HAW HAMBURG

BACHELOR THESIS
Erik Hildebrandt

Entwicklung eines Open-Source-Systems für die Multi-Kamera Human Pose Estimation

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Erik Hildebrandt

Entwicklung eines Open-Source-Systems für die Multi-Kamera Human Pose Estimation

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

Betreuender Prüfer: Prof. Dr. Christian Lins
Zweitgutachter: Prof. Dr. Jan Sudeikat

Eingereicht am: 27. Juli 2023

Erik Hildebrandt

Thema der Arbeit

Entwicklung eines Open-Source-Systems für die Multi-Kamera Human Pose Estimation

Stichworte

Human Pose Estimation, Open-Source, Multi-Kamera

Kurzzusammenfassung

In diesem Projekt wurde darauf abgeziehlt, ein Open-Source-System zur Human Pose Estimation zu entwickeln, das auf den Frameworks OpenCV und MediaPipe basiert und als Grundlage für zukünftige Entwicklungen dient. Das System wird auch mit einem anderen Pose Estimation System verglichen, um seine Leistungsfähigkeit und Genauigkeit zu evaluieren.

Erik Hildebrandt

Title of Thesis

Development of an open-source system for multi-camera human pose estimation

Keywords

Human Pose Estimation, Open-Source, Multi-Camera

Abstract

In this project, I aim to develop an open-source human pose estimation system based on the OpenCV and MediaPipe frameworks to serve as a foundation for future developments. The system will also be compared to another pose estimation system to evaluate its performance and accuracy.

Inhaltsverzeichnis

Abbildungsverzeichnis **vii**

Tabellenverzeichnis **viii**

1 Einleitung **1**
- 1.1 Hintergrund und Motivation . . . 1
- 1.2 Zielsetzung . . . 2
- 1.3 Methodik . . . 2
- 1.4 Stand der Forschung . . . 3
- 1.5 Struktur der Arbeit . . . 5

2 Literaturübersicht **7**
- 2.1 Grundlegende Konzepte der Human Pose Estimation . . . 7
- 2.2 Frühe Arbeiten zur Human Pose Estimation . . . 8
- 2.3 Aktuelle Forschung im Bereich der Human Pose Estimation . . . 9
- 2.4 Anwendungen der Human Pose Estimation . . . 10
- 2.5 Limitationen und Herausforderungen . . . 11

3 Methodik **13**
- 3.1 2D und 3D Pose Estimation . . . 13
- 3.2 Neuronale Netze und Deep Learning Modelle . . . 14
- 3.3 Datensätze . . . 15
- 3.4 Architekturen . . . 16
  - 3.4.1 Fully-Connected Network (FCN) . . . 16
  - 3.4.2 Region-based Convolutional Neural Network (R-CNN) . . . 16
  - 3.4.3 Single Shot MultiBox Detector (SSD) . . . 17
  - 3.4.4 You-Only-Look-Once (YOLO) . . . 17
  - 3.4.5 BlazePose . . . 18

4 Implementierung **20**
4.1 MediaPipe . . . 20
4.1.1 Einführung in MediaPipe . . . 20
4.1.2 Erläuterung . . . 21
4.1.3 Beschreibung . . . 21
4.1.4 Vorteile von MediaPipe . . . 22
4.1.5 Installation von MediaPipe . . . 23
4.2 OpenCV . . . 23
4.2.1 Einführung in OpenCV . . . 23
4.2.2 Funktionen . . . 24
4.2.3 OpenCV in der Pose Estimation . . . 25
4.2.4 Vorteile von OpenCV . . . 25
4.2.5 Installation von OpenCV . . . 26
4.3 Architektur . . . 27
4.4 Code Erklärung . . . 33
4.4.1 Interface Klassen . . . 33
4.4.2 DataStreamModule Klasse . . . 34
4.4.3 HPEModule Klasse . . . 36
4.4.4 PoseEstimation GUI Klasse . . . 39

5 Ergebnisse **45**
5.0.1 Leistungsvergleich . . . 45
5.0.2 Frame Vergleich . . . 46
5.0.3 Benchmark Ergebnisse . . . 52

6 Zusammenfassung **55**

7 Ausblick **58**
7.0.1 Verbesserung der Genauigkeit durch die Verwendung mehrerer Kameras und Tiefenkameras . . . 58
7.0.2 Virtuelle Realität (VR) und erweiterte Realität (AR) . . . 59
7.0.3 Gesundheit und Fitness . . . 60
7.0.4 Überwachungssysteme . . . 61
7.0.5 Robotik . . . 62
7.0.6 Fazit . . . 62

Literaturverzeichnis **64**

Selbstständigkeitserklärung 68

Abbildungsverzeichnis

3.1 Die 33 Landmarks, die mithilfe von Blazepose detektiert werden.[1] 19

4.1 Technischer Kontext 27
4.2 Systemkontextdiagramm 28
4.3 Use Case Diagramm 29
4.4 Komponentendiagramm 30
4.5 Sequenzdiagramm live Kamera 31
4.6 Sequenzdiagramm Video Datei 32
4.7 GUI 43

5.1 Yoga Pose OpenPose 46
5.2 Yoga Pose erarbeitetes System 47
5.3 Yoga Pose OpenPose 48
5.4 Yoga Pose des erarbeitetem Systems 48
5.5 Breakdance Pose OpenPose 49
5.6 Breakdance Pose erarbeitetes System 50
5.7 Breakdance Pose OpenPose 51
5.8 Breakdance Pose erarbeitetes System 51

Tabellenverzeichnis

5.1 Model FPS und PCK@0.2 auf verschiedenen Datensätzen 53

1 Einleitung

1.1 Hintergrund und Motivation

Die Human Pose Estimation hat in den letzten Jahren große Fortschritte gemacht, besonders durch den Einsatz von Deep Neural Networks (DNNs) und anderen Machine-Learning-Techniken. Diese Fortschritte tragen dazu bei, die Funktionalität der Human Pose Estimation zu optimieren und ihren Einsatzbereich in verschiedenen Bereichen zu erweitern. Deswegen findet die Human Pose Estimation immer mehr Verwendung in verschiedenen Bereichen wie Robotik, virtuelle Realität, Filmproduktion, Bewegungsanalyse und medizinische Diagnose. Beispielsweise kann in der Robotik die Human Pose Estimation Robotern dabei behilflich sein, menschenähnliche Bewegungen auszuführen, und sogar bei der Pflege älterer oder behinderter Menschen zu helfen. In der virtuellen Realität kann Human Pose Estimation dabei helfen, ein realistischeres und immersives Erlebnis zu erreichen, indem die Bewegungen des Benutzers in Echtzeit verfolgt und in die virtuelle Realität eingebaut werden. Bei der Sportanalyse kann die Human Pose Estimation verwendet werden, um die Technik eines Sportlers zu korrigieren und Verletzungen vorzubeugen, indem Bewegungsmuster und Bewegungssequenzen genau ausgewertet werden. Außerdem kann in der medizinischen Diagnose die Human Pose Estimation helfen, die Beweglichkeit der Gelenke und die Körperhaltung des Patienten zu beurteilen. Das Tracking kann dann zusätzlich noch verbessert werden durch die Verwendung mehrerer Kameras, da diese zusätzliche Information über Körperbewegung und Position aus verschiedenen Winkeln liefern, was dazu beitragen kann, die Genauigkeit und Zuverlässigkeit der Human Pose Estimation zu verbessern. Die Human Pose Estimation über mehrere Kameras steht jedoch vor verschiedenen Herausforderungen, wie z.B. der Synchronisierung von Kameras, der Integration von Daten aus verschiedenen Quellen und der Verarbeitung großer Datenmengen. Aus diesem Grund ist die Entwicklung von Systemen zur Human Pose Estimation mit mehreren Kameras ein relativ neuer und wichtiger Bereich der Forschung und Entwicklung. Wenn man an solch großen Projekten arbeitet, ist es immer von

Vorteil, wenn diese Open-Source gemacht werden, da Open-Source-Systeme sich in der Computer-Vision-Forschung und Entwicklung als sehr nützlich erwiesen haben, weil sie die Zusammenarbeit und den Wissensaustausch fördern. Das Open-Source-System bietet eine transparente, modulare und skalierbare Architektur, mit der Forscher und Entwickler ihre eigenen Algorithmen integrieren und ihre eigenen Daten verwenden können.[5] [27] [29] [34]

1.2 Zielsetzung

Das Ziel dieser Arbeit ist es, ein System zur Human Pose Estimation mit mehreren Kameras zu entwickeln. Das System soll eine benutzerfreundliche GUI (Graphical User Interface) bieten, mit der der Benutzer Kameras auswählen und die Human Pose Estimation auf den ausgewählten Kameras durchführen kann. Das System sollte auch in der Lage sein, Daten von einer Vielzahl von Kameras und Quellen zu unterstützen. Um dieses Ziel zu erreichen, wurden in dieser Arbeit mehrere Schritte durchgeführt. Zunächst wird eine umfassende Literaturrecherche durchgeführt, um den Forschungsstand auf dem Gebiet der Human Pose Estimation zu verstehen. Anschließend wird unter Verwendung von Open-Source-Softwarebibliotheken und vorhandenen Algorithmen eine für das System geeignete Architektur entworfen und implementiert. Das entwickelte System wird dann durch Experimente und Tests bewertet, um seine Leistung und Genauigkeit zu beurteilen. Die Bewertung wird auch dazu beitragen, die Stärken und Schwächen des Systems zu identifizieren und mögliche Verbesserungen zu empfehlen. Durch diese Arbeit soll es möglich sein, in Zukunft an diesem Projekt weiterzuarbeiten, um dann die Sensorfusion der verschiedenen Kameras zu unterstützen, um ein präziseres Ergebnis zu erschaffen.

1.3 Methodik

Bei der Entwicklung dieses Systems wurden zwei bewährte und leistungsstarke Technologien verwendet: OpenCV und MediaPipe. OpenCV ist eine Open-Source-Computer-Vision- und Machine-Vision-Bibliothek, die komplexe Bildverarbeitungsaufgaben löst. Es stehen verschiedene Funktionen und Algorithmen für Bildfilterung, Merkmalserkennung, Objekterkennung, Kamerakalibrierung, Stereo- und Strukturberechnungen zur Verfügung. Die Entscheidung fiel auf OpenCV, weil es in C++ geschrieben ist und APIs für Python und andere Sprachen bereitstellt. Auf der anderen Seite ist MediaPipe ein

leistungsstarkes Open-Source-Echtzeit-Multimedia-Verarbeitungsframework, das Deep-Learning-Modelle zur Analyse von Bilddaten nutzt. Für die Erkennung menschlicher Posen in Bildern wurde MediaPipe eingesetzt. Es bietet eine große Auswahl an vortrainierten Modellen und Pipeline-Komponenten zur Erstellung von Anwendungen für Gesichts-, Gesten- und Posenerkennung. Die Kombination von OpenCV und MediaPipe ermöglichte die Entwicklung eines Human Pose Estimation Systems mit hoher Genauigkeit und Leistung. Die verwendeten Techniken erlauben es, ein einfaches, zuverlässiges und skalierbares System zur Schätzung menschlicher Posen zu entwickeln, das auf einer Vielzahl von Geräten und Plattformen ausgeführt werden kann. [4] [6]

1.4 Stand der Forschung

Die Forschung im Bereich der Human Pose Estimation hat in den letzten Jahren erhebliche Fortschritte gemacht. Hier sind einige wichtige Entwicklungen:

- **Deep Learning-Modelle**: Human Pose Estimation-Systeme haben durch den Einsatz von Deep Learning-Modellen enorme Fortschritte gemacht. Im Vergleich zu traditionellen Methoden, wie z.B. dem Random Forest-basierten Ansatz, können Deep Learning-basierte Human Pose Estimation-Modelle wesentlich genauere Ergebnisse liefern. Convolutional Neural Networks (CNNs) und ihre Varianten wie Region-based Convolutional Neural Networks (R-CNNs), Fully Convolutional Networks (FCNs) und Hourglass Networks sind einige der am häufigsten verwendeten Deep Learning-Architekturen für Human Pose Estimation. Diese Architekturen werden typischerweise mit einer bestimmten Verlustfunktion trainiert, wie zum Beispiel Mean Squared Error (MSE) oder Mean Average Precision (mAP). Ein herausragendes Beispiel für ein Deep Learning-Modell zur Human Pose Estimation ist das OpenPose-Modell, das von Carnegie Mellon University und Facebook AI Research entwickelt wurde. Dieses Modell verwendet ein Multi-Stage-Verfahren zur Detektion von Körperteilen und hat gezeigt, dass es in der Lage ist, die 2D-Positionen von bis zu 135 Körperteilen gleichzeitig zu erkennen. Trotz der vielversprechenden Ergebnisse von Deep Learning-Modellen gibt es auch einige Einschränkungen, wie beispielsweise den Bedarf an großen Trainingsdatensätzen und leistungsstarken Rechenressourcen. Dennoch sind Deep Learning-Modelle derzeit einer der vielversprechendsten Ansätze zur Verbesserung der Leistung von Human Pose Estimation-Systemen. Durch den Einsatz dieser Technologie können zuverlässigere und genaue-

re Ergebnisse erzielen werden, was zu einer breiteren Anwendung von Human Pose Estimation-Systemen in verschiedenen Bereichen führt. [5] [28]

- **3D Pose Estimation**: Neuere Forschungen zur Human Pose Estimation konzentrieren sich auf die Entwicklung von Systemen zur Schätzung der 3D-Positionen von Körperteilen. Solche Systeme haben Anwendungen in unterschiedlichen Bereichen wie Virtual Reality und Gesundheitswesen. Im Gegensatz zur 2D-Lokalisierung ermöglicht die 3D-Posenschätzung eine präzisere Messung von Posen und Bewegungen, was zu einem realistischeren und immersiveren Erlebnis in der virtuellen Realität führt und im Gesundheitswesen zur Diagnose und Behandlung von Erkrankungen eingesetzt werden kann. Die aktuelle Forschung untersucht verschiedene Ansätze zur 3D-Posenschätzung, einschließlich der Verwendung von Stereokameras, Tiefenkameras und der Kombination mehrerer 2D-Bilder. Zu den jüngsten Arbeiten in diesem Bereich gehören „Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification“ von A. Diba et al., 2017, „3D Human Pose Estimation in the Wild by Adversarial Learning“ von W. Yang et al., 2018 und „Monocular 3D Human Pose Estimation In the Wild Using Improved CNN Supervision“ von D. Mehta et al., 2017. Diese Arbeiten stellen einen wichtigen Fortschritt in der Entwicklung der 3D-Posenschätzung dar und demonstrieren das Potenzial dieser Technik in verschiedenen Anwendungsbereichen. Obwohl die Entwicklung der 3D-Posenschätzung noch in den Kinderschuhen steckt und einige Herausforderungen mit sich bringt, wie z. B. die Notwendigkeit großer Trainingsdatensätze und leistungsstarker Rechenressourcen, sind die Möglichkeiten, die sie bieten, vielversprechend. [8] [33] [20]

- **Multi-Personen-Tracking**: Fortschritte in der Bildverarbeitung und im maschinellen Lernen haben auch zur Entwicklung von Systemen geführt, die mehrere Personen in Bildern verfolgen. Solche Systeme können in vielen Anwendungen wie Überwachung, Crowd-Analyse oder der Unterhaltungsindustrie eingesetzt werden. Ein Beispiel ist das PoseFlow-System von Y. Xiu et al. Es verwendet eine neuronale Netzwerkarchitektur, die für die Verfolgung mehrerer Personen entwickelt wurde. Es kann die Bewegung von bis zu 100 Personen in Echtzeit verfolgen und eine 2D-Positions- und Orientierungsschätzung jedes Körperteils liefern. Diese und andere Studien demonstrieren, dass es möglich ist, zuverlässige und effiziente Mehrpersonen-Tracking und Human Pose Estimation Syteme zu entwickeln. Ein solches System hat das Potenzial, in einer Vielzahl von Anwendungen eingesetzt

zu werden, und stellt einen wichtigen Fortschritt in der umfassenden Verarbeitung und Analyse visueller Daten dar. [32]

1.5 Struktur der Arbeit

Die vorliegende Bachelorarbeit ist wie folgt strukturiert:

- Kapitel 1 - Einleitung: In diesem Kapitel wird die Motivation und Zielsetzung der Arbeit erläutert. Zudem werden relevante Hintergrundinformationen zum Thema Human Pose Estimation, der Stand der Forschung und den verwendeten Technologien wie OpenCV und MediaPipe kurz angeschnitten und vorgestellt.
- Kapitel 2 - Literaturübersicht: In diesem Abschnitt wird eine Übersicht über die relevanten Literaturquellen gegeben, um den aktuellen Stand der Technik in Bezug auf die behandelten Themen zu erfassen. Es werden verschiedene Forschungsprojekte und Publikationen untersucht, um eine solide Basis für die eigene Forschung zu schaffen. Die Recherche wird über wissenschaftliche Datenbanken und Suchmaschinen durchgeführt, um eine möglichst vollständige Liste relevanter Quellen zu erhalten. Zusätzlich werden auch verwandte Arbeiten und Projekte in diesem Bereich berücksichtigt, um einen ganzheitlichen Überblick über den aktuellen Stand der Forschung zu geben. Es werden auch die bestehenden Herausforderungen und Limitationen der Systeme diskutiert, um einen Leitfaden für die eigene Forschung zu erhalten.
- Kapitel 3 - Methodik: In diesem Abschnitt wird die Vorgehensweise und Methodik zur Implementierung von Human Pose Estimation Systemen beschrieben. Es wird detailliert erläutert, welche Technologien zur Umsetzung verwendet werden können, darunter werden auch verschiedene Architekturen erklärt.
- Kapitel 4 - Implementierung: Im folgenden Kapitel wird die praktische Umsetzung des Systems beschrieben. Dabei werden alle notwendigen Schritte von der Beschaffung der Daten über die Datenverarbeitung bis hin zur Ausgabe der Ergebnisse erläutert. Hierbei werden auch die verwendeten Tools und Technologien detailliert beschrieben und erläutert, wie sie in das System integriert wurden.
- Kapitel 5 - Ergebnisse: In diesem Abschnitt werden die Ergebnisse der Human Pose Estimation des implementierten Systems evaluiert. Dabei wird untersucht,

wie genau das System die Körperhaltung der Personen in verschiedenen Szenen erkennt. Zudem wird das System mit anderen existierenden Systemen wie OpenPose verglichen, um die Effektivität und Genauigkeit des Systems zu bewerten.

- Kapitel 6 - Zusammenfassung: Im vorletzten Kapitel werden die Ergebnisse der Arbeit noch einmal zusammengefasst und auf den Punkt gebracht.
- Kapitel 7 - Ausblick: In diesem abschließenden Kapitel werden potenzielle zukünftige Forschungsrichtungen und Anwendungsbereiche des Systems diskutiert. Es werden auch die Stärken und Schwächen des Systems reflektiert und mögliche Verbesserungen und Erweiterungen aufgezeigt.

2 Literaturübersicht

In diesem Kapitel wird eine Literaturrecherche zur Human Pose Estimation durchgeführt. Dazu werden grundlegende Konzepte und Definitionen erarbeitet, um eine solide Grundlage für das Verständnis verschiedener Forschungsansätze zu schaffen. Frühere Arbeiten und aktuelle Forschungsergebnisse werden analysiert. Ebenso werden Anwendungen zur Human Pose Estimation untersucht sowie die Herausforderungen und Grenzen der Implementierung solcher Systeme diskutiert. Eine umfassende Literaturrecherche ist in diesem Zusammenhang besonders wichtig, um das Verständnis der Human Pose Estimation und ihrer Chancen und Herausforderungen zu verbessern.

