der Bundeswehr
Universität München

Analyse und Implementierung von MediaPipe BlazePose für 3D Body-Tracking

Bachelorarbeit

James Daniel Beutler

1202523

Erstprüfer:	Prof. Dr. Michael Koch
Zweitprüfer:	Prof. Dr. Florian Alt
Betreuer:	Julian Fietkau
Abgabetermin:	06.02.2023

Universität der Bundeswehr München

Fakultät für Informatik

Kurzfassung

In der Forschung im Themenbereich Mensch-Computer-Interaktion gibt es für ein großes Spektrum an Problemfeldern eine Vielzahl an Lösungen. Sobald eine Person im (halb-)öffentlichen Raum mit einem interaktiven Bildschirm interagiert, kann das eventuell Auswirkungen auf das Verhalten von Passanten haben. Dieses Phänomen wird Honeypot-Effekt genannt. Um das Verfahren zur Untersuchung dieses Effekts zu vereinfachen, nutzt die Universität der Bundeswehr in München in Kollaboration mit der Hochschule für Angewandte Wissenschaften Hamburg das sogenannte 3D Body-Tracking. Hierbei ist es bisher unumgänglich gewesen, sich einer speziell für diesen Zweck entwickelten Hardware zu bedienen. Dieser Faktor kann einschränkend wirken und daher ist es Ziel dieser Arbeit ein System zu implementieren, welches nicht auf spezielle Sensorik angewiesen ist. Die Integration eines neuen 3D Body-Tracking Systems in die vorhandene Software zum abspielen der Tracking-Aufnahmen dient zur Steigerung der Flexibilität und Erweiterbarkeit des Programms. Tiefere Einblicke in die Software ergaben jedoch, dass die Erweiterung um das neue Aufzeichnungsformat mit einigen Abzügen hinsichtlich der Funktionalität und der Qualität der Tiefenerkennung verbunden ist. Positiv kann angemerkt werden, dass die implementierte Methode unabhängig vom Betriebssystem operieren kann. Des weiteren ist es möglich, eine Vielzahl an weiteren Funktionen auf deren Basis hinzuzufügen. Dieser letzte Faktor eröffnet neue Möglichkeiten hinsichtlich der Forschung rund um die Mensch-Computer-Interaktionen und vereinfacht zukünftige Arbeiten und Wartungen an der Software.

Abstract

In the field of human-computer interaction research, there are a multitude of solutions for a wide range of problem areas. When a person interacts with an interactive screen in a (semi-)public space, it may affect the behavior of passers-by. This phenomenon is called the Honeypot effect. To simplify the procedure for investigating this effect, the University of the Bundeswehr in Munich in collaboration with the University of Applied Sciences Hamburg is using 3D body tracking. So far, it has been unavoidable to use specially developed hardware for this purpose. This factor can be restrictive, and therefore the aim of this work is to implement a system that is not dependent on special sensors. Integrating a new 3D body tracking system into the existing software for playing the tracking recordings is intended to increase the flexibility and expandability of the program. However, deeper insights into the software revealed that the extension to the new recording format is associated with some limitations in terms of functionality and depth recognition quality. On the other hand, the implemented method can operate independently of the operating system. Furthermore, a multitude of additional functions can be added on this basis. This last factor opens up new possibilities for research on human-computer interactions and simplifies future work and maintenance on the software.

Inhaltsverzeichnis

1 Einleitung

Honigtöpfe und Bildschirme sind zwei Wörter die an und für sich jeder kennt und die man im Alltag nicht miteinander in Verbindung bringt. Im Forschungsprojekt **HoPE**[1], welches von der Universität der Bundeswehr München in Zusammenarbeit mit der Hochschule für Angewandte Wissenschaften Hamburg durchgeführt wird, bekommen beide Begriffe jedoch eine ganz neue Bedeutung und im Zusammenhang mit der Forschung im Bereich der Mensch-Computer-Interaktion öffnen diese Wörter den Blick auf ein interessantes und breit gefächertes Themengebiet. Dieses Projekt untersucht den Honeypot-Effekt an (halb-)öffentlichen interaktiven Ambient Displays in einer Langzeitfeldstudie. Auf die Bedeutung der einzelnen Begrifflichkeiten werde ich im nächsten Kapitel näher eingehen. Ins Leben gerufen wurde das Projekt im September 2021 und läuft voraussichtlich bis Ende August 2024. Es werden Erkenntnisse zu verschiedenen Faktoren der Nutzung von interaktiven Bildschirmen gesammelt und wobei ein wichtiger Aspekt, die Steigerung der Aufmerksamkeit von potenziellen Nutzern noch Forschungsbedarf aufweist. Hierzu gehört auch der sogenannte Honeypot-Effekt (Honeypot, auf deutsch Honigtopf), welcher auf dem Gebiet der Mensch-Computer Interaktionen beschreiben soll, inwiefern die Interaktion einer Person mit dem Testobjekt (in diesem Fall ein Ambient Display) andere Personen in unmittelbarer Nähe dazu anregt das Geschehen entweder zu beobachten oder daran teilzunehmen.

