "Mocap suit: How does a motion capture suit work?" <https://www.mimicproductions.com/post/motion-capture-suits>, 2024. Zugriff am: 15.01.2026.
- [3] A. C. Bullinger-Hoffmann, ed., *Homo Sapiens Digitalis – Virtuelle Ergonomie und digitale Menschmodelle*. Berlin, Heidelberg: Springer Vieweg, 2017.
- [4] P. Grimm and Y. Jung, "Webbasierte Anwendungen virtueller Techniken," in *Virtual Reality und Augmented Reality (VR/AR)* (R. Dörner, W. Broll, P. Grimm, and B. Jung, eds.), Berlin, Heidelberg: Springer Vieweg, 2. ed., 2019.
- [5] H. Greaves, A. Eleuteri, G. Barton, M. Robinson, K. Gibbon, and R. Foster, "Comparison of marker- and markerless-derived lower body three-dimensional gait kinematics in typically developing children," *Sensors*, vol. 25, p. 4249, 07 2025.
- [6] Y. Desmarais, "A review of 3D human pose estimation algorithms for markerless motion capture. Computer Vision and Image Understanding." <https://doi.org/10.1016/j.cviu.2021.103275>, 2021. Zugriff am: 15.01.2026.
- [7] A. Avogaro, F. Cunico, B. Rosenhahn, and F. Setti, "Markerless human pose estimation for biomedical applications: A survey. Frontiers in Computer Science,5." <https://doi.org/10.3389/fcomp.2023.1153160>, 2023. Zugriff am: 15.01.2026.
- [8] K. Song, "Accuracy of markerless motion capture for estimating lower-extremity kinematics and kineticsJournal of Biomechanics, 149, 111406.." <https://www.sciencedirect.com/science/article/abs/pii/S0021929023003214?via%3Dihub>, 2023. Zugriff am: 15.01.2026.
- [9] Y. Guo, " A survey of contemporary deep learning-based techniques for 3D human pose estimation. Sensors,25(8), 2904.." <https://doi.org/10.3390/s25082409>, 2025. Zugriff am: 15.01.2026.
- [10] OpenSim, "Musculoskeletal models - opensim documentation." <https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53090000/Scaling>, 2024. Zugriff am: 13.01.2026.

- [11] OpenSim, "Musculoskeletal models - opensim documentation." <https://opensimconfluence.atlassian.net/wiki/spaces/OpenSim/pages/53090037/Inverse+Kinematics>, 2024. Zugriff am: 13.01.2026.
- [12] PerfAnalytics, "Pose2sim: Robust markerless kinematics from open-source pose estimation." <https://github.com/perfanalytics/pose2sim>, 2024. Zugriff am: 13.01.2026.
- [13] Projekt Markerlose Bewegungserfassung, "GitHub-Repository des Projektes." <https://github.com/PhilippSG/Markerlose-Bewegungserfassung/blob/main>, 2026. GitHub-Repository, Zugriff am: 16.01.2026.
- [14] D. Rode, A. Dunkel, R. Willi, P. Wolf, M. Xiloyannis, and R. Riener, "Assessment of monocular human pose estimation models for clinical movement analysis," *Scientific Reports*, vol. 15, no. 1, p. 38767, 2025.