2.1 Grundlegende Konzepte der Human Pose Estimation

Die Human Pose Estimation umfasst das Schätzen der Position eines menschlichen Körpers in einem Bild oder Video. Es gibt zwei grundlegende Ansätze zum Schätzen der Position des menschlichen Körpers: 2D- und 3D-Positionsschätzung. Die 2D-Positionsschätzung schätzt die Position eines Körperteils in einem zweidimensionalen Bild, während die 3D-Positionsschätzung die dreidimensionale Position eines Körperteils schätzt. In den letzten Jahren hat sich die Human Pose Estimation von der Verwendung traditioneller Computer-Vision-Methoden zur Verwendung von Deep-Learning-Modellen verlagert. Diese Modelle sind in der Lage, komplexe Beziehungen zwischen Eingabe- und Ausgabevariablen zu lernen und dadurch die Genauigkeit und Robustheit der Schätzung von Körperpositionen zu verbessern. Eine der erfolgreichsten Anwendungen von Deep-Learning-Modellen zur Human Pose Estimation ist das Convolutional Neural Network (CNN). CNNs bestehen aus mehreren Schichten von Neuronen, wobei jede Schicht Filter enthält, die auf die Eingabedaten angewendet werden, um Merkmale auf verschiedenen Abstraktionsebenen zu extrahieren. Diese Merkmale werden dann verwendet, um die Körperposition zu schätzen. Ein weiteres wichtiges Konzept bei der Human Pose Estimation ist die Verfolgung mehrerer Personen. Dazu gehört, mehreren Personen in einem Bild

oder Video zu folgen und ihre Bewegung im Laufe der Zeit zu verfolgen. Diese Technologie wird in verschiedenen Anwendungen wie Überwachungssystemen oder interaktiven Spielen verwendet. Zusammenfassend lässt sich sagen, dass die Human Pose Estimation aufgrund von Fortschritten bei Deep-Learning-Techniken und Bildverarbeitung zu einem wichtigen Forschungsgebiet geworden ist. Die Verwendung von Deep-Learning-Modellen wie CNN und die Entwicklung der 3D-Posenschätzung und der Mehrpersonen-Tracking-Technologie haben die Fähigkeit und Genauigkeit der Human Pose Estimation erheblich verbessert. [35] [28]

2.2 Frühe Arbeiten zur Human Pose Estimation

Frühe Arbeiten zur Human Pose Estimation konzentrierten sich auf die Erkennung und Lokalisierung menschlicher Körper in Bildern unter Verwendung von Computer-Vision-Techniken wie Edge Detection und Template Matching. Diese Methoden sind jedoch fehleranfällig, da sie empfindlich auf schlechte Beleuchtung, Okklusionen und Hintergrundablenkungen reagieren. Mit dem Aufkommen von Deep-Learning-Modellen haben sich die Methoden zur Human Pose Estimation jedoch erheblich weiterentwickelt. Convolutional Neural Networks (CNNs) ermöglichen es diesen Modellen, menschliche Körperteile in Bildern zu erkennen und zu lokalisieren. Die meisten der aktuellen State-of-the-Art-Methoden verwenden Deep-Learning-Modelle für die Human Pose Estimation. Einige neuere Forschungen im Bereich Human Pose Estimation konzentrierten sich auf die Entwicklung von Systemen zur Schätzung der 3D-Positionen von Körperteilen, die in Bereichen wie Virtual Reality oder Gesundheitswesen eingesetzt werden könnten. Fortschritte in der Bildverarbeitung und im maschinellen Lernen haben auch zur Entwicklung von Tracking-Systemen für mehrere Personen geführt, die mehrere Personen in einem einzigen Bild verfolgen können. Trotz der Fortschritte nehmen frühere Arbeiten zur Human Pose Estimation immer noch einen herausragenden Platz in der Forschung ein. Ein Beispiel ist der von Felzenszwalb et al, 2005 entwickelte "Pictorial StructuresÄnsatz. Das Verfahren verwendet ein probabilistisches Modell, um den menschlichen Körper als eine Sammlung miteinander verbundener starrer Teile zu modellieren. Die Methode wurde später weiterentwickelt, um Kontextinformationen und Bewegungsmodelle zu integrieren. Ein weiteres Beispiel ist der von Felzenszwalb et al, 2010 entwickelte „Deformable Part Models“ Ansatz. Das Verfahren verwendet einen Satz verformbarer Teilmodelle, um den menschlichen Körper zu simulieren. Jedes Teilmodell ist in der Lage, lokale Informationen

über den Bereich um das Teil herum zu erfassen und zu integrieren, um eine robustere Posenschätzung bereitzustellen. Insgesamt haben frühere Arbeiten zur Human Pose Estimation wichtige Beiträge zur Entwicklung der heutigen State-of-the-Art-Methoden geleistet und werden weiterhin als Grundlage für zukünftige Forschung auf diesem Gebiet dienen. [11] [10] [35]

2.3 Aktuelle Forschung im Bereich der Human Pose Estimation

Wie eingangs erwähnt, hat die Forschung auf dem Gebiet der Human Pose Estimation in den letzten Jahren bemerkenswerte Fortschritte erzielt. Dies liegt an Fortschritten in der Entwicklung von Deep-Learning-Modellen, einer Art maschineller Lernmethode, die auf künstlichen neuronalen Netzen (KNNs) basiert. Diese Netzwerke können komplexe Datenstrukturen verarbeiten und lernen, indem sie unter Verwendung hierarchischer Schichten von Neuronen Merkmale aus den Daten extrahieren. Der Einsatz von Deep-Learning-Modellen hat erhebliche Fortschritte in der Bild- und Sprachverarbeitung ermöglicht. Die Human Pose Estimation verwendet typischerweise Convolutional Neural Networks (CNNs), um Bilder in Patches zu segmentieren und Merkmale wie Kanten, Farben und Texturen für die Lokalisierung von Körperteilen zu identifizieren. Eine der Herausforderungen bei der Anwendung von Deep-Learning-Modellen auf die Human Pose Estimation besteht darin, dass das Modell nicht nur die Position jedes Körperteils erkennt, sondern auch die Verbindungen zwischen ihnen versteht. Um dieses Problem anzugehen, haben sich einige neuere Forschungen auf die Entwicklung von Modellen konzentriert, die den menschlichen Körper als Ganzes verstehen und analysieren können. Trotz des Erfolgs von Deep-Learning-Modellen bei der Human Pose Estimation gibt es Einschränkungen, z. B. nur die Posen von Personen zu schätzen, deren Bewegung dem Trainingsdatensatz ähnlich ist. In einigen Fällen können Deep-Learning-Modelle auch Schwierigkeiten haben, die Pose einer Person einzuschätzen, die sich außerhalb des Fokus oder außerhalb des Kamerabereichs befindet. Um die Genauigkeit und Geschwindigkeit von Pose Estimation Systemen zu verbessern, konzentrierten sich einige neuere Forschungen auf die Verbesserung der Systeme durch die Verwendung sogenannter „Skip Connections“ in CNNs. Diese Verbindungen ermöglichen es, Informationen von früheren Schichten direkt an spätere Schichten weiterzugeben, um das Problem des verschwindenden Gradienten anzugehen. Dies geschieht, wenn die während des Trainings durch das Netzwerk zurückgereichten

Gradienten in den frühen Schichten des Netzwerks zu klein werden und schließlich verschwinden. Transfer-Lerntechniken können auch verwendet werden, um die Genauigkeit von Pose Estimation Systemen zu verbessern, insbesondere wenn das Trainingsset begrenzt ist. Einige der fortschrittlichsten Techniken zur Human Pose Estimation werden in der Forschung entwickelt und in Arbeiten wie „Real-time 3D Multi-person Pose Estimation from Multiple Views“ von A Elmi et al. (2020), „Simple Baselines for Human Pose Estimation and Tracking“ von B Xiao et al. (2018) und „Associative Embedding: End-to-end Learning for Joint Detection and Grouping“ von A Newell et al. (2017) demonstriert [9] [31] [21]

2.4 Anwendungen der Human Pose Estimation

Die Anwendungsmöglichkeiten der Human Pose Estimation sind vielfältig und reichen von der Unterhaltungsindustrie bis hin zur medizinischen Diagnostik. Im Folgenden werden einige Beispiele vorgestellt:

- Unterhaltungsindustrie: Die Human Pose Estimation wird in der Filmindustrie verwendet, um die Bewegungen von Schauspielern auf digitale Charaktere abzubilden. Dadurch entstehen realistische Bewegungen animierter Charaktere. Die Technologie wird auch in der Entwicklung von Videospielen verwendet, um virtuelle Charaktere lebensechter zu machen.

- Fitness- und Gesundheitsanwendungen: Human Pose Estimation findet auch Anwendung in der Fitness- und Wellnessbranche. Mithilfe von Wearables und Mobilgeräten können die Bewegungen von Personen verfolgt und analysiert werden, um personalisierte Trainingspläne für Fitness-Apps und Reha-Einrichtungen zu erstellen.

- Robotik: Die Human Pose Estimation ist auch in der Robotik von großem Nutzen, da sie bei der Entwicklung humanoider Roboter helfen kann. Mithilfe von Sensoren und Kameras können Roboter menschenähnliche Bewegungen erkennen und nachahmen. Diese Technologie ermöglicht den Einsatz von Robotern in verschiedenen Bereichen wie Pflege oder Transport. Gerade in der Pflege können humanoide Roboter als Unterstützung für ältere Menschen oder als Assistenten von Pflegekräften dienen.

- Medizinische Diagnostik: Die Human Pose Estimation kann in der medizinischen Diagnose verwendet werden, um die Bewegungsmuster von Patienten zu analysieren und mögliche Krankheiten zu identifizieren. Mit der Technologie lassen sich beispielsweise Gelenkprobleme oder Haltungsschäden erkennen. Dies ermöglicht eine frühzeitige Diagnose und gezielte Therapie zur Verbesserung der Gesundheit und des Wohlbefindens der Patienten.
- Sicherheits- und Überwachungsanwendungen: Human Pose Estimation kann in der Sicherheits- und Überwachungsbranche verwendet werden, um die Bewegungen von Personen in Echtzeit zu verfolgen. Durch den Einsatz von Kameras können beispielsweise Unfälle vermieden oder Straftaten verhindert werden. Die Technologie könnte auch in der Automobilindustrie zur Überwachung des Fahrerverhaltens während der Fahrt eingesetzt werden, um die Verkehrssicherheit zu verbessern.
- Virtual Reality: Die Human Pose Estimation ist ein wichtiges Werkzeug in der virtuellen Realität (VR). Bei VR-Anwendungen muss die Bewegung des Benutzers in Echtzeit genau erfasst werden, um eine natürliche Interaktion mit der virtuellen Umgebung zu erreichen. Durch die Verwendung von Algorithmen zur Human Pose Estimation können die Position und Bewegung des Benutzers in Echtzeit erkannt und verfolgt werden. Die Technologie ermöglicht es dem System, die Position und Ausrichtung virtueller Objekte relativ zum Benutzer zu bestimmen, was eine natürliche Interaktion mit der virtuellen Umgebung ermöglicht. Einige Beispiele für VR-Anwendungen, die die Human Pose Estimation verwenden, umfassen Spiele, Simulatoren, Trainingsanwendungen und virtuelle Umgebungen für Therapie- und Rehabilitationszwecke.

[14] [26] [27]

2.5 Limitationen und Herausforderungen

Eine der größten Herausforderungen bei der Human Pose Estimation ist die Komplexität des menschlichen Körpers und seiner Bewegung. Alle Details der menschlichen Anatomie und Bewegungsabläufe in das Modell zu integrieren, war eine Herausforderung. Unterschiede in der Körpergröße, Körperform und Körperbeweglichkeit zwischen Personen können sich ebenfalls auf die Genauigkeit auswirken. Das Erfordernis einer ausreichenden Menge an Trainingsdaten zum Trainieren eines genauen Modells ist ein weiteres Problem,

und das Erhalten einer ausreichenden Menge an Trainingsdaten kann schwierig sein, insbesondere bei seltenen oder ungewöhnlichen Bewegungen. Auch die Datenqualität kann sich auf die Ergebnisse auswirken, insbesondere bei verschwommenen oder überbelichteten Bildern. Die Pose Estimation wird auch schwieriger, wenn die Kleidung des Benutzers ungewöhnlich oder zu weit ist. Das Tragen enger Kleidung kann jedoch die Genauigkeit der Schätzungen verbessern. Es gibt auch schwierige Situationen, wie das Drehen eines Hula-Hoop-Reifens oder Yoga-Posen, die von Standard-Posen abweichen. In diesem Fall kann es schwierig sein, die Pose genau einzuschätzen. Haltungsverzerrungen aufgrund körperlicher Einschränkungen oder Verletzungen erfordern spezielle Modelle oder Anpassungen, um genaue Haltungsschätzungen zu liefern. Echtzeitdaten schnell und effizient zu verarbeiten, ist eine weitere Herausforderung, insbesondere in Anwendungen wie Bewegungserkennung oder Mehrpersonen-Tracking. Hier müssen Modelle entwickelt werden, die schnell und genau genug sind, um in Echtzeit zu arbeiten. Ethische Fragen müssen ebenfalls angegangen werden, da die Human Pose Estimation zur Überwachung oder Verfolgung von Personen verwendet werden kann, was Datenschutz- und Datenschutzbedenken aufwirft. Insgesamt gibt es viele Herausforderungen und Einschränkungen bei der Anwendung der Human Pose Estimation, die berücksichtigt und überwunden werden müssen. Weitere Forschung und Entwicklung in diesem Bereich werden dazu beitragen, diese Herausforderungen anzugehen und die Anwendung der Human Pose Estimation in verschiedenen Bereichen zu verbessern. [30] [5] [28]

3 Methodik

In diesem Abschnitt werden die Methoden zur Erkennung und Analyse von menschlicher Körperhaltung besprochen, die als Human Pose Estimation bekannt sind. Die Human Pose Estimation ist ein wichtiger Teil der Computer Vision und wird in vielen verschiedenen Anwendungen verwendet. So lassen sich beispielsweise Bewegungsanalysen automatisieren, sichere Bereiche überwachen oder virtuelle Anprobemöglichkeiten für die Modebranche entwickeln. In diesem Abschnitt werden verschiedene Techniken zur Human Pose Estimation vorgestellt, einschließlich der Verwendung von Deep Learning und der Kombination verschiedener Bildverarbeitungsmethoden. Darüber hinaus werden die Vor- und Nachteile der verschiedenen Ansätze diskutiert und Empfehlungen gegeben, um festzustellen, welcher Ansatz für welche Situation am besten geeignet ist. [35]

3.1 2D und 3D Pose Estimation

Dieser Abschnitt stellt zwei grundlegende Methoden der Human Pose Estimation vor: 2D-Posenschätzung und 3D-Posenschätzung. Die 2D-Posenschätzung erfasst die Position von Körperteilen in zweidimensionalen Bildern und wird häufig zur Überwachung und Erkennung menschlicher Aktivitäten in der Videoüberwachung oder Spieleentwicklung verwendet. Im Gegensatz dazu erfasst die 3D-Pose-Schätzung die Position menschlicher Körperteile im dreidimensionalen Raum und kann die räumliche Position menschlicher Körperteile schätzen. Dies ist besonders wichtig bei der Analyse räumlicher Beziehungen zwischen Körperteilen, beispielsweise in der medizinischen Diagnostik oder Bewegungsanalyse. Im Vergleich zur 3D-Posenschätzung ist die 2D-Posenschätzung relativ einfach durchzuführen, da nur zweidimensionale Bilder erforderlich sind. Daher ist es normalerweise schneller und einfacher als die 3D-Posenschätzung. Allerdings gibt es auch einige Nachteile. Einerseits ist die 2D-Posenschätzung anfälliger für Verzerrungen und Ungenauigkeiten, die durch unterschiedliche Kamerapositionen, Blickwinkel oder Lichtverhältnisse verursacht werden können. Andererseits sind bestimmte Posen und Handlun-

gen anhand nur eines Bildes nicht immer eindeutig erkennbar. Im Gegensatz dazu ist die 3D-Posenschätzung genauer und besser in der Lage, die räumliche Beziehung zwischen Körperteilen zu erfassen. Es erfordert jedoch normalerweise mehr Kameras oder umfangreichere Kalibrierungen, um eine genaue räumliche Rekonstruktion durchzuführen. Wie schon in Kapitel 2.5 Limitationen und Herausforderungen erwähnt, kann auch die Kleidung oder Körperhaltung einer Person die Ergebnisse beeinflussen. [18] [20] [32]

3.2 Neuronale Netze und Deep Learning Modelle

Ein wichtiger Aspekt der menschlichen Posenschätzung ist die Verwendung von neuronalen Netzen und Deep-Learning-Modellen. Diese Modelle haben in letzter Zeit erhebliche Fortschritte in der Genauigkeit und Effizienz von Posenschätzungsalgorithmen gemacht. Ein neuronales Netzwerk besteht aus miteinander verbundenen Neuronen, die Informationen von einer Neuronenschicht zur nächsten weitergeben. Bei der menschlichen Haltungsschätzung werden neuronale Netze häufig verwendet, um die 2D- oder 3D-Positionen menschlicher Gelenke oder Punkte vorherzusagen. Es gibt verschiedene Ansätze zur Erreichung dieses Ziels, wie z.B. das Fully-Connected Network (FCN), das Region-based Convolutional Neural Network (R-CNN), das Single Shot MultiBox Detector (SSD) und das You-Only-Look-Once (YOLO) Netzwerk. FCN ist ein typisches Beispiel für ein Deep-Learning-Modell, bei dem vollständig verbundene Schichten durch Convolutional Layers ersetzt werden, wodurch das Modell räumliche Informationen besser berücksichtigen kann. R-CNN verwendet Region Proposals, um Regionen von Interesse in einem Bild für die Vorhersage zu identifizieren. SSD und YOLO sind auf schnelle Objekterkennung ausgelegt und arbeiten somit in Echtzeit. Ein Deep-Learning-Modell ist ein mehrschichtiges neuronales Netzwerk, das komplexe Beziehungen zwischen Eingabe- und Ausgabedaten lernen kann. Einige der am häufigsten verwendeten Deep-Learning-Modelle, die bei der menschlichen Posenschätzung verwendet werden, sind Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) und Markov Random Fields (MRFs). CNNs sind auf die Verarbeitung von Bildern spezialisiert und können Muster in visuellen Daten erkennen. Die Fähigkeit von RNNs, Datenfolgen zu verarbeiten, macht sie besonders geeignet, um mit dem zeitlichen Verlauf der Körperbewegung umzugehen. Bei der Human Pose Estimation können RNNs verwendet werden, um die Körperbewegung zu verfolgen. MRF (Markov Random Field) ist ein statistisches Modell, das verwendet wird, um komplexe Beziehungen zwischen mehreren Variablen zu modellieren. Bei der Human Pose Estimation kann MRF verwendet werden, um die räumlichen

Verbindungen zwischen verschiedenen Körperteilen und deren Wechselbeziehungen zu modellieren. Es ist jedoch wichtig zu beachten, dass die Arbeit mit neuronalen Netzen und Deep-Learning-Modellen Herausforderungen mit sich bringt. Komplexität und der Bedarf an großen Datensätzen und Rechenleistung können Herausforderungen darstellen. Die Optimierung von Modellen kann ebenfalls schwierig sein und erfordert oft Erfahrung und Fachwissen in der Modellierung und Programmierung. [23] [19] [24] [16]

3.3 Datensätze

Datensätze und Benchmarks sind unerlässlich, wenn Algorithmen zur Erkennung menschlicher Posen entwickelt und evaluiert werden. Der COCO-Datensatz ist einer der bekanntesten und am häufigsten verwendeten Datensätze in diesem Anwendungsbereich. Es enthält mehr als 330.000 Bilder von insgesamt 2,5 Millionen Menschen, von denen jedes manuell von mindestens fünf Annotatoren beschriftet wurde, um die 2D-Positionen von 17 verschiedenen Gelenken zu bestimmen. Die Forschung zur Human Pose Estimation hat durch die Verwendung des COCO-Datensatzes erhebliche Fortschritte gemacht, da er groß und vielfältig genug ist, um genaue Vorhersagen in verschiedenen Kontexten und Umgebungen zu ermöglichen. Ein weiterer wichtiger Datensatz ist der MPII-Datensatz des Max-Planck-Instituts für Informatik. Mit mehr als 25.000 Bildern und insgesamt 40.000 annotierten menschlichen Posen ist es besonders nützlich für die Entwicklung und Bewertung von Methoden zur Posenschätzung in Alltagssituationen und von Algorithmen zur Bewertung bestimmter Körperteile wie Hände und Füße. Der MPII-Datensatz enthält Bilder in verschiedenen Situationen, darunter körperliche Aktivität, Tanz, Yoga und alltägliche Aktivitäten. Dieser wurde in vielen wissenschaftlichen Arbeiten zur Posenschätzung verwendet und hat zu großen Fortschritten auf diesem Gebiet beigetragen. Einige der verwendeten Methoden sind Convolutional Pose Machine (CPM), Multi-Context Attention for Human Pose Estimation (MCA-Net) und Spatial Temporal Graph Convolutional Networks (ST-GCN). Durch die Verwendung von Datensätzen wie COCO und MPII lässt sich die Genauigkeit und Leistung von Algorithmen zur Human Pose Estimation verbessern, was zum Fortschritt auf diesem Gebiet beitragen kann. Ein Beispiel hierfür ist OpenPose, ein auf Deep Learning basierendes Modell, das auf dem COCO-Datensatz trainiert wurde und menschliche Posen in Echtzeit mit hoher Genauigkeit erkennt. [15] [2] [30]

3.4 Architekturen

3.4.1 Fully-Connected Network (FCN)

Der Fully-Connected Network-Ansatz (FCN) wird genutzt, um Convolutional Neural Networks (CNNs) für Bildsegmentierungsaufgaben und die Human Pose Estimation umzusetzen. Im Gegensatz zu herkömmlichen CNNs, die am Ende des Netzes vollständig verbundene Schichten verwenden, um eine Klassifizierung für ein gegebenes Bild zu liefern, nutzt FCN eine Kombination aus Convolutional-Schichten und Upsampling-Schichten, um Vorhersagen für jedes Pixel im Eingabebild zu treffen. Die Architektur von FCN besteht aus mehreren Convolutional-Schichten zur Merkmalsextraktion, gefolgt von Pooling-Schichten, die die Größe des Eingabebildes reduzieren und so Rechenressourcen einsparen. Anschließend werden Upsampling-Schichten verwendet, um das Ausgabebild auf die gleiche Größe wie das Eingabebild zu bringen. Durch die Verwendung von Upsampling-Schichten bleiben im Ausgabebild Details erhalten, die in der Pooling-Phase normalerweise verloren gehen. FCN kann genutzt werden, um eine Vorhersage für jedes Gelenk im Körper zu treffen, indem das Eingabebild in das Netzwerk eingespeist wird und eine Vorhersage für jedes Pixel im Bild getroffen wird, welches ein Gelenk repräsentiert. Durch die Verwendung von FCN können präzise Vorhersagen für jedes Gelenk getroffen werden, aber es hat auch einige Nachteile, wie beispielsweise die Anfälligkeit für räumliche Ungenauigkeiten, da es schwierig ist, die exakte Position jedes Gelenks im Eingabebild zu bestimmen. Darüber hinaus können lange Trainingszeiten und hohe Rechenleistung erforderlich sein, um ein optimales Modell zu trainieren. [19] [30]

3.4.2 Region-based Convolutional Neural Network (R-CNN)

Das Region-based Convolutional Neural Network (R-CNN) ist ein Deep-Learning-Modell, das in der Objekterkennung und Objektsegmentierung eingesetzt wird. Im Gegensatz zu anderen Ansätzen, die das gesamte Bild auf einmal betrachten, arbeitet R-CNN auf einer Regionen-basierten Methode. Hierbei werden zunächst potenzielle Regionen im Bild identifiziert, die ein Objekt enthalten könnten, bevor diese Regionen mit einem Convolutional Neural Network (CNN) verarbeitet werden, um die tatsächlichen Objekte in der Region zu identifizieren und zu klassifizieren. Das R-CNN-Modell setzt sich aus drei Hauptkomponenten zusammen: Der Region Proposal Network (RPN), dem Feature Extraction Network und dem Classifier. Der RPN identifiziert potenzielle Regionen im Bild

und generiert Bounding-Box-Vorschläge. Diese Vorschläge werden an das Feature Extraction Network weitergeleitet, das für jede Bounding-Box eine Feature-Map extrahiert. Jede Feature-Map wird schließlich von einem Classifier-Netzwerk verarbeitet, das die tatsächlichen Objekte in jeder Region identifiziert und klassifiziert. Der R-CNN-Ansatz hat in verschiedenen Computer-Vision-Anwendungen beeindruckende Ergebnisse erzielt, wie z.B. in der Objekterkennung und Segmentierung. Allerdings hat R-CNN auch einige Nachteile, wie z.B. eine lange Trainingszeit und hohe Speicheranforderungen aufgrund der Verarbeitung vieler Regionen innerhalb des Bildes. Aus diesem Grund wurden schnellere Varianten entwickelt, wie z.B. Fast R-CNN und Faster R-CNN. [7] [12]

3.4.3 Single Shot MultiBox Detector (SSD)

Der Single Shot MultiBox Detector (SSD) ist ein Deep-Learning-Modell, das für Echtzeit-Objekterkennung und Objektlokalisierung entwickelt wurde. Im Vergleich zu anderen Modellen, die einen zweistufigen Prozess nutzen, verwendet SSD eine einzige Vorwärtsdurchlauf-Operation, um gleichzeitig Objekte zu erkennen und zu lokalisieren. Das Modell besteht aus einer Convolutional Neural Network (CNN)-Architektur, gefolgt von mehreren Convolutional-Layern und einer Reihe von Multibox-Layern. Die Multibox-Layer generieren eine Vielzahl von Vorschlägen für Objektlokalisierungen, die dann mithilfe von Regressionsanalyse verbessert werden. Anschließend wird jede Vorschlagsbox einem Score zugeordnet, der angibt, wie wahrscheinlich es ist, dass die Box ein Objekt enthält. SSD hat eine hohe Genauigkeit bei der Erkennung von Objekten in Echtzeit und wird häufig in Anwendungen wie autonomen Fahrzeugen, Gesichtserkennung und Videoüberwachung eingesetzt. [17]

3.4.4 You-Only-Look-Once (YOLO)

Das YOLO-Netzwerk ist ein weiterer Ansatz zur Objekterkennung und Objektverfolgung sowie zur Human Pose Estimation, der von Forschern entwickelt wurde. Im Gegensatz zu anderen Ansätzen betrachtet YOLO das Eingabebild nur einmal, um Objekte zu erkennen und ihre Positionen zu bestimmen. Das Netzwerk unterteilt das Bild in ein Gitter und weist jedem Gitter eine Vorhersage zu, welche mehrere überlappende Objekte enthalten kann. Zudem werden verschiedene Skalen des Bildes genutzt, um Objekte in unterschiedlichen Größen zu erkennen. YOLO durchläuft mehrere Faltungs- und Pooling-Schichten und gibt als Ausgabe eine Liste von Vorhersagen aus, die jedes Gitter und jede

Skala enthalten. Diese Vorhersagen enthalten Wahrscheinlichkeitswerte sowie Positionen und Größen der erkannten Objekte. YOLO ist bekannt für seine schnelle Verarbeitung und Echtzeitfähigkeit bei der Anwendung auf Videodaten. Es hat jedoch Schwierigkeiten bei der Erkennung kleiner Objekte und der Unterscheidung von eng anliegenden Objekten sowie bei der Erfassung feiner Details. [25]

3.4.5 BlazePose

BlazePose ist ein neuer, von Google entwickelter Ansatz zur Human Pose Estimation. Das Verfahren ermöglicht die Verfolgung menschlicher Posen mithilfe von maschinellem Lernen (ML), um 33 2D-Körper-Orientierungspunkte aus einem einzigen Frame abzuleiten. BlazePose lokalisiert mehr Schlüsselpunkte als aktuelle Posenmodelle, was es einzigartig für Fitnessanwendungen macht. Pose Estimation ist besonders für Fitnessanwendungen herausfordernd aufgrund der Vielzahl möglicher Posen, Okklusionen (z.B. der Körper oder andere Objekte verdecken Gliedmaßen aus Sicht der Kamera) und einer großen Anzahl von Looks oder Kleidung. BlazePose verwendet einen bewährten zweistufigen Detektor-Tracker-ML-Pipeline-Ansatz zur Posenerkennung. Unter Verwendung eines Detektors wird zunächst ein Pose-Region-of-Interest (ROI) innerhalb des Rahmens lokalisiert. Der Tracker sagt dann alle 33 Schlüsselpunkte aus diesem ROI voraus. Im Vergleich zu aktuellen Methoden, die für die Inferenz auf leistungsstarke Desktop-Umgebungen angewiesen sind, erreicht diese Methode Echtzeitleistung auf Mobiltelefonen durch CPU-Inferenz. Mithilfe von GPU-Inferenz erreicht BlazePose Ultra-Echtzeitleistung und ermöglicht die anschließende Ausführung von ML-Modellen wie Gesichts- oder Handverfolgung. BlazePose hat eine neue Topologie mit 33 Schlüsselkörperpunkten, die eine Übermenge von COCO-, BlazeFace- und BlazePalm-Topologien darstellt. Dies ermöglicht es, die Körpersemantik nur aus Posenvorhersagen zu bestimmen, was mit Gesichts- und Handmodellen konsistent ist. Die Posenerkennungskomponente der Pipeline sagt die Positionen aller 33 Personenschlüsselpunkte mit vier Variablen (x, y, z und Sichtbarkeit) plus den beiden virtuellen Ausrichtungsschlüsselpunkten voraus. Im Gegensatz zu aktuellen Methoden, die rechenintensive Heatmap-Vorhersagen verwenden, verwendet dieses Modell einen Regressionsansatz, der durch kombinierte Heatmap/Offset-Vorhersagen aller Schlüsselpunkte überwacht wird. BlazePose ist in MediaPipe für Android, iOS und Python verfügbar und wird in einer kommenden Version von ML Kit auch der breiteren Community mobiler Entwickler über die Posenerkennungs-API zur Verfügung gestellt. [4] [3]

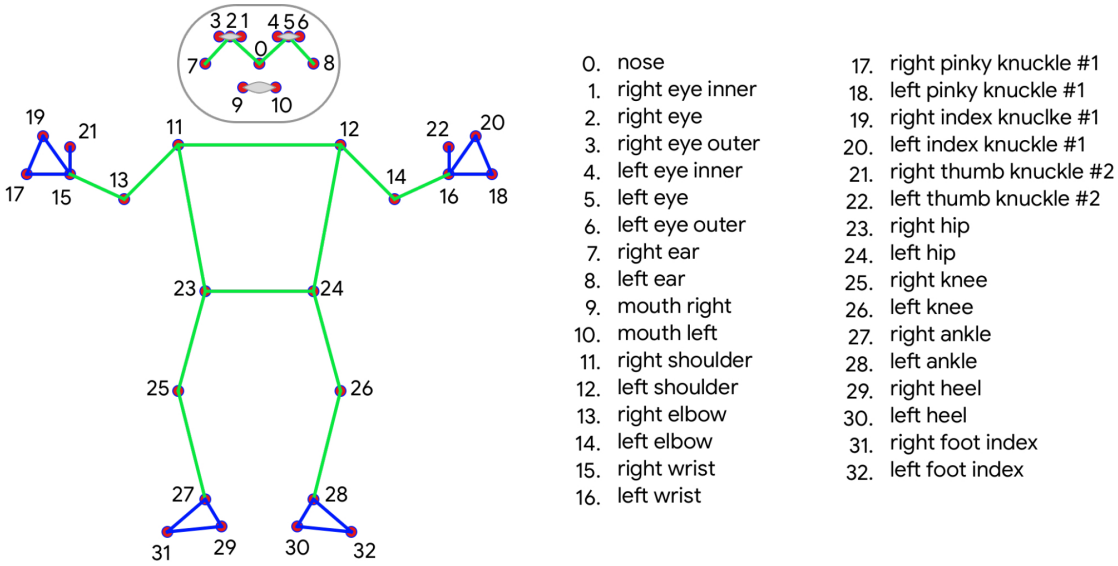

Abbildung 3.1: Die 33 Landmarks, die mithilfe von Blazepose detektiert werden.[1]

4 Implementierung

Im vorigen Kapitel wurden die theoretischen Grundlagen der Human Pose Estimation erläutert. Nun erfolgt die praktische Umsetzung dieser Konzepte. In diesem Kapitel werden die verwendeten Techniken und Werkzeuge vorgestellt, und die verschiedenen Schritte der Umsetzung werden detailliert erläutert – von der Datenvorverarbeitung bis zur Visualisierung der Ergebnisse. Die wesentlichen Features von MediaPipe und OpenCV werden vorgestellt und gezeigt, wie sie in der Umsetzung eingesetzt werden. Das Hauptziel der Implementierung besteht darin, eine solide Grundlage für die weitere Entwicklung des Systems zu schaffen und einen Einblick in die praktischen Herausforderungen bei der Implementierung dieser Technik zu vermitteln.

4.1 MediaPipe

4.1.1 Einführung in MediaPipe

MediaPipe ist ein von Google entwickeltes Framework, das Entwicklern verschiedene Tools und Funktionen zum Erstellen von Modellen für maschinelles Lernen zur Verfügung stellt. Diese Modelle können in verschiedenen Anwendungen wie Computer Vision, Audioverarbeitung und Sensorfusion verwendet werden. MediaPipe läuft nicht nur auf Smartphones, sondern auch auf anderen IoT-Geräten und kann für diese optimiert werden. Die modulare Struktur von MediaPipe ermöglicht es Entwicklern, verschiedene Komponenten und Modelle zu verbinden, um individuelle Lösungen für ihre spezifischen Anwendungsfälle zu erstellen. Es bietet auch eine Vielzahl vorgefertigter Modelle und Komponenten, mit denen Entwickler ihre Entwicklung in Projekten beschleunigen können. Alles in allem ist MediaPipe ein leistungsstarkes Framework, das Entwicklern hilft, fortschrittliche Anwendungen schnell und effizient zu erstellen. [13] [4]

4.1.2 Erläuterung

MediaPipe besteht aus verschiedenen Komponenten, die zusammenarbeiten, um Modelle zu erstellen. Eine Schlüsselkomponente ist ein Framework zum Erstellen von Pipelines, mit dem Entwickler den Datenfluss definieren können, der durch verschiedene Komponenten fließt, und bestimmte Vorgänge ausführen kann, z. B. die Vorverarbeitung von Daten oder die Anwendung von Modellen für maschinelles Lernen. Ein weiterer wichtiger Bestandteil ist die Modellbibliothek, die bereits vortrainierte Modelle für verschiedene Aufgaben enthält. Diese Modelle können in Ihrer eigenen Pipeline verwendet und angepasst werden. Entwickler können das Kalkulator-Framework auch verwenden, um benutzerdefinierte Komponenten zu implementieren und in ihre Pipelines zu integrieren. Mit dem MediaPipe-Visualizer können Entwickler die Ausgabe ihrer Pipelines in Echtzeit visualisieren, um Fehler schnell zu identifizieren und zu beheben. Somit bietet MediaPipe Entwicklern eine Plattform zum Erstellen von Anwendungen für maschinelles Lernen im Bereich Computer Vision und Multimedia, die verschiedene Komponenten wie vortrainierte Modelle, ein Framework zum Erstellen von Pipelines und Tools zum Visualisieren von Ergebnissen umfasst. [13] [4]

4.1.3 Beschreibung

MediaPipe bietet eine Vielzahl von Funktionen, die bei der Implementierung der Human Pose Estimation verwendet werden können.

- Verarbeitungsknoten: MediaPipe bietet eine Vielzahl von Knoten, mit denen Entwickler komplexe Verarbeitungen wie Bildverarbeitung, maschinelles Lernen und Gesichtserkennung implementieren können. Diese Knoten wurden optimiert, um die Leistung zu maximieren, und können problemlos in benutzerdefinierte Graphen integriert werden.
- Tensorflow Lite Inference: MediaPipe verwendet Tensorflow Lite, um maschinelle Lernmodelle effizient auf Mobilgeräten auszuführen. Dies ist besonders nützlich bei der Implementierung der Posenschätzung, da Deep-Learning-Modelle verwendet werden, um die menschliche Pose aus Bildern zu extrahieren.
- GlCompute: MediaPipe unterstützt die GPU-beschleunigte Verarbeitung von Bilddaten mit OpenGL ES, was die Verarbeitung von Echtzeit-Videoströmen auf mobilen Geräten beschleunigt.

- OpenCV: MediaPipe verwendet OpenCV zur Bildverarbeitung für Funktionen wie Rauschunterdrückung, Kantenerkennung und Farbkonvertierung. Diese Funktionen sind besonders nützlich bei der Implementierung der Human Pose Estimation, da sie dazu beitragen können, die Genauigkeit der Ergebnisse zu verbessern.

[13] [4]

4.1.4 Vorteile von MediaPipe

MediaPipe hat viele Vorteile gegenüber anderen Frameworks und Tools für die Entwicklung von Modellen für maschinelles Lernen. MediaPipe ist einfach zu bedienen, bietet eine intuitive Benutzeroberfläche und eine Vielzahl vorgefertigter Verarbeitungsknoten, mit denen Entwickler komplexe Verarbeitungen einfach implementieren können. Es ist leicht zu erlernen und schnell anzuwenden. Es bietet auch eine große Flexibilität für komplexe Verarbeitung, indem benutzerdefinierte Verarbeitungsknoten und Graphen erstellt werden. Es unterstützt mehrere Datenformate und ist mit anderen Frameworks für maschinelles Lernen wie TensorFlow und PyTorch kompatibel. Darüber hinaus unterstützt MediaPipe die Echtzeitverarbeitung von Daten. Es bietet Funktionen wie GPU-beschleunigte Verarbeitung und optimierte Modelle für maschinelles Lernen, um eine hohe Leistung und schnelle Verarbeitung von Echtzeitdaten zu gewährleisten. Zusätzlich ist MediaPipe ein Open-Source-Framework. Es bietet eine offene Architektur, die es Entwicklern ermöglicht, ihre Anwendungen zu erweitern und anzupassen. Die offene Architektur von MediaPipe fördert auch die Zusammenarbeit und den Austausch von Ideen und Innovationen innerhalb der Entwicklergemeinschaft. MediaPipe hat eine breite Anwendbarkeit und kann in einer Vielzahl von Anwendungen verwendet werden, einschließlich Pose Estimation, Objekterkennung, Gesichtserkennung und Augmented Reality. Es bietet eine breite Palette von Funktionen und Tools, mit denen Entwickler die Genauigkeit und Leistung ihrer Anwendungen maximieren können. Aufgrund dieser Vorteile wurde sich für MediaPipe als Lösung für dieses Projekt entschieden. Im späteren Abschnitt werden detaillierte Benchmarks präsentiert, die die besondere Performance von MediaPipe unterstreichen und zeigen, dass es besonders schnell im Vergleich zu anderen Optionen ist. [13] [4]

4.1.5 Installation von MediaPipe

Das Framework bietet eine Vielzahl von Funktionen, einschließlich der Pose Estimation. Um MediaPipe zur Pose Estimation in Python zu nutzen, muss zunächst die MediaPipe Python-Bibliothek in das Projekt eingebunden werden. Die MediaPipe-Bibliothek kann durch Hinzufügen der Bibliothek als Abhängigkeit zum Projekt oder durch Herunterladen der Bibliothek und manuelles Hinzufügen zum Projekt erfolgen. Sobald die Bibliothek hinzugefügt wurde, muss eine Instanz des Pose Estimators erstellt werden und die Eingabedaten müssen bereitgestellt werden. Dazu kann beispielsweise einen Kamerastrom oder ein Videoframe verwenden werden. Anschließend kann man die Pose Estimation durchführen, indem der Estimator auf die Eingabedaten angewendet wird. Das Ergebnis der Pose Estimation wird als Liste von Gelenkpositionen zurückgegeben. Diese Liste enthält die Positionen der verschiedenen Gelenke des Körpers, wie z.B. der Hüfte, der Schultern, der Knie, der Ellbogen und der Handgelenke. Die Gelenkpositionen können verwendet werden, um die Körperhaltung und Bewegungen der Person zu bestimmen. Nachdem die Pose Estimation durchgeführt wurde, können die Ergebnisse weiterverarbeitet werden, um zusätzliche Informationen zu extrahieren. Zum Beispiel kann die Geschwindigkeit oder Richtung der Bewegungen der Person berechnet werden, oder bestimmte Körperhaltungen identifiziert werden, die für die Anwendung wichtig sind. MediaPipe bietet auch Möglichkeiten zur Visualisierung der Ergebnisse, wie z.B. das Zeichnen der Körperhaltung auf dem Video oder die Anzeige der Gelenkpositionen als 3D-Modell. Diese Visualisierungen können dazu beitragen, die Ergebnisse der Pose Estimation zu verstehen und zu analysieren. Dieses fertige Video wird dann in diesem Projekt in der GUI angezeigt. [13] [4]

4.2 OpenCV

4.2.1 Einführung in OpenCV

OpenCV ist eine Open-Source-Computer-Vision-Bibliothek, die in C++, Python und anderen Programmiersprachen verfügbar ist. Sie wurde speziell zur Unterstützung von Bild- und Videobearbeitungsanwendungen entwickelt und kann Aufgaben wie Gesichtserkennung, Objekterkennung, Farbkorrektur und sogar Augmented-Reality-Anwendungen übernehmen. Die OpenCV-Bibliothek bietet eine Vielzahl von Funktionen und Tools zur Bearbeitung von Bildern und Videos. Einige Schlüsselfunktionen von OpenCV sind

die Fähigkeit, Merkmale aus Bildern zu extrahieren, wie z. B. Kantenerkennung, Formerkennung und Textextraktion. Darüber hinaus bietet OpenCV auch eine Vielzahl von Gesichtserkennungs- und Objekterkennungsalgorithmen, einschließlich Haar-Cascades und Support Vector-Machine (SVM)-Algorithmen. OpenCV ist auch in der Lage, 3D-Bildverarbeitungsaufgaben zu bewältigen und bietet verschiedene Tools und Funktionen für die Stereobildverarbeitung, die 3D-Kamerakalibrierung und die 3D-Objekterkennung. Die Bibliothek ist auch zur Echtzeit-Videoverarbeitung fähig und bietet verschiedene Bewegungserkennungs- und Verfolgungsalgorithmen. [22] [6]

4.2.2 Funktionen

OpenCV ist ein leistungsstarkes Bildverarbeitungstool, das eine breite Palette an Funktionen bietet. Eines der Kernkonzepte in der Bildverarbeitung ist die Filterung, die zur Verbesserung der Bildqualität und zum Entfernen von Rauschen verwendet wird. Mit OpenCV können verschiedene Filterfunktionen wie der Medianfilter, der Gauß-Filter und der Bilateral-Filter verwendet werden. Der Medianfilter reduziert Ausreißer im Bild, während der Gauß-Filter Rauschen reduziert und Kanten erhält. Der Bilateral-Filter ist ein adaptiver Filter, der das Rauschen reduziert und die Kanten schärft. Ein weiteres wichtiges Konzept in der Bildverarbeitung ist die Kantenerkennung, die verwendet wird, um Kanten oder Konturen in einem Bild zu identifizieren. OpenCV bietet verschiedene Kantenerkennungsfunktionen wie Sobel-, Canny- und Laplace-Operatoren. Der Sobel-Operator ist ein einfacher Gradientenoperator, der zum Erkennen von Kanten in einer bestimmten Richtung verwendet wird. Der canny-Operator ist ein komplexerer Operator, der Kanten mit höherer Genauigkeit erkennt. Der Laplace-Operator wird eingesetzt, um die zweite Ableitung des Bildes zu berechnen und somit Kanten zu finden. Bei der Bildsegmentierung wird ein Bild in verschiedene Bereiche oder Segmente unterteilt, um Objekte oder Bereiche von Interesse zu identifizieren. OpenCV bietet verschiedene Bildsegmentierungsfunktionen wie die Schwellwertsegmentierung, die Watershed-Segmentierung und die Konturfindung. Die Schwellwertsegmentierung ist eine der einfachsten Bildsegmentierungstechniken, die auf einem festgelegten Schwellenwert basiert, der ein Bild in Vordergrund- und Hintergrundpixel unterteilt. Die Watershed-Segmentierung basiert auf dem Konzept topografischer Kartenmodelle zum Segmentieren von Bildern. Die Konturfindung wird verwendet, um Objektkonturen in Bildern zu identifizieren. Schließlich ist die Merkmalsextraktion ein wichtiger Schritt in der Bildverarbeitung, um einzigartige Merkmale in einem Bild zu identifizieren. Mit OpenCV können verschiedene Merkmalsex-

traktionsfunktionen wie SIFT, SURF und ORB verwendet werden. SIFT (Scale-Invariant Feature Transform) ist ein Algorithmus, der einzigartige Merkmale in einem Bild extrahiert und sie robust gegenüber Skalierung, Rotation und Helligkeitsänderungen macht. SURF (Speeded Up Robust Features) ist eine schnellere Variante von SIFT, die auch robust gegenüber Skalierung, Rotation und Helligkeitsänderungen ist. ORB (Oriented FAST and Rotated BRIEF) ist eine weitere schnelle Variante von SIFT, die die FAST- und BRIEF-Algorithmen kombiniert. [22] [6]

4.2.3 OpenCV in der Pose Estimation

OpenCV spielt eine wichtige Rolle bei der Pose Estimation von MediaPipe. OpenCV dient zum Laden, Verarbeiten und Anzeigen von Bildern und Videos. Insbesondere in MediaPipe wird OpenCV verwendet, um Eingabe- und Ausgabeströme zu verwalten und zu manipulieren sowie erfasste Bilder aufzubereiten und sie an das Pose Estimation Modul weiterzuleiten. Darüber hinaus bietet OpenCV verschiedene Bildverarbeitungs- und geometrische Transformationsfunktionen, die für die Posenschätzung erforderlich sind. Beispielsweise kann OpenCV zum Skalieren, Drehen, Zuschneiden oder Transformieren von Bildern verwendet werden, um die Bildqualität zu verbessern oder die Genauigkeit der Posenschätzung zu erhöhen. In diesem konkreten Anwendungsfall wird OpenCV verwendet werden, um eine Videodatei zu laden, die den Benutzer zeigt, und MediaPipe, um die Pose des Benutzers im Video zu schätzen. Die geschätzte Pose könnte dann in Echtzeit angezeigt oder in eine Datei gespeichert werden, die später zur Analyse verwendet werden kann. [22] [6]

4.2.4 Vorteile von OpenCV

OpenCV hat gegenüber anderen Tools eine Fülle von Vorteilen, die es zu einem der wichtigsten Tools für Bildverarbeitung und Computer Vision machen. Eine der größten Stärken von OpenCV ist die breite Unterstützung für Betriebssysteme und Programmiersprachen. Im Vergleich zu anderen Tools unterstützt OpenCV Windows, Linux, Mac OS X und Android. Darüber hinaus kann es in C++, Python und Java programmiert werden, sodass Entwickler ihre bevorzugte Sprache auswählen und das Framework auf verschiedenen Betriebssystemen verwenden können. Ein weiterer Vorteil von OpenCV ist die einfache Integration mit anderen Bibliotheken und Tools. OpenCV bietet APIs für die Integration mit TensorFlow, PyTorch und anderen Deep-Learning-Frameworks, die

es Entwicklern ermöglichen, Computer-Vision-Anwendungen nahtlos in Deep-Learning-Modelle zu integrieren und beide zu nutzen. OpenCV bietet auch eine breite Palette von Bildverarbeitungsfunktionen, einschließlich Bildfilterung, Morphologie, geometrische Transformationen, Schwellenwertverarbeitung und Objekterkennung. Diese Fähigkeiten sind entscheidend für die Entwicklung von Computer-Vision-Anwendungen und können zur Verbesserung der Bildqualität und Bildgenauigkeit beitragen. Ein weiterer Vorteil von OpenCV ist seine Geschwindigkeit und Effizienz. OpenCV ist eine optimierte Bibliothek, die für die Echtzeit-Bildverarbeitung entwickelt wurde. Die Bibliothek nutzt die Hardwarebeschleunigung von GPUs und FPGAs, um die Verarbeitungsleistung zu maximieren. Dadurch können Entwickler schnellere und effizientere Computer-Vision-Anwendungen erstellen. OpenCV ist auch sehr flexibel und unterstützt eine breite Palette von Anwendungen. Damit können Anwendungen wie Gesichtserkennung, Posenschätzung, Objekterkennung und -verfolgung, Augmented Reality und sogar autonomes Fahren entwickelt werden. Ein weiterer Vorteil von OpenCV ist die große und engagierte Community von Entwicklern. Da OpenCV Open Source ist, hat es eine aktive Community von Entwicklern, die ständig neue Funktionen und Verbesserungen hinzufügen. Die Community bietet auch eine Fülle von Tutorials, Beispielen und Support, die es Entwicklern erleichtern, das Framework zu erlernen und zu verwenden. All diese Vorteile machen OpenCV zur ersten Wahl für Entwickler, die fortschrittliche Computer-Vision-Anwendungen erstellen möchten. Die Flexibilität, Geschwindigkeit und Leistung von OpenCV ermöglichen es Entwicklern, schnellere und genauere Computer-Vision-Anwendungen zu erstellen, die in einer Vielzahl von Branchen eingesetzt werden können. [22] [6]

4.2.5 Installation von OpenCV

Um einen Kamerastream mit OpenCV in Python zu erfassen und anzuzeigen, besteht der erste Schritt darin, die OpenCV-Bibliothek in das Projekt zu importieren. Dies kann durch Hinzufügen der Bibliothek als Abhängigkeit zum Projekt oder durch Herunterladen der Bibliothek und manuelles Hinzufügen zum Projekt erfolgen. Nachdem die Bibliothek hinzugefügt wurde, kann der Kamerastream abgerufen werden, indem eine Instanz des VideoCapture-Objekts erstellt und die gewünschte Kameraquelle angegeben wird. Der Kamerastream kann dann abgerufen und in einem Fenster angezeigt werden. Um den Kamerastream anzuzeigen, kann dann eine OpenCV-Funktion verwenden werden, die Bilder in einer Schleife aufnimmt und sie in einer Endlosschleife anzeigt, bis die Schleife endet. Das Betrachten eines Videostreams in einem Fenster bietet viele Möglichkeiten

zum Verarbeiten und Analysieren von Einzelbildern. Beispielsweise können Filter auf die Frames angewendet werden, um die Bildqualität zu verbessern oder bestimmte Merkmale hervorzuheben. Auch können verschiedene Algorithmen genutzt werden, um bestimmte Objekte in den Frames zu erkennen und zu verfolgen. [22] [6]

4.3 Architektur

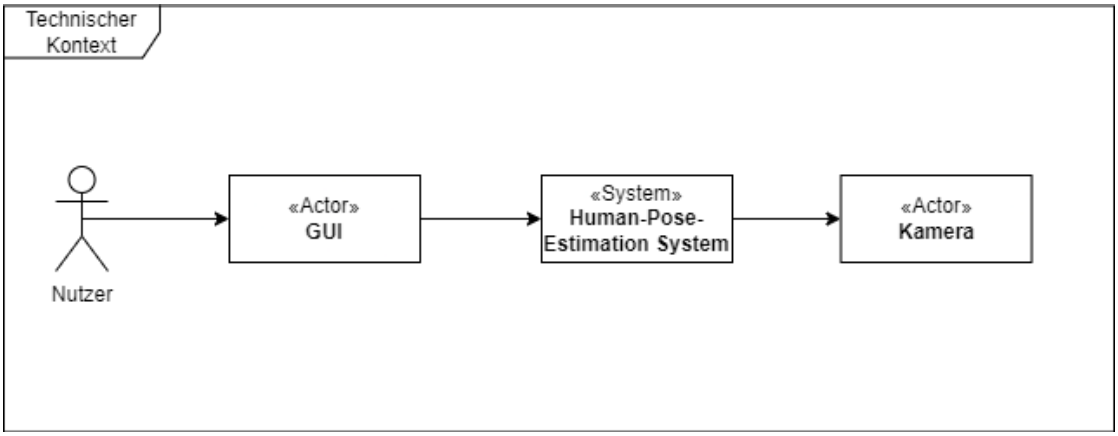

Abbildung 4.1: Technischer Kontext

Das technische Konzept Diagramm zeigt, dass bei diesem System der Nutzer mit einer grafischen Benutzeroberfläche interagiert, um das Human-Pose-Estimation System zu steuern. Das Human-Pose-Estimation System kann dann die Pose Estimation für die angeschlossenen Kameras ausführen. Mithilfe der grafischen Benutzeroberfläche können Benutzer daher die Pose Estimation für ihre Kamera problemlos und ohne Vorkenntnisse verwenden.

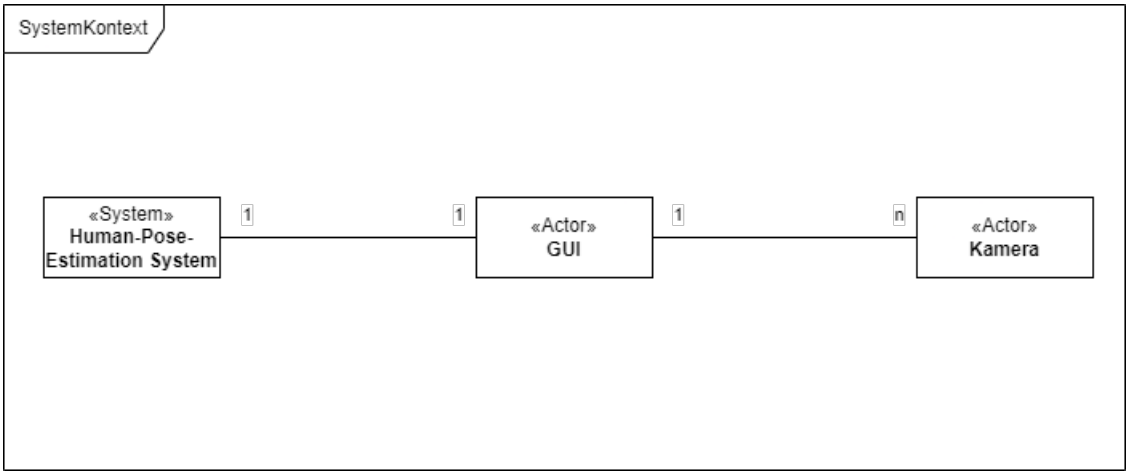

Abbildung 4.2: Systemkontextdiagramm

In diesem Systemkontextdiagramm ist auf einen Blick erkennbar, dass das System nur aus drei Komponenten besteht. Die erste Komponente, das Human-Pose-Estimation System, verarbeitet Bilder der ausgewählten Kamera und stellt dem Benutzer eine Liste der verfügbaren verbundenen Kameras zur Verfügung. Außerdem werden verarbeitete Livebilder der Kamera erstellt. Die zweite Komponente ist die grafische Benutzeroberfläche, die es dem Benutzer ermöglicht, einfach zu interagieren und das verarbeitete Livebild der Kamera anzuzeigen. Die letzte Komponente ist die angeschlossene Kamera, von der es eine oder mehrere geben kann.

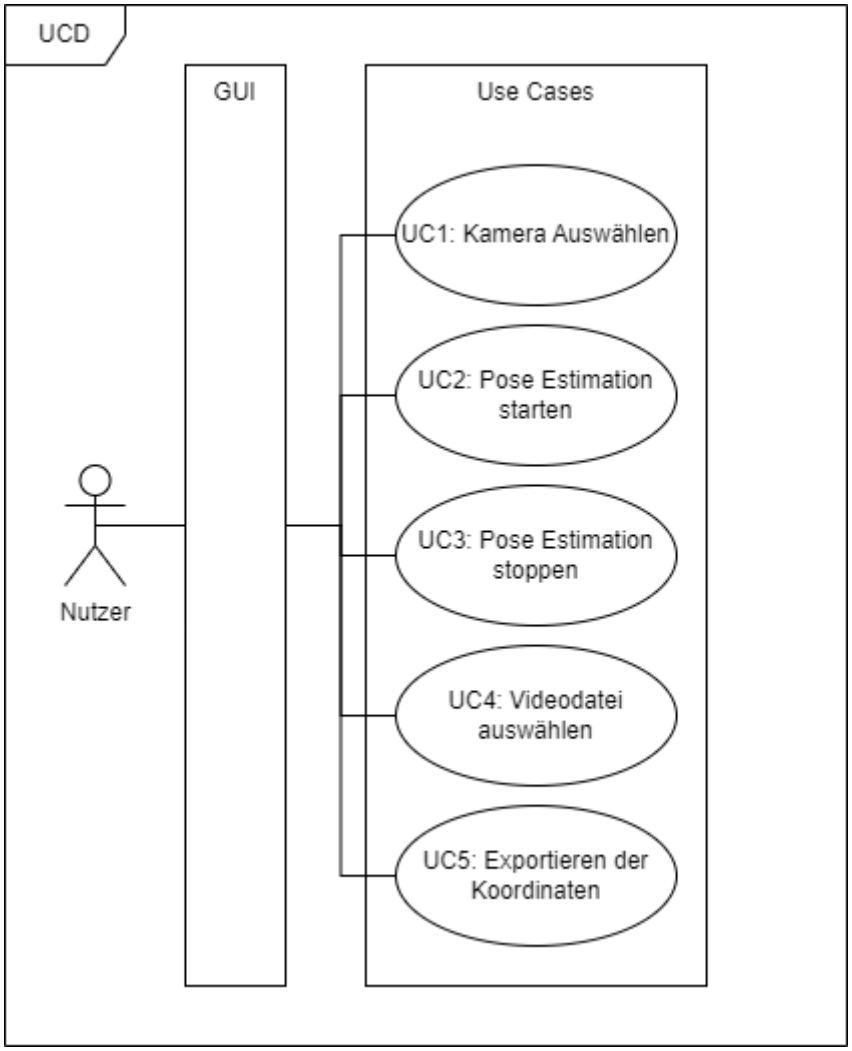

Abbildung 4.3: Use Case Diagramm

In diesem Use Case Diagramm kann man sehen, dass das System über vier Anwendungsfälle verfügt. Der erste Use Case besteht darin, eine Kamera auszuwählen, um die Human Pose Estimation zu starten, was über ein Dropdown-Menü auf der grafischen Benutzeroberfläche erfolgt. Der zweite Use Case besteht darin, die Pose Estimation zu starten, was zur Verarbeitung des Livebilds der ausgewählten Kamera und zur anschließenden Anzeige des verarbeiteten Bilds und des aufgezeichneten Skeletts in Echtzeit führt. Dies geschieht durch einen einfachen Knopfdruck auf der GUI. Der Dritte Use Case besteht darin, die Pose Estimation für eine bestimmte Kamera zu stoppen. Dies geschieht durch das Schließen des Fensters, in dem das Livebild der verarbeiteten Kamera angezeigt wird.

Dann gibt es noch den vierten Use Case, das Auswählen einer Videodatei, dies ist dafür da, um die Pose Estimation auf ein schon voraufgezeichnetes Video Stattfinden zu lassen, um eine reproduzierbares und vergleichbares Ergebnis zu schaffen. Dies kann auch über einen Button über die GUI erfolgen, der dann den Datei-Explorer des Systems öffnet, wo dann die Datei ausgewählt werden kann. Der letzte Use Case ist das Exportieren der Koordinaten. Hier wird dann eine XML-Datei erstellt, in welcher man dann im Anschluss nachschauen kann, wo welches Gelenk in einem bestimmten Frame relativ zur Kamera war, dazu steht dann auch noch die Information wie wahrscheinlich dieses Gelenk von der Kamera zu sehen und nicht verdeckt war. Diese Funktion ist verfügbar über ein Kontrollkästchen auf der grafischen Benutzeroberfläche zum Auswählen.

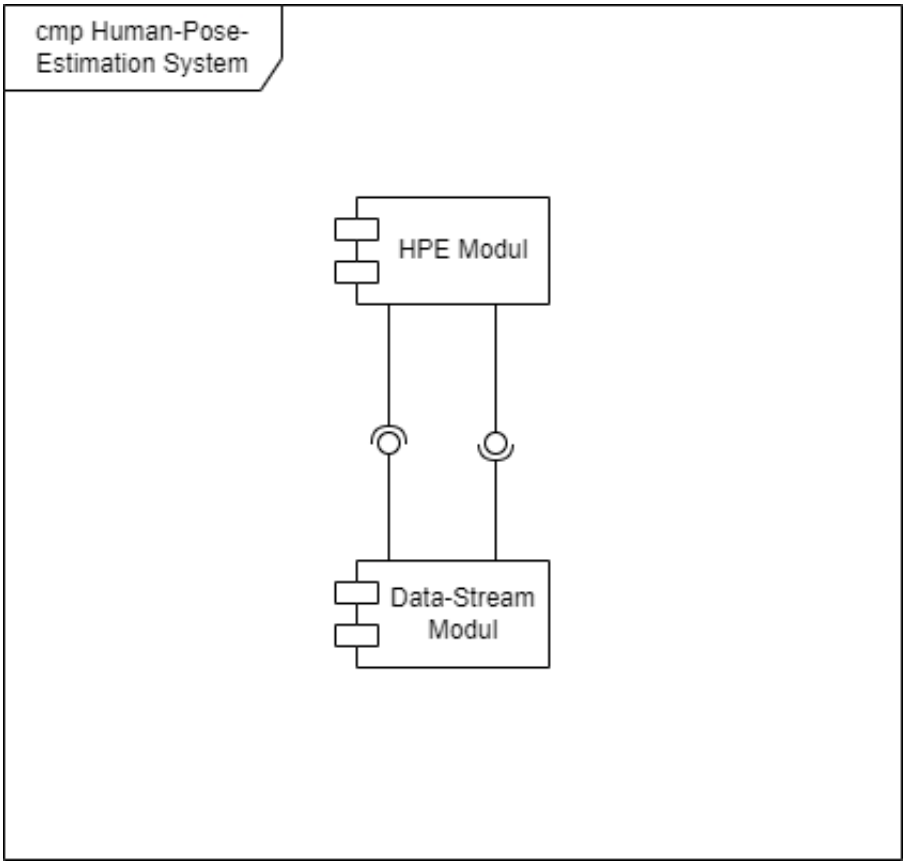

Abbildung 4.4: Komponentendiagramm

In diesem Komponentendiagramm werden die Komponenten des Human-Pose-Estimation Systems aufgezeichnet. Wie hier zu sehen ist, besteht das System aus zwei Komponenten, dem HPE Modul und dem Data-Stream Modul. Das HPE-Modul ist für die eigentliche

Berechnung der Posen verantwortlich und stellt somit das Herzstück des Projekts dar, in dem die gesamte Logik stattfindet. Damit dieses Modul aber funktioniert, braucht es den Stream einer Kamera, wozu das Data-Stream Modul dient. Das Data-Stream Modul nimmt Daten von der Kamera auf und leitet sie zur anschließenden Pose Estimation an das HPE-Modul weiter. Darüber hinaus überprüft das Data-Stream Modul auch die tatsächlich verfügbaren Kameras im System, um die Liste der Kameras in der GUI anzuzeigen.

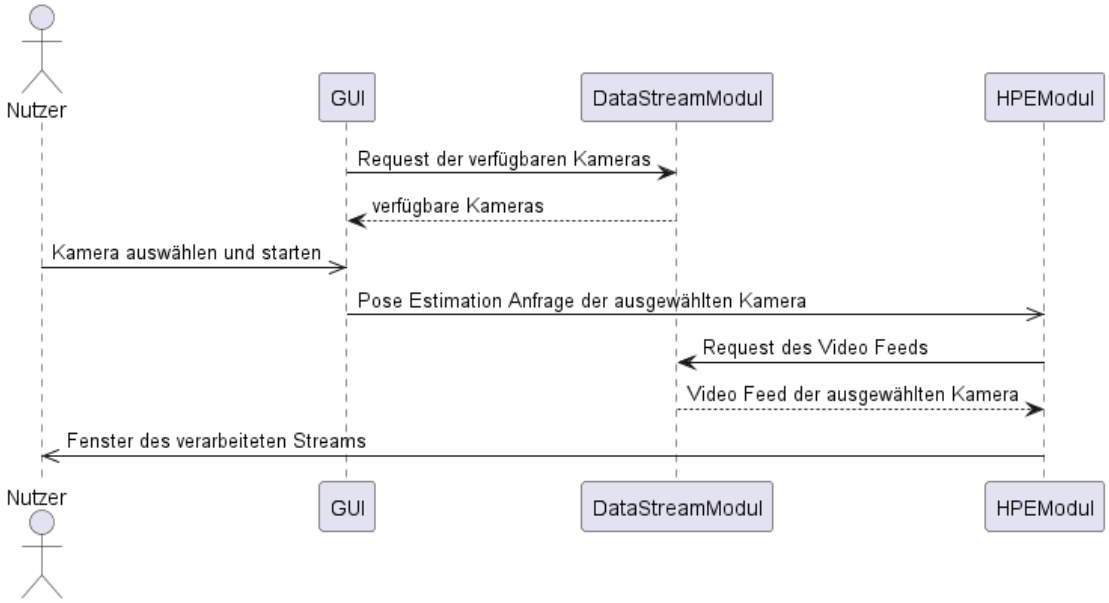

Abbildung 4.5: Sequenzdiagramm live Kamera

Im Sequenzdiagramm wird beschrieben, wie der Ablauf ist, wenn ein Nutzer die Pose Estimation für eine live Kamera starten möchte. Der Ablauf beginnt damit, dass die grafische Benutzeroberfläche die verfügbaren Kameras vom Data-Stream Modul anfordert. Das Data-Stream Modul verarbeitet die Anfrage und sendet eine Antwort zurück an die GUI, in der die verfügbaren Kameras aufgelistet sind. Der Nutzer wählt nun eine Kamera aus den verfügbaren Optionen aus und startet sie. Dazu sendet der Nutzer über die GUI eine Anfrage an das HPE Modul für die ausgewählte Kamera. Das HPE Modul benötigt jedoch den Video-Feed der ausgewählten Kamera, um die Pose Estimation durchzuführen. Daher sendet das HPE Modul eine Anfrage an das Data-Stream Modul, um den Video-Feed zu erhalten. Das Data-Stream Modul empfängt die Anfrage und liefert den Video-Feed der ausgewählten Kamera an das HPE Modul. Das HPE Modul verarbeitet

nun den empfangenen Video-Feed und generiert einen verarbeiteten Stream, der die Ergebnisse der Pose Estimation enthält. Anschließend startet das HPE Modul ein Fenster mit dem verarbeiteten Stream. Der Nutzer erhält ein Fenster, in dem der verarbeitete Stream angezeigt wird. In diesem Fenster kann der Nutzer die Ergebnisse der Pose Estimation betrachten. Auf diese Weise erfolgt die Kommunikation zwischen dem Nutzer, der GUI, dem Data-Stream Modul und dem HPE Modul, um die verfügbaren Kameras abzurufen, eine Kamera auszuwählen, den Video-Feed zu erhalten und schließlich den verarbeiteten Stream mit den Pose Estimation-Ergebnissen anzuzeigen.

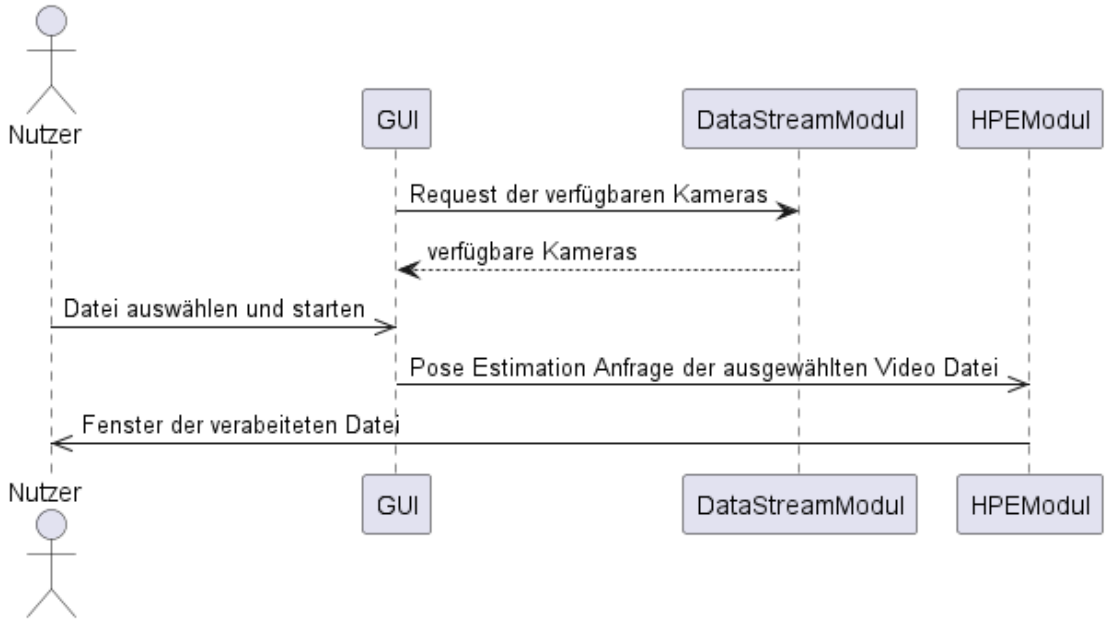

Abbildung 4.6: Sequenzdiagramm Video Datei

In diesem Sequenzdiagramm wird der Ablauf beschrieben, wenn der Nutzer anstatt einer Kamera eine Videodatei nutzen möchte. Der Ablauf beginnt damit, dass die grafische Benutzeroberfläche die verfügbaren Kameras vom Data-Stream Modul anfordert. Das Data-Stream Modul verarbeitet die Anfrage und sendet eine Antwort zurück an die GUI, in der die verfügbaren Kameras aufgelistet sind. Als Nächstes wählt der Nutzer eine Video-Datei aus und startet sie. Dazu sendet der Nutzer über die GUI eine Anfrage an das HPE Modul für die ausgewählte Video-Datei. Das HPE Modul verarbeitet die Anfrage und generiert einen verarbeiteten Stream basierend auf der ausgewählten Video-Datei. Anschließend startet das HPE Modul ein Fenster mit der verarbeiteten Datei.

Der Nutzer erhält ein Fenster, in dem die verarbeitete Datei angezeigt wird. In diesem Fenster kann der Nutzer das Ergebnis der Pose Estimation betrachten.

4.4 Code Erklärung

4.4.1 Interface Klassen

```
class IDataStreamModule(ABC):

    # Method to find available cameras and return their IDs
    @abstractmethod
    def findCameras(self):
        pass

    # Method to get the camera stream from a given camera
    @abstractmethod
    def get_camera_stream(self, camera_name):
        pass
```

Die Klasse „IDataStreamModule“ definiert ein abstraktes Basisklassen-Interface. Es enthält zwei abstrakte Methoden, die von den abgeleiteten Klassen implementiert werden müssen. Die erste abstrakte Methode „findCameras“ hat den Auftrag, verfügbare Kameras zu finden und deren IDs zurückzugeben. Der genaue Mechanismus zum Auffinden der Kameras wird nicht in dieser abstrakten Methode definiert, sondern in den abgeleiteten Klassen implementiert. Diese Methode gibt eine Liste der gefundenen Kamera-IDs zurück. Die zweite abstrakte Methode „get_camera_stream“ hat den Zweck, den Kamerastrom einer bestimmten Kamera zu erhalten. Der Parameter „camera_name“ wird verwendet, um die Kamera zu identifizieren, von der der Strom erhalten werden soll. Wie genau der Kamerastrom abgerufen wird, wird ebenfalls in den abgeleiteten Klassen implementiert. Diese Methode liefert den Kamerastrom zurück, der in der Regel als Sequenz von Frames repräsentiert wird. Das Interface „IDataStreamModule“ dient als Rahmen und stellt sicher, dass alle abgeleiteten Klassen die erforderlichen Methoden bereitstellen. Durch die Verwendung dieses Interfaces können verschiedene Implementierungen von DataStreamModules nahtlos ausgetauscht werden, solange sie das Interface einhalten. Dadurch wird die Flexibilität und Wartbarkeit des Codes verbessert.

```python
class IHPEModule(ABC):

    # This method starts HPE using a camera specified by its name
    @abstractmethod
    def startHPEwithCamera(self, camera_name, export_landmarks):
        pass
```

Die Klasse „IHPEModule“ definiert ein abstraktes Basisklassen-Interface für das HPE-Module. Es enthält eine abstrakte Methode, die von den abgeleiteten Klassen implementiert werden muss. Die abstrakte Methode „startHPEwithCamera“ hat den Zweck, die Haltungsschätzung mit einer Kamera zu starten, die durch ihren Namen spezifiziert wird. Es werden zwei Parameter übergeben: „camera_name“ und „export_landmarks“. Der Parameter „camera_name“ identifiziert die Kamera, mit der die Haltungsschätzung durchgeführt werden soll. Der Parameter „export_landmarks“ gibt an, ob die erkannten Landmarken exportiert werden sollen. Durch die Verwendung des Interfaces „IHPEModule“ können verschiedene Implementierungen von HPE-Modulen erstellt werden, solange sie das Interface einhalten. Dadurch wird die Flexibilität und Erweiterbarkeit des Codes verbessert und die Möglichkeit geschaffen, verschiedene HPE-Module auszutauschen und zu erweitern, ohne den gesamten Code ändern zu müssen.

4.4.2 DataStreamModule Klasse

```python
class DataStreamModule(IDataStreamModule):

    # Method to find available cameras and return their IDs
    def findCameras(self):
        # List to store the IDs of the found cameras
        cameras = []
        # Iterating through all possible camera IDs
        i = 0
        while True:
            cap = cv2.VideoCapture(i)
            if not cap.isOpened():
                # No further cameras found, break the loop
                break
            # Camera found, store ID in the list
            cameras.append(i)
```

```
16              cap.release()
17              i += 1
18          # Return the IDs of the found cameras
19          return cameras
20
21      # Method to get the camera stream from a given camera
22      def get_camera_stream(self, camera_name):
23          cap = cv2.VideoCapture(camera_name)
24          while cap.isOpened():
25              ret, frame = cap.read()
26              # Yield each frame as a numpy array
27              yield np.array(frame)
28          cap.release()
```

Die Klasse enthält zwei Methoden: „findCameras()“ und „get_camera_stream(camera_name)“. Die Methode „findCameras()“ sucht nach verfügbaren Kameras und gibt die IDs dieser Kameras zurück. Dazu wird eine leere Liste namens „cameras“ initialisiert. Dann wird eine Schleife gestartet, in der verschiedene Kamera-IDs ausprobiert werden. Für jede Kamera-ID wird versucht, eine Verbindung zur Kamera herzustellen, indem die Methode „cv2.VideoCapture(i)“ aufgerufen wird, wobei „i“ die aktuelle Kamera-ID ist. Wenn die Verbindung nicht erfolgreich ist (d.h. „cap.isOpened()“ ist False), wird die Schleife abgebrochen, da keine weiteren Kameras gefunden wurden. Wenn die Kamera erfolgreich geöffnet wird, wird die ID zur Liste „cameras“ hinzugefügt und die Verbindung zur Kamera wird freigegeben, indem „cap.release()“ aufgerufen wird. Dann wird die Kamera-ID inkrementiert und der Schleifenprozess beginnt von vorne. Am Ende werden die IDs der gefundenen Kameras zurückgegeben. Die Methode „get_camera_stream(camera_name)“ erhält den Namen der Kamera als Eingabeparameter. Zunächst wird eine Verbindung zur Kamera hergestellt, indem „cv2.VideoCapture(camera_name)“ aufgerufen wird. Die Variable „cap“ repräsentiert das Video-Capture-Objekt, das den Kamerastream repräsentiert. Anschließend wird eine Schleife gestartet, die so lange läuft, wie die Kamera geöffnet ist. In jeder Iteration wird versucht, einen Frame von der Kamera zu lesen, indem „cap.read()“ aufgerufen wird. Der Rückgabewert „ret“ gibt an, ob das Lesen erfolgreich war, und der Frame wird in der Variable „frame“ gespeichert. Der interessante Teil ist, dass jeder Frame als ein NumPy-Array zurückgegeben wird, indem „yield np.array(frame)“ verwendet wird. Das „yield“-Schlüsselwort ermöglicht es, den Frame als Generator zu liefern, der von außen aufgerufen werden kann, um auf jeden Frame zuzugreifen. Dadurch können andere Teile des Codes den Kamerastream verwenden und

die Frames weiterverarbeiten, ohne den gesamten Stream im Voraus laden zu müssen. Am Ende wird die Verbindung zur Kamera freigegeben, indem „cap.release()“ aufgerufen wird, um die Ressourcen freizugeben und die Verbindung zu schließen.

4.4.3 HPEModule Klasse

```
class HPEModule(IHPEModule):

    # This method starts HPE using a camera specified by its name
    def startHPEwithCamera(self, camera_name, export_landmarks):
        out = None
        # check if the camera_name is a file path or not
        if os.path.isfile(camera_name):
            # open the video file using cv2.VideoCapture
            cap = cv2.VideoCapture(camera_name)
            # set the output video file path
            output_path = os.path.splitext(camera_name)[0] + "_output
                .mp4"
            # get the frame rate and size of the input video
            fps = cap.get(cv2.CAP_PROP_FPS)
            width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
            height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))

            # initialize video writer to save the output video
            fourcc = cv2.VideoWriter_fourcc(*'mp4v')
            out = cv2.VideoWriter(output_path, fourcc, fps, (width,
                height))
        else:
            # initialize data stream and camera object
            data_stream = DataStreamModule()
            cap = data_stream.get_camera_stream(camera_name)

        # set the window name using the camera name
        window_name = f"Pose Estimation on Camera {camera_name}"
        print(export_landmarks.get())
        if export_landmarks.get() is True:
            # create the root element
            root = ET.Element("Landmarks")

            # loop through each landmark
```

```
33              for i in range(33):
34                  ET.SubElement(root, f'landmark{i}')
35
36              framecounter = 0
37
38
39          # start pose detection
40          with mp_pose.Pose(min_detection_confidence=0.5,
                min_tracking_confidence=0.5) as pose:
41
42              while True:
43                  # get the next frame from the camera
44                  if out is not None:
45                      ret, frame = cap.read()
46                      if not ret:
47                          break
48                  else:
49                      frame = next(cap)
50
51                  # Recolor image to RGB
52                  image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
53                  image.flags.writeable = False
54
55                  # Make detection
56                  results = pose.process(image)
57
58                  # Recolor back to BGR
59                  image.flags.writeable = True
60                  image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)
61
62                  # Extract landmarks
63                  try:
64                      landmarks = results.pose_landmarks.landmark
65                      if export_landmarks.get() is True:
66                          for j, landmark in enumerate(landmarks):
67                              frame_element = ET.SubElement(root.find(f
                                    'landmark{j}'), f'frame{framecounter}'
                                    )
68                              frame_element.set("X", str(landmark.x))
69                              frame_element.set("Y", str(landmark.y))
```

```
70                              frame_element.set("Z", str(landmark.z))
71                              frame_element.set("visibility", str(
                                    landmark.visibility))
72                          framecounter += 1
73                      # print(landmarks)
74                  except:
75                      pass
76
77                  # Render detections
78                  mp_drawing.draw_landmarks(image, results.
                        pose_landmarks, mp_pose.POSE_CONNECTIONS,
79                                            mp_drawing.DrawingSpec(
                                                  color=(245, 117, 66),
                                                  thickness=2,
                                                  circle_radius=2),
80                                            mp_drawing.DrawingSpec(
                                                  color=(245, 66, 230),
                                                  thickness=2,
                                                  circle_radius=2)
81                                            )
82
83                  cv2.imshow(window_name, image)
84
85                  if out is not None:
86                      out.write(image)
87
88                  keyCode = cv2.waitKey(1)
89                  if cv2.getWindowProperty(window_name, cv2.
                        WND_PROP_VISIBLE) < 1:
90                      break
91
92          if export_landmarks.get() is True:
93              xml_string = ET.tostring(root, encoding="UTF-8")
94              parsed_xml = minidom.parseString(xml_string)
95              pretty_xml_string = parsed_xml.toprettyxml(indent="    ")
96              with open(f"{camera_name}_output.xml", "w") as xml_file:
97                  xml_file.write(pretty_xml_string)
```

Die Klasse „HPEModule“ erbt von der Klasse „IHPEModule“. Diese Klasse enthält die Methode „startHPEwithCamera“, die die Pose Estimation mit einer Kamera startet und

optional die Landmarks exportiert. Zu Beginn wird eine Variable „out“initialisiert, die später für die Ausgabedatei verwendet wird, falls es sich um eine Datei handelt. Der Code überprüft, ob der „camera_name“ ein Dateipfad ist. Wenn ja, wird angenommen, dass es sich um eine Videodatei handelt. Das Video wird mit „cv2.VideoCapture“ geöffnet und Informationen wie Framerate und die Größe des Eingabevideos werden abgerufen. Wenn „camera_name“ kein Dateipfad ist, wird angenommen, dass es sich um den Namen einer Kamera handelt. Es wird eine Instanz der Klasse „DataStreamModule“ erstellt und der Kamerastream wird über die Methode „get_camera_stream(camera_name)“ abgerufen. Danach wird der Fenstername für die Anzeige festgelegt, der auf dem Namen der Kamera basiert. Der Code überprüft, ob die Option „export_landmarks“ aktiviert ist. Wenn ja, wird ein XML-Element „root“ erstellt, das als Container für die Landmarks dient. Es werden 33 „landmark“-Unterelemente erstellt. Der Code betritt als Nächstes eine endlose Schleife. In jeder Iteration wird der nächste Frame von der Kamera gelesen. Wenn „out“ nicht None ist, handelt es sich um eine Videodatei, und „ret“ und „frame“ werden über „cap.read()“ abgerufen. Andernfalls wird „frame“ durch den Aufruf von „next(cap)“ erhalten, da „cap“ ein Generator ist, der den Kamerastream repräsentiert. Das Bild wird von BGR zu RGB umgewandelt, da dies das erwartete Format für die Pose Estimation von MediaPipe ist. Anschließend wird die Pose Estimation auf dem Bild mithilfe von MediaPipe durchgeführt und die Ergebnisse werden in der Variable „results“ gespeichert. Die Landmarks werden extrahiert und in das XML-Element „root“ eingefügt, sofern die Option „export_landmarks“ aktiviert ist. Dabei werden die Koordinaten der Landmarks sowie deren Sichtbarkeit in jedem Frame gespeichert. Die erkannten Landmarks werden auf dem Bild visualisiert, indem „mp_drawing.draw_landmarks“ verwendet wird. Das Bild wird in einem Fenster mit dem Namen „window_name“ angezeigt. Wenn „out“ nicht None ist, handelt es sich um eine Videodatei, und der aktuelle Frame wird in die Ausgabedatei geschrieben. Die Schleife wird fortgesetzt, bis das Anzeigefenster geschlossen wird oder die Videodatei fertig eingelesen wurde. Danach wird überprüft, ob die Option „export_landmarks“ aktiviert ist. Wenn ja, wird das XML-Element „root“ in eine XML-Datei geschrieben, indem es zuerst in einen XML-String umgewandelt, dann analysiert und schließlich in ein lesbares Format konvertiert wird. Das Konvertieren und Schreiben in die Datei wird erst zum Ende gemacht, da sonst das ständige schreiben zu massiven Performanceeinbrüchen führen würde.

4.4.4 PoseEstimation GUI Klasse

```
1 class PoseEstimationGUI:
```

```
2
3       def __init__(self):
4           self.mp4_file_path = None
5           self.root = tk.Tk()
6           self.root.title("Pose Estimation GUI")
7           self.root.geometry("400x400")
8           self.root.config(bg="#F9F9F9")
9
10          self.camera_options = []  # Initialize camera options list
11
12          # Create a drop-down menu with available cameras
13          self.camera_options = DataStreamModule().findCameras()
14          self.selected_camera = tk.StringVar()
15          self.selected_camera.set(self.camera_options[0])  # default
                value
16          camera_label = tk.Label(self.root, text="Select a camera:",
                bg="#F9F9F9", font=("Arial", 12, "bold"))
17          camera_label.pack(pady=(20, 0))
18          camera_menu = tk.OptionMenu(self.root, self.selected_camera,
                *self.camera_options)
19          camera_menu.config(font=("Arial", 12))
20          camera_menu.pack()
21
22          # Create a button to select an mp4 file
23          self.select_file_button = tk.Button(self.root, text="Select
                MP4 File to Start HPE on", bg="#2196F3", fg="white",
24                                              font=("Arial", 12, "bold
                                                    "), command=self.
                                                    select_file)
25          self.select_file_button.pack(pady=(10, 20))
26
27          # Create label widget to indicate file selection status
28          self.file_selected_label = Label(self.root, text="No file
                selected", fg="red", bg="#F9F9F9",
29                                           font=("Arial", 12))
30          self.file_selected_label.pack(pady=(10, 0))
31
32          # Create a button to start pose estimation
33          self.start_button = tk.Button(self.root, text="Start Pose
                Estimation", bg="#4CAF50", fg="white",
```

```
34                                      font=("Arial", 12, "bold"),
                                            command=self.
                                            start_pose_estimation)
35          self.start_button.pack(pady=(20, 10))
36
37          # Create a checkbox to enable/disable Export of Landmarks
38          self.export_Landmarks = tk.BooleanVar()
39          self.export_Landmarks.set(False)
40          export_Landmarks_checkbox = tk.Checkbutton(self.root, text="
                Export Landmarks", variable=self.export_Landmarks,
41                                                  bg="#F9F9F9",
42                                                  font=("Arial", 12, "bold
                                                        "), onvalue=True,
                                                        offvalue=False)
43          export_Landmarks_checkbox.pack()
44
45          self.root.mainloop()
46
47      def start_pose_estimation(self):
48          if self.mp4_file_path is not None:
49              # Start pose estimation on the selected mp4 file
50              pose_estimator = HPEModule()
51              pose_thread_file = threading.Thread(target=pose_estimator
                    .startHPEwithCamera, args=(self.mp4_file_path, self.
                    export_Landmarks,))
52              self.mp4_file_path = None
53              self.file_selected_label.config(text="No File selected",
                    fg="red")
54              pose_thread_file.start()
55          else:
56              # Start pose estimation on the selected camera
57              # print(self.selected_camera.get())
58              pose_estimator = HPEModule()
59              camera = int(self.selected_camera.get())
60              pose_thread_camera = threading.Thread(target=
                    pose_estimator.startHPEwithCamera, args=(camera, self.
                    export_Landmarks,))
61              pose_thread_camera.start()
62
63      def select_file(self):
```

```
64          self.mp4_file_path = filedialog.askopenfilename(initialdir="/
                ", title="Select MP4 File to Start HPE on",
65                                                          filetypes=(("
                                                                mp4 files"
                                                                , "*.mp4")
                                                                , ("all
                                                                files", "
                                                                *.*")))
66          # Update file selection status label
67          if self.mp4_file_path:
68              self.file_selected_label.config(text="File selected", fg=
                    "green")
69          else:
70              self.file_selected_label.config(text="No File selected",
                    fg="red")
71              self.mp4_file_path = None
```

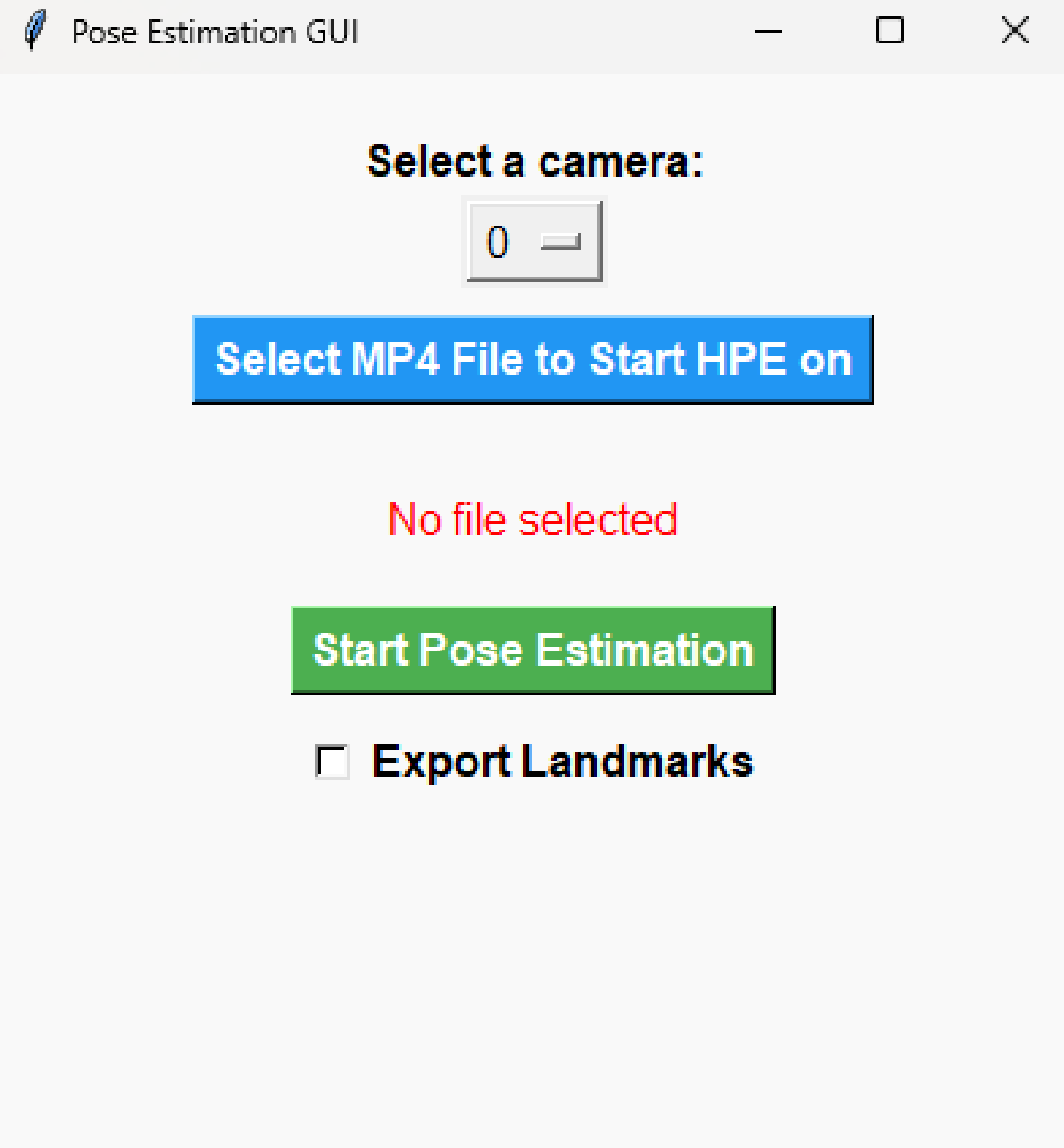

Abbildung 4.7: GUI

Die Klasse „PoseEstimationGUI“ stellt eine grafische Benutzeroberfläche (GUI) für die Pose Estimation bereit. Die GUI wird mithilfe des Tkinter-Moduls erstellt. Im Konstruktor „init“ werden die GUI-Elemente initialisiert. Zunächst wird das Hauptfenster erstellt und konfiguriert, einschließlich des Titels, der Größe und der Hintergrundfarbe. Anschließend wird eine Dropdown-Liste erstellt, um die verfügbaren Kameras anzuzeigen. Hierbei

wird die Methode „findCameras“ aus der Klasse „DataStreamModule“ aufgerufen, um die IDs der verfügbaren Kameras zu erhalten. Die ausgewählte Kamera wird in einer Variablen gespeichert. Des Weiteren wird ein Button („select_file_button“) erstellt, der es dem Benutzer ermöglicht, eine MP4-Datei auszuwählen. Wenn der Button geklickt wird, wird die Methode „select_file“ aufgerufen, um den Dateiauswahldialog anzuzeigen. Der ausgewählte Dateipfad wird in der Variable „mp4_file_path“ gespeichert. Ein Label („file_selected_label“) wird erstellt, um den Status der Dateiauswahl anzuzeigen. Wenn eine Datei ausgewählt wurde, wird der Text des Labels entsprechend aktualisiert. Ein weiterer Button („start_button“) wird erstellt, um die Pose Estimation zu starten. Wenn der Button geklickt wird, wird die Methode „start_pose_estimation“ aufgerufen. Abhängig davon, ob eine MP4-Datei ausgewählt wurde oder nicht, wird entweder die ausgewählte Kamera oder die ausgewählte MP4-Datei übergeben und die Methode „startHPEwithCamera“ der Klasse „HPEModule“ aufgerufen. Zusätzlich wird ein Kontrollkästchen („export_Landmarks_checkbox“) erstellt, um das Exportieren von Landmarken zu aktivieren oder zu deaktivieren. Der Status des Kontrollkästchens wird in der Variable „export_Landmarks“ gespeichert. Die Methode „start_pose_estimation“ überprüft, ob eine MP4-Datei ausgewählt wurde. Wenn dies der Fall ist, wird die Pose Estimation auf der ausgewählten MP4-Datei gestartet, indem ein separater Thread erstellt wird, der die Methode „startHPEwithCamera“ der Klasse „HPEModule“ aufruft. Danach wird der ausgewählte Dateipfad auf „None“ gesetzt und der Text des Dateiauswahl-Labels entsprechend aktualisiert. Falls keine MP4-Datei ausgewählt wurde, wird die Pose Estimation mit der ausgewählten Kamera gestartet. Dafür wird ein separater Thread erstellt, der die Methode „startHPEwithCamera“ der Klasse „HPEModule“ aufruft und die ausgewählte Kamera übergibt. Die Methode „select_file“ öffnet einen Datei Explorer, um eine MP4-Datei auszuwählen. Der ausgewählte Dateipfad wird in der Variable „mp4_file_path“ gespeichert und der Text des Dateiauswahl-Labels entsprechend aktualisiert.

5 Ergebnisse

In diesem Abschnitt werden die Ergebnisse einer durchgeführten Vergleichsanalyse zwischen dem entwickelten Pose Estimation System, das auf MediaPipe basiert, und dem weit verbreiteten System OpenPose präsentiert. Zunächst wurden Experimente durchgeführt, um qualitatives und quantitatives Feedback zu den beiden Systemen zu erhalten. Die Ergebnisse basieren auf zwei ausgewählten Videos mit unterschiedlichen Posen und unterschiedlichem Schwierigkeitsgrad. Die Daten wurden mithilfe von dem erarbeitetem System und OpenPose verarbeitet, wobei die Ausgabeergebnisse der beiden Systeme verglichen und analysiert wurden. Die Ergebnisse zeigen, dass das Pose Estimation System auf Basis von MediaPipe eine hohe Genauigkeit bei der Erkennung von Körperhaltungen aufweist. Es wurde eine detaillierte Auswertung der Positionen der erkannten Gelenkpunkte durchgeführt, um die Übereinstimmung mit den tatsächlichen Positionen zu überprüfen. Dabei wurde eine durchschnittliche Abweichung festgestellt, was auf eine präzise Schätzung der Pose hinweist. Im Vergleich dazu wurden die Ergebnisse des OpenPose-Systems analysiert. Es zeigte ebenfalls eine gute Leistung bei der Pose Estimation, jedoch wurden in einigen Fällen leichte Abweichungen in den erkannten Gelenkpositionen festgestellt. Eine detaillierte Bewertung dieser Abweichungen wurde durchgeführt, um die spezifischen Stärken und Schwächen von OpenPose im Vergleich zu MediaPipe zu identifizieren. Die präsentierten Ergebnisse werden durch visuelle Darstellungen, wie zum Beispiel Bildern und Vergleichstabellen, unterstützt, um die Unterschiede und Ähnlichkeiten zwischen den beiden Systemen deutlich zu machen. Die Ergebnisse dieser Vergleichsanalyse liefern wertvolle Erkenntnisse für die Weiterentwicklung und Anwendung des Pose Estimation Systems. Darüber hinaus werden Hinweise auf mögliche Verbesserungen und Optimierungen gegeben.

5.0.1 Leistungsvergleich

Der entscheidende Vorteil des Systems gegenüber OpenPose zeigt sich beim Bearbeiten der Beispielvideos. Während OpenPose mit nur 10 FPS (Bildern pro Sekunde) lief, zeigte

das erabeitete System deutlich mehr Effizienz und konnte höhere Bildraten aufrechterhalten. Die niedrige Bildrate von OpenPose kann zu Verzögerungen und Trägheit bei der Echtzeitverarbeitung von Videodaten führen. Dies ist insbesondere bei Anwendungen problematisch, die eine Reaktion in Echtzeit erfordern, beispielsweise in der Spielebranche oder bei interaktiven Anwendungen. Im Gegensatz dazu bietet das System eine verbesserte Leistung und kann höhere Bildraten erreichen, beispielsweise 30 FPS. Dies ermöglicht eine reibungslose Echtzeitverarbeitung der Pose Estimation ohne spürbare Verzögerung oder Beeinträchtigung der Benutzererfahrung. Die verbesserte Systemleistung des Systems macht es zu einer geeigneten Wahl für Anwendungen, die Echtzeitverarbeitung und hohe Bildraten erfordern. Dies ist beispielsweise für Live-Streaming-Plattformen, Virtual-Reality-Anwendungen oder interaktive Installationen wichtig. Die optimierte Systemleistung des Systems trägt dazu bei, die Effizienz und Reaktionsfähigkeit des Pose Estimation Systems zu verbessern. Dies ermöglicht eine reibungslose und leistungsstarke Durchführung der Pose Estimation in Echtzeit ohne großen Einsatz von Systemressourcen. Außerdem kann das System die Pose Estimation auf einem System mit geringer Leistung starten, ohne dass das Problem auftritt, dass das Video mit nur einem FPS läuft.

5.0.2 Frame Vergleich

Yoga Vergleich

Abbildung 5.1: Yoga Pose OpenPose

Abbildung 5.2: Yoga Pose erarbeitetes System

Die beiden Bilder, eines generiert durch OpenPose und das andere durch das erarbeitete System, wurden aus einem Yoga-Beispiel-Video gezogen, um die Leistung und Genauigkeit beider Pose Estimation Systeme zu bewerten. Bei der Untersuchung des Bildes, das mit OpenPose erstellt wurde, fiel auf, dass das System die Körperhaltung des dargestellten Individuums insgesamt korrekt erkannt hat. Alle Gelenkpositionen wurden akkurat geschätzt. Im Vergleich dazu zeigt das Bild, das mit dem erarbeitetem System erstellt wurde, eine genauso gute Präzision bei der Erkennung der Körperhaltung. Die Gelenkpositionen wurden genauso fehlerfrei erfasst und entsprachen den tatsächlichen Positionen des dargestellten Individuums. Sowohl die Arm- als auch die Beinpositionen wurden genau erkannt, was auf eine hohe Leistungsfähigkeit des Systems hinweist. Es ist wichtig anzumerken, dass beide Systeme insgesamt gute Ergebnisse liefern und zuverlässige Pose Estimation bieten.

Abbildung 5.3: Yoga Pose OpenPose

Abbildung 5.4: Yoga Pose des erarbeitetem Systems

Die beiden Bilder wurden aus dem gleichen Yoga-Video wie zuvor entnommen und sind aus einem Frame vom Video, welches für die Pose Estimation etwas schwieriger zu schätzen sein sollte. Beim Betrachten des Yoga-Bildes, das mit OpenPose generiert wurde, konnte man feststellen, dass das System die Körperhaltung des dargestellten Individuums sehr genau erkannt hat. Im Vergleich dazu hat das Yoga-Bild, das mit meinem System erstellt wurde, genauso präzise die Körperhaltung erfasst. Die Gelenkpositionen wurden nahezu fehlerfrei erkannt und stimmten größtenteils mit den tatsächlichen Positionen des dargestellten Individuums überein. Nur wenn man genau auf die Handgelenke schaut, sieht man, dass diese leicht abweichen von der tatsächlichen Position der Gelenke. Die genaueren Handgelenkpositionen fehlen jedoch bei OpenPose komplett, daher wird

diese Ungenauigkeit als nicht so großes Problem betrachtet. Zusammenfassend lässt sich sagen, dass das System gegenüber OpenPose keinen Vorteil in Bezug auf die Genauigkeit und Präzision bei der Erkennung von Yoga-Posen aufweist.

Breakdance Vergleich

Für die Untersuchung der Pose Estimation wurde ein Breakdance-Video als Testmaterial gewählt, um eine Herausforderung für das System zu schaffen. Breakdance zeichnet sich durch schnelle und komplexe Bewegungen aus, die eine genaue und robuste Pose Estimation erfordern. Das Breakdance-Video wurde als Testmaterial ausgewählt, um die Leistung des Pose Estimation Systems unter realistischen Bedingungen zu bewerten. Die schnellen Bewegungen, Drehungen, Sprünge und akrobatischen Elemente beim Breakdance stellen eine große Herausforderung dar, da sie eine genaue Posenerfassung und Posenverfolgung erfordern. Mithilfe anspruchsvoller Testvideos werden die Grenzen der Systeme zur Posenschätzung ausgelotet und ihre Fähigkeit analysiert, komplexe Bewegungen beim Breakdance genau zu erkennen und zu verfolgen. Dies ermöglicht eine objektive Bewertung des Systems und gibt Aufschluss über seine Leistung und sein Potenzial. Die Ergebnisse dieser Untersuchung sind nicht nur für die Entwicklung von Pose Estimation Systemen im Breakdance-Bereich relevant, sondern auch für andere Anwendungen, die schnelle und komplexe Bewegungen erfordern, wie beispielsweise Sportanalysen, Bewegungserkennung oder Rehabilitationstechniken.

Abbildung 5.5: Breakdance Pose OpenPose

Abbildung 5.6: Breakdance Pose erarbeitetes System

Bei der Verwendung von OpenPose zur Analyse von Breakdance-Videos wurde eine Reihe von Problemen festgestellt. Das System hatte Schwierigkeiten, schnelle Bewegungen und akrobatische Elemente genau zu erfassen. Es gab Ungenauigkeiten bei der Aufzeichnung von Gelenkwinkeln und Verbindungslinien zwischen Körperteilen, insbesondere bei schnellen Bewegungen. Dies führte dazu, dass ungenaue Posen von OpenPose erkannt wurden. Im Gegensatz dazu schneidet das erarbeitete System bei der Schätzung von Posen in Breakdance-Videos sehr gut ab. Das System ist in der Lage, schnelle Bewegungen und akrobatische Elemente sehr genau zu erfassen und zu verfolgen. Gelenkwinkel werden präzise identifiziert und die Verbindungslinien zwischen Körperteilen sind selbst bei schnellen und komplexen Bewegungen deutlich sichtbar. Auf dem Bild ist nur ein Gelenk im Arm zu sehen, das nicht ganz in der richtigen Position ist. Der Vergleich der Ergebnisse zeigt, dass das System bei der Pose Estimation im Breakdance-Video etwas besser abschneidet als OpenPose. Die präzise Erfassung der Körperhaltung und die robuste Verfolgung der Bewegungen ermöglichen eine genaue Analyse der Breakdance-Performance.

Abbildung 5.7: Breakdance Pose OpenPose

Abbildung 5.8: Breakdance Pose erarbeitetes System

In diesem weiteren Ausschnitt aus dem Breakdance Video lässt sich feststellen, dass OpenPose im Vergleich zum vorherigen Beispiel eine etwas verbesserte Leistung gezeigt hat. Das System war in der Lage, die Bewegungen im Foto genauer zu erfassen und den komplexen Posen des Tänzers präziser zu verfolgen. Es gab jedoch immer noch einige Unschärfen und Ungenauigkeiten in den erkannten Gelenkwinkeln und Verbindungslinien. Im Gegensatz dazu hatte das erarbeitete System wenig bis gar keine Probleme bei der Verarbeitung dieses Breakdance-Materials. Selbst wenn das Gesicht verdeckt war, wurden Teile des Gesichts korrekt erkannt. Die Leistung von dem System bei der Pose Estimation im Breakdance-Ausschnitt bestätigt seine Fähigkeit, auch in anspruchsvollen Szenarien hohe Genauigkeit und Präzision zu liefern. Die präzise Erfassung der Körperhaltung

und die klaren Verbindungslinien zwischen den Körperteilen ermöglichen eine genaue Analyse. Diese Beobachtungen zeigen deutlich, dass das erarbeitete System in diesem spezifischen Breakdance-Ausschnitt eine überlegene Leistung gegenüber OpenPose zeigt. Es ist wichtig anzumerken, dass die Leistung der Pose Estimation Systeme stark von den spezifischen Bewegungen und Pose-Charakteristika abhängt. In diesem Fall hat das System jedoch eine präzisere Pose Estimation im Vergleich zu OpenPose erreicht.

5.0.3 Benchmark Ergebnisse

Im folgenden Abschnitt werden die Ergebnisse von Benchmark-Tests zwischen OpenPose und MediaPipe zur Pose Estimation verglichen. Diese Untersuchung beruht auf den Informationen aus dem Paper zu BlazePose (BlazePose: On-device Real-time Body Pose tracking. 2020), einer fortschrittlichen Technologie, die in MediaPipe für die Pose Estimation implementiert ist. In dieser Studie wurde eine Reihe von Tests und Messungen durchgeführt, um die Leistungsfähigkeit und Genauigkeit beider Systeme zu bewerten. Die Ergebnisse dieser Benchmarks bieten wertvolle Einblicke in die Fähigkeiten und Stärken der beiden Pose Estimation-Technologien. Durch den Vergleich der Leistungswerte wird ein detaillierter Überblick darüber erhalten, wie sich OpenPose und MediaPipe in verschiedenen Szenarien und Anwendungsbereichen behaupten und welche Systemeigenschaften ihre Leistung beeinflussen. Die Benchmarking-Ergebnisse werden helfen, die Stärken und Schwächen beider Systeme besser zu verstehen und eine fundierte Einschätzung ihrer Eignung für bestimmte Aufgaben vorzunehmen.

Tabelle 5.1: Model FPS und PCK@0.2 auf verschiedenen Datensätzen

Model	FPS	AR Dataset PCK@0.2	Yoga Dataset PCK@0.2
OpenPose (body only)	0.4	87.8	83.4
BlazePose Full	10	84.1	84.5
BlazePose Lite	31	79.6	77.6

BlazePose lief auf einem Pixel 2 Single Core via XNNPACK backend

Openpose lief auf einer Desktop CPU with 20 cores (Intel i9-7900X)

Um die Qualität ihres Modells zu bewerten, haben die Autoren OpenPose als Baseline ausgewählt. Dafür haben sie manuell zwei eigene Datensätze mit jeweils 1000 Bildern annotiert. Jeder Datensatz enthält 1-2 Personen in verschiedenen Szenarien. Der erste Datensatz, AR-Datensatz genannt, besteht aus verschiedenen menschlichen Posen in einer natürlichen Umgebung, während der zweite Datensatz nur Yoga-/Fitness-Posen enthält. Als Auswertungsmetrik verwendeten sie den „Percent of Correct Points with 20% tolerance“ (PCK@0.2). Dabei galt ein Punkt als korrekt erkannt, wenn der 2D-euklidische Fehler kleiner als 20% der Torso-Größe der entsprechenden Person war. Um die menschliche Baseline zu überprüfen, wurden zwei Annotatoren gebeten, den AR-Datensatz unabhängig voneinander zu annotieren. Dabei erzielten sie einen durchschnittlichen PCK@0.2 von 97,2. Die Autoren trainierten zwei Modelle mit unterschiedlichen Kapazitäten: BlazePose Full (6,9 MFlop, 3,5 Millionen Parameter) und BlazePose Lite (2,7 MFlop, 1,3 Millionen Parameter). Obwohl ihre Modelle auf dem AR-Datensatz eine etwas schlechtere Leistung als das OpenPose-Modell zeigten, übertraf BlazePose Full OpenPose bei Yoga-/Fitness-Anwendungsfällen. Gleichzeitig war BlazePose je nach gewünschter Qualität 25-75 Mal schneller auf einem einzelnen Mittelklasse-Smartphone-Prozessor im Vergleich zu OpenPose auf einem Desktop-Prozessor mit 20 Kernen. Die Tabelle vergleicht BlazePose mit OpenPose und zeigt die Werte für die Bilder pro Sekunde (FPS) und PCK@0.2 sowohl für den AR-Datensatz als auch für den Yoga-Datensatz. Insgesamt zeigte BlazePose im

Vergleich zu OpenPose eine wettbewerbsfähige Leistung, insbesondere bei Yoga-/Fitness-Anwendungsfällen. Es bot auch erhebliche Geschwindigkeitsverbesserungen, was es effizienter für echtzeitfähige Anwendungen auf mobilen Geräten macht. [3]

6 Zusammenfassung

Zusammenfassend lässt sich sagen, dass Pose Estimation eine vielversprechende Technologie ist, die eine genaue und präzise Erfassung von Körperhaltungen ermöglicht. Die hier vorgestellten Ergebnisse zeigen, dass sowohl das Pose Estimation System auf Basis von MediaPipe als auch das weit verbreitete System OpenPose in der Lage sind, gute Ergebnisse bei der Erkennung von Körperhaltungen zu liefern. Beide Systeme zeigen eine hohe Genauigkeit und Präzision, wobei MediaPipe in Echtzeit Szenarien und Anwendungsbereichen eine überlegene Leistung zeigt. Die durchgeführten Vergleichsanalysen lieferten wertvolle Einblicke in die Leistung und Genauigkeit meines MediaPipe-basierten Pose Estimation Systems im Vergleich zum weit verbreiteten System OpenPose. Die Ergebnisse basieren auf Experimenten mit unterschiedlichen Posen und Schwierigkeitsgraden an ausgewählten Videos. Das erarbeitete System weist eine hohe Genauigkeit bei der Erkennung von Körperhaltungen auf, wie eine detaillierte Auswertung der erkannten Gelenkpunkte und deren Übereinstimmung mit tatsächlichen Positionen zeigt. OpenPose schnitt ebenfalls gut ab, allerdings wurden in einigen Fällen leichte Unterschiede in den erkannten Gelenkpositionen festgestellt, wie in dem Vergleich des Breakdance Videos zu lesen war. Die präsentierten Ergebnisse werden durch visuelle Darstellungen unterstützt, um die Unterschiede und Gemeinsamkeiten zwischen den beiden Systemen zu verdeutlichen. Für die Untersuchung der Pose Estimation wurden sowohl ein Yoga-Beispielvideo als auch ein Breakdance-Video verwendet, um die Leistung und Genauigkeit beider Systeme zu bewerten. Bei der Pose Estimation im Yoga-Video zeigen sowohl das erarbeitete System als auch OpenPose eine genaue Erfassung der Körperhaltung und der Gelenkpositionen. Während das erarbeitete System eine hohe Genauigkeit bei der Erkennung der Armpositionen und Beinpositionen zeigte, zeigte OpenPose in einigen Fällen leichte Abweichungen bei den erkannten Gelenkpositionen. Im Breakdance-Video erzielt das erarbeitete System eine überlegene Leistung gegenüber OpenPose, da es schnelle Bewegungen und akrobatische Elemente genau erfasst und verfolgt. OpenPose hingegen zeigt Ungenauigkeiten und Unschärfen in den erkannten Gelenkwinkeln und Verbindungslinien. Die Ergebnisse dieser Vergleichsanalyse liefern wertvolle Erkenntnisse für die Weiterentwick-

lung und Anwendung meines Pose Estimation Systems. Gerade bei Anwendungen mit schnellen und komplexen Bewegungen, wie Sportanalysen, Bewegungserkennung oder Rehabilitationstechniken können die präzise Erfassung der Körperhaltung und die robuste Verfolgung der Bewegungen meines Systems von Vorteil sein. Die Benchmarks zwischen OpenPose und MediaPipe zeigen, dass BlazePose, eine Technologie in MediaPipe implementiert, eine wettbewerbsfähige Leistung und erhebliche Geschwindigkeitsverbesserungen im Vergleich zu OpenPose bietet, insbesondere bei Yoga-/Fitness-Anwendungsfällen. Dies macht es effizienter für echtzeitfähige Anwendungen auf mobilen Geräten. Insgesamt kann festgestellt werden, dass das erarbeitete Pose Estimation System eine hohe Genauigkeit und Präzision bei der Erkennung von Körperhaltungen aufweist und in bestimmten Szenarien, wie bei schnellen und komplexen Bewegungen, überlegen sein kann. Durch den modularen Aufbau meines Systems habe wurde zudem eine solide Grundlage geschaffen, um das System in Zukunft effektiv zu erweitern und anzupassen. Diese Flexibilität ermöglicht es, neue Funktionen und Module nahtlos hinzuzufügen und so die Leistung und Funktionalität des Systems kontinuierlich zu verbessern. Der Vorteil des modularen Ansatzes besteht darin, dass neue Algorithmen und Modelle einfach in das System integriert werden können, um die Möglichkeiten zur Pose Estimation zu erweitern. Durch die kontinuierliche Integration neuer Algorithmen ist es stets auf dem neuesten Stand der Technik. Außerdem ermöglichte mir die Struktur die reibungslose Integration zusätzlicher Sensoren in das System. Die Integration von Inertialsensoren oder Tiefenkameras kann zusätzliche Informationen über die Körperhaltung sammeln und dabei helfen, die Pose Estimation zu verbessern. Aufgrund der Modularität können diese Sensoren einfach in den Posenschätzungsprozess integriert und die gewonnenen Daten effizient genutzt werden. Neben der Anpassungsfähigkeit sind modulare Systeme skalierbar und erweiterbar. Das System kann relativ einfach erweitert werden, indem weitere Instanzen bestehender Module hinzugefügt oder neue Module für zusätzliche Funktionalität entwickelt werden. Dadurch kann das System an wachsende Bedürfnisse angepasst und auf verschiedenen Plattformen und in unterschiedlichen Umgebungen eingesetzt werden. Ein weiterer Vorteil besteht darin, dass es einfacher zu warten und zu aktualisieren ist. Dies erleichtert die Fehlerbehebung, die Integration neuer Funktionen und die Anpassung an neue Technologien. Insgesamt bietet der modulare Aufbau des Systems viele Vorteile für zukünftige Erweiterungen und Anpassungen. Durch die nahtlose Integration neuer Algorithmen, Sensoren und Anwendungsmodule ist es möglich, die Systemleistung kontinuierlich zu verbessern und an unterschiedliche Anwendungsfälle anzupassen. Auf diese Weise kann das System immer auf dem neuesten Stand der Technik bleiben und präzise sowie vielseitige Pose Estimation Funktionen bieten. Zusammenfassend lässt sich sagen, dass die

Posenschätzung das Potenzial hat, unsere Interaktionen mit Technologie zu verbessern, neue Anwendungen in Bereichen wie Gesundheit, Sport, Animation und Sicherheit zu ermöglichen und ein breites Spektrum an Einsatzmöglichkeiten zu eröffnen.

7 Ausblick

Im vorangegangenen Kapitel wurde ein modulares Pose Estimation-System vorgestellt, und die verschiedenen Komponenten sowie die Vorteile seines modularen Aufbaus wurden erläutert. Nun wird ein Blick in die Zukunft geworfen, um potenzielle Entwicklungen und Anwendungsmöglichkeiten des Systems zu diskutieren. Der Ausblick befasst sich mit den zukünftigen Forschungsrichtungen, die darauf abzielen, die Genauigkeit und Leistung des Systems weiter zu verbessern und es in verschiedenen Bereichen einzusetzen. Darüber hinaus wird das Potenzial des Systems zur Förderung der barrierefreien Technologie und zur Verbesserung der Mensch-Maschine-Interaktion beleuchtet. Die Erforschung dieser Aspekte könnte zu Innovationen führen, die unser Verständnis von Bewegung und Körperhaltung erweitern und unsere Lebensqualität bereichern. Im Folgenden werden mögliche Wege aufgezeigt, wie das System in Zukunft erweitert und angewendet werden kann.

7.0.1 Verbesserung der Genauigkeit durch die Verwendung mehrerer Kameras und Tiefenkameras

Ein vielversprechender Ansatz zur weiteren Verbesserung der Genauigkeit der Pose Estimation ist der Einsatz mehrerer Kameras oder sogar spezieller Tiefenkameras. Durch die gleichzeitige Verwendung mehrerer Kameras können mehr Winkel und Perspektiven erfasst werden. Dies verbessert die Genauigkeit der 3D-Rekonstruktion und damit die Genauigkeit der Pose Estimation. Durch die Integration mehrerer Kameras können verschiedene Aspekte der Pose Estimation optimiert werden. Beispielsweise können Kameras mit unterschiedlichen Brennweiten oder Blickwinkeln verwendet werden, um eine umfassendere Pose Estimation zu ermöglichen. Dies ist besonders nützlich, wenn bestimmte Körperteile wie Hände oder Füße schwer zu erkennen sind oder wenn sich die Person schnell bewegt. Die Kombination verschiedener Ansichten verschiedener Kameras ermöglicht eine robustere und genauere Schätzung der Pose. Ein weiterer vielversprechender Ansatz zur Verbesserung der Genauigkeit ist der Einsatz von Tiefenkameras, die neben

Farbinformationen auch Tiefeninformationen für Bereiche der Szene erfassen. Durch die Kombination von Farbinformationen und Tiefeninformationen kann die Pose Estimation ein neues Maß an Genauigkeit erreichen. Tiefenkameras sind in der Lage, die räumliche Tiefe genauer zu erfassen und somit Posen genauer zu rekonstruieren. Sie sind besonders nützlich, um die Genauigkeit in Bereichen zu verbessern, die für herkömmliche Kameras aufgrund von Unschärfe, geringem Kontrast oder großen Schwankungen in der Lichtqualität schwierig sind. Die Integration mehrerer Kameras bzw. Tiefenkameras eröffnet auch neue Möglichkeiten zur Verbesserung anderer Aspekte des Systems. Durch die parallele Verarbeitung der Daten aus mehreren Kameras können Echtzeit-Anwendungen ermöglicht werden, bei denen eine schnelle und präzise Pose Estimation erforderlich ist. Darüber hinaus können durch den Einsatz einer Tiefenkamera zusätzliche Informationen wie die 3D-Form des Körpers oder die Position von Objekten in der Umgebung erfasst werden, was dem System neue Anwendungsmöglichkeiten eröffnet.

7.0.2 Virtuelle Realität (VR) und erweiterte Realität (AR)

Die Pose Estimation, insbesondere in Kombination mit Virtual Reality (VR) und Augmented Reality (AR), eröffnet eine Fülle von Möglichkeiten und Anwendungen. Sie trägt dazu bei, das immersive Erlebnis und die Interaktion zwischen Benutzern und virtuellen oder erweiterten Umgebungen zu verbessern. In der virtuellen Realität ermöglicht die Pose Estimation eine realistischere Darstellung des eigenen Körpers in der virtuellen Welt. Durch die Verfolgung von Körperbewegungen und Körperpositionen kann das System den virtuellen Charakter oder Avatar des Benutzers entsprechend animieren. Dadurch entsteht ein noch intensiveres Erlebnis, da Benutzer ihre eigenen Posen und Bewegungen im virtuellen Raum erkennen. Dadurch entsteht ein tieferes Gefühl der Präsenz und ein stärkeres Gefühl des Eintauchens in die virtuelle Umgebung. Darüber hinaus ermöglicht die Pose Estimation in VR natürliche und intuitive Interaktionen. Benutzer können mit der virtuellen Welt mithilfe der Hände oder dem ganzen Körper interagieren, ohne auf spezielle Controller oder Eingabegeräte angewiesen zu sein. Dies eröffnet neue Möglichkeiten für virtuelle Spiele, Simulationen, Trainingsanwendungen und kreative Designumgebungen. In VR-Spielen können Benutzer beispielsweise virtuelle Objekte mit ihren Händen aufnehmen, werfen oder manipulieren und so ein realistischeres und intuitiveres Spielerlebnis schaffen. Auch in der Augmented Reality eröffnet die Pose Estimation vielfältige Anwendungsmöglichkeiten. Durch die genaue Erfassung von Pose und Körperposition kann das System virtuelle Inhalte nahtlos in die reale Umgebung integrieren.

Dies könnte die Interaktion mit der realen Welt erweitern und neue Anwendungen in Bereichen wie Architektur, Design, Bildung und Ausbildung eröffnen. Darüber hinaus kann die Pose Estimation auch in der Augmented Reality zur Verbesserung der menschlichen Wahrnehmung und zur Unterstützung von Aufgaben eingesetzt werden. Beispielsweise können AR-Systeme Medizinern bei der präzisen Positionierung oder Bewegung während einer Operation helfen, indem sie ihnen während des Eingriffs Echtzeit-Visualisierungen von relevanten anatomischen Strukturen und medizinischen Informationen bieten. Dies ermöglicht es dem Chirurgen, eine klarere und detailliertere Sicht auf das Operationsgebiet zu haben, was wiederum die Genauigkeit der Platzierung von Instrumenten oder Implantaten verbessert und das Risiko von Fehlern minimiert. Außerdem können AR-Anwendungen in der Logistik oder im Handel bei der Optimierung von Arbeitsabläufen und bei der Durchführung von Aufgaben wie Kommissionierung, Verpackung oder Lagerhaltung unterstützen, indem sie den Mitarbeitern in Echtzeit visuelle Informationen überlagern. Dadurch können sie Produkte im Lager schneller lokalisieren, den besten Weg zur Entnahme anzeigen, Verpackungsanweisungen anzeigen und sogar die optimale Anordnung von Waren in Kartons oder Regalen visualisieren, um den Platz effizient zu nutzen. Diese immersive und informative Technologie trägt dazu bei, die Produktivität zu steigern und Fehler zu minimieren.

7.0.3 Gesundheit und Fitness

Die Pose Estimation hat das Potenzial, in der Gesundheit und Fitness neue Möglichkeiten zu eröffnen, indem es präzise Informationen über Körperhaltung und Bewegungen liefert. Diese Technologie kann in verschiedenen Anwendungen eingesetzt werden, um die Körperwahrnehmung, das Training und die Rehabilitation zu verbessern. Im Fitnessbereich kann die Pose Estimation ein genaues Feedback für die korrekte Durchführung von Übungen liefern. Durch die Verfolgung der Körperhaltung in Echtzeit kann das System Fehler oder Ungenauigkeiten erkennen und den Benutzer entsprechend korrigieren. Dies ist besonders nützlich für diejenigen, die alleine oder ohne persönliche Anleitung trainieren. Indem sie Feedback zu ihrer Haltung erhalten, können sie sicherstellen, dass sie ihre Übungen korrekt ausführen und Verletzungen so vorbeugen. Darüber hinaus können Fortschritte und Verbesserungen im Laufe der Zeit verfolgt werden, wodurch die Trainingsmotivation und das Engagement gesteigert werden können. Die Pose Estimation kann auch einen wesentlichen Beitrag zur physiotherapeutischen Rehabilitation leisten. Es ermöglicht eine präzise Bewegungsüberwachung während der Rehabilitation. Thera-

peuten können den Fortschritt des Patienten genau überwachen und einen individuellen Behandlungsplan erstellen. Darüber hinaus können Patienten mobile Geräte oder Wearables verwenden, um Übungen durchzuführen und die Genauigkeit ihrer Bewegungen zu Hause oder außerhalb der Klinik zu überwachen. Dies ermöglicht eine effektivere Rehabilitation und eine schnellere Genesung. Ein weiterer Bereich, in dem die Pose Estimation im Gesundheitswesen nützlich ist, ist die Ergonomie und die Prävention von Verletzungen am Arbeitsplatz. Durch eine genaue Überwachung der Körperhaltung können potenziell schädliche Bewegungen oder eine schlechte Körperhaltung erkannt werden. Dies ist besonders wichtig für Jobs, die sich wiederholende Bewegungen erfordern oder bei denen Mitarbeiter längere Zeit in einer bestimmten Position verbringen, beispielsweise in Büros, in der Fertigung oder im Gesundheitswesen. Durch die Identifizierung von Risikofaktoren können geeignete Maßnahmen ergriffen werden, um Verletzungen und muskuloskelettale Beschwerden zu vermeiden.

7.0.4 Überwachungssysteme

Die Integration der Pose Estimation in Sicherheitssystemen und Überwachungssystemen kann einen wesentlichen Beitrag zur Verbesserung der Wirksamkeit und Genauigkeit dieser Systeme leisten. Durch die Erfassung von Informationen über die Pose und Bewegung einer Person ermöglicht die Pose Estimation eine präzise und kontextbezogene Überwachung, die über die herkömmliche Videoüberwachung hinausgeht. Ein Bereich, in dem die Pose Estimation in Sicherheitssystemen eine wichtige Rolle spielen kann, ist die Erkennung verdächtigen Verhaltens. Durch die Verfolgung von Körperhaltung und Bewegung kann das System ungewöhnliche oder potenziell gefährliche Aktivitäten erkennen, wie etwa das Abstellen von Gepäck an öffentlichen Orten oder das Betreten von Sperrbereichen ohne Erlaubnis. Dies ermöglicht eine frühzeitige Warnung und Intervention zur Eindämmung potenzieller Bedrohungen. Ein weiterer Anwendungsfall ist die Überwachung von Menschenmengen. In stark frequentierten Bereichen wie Bahnhöfen, Stadien oder Einkaufszentren hilft die Pose Estimation dabei, Bewegungsmuster zu analysieren und potenzielle Engpässe oder gefährliche Situationen zu erkennen. Durch die Erkennung von Staus oder Menschenansammlungen können vorbeugende Maßnahmen ergriffen werden, um einen sicheren und reibungslosen Betrieb zu gewährleisten. Darüber hinaus kann die Pose Estimation auch in Sicherheitssystemen und Überwachungssystemen zur Überwachung von Mitarbeitern in Industrie und Arbeitsumgebungen eingesetzt werden. Durch die genaue Erfassung von Körperhaltung und Bewegungen können potenziell gefährliche

Situationen oder Verstöße gegen Sicherheitsrichtlinien erkannt werden. Dies hilft Unternehmen, Arbeitsunfälle zu reduzieren und die Einhaltung von Sicherheitsstandards zu Verbessern.

7.0.5 Robotik

Die Integration der Pose Estimation in die Robotik eröffnet vielfältige Möglichkeiten und bietet viele Vorteile. Durch die genaue Erfassung menschlicher Körperhaltung und Bewegungen ermöglicht die Pose Estimation Robotern, ihre Umgebung und ihre Interaktion mit Menschen besser zu verstehen. Eine Kernanwendung ist die menschenähnliche Bewegungssteuerung von Robotern. Mithilfe der Pose Estimation können Roboter menschenähnliche Bewegungen erlernen und reproduzieren. Dies ermöglicht eine natürlichere und intuitivere Interaktion zwischen Menschen und Robotern. Roboter können lernen und sich an Aktionen wie Gehen, Greifen von Objekten und sogar komplexe Aktionen wie die Verwendung von Werkzeugen anpassen. Dies hat das Potenzial, die Einsatzmöglichkeiten von Robotern zu erweitern, sei es in der Industrie, im Gesundheitswesen oder im Haushalt. Darüber hinaus kann die Pose Estimation auch eine sichere Zusammenarbeit zwischen Mensch und Roboter ermöglichen. Durch die genaue Erfassung von Körperhaltungen und Körperbewegungen kann der Roboter in Echtzeit auf menschliche Bewegungen und Positionen reagieren. Dies ist besonders wichtig in Umgebungen, in denen Menschen und Roboter eng zusammenarbeiten, beispielsweise in der Montage oder bei der Rehabilitation. Roboter können menschliche Bewegungen vorhersagen und entsprechend reagieren, um Kollisionen oder Verletzungen zu vermeiden. Darüber hinaus eröffnet die Integration der Pose Estimation in die Robotik auch neue Möglichkeiten für die Mensch-Roboter-Kommunikation. Durch die Erfassung von Körperhaltung und Bewegungen können Roboter menschliche Gesten und nonverbale Signale interpretieren und darauf reagieren. Dies ermöglicht eine natürlichere und effizientere Kommunikation zwischen Menschen und Robotern, sei es bei der Steuerung von Aufgaben oder beim Austausch von Informationen.

7.0.6 Fazit

Zusammenfassend gesagt, kann das modulare Pose Estimation System in einer Vielzahl von Bereichen weiterentwickelt und angewendet werden. Die Verbesserung der Genauigkeit, die Anwendung in der virtuellen und erweiterten Realität, die Nutzung im

Gesundheits- und Fitnessbereich sowie die Integration in Sicherheits- und Überwachungssysteme sind nur einige der möglichen Forschungsrichtungen und Anwendungen. Durch die kontinuierliche Forschung und Weiterentwicklung können Innovationen und Lösungen entstehen, die unser Verständnis von Bewegung und Körperhaltung erweitern und unsere Lebensqualität verbessern können.

Literaturverzeichnis

[1] *Post Tracking Full Body Landmarks.* https://camo.githubusercontent.com/7fbec98ddbc1dc4186852d1c29487efd7b1eb820c8b6ef34e113fcde40746be2/68747470733a2f2f6d6564696170970652e6465762f696d616765732f6d6f62696c652f70 – Accessed on July 3, 2023

[2] Andriluka, Mykhaylo ; Pishchulin, Leonid ; Gehler, Peter ; Schiele, Bernt: 2D Human Pose Estimation: New Benchmark and State of the Art Analysis. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014

[3] Bazarevsky, Valentin ; Grishchenko, Ivan ; Raveendran, Karthik ; Zhu, Tyler ; Zhang, Fan ; Grundmann, Matthias: *BlazePose: On-device Real-time Body Pose tracking.* 2020

[4] Blog, Google A.: *On-Device Real-time Body Pose Tracking.* https://ai.googleblog.com/2020/08/on-device-real-time-body-pose-tracking.html. Aug 2020. – Accessed: May 7, 2023

[5] Chen, Haoming ; Feng, Runyang ; Wu, Sifan ; Xu, Hao ; Zhou, Fengcheng ; Liu, Zhenguang: *2D Human Pose Estimation: A Survey.* 2022

[6] Culjak, Ivan ; Abram, David ; Pribanic, Tomislav ; Dzapo, Hrvoje ; Cifrek, Mario: A brief introduction to OpenCV. In: *2012 Proceedings of the 35th International Convention MIPRO*, 2012, S. 1725–1730

[7] Dai, Jifeng ; Li, Yi ; He, Kaiming ; Sun, Jian: *R-FCN: Object Detection via Region-based Fully Convolutional Networks.* 2016

[8] Diba, Ali ; Fayyaz, Mohsen ; Sharma, Vivek ; Karami, Amir H. ; Arzani, Mohammad M. ; Yousefzadeh, Rahman ; Gool, Luc V.: Temporal 3D ConvNets: New Architecture and Transfer Learning for Video Classification. In: *CoRR* abs/1711.08200 (2017). – URL http://arxiv.org/abs/1711.08200

[9] Elmi, Alessio ; Mazzini, Davide ; Tortella, Pietro: *Light3DPose: Real-time Multi-Person 3D PoseEstimation from Multiple Views*. 2020

[10] Felzenszwalb, Pedro F. ; Girshick, Ross B. ; McAllester, David A. ; Ramanan, Deva: Visual object detection with deformable part models. In: *Commun. ACM* 56 (2013), S. 97–105

[11] Felzenszwalb, Pedro F. ; Huttenlocher, Daniel P.: Pictorial Structures for Object Recognition. In: *International Journal of Computer Vision* 61 (2004), S. 55–79

[12] Girshick, Ross: *Fast R-CNN*. 2015

[13] Google: *pipe Framework*. https://developers.google.com/mediapipe/framework

[14] K, Amrutha ; P, Prabu ; Paulose, Joy: Human Body Pose Estimation and Applications. In: *2021 Innovations in Power and Advanced Computing Technologies (i-PACT)*, 2021, S. 1–6

[15] Lin, Tsung-Yi ; Maire, Michael ; Belongie, Serge ; Hays, James ; Perona, Pietro ; Ramanan, Deva ; Dollár, Piotr ; Zitnick, C. L.: Microsoft COCO: Common Objects in Context. In: Fleet, David (Hrsg.) ; Pajdla, Tomas (Hrsg.) ; Schiele, Bernt (Hrsg.) ; Tuytelaars, Tinne (Hrsg.): *Computer Vision – ECCV 2014*. Cham : Springer International Publishing, 2014. – ISBN 978-3-319-10602-1

[16] Liu, Wei ; Anguelov, Dragomir ; Erhan, Dumitru ; Szegedy, Christian ; Reed, Scott ; Fu, Cheng-Yang ; Berg, Alexander C.: SSD: Single Shot MultiBox Detector. In: Leibe, Bastian (Hrsg.) ; Matas, Jiri (Hrsg.) ; Sebe, Nicu (Hrsg.) ; Welling, Max (Hrsg.): *Computer Vision – ECCV 2016*. Cham : Springer International Publishing, 2016. – ISBN 978-3-319-46448-0

[17] Liu, Wei ; Anguelov, Dragomir ; Erhan, Dumitru ; Szegedy, Christian ; Reed, Scott ; Fu, Cheng-Yang ; Berg, Alexander C.: SSD: Single Shot MultiBox Detector. In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, S. 21–37. – URL https://doi.org/10.1007%2F978-3-319-46448-0_2

[18] Liu, Wu ; Bao, Qian ; Sun, Yu ; Mei, Tao: Recent Advances in Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective. In: *CoRR* abs/2104.11536 (2021). – URL https://arxiv.org/abs/2104.11536

[19] Long, Jonathan ; Shelhamer, Evan ; Darrell, Trevor: Fully convolutional networks for semantic segmentation. In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, S. 3431–3440

[20] Mehta, Dushyant ; Rhodin, Helge ; Casas, Dan ; Sotnychenko, Oleksandr ; Xu, Weipeng ; Theobalt, Christian: Monocular 3D Human Pose Estimation Using Transfer Learning and Improved CNN Supervision. In: *CoRR* abs/1611.09813 (2016). – URL http://arxiv.org/abs/1611.09813

[21] Newell, Alejandro ; Huang, Zhiao ; Deng, Jia: *Associative Embedding: End-to-End Learning for Joint Detection and Grouping*. 2017

[22] OpenCV: *OpenCV Documentation*. https://docs.opencv.org/. Date accessed

[23] Pouly, Marc ; Koller, Thomas ; Gottfrois, Philippe ; Lionetti, Simone: Künstliche Intelligenz in der Bildanalyse–Grundlagen und neue Entwicklungen. In: *Der Hautarzt* 71 (2020), Nr. 9, S. 660–668. – URL https://doi.org/10.1007/s00105-020-04663-7. – ISSN 1432-1173

[24] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: You Only Look Once: Unified, Real-Time Object Detection. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, S. 779–788

[25] Redmon, Joseph ; Divvala, Santosh ; Girshick, Ross ; Farhadi, Ali: *You Only Look Once: Unified, Real-Time Object Detection*. 2016

[26] Stenum, Jan ; Cherry-Allen, Kendra ; Pyles, Connor ; Reetzke, Rachel ; Vignos, Michael ; Roemmich, Ryan: Applications of Pose Estimation in Human Health and Performance across the Lifespan. In: *Sensors* 21 (2021), 11, S. 7315

[27] Stenum, Jan ; Cherry-Allen, Kendra M. ; Pyles, Connor O. ; Reetzke, Rachel D. ; Vignos, Michael F. ; Roemmich, Ryan T.: Applications of Pose Estimation in Human Health and Performance across the Lifespan. In: *Sensors* 21 (2021), Nr. 21. – URL https://www.mdpi.com/1424-8220/21/21/7315. – ISSN 1424-8220

[28] Toshev, Alexander ; Szegedy, Christian: DeepPose: Human Pose Estimation via Deep Neural Networks. In: *CoRR* abs/1312.4659 (2013). – URL http://arxiv.org/abs/1312.4659

[29] Tu, Hanyue ; Wang, Chunyu ; Zeng, Wenjun: *VoxelPose: Towards Multi-Camera 3D Human Pose Estimation in Wild Environment.* 2020

[30] Wang, Jiahang ; Jin, Sheng ; Liu, Wentao ; Liu, Weizhong ; Qian, Chen ; Luo, Ping: *When Human Pose Estimation Meets Robustness: Adversarial Algorithms and Benchmarks.* 2021

[31] Xiao, Bin ; Wu, Haiping ; Wei, Yichen: *Simple Baselines for Human Pose Estimation and Tracking.* 2018

[32] Xiu, Yuliang ; Li, Jiefeng ; Wang, Haoyu ; Fang, Yinghong ; Lu, Cewu: Pose Flow: Efficient Online Pose Tracking. In: *CoRR* abs/1802.00977 (2018). – URL http://arxiv.org/abs/1802.00977

[33] Yang, Wei ; Ouyang, Wanli ; Wang, Xiaolong ; Ren, Jimmy S. J. ; Li, Hongsheng ; Wang, Xiaogang: 3D Human Pose Estimation in the Wild by Adversarial Learning. In: *CoRR* abs/1803.09722 (2018). – URL http://arxiv.org/abs/1803.09722

[34] Zhang, Letian ; Xu, Jie: *E^3Pose: Energy-Efficient Edge-assisted Multi-camera System for Multi-human 3D Pose Estimation.* 2023

[35] Zheng, Ce ; Wu, Wenhan ; Yang, Taojiannan ; Zhu, Sijie ; Chen, Chen ; Liu, Ruixu ; Shen, Ju ; Kehtarnavaz, Nasser ; Shah, Mubarak: Deep Learning-Based Human Pose Estimation: A Survey. In: *CoRR* abs/2012.13392 (2020). – URL https://arxiv.org/abs/2012.13392

Erklärung zur selbstständigen Bearbeitung

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

Ort	Datum	Unterschrift im Original