Ohne Hilfsmittel kann man dies nur in Echtzeitbeobachtung feststellen und analysieren. An der Universität der Bundeswehr in München wird diesbezüglich jedoch eine Methode verwendet, die diesen Prozess deutlich vereinfacht. Es wird eine Stereokamera mit zwei optischen Sensoren verwendet, mit der man 3D Body-Tracking Aufnahmen generieren kann. Diese Aufnahmen können abgespeichert werden um im Nachhinein dann analysiert zu werden, sodass man nicht auf eventuelle Passanten warten muss, da eine Aufzeichnung nur dann erstellt wird, wenn eine Interaktion mit dem System stattfindet. Eine Schwierigkeit hierbei ist bislang aber die Abhängigkeit von einem bestimmten Kameratypen. Diese Schwierigkeit kann durch die Möglichkeit umgangen werden, Tiefenaufnahmen auch mit herkömmlichen Kameras, die nur über einen optischen Sensor verfügen, zu erstellen. Wie eine mögliche Lösung dieser Problematik aussehen kann behandle ich in den folgenden Kapiteln. Ziel der Arbeit ist es nicht, eine komplett neue Lösung auf die Beine zu stellen, sondern eine messbare Verbesserung der Flexibiliät bei deren Verwendung, mittels Implementierung einer weiteren Methode zur Aufnahme von 3D Body-Tracking Daten, herzustellen. Das gewünschte Ergebnis soll folglich sein, mit einer neuen

[1] `https://www.unibw.de/inf2/forschung/projekte/hope`

Methode qualitativ vergleichbare Aufnahmen erstellen zu können.

Um die Zweckmäßigkeit zu belegen, werden beide Methoden analysiert, um hinterher einen Vergleich aufstellen zu können. Weiterführend sollen das Vorgehen und die getroffenen Entscheidungen besser verständlich gemacht werden, dafür werden die Lösungen in Hardwaregestützte und Softwaregestützte Tiefenerkennung unterteilt. Es ist an dieser Stelle wichtig anzumerken, dass beide Methoden gleichzeitig sowohl Hardware als auch Software benötigen um funktionieren zu können und die hier erwähnte Unterteilung lediglich den Hauptaspekt der jeweiligen Prozedur zur Tiefenerkennung beschreibt.

Um den Vorgang übersichtlicher zu gestalten, ist die Arbeit wie folgt gegliedert: Diese Einleitung stellt das erste Kapitel dar. Um den Bezug der Thematik zur Aktualität zu festigen, werden darauf folgend in Kapitel 2 einige verwandte Arbeiten vorgestellt. Hinterher werden dann im Kapitel 3 alle themenrelevanten Grundlagen, Begriffe und Definitionen, sowie das Konzept von 3D Body-Tracking selbst erläutert. Im nächsten Schritt wird die bisherig genutzte Lösung und deren Funktionsweise erklärt. Um das Kapitel 3 dann abzuschließen wird die Lösung vorgestellt, die implementiert werden soll um das Programm zu erweitern und die somit im Fokus der Arbeit steht. Im Kapitel 4 wird die Anforderungsanalyse behandelt, um auf einzelne Anforderungen die an die Software gestellt werden einzugehen und gleichzeitig die Argumente, welche für die Implementierung des gewählten Formats sprechen, zu untermauern. Das fünfte Kapitel behandelt die Umsetzung der hier erarbeiteten Lösung und ermöglicht einen Einblick in den Programmablauf im Detail. In Kapitel 6 dann die Lösung bewertet, unter anderem basierend auf den aufgestellten Anforderungen. Hier wird die Frage beantwortet, ob es möglich war, alle gewünschten Anforderungen zu erfüllen, sowohl funktional als auch qualitativ und ob die Möglichkeit zu einer Verbesserung besteht. Dafür werden die Ergebnisse verglichen, die mit beiden verschiedenen Methoden erzielt werden konnten. Das abschließende Fazit der Arbeit erfolgt im letzten Kapitel zusammen mit einem Ausblick auf eventuelle Weiterentwicklungen der Software durch eventuelle weiterführende Arbeiten.

2 Verwandte Arbeiten

Diese Bachelorarbeit wurde mit Hilfe eines Computers verfasst. Durch tippen auf Buchstabentasten einer Tastatur als Mensch-Computer-Schnittstelle können Gedanken eingegeben, festgehalten und in Textform auf einem Bildschirm wiedergegeben werden. Anders könnte man auch ausdrücken, dass diese Bachelorarbeit digital aufgezeichnet wurde. Heutzutage sind derartige Vorgänge normal und aus dem Alltag nicht wegzudenken. Noch vor 50 Jahren hätte man zum Verfassen dieser Worte eine Schreibmaschine benutzen müssen oder diesen sogar von Hand geschrieben. Diese Aussage soll eine grob gerichtete Analogie zu der hier vorgestellten Thematik im Bereich Body-Tracking darstellen. In Ihrer Diplomarbeit zum Thema Body-Tracking aus dem Jahr 2012 eröffnet Frau Keil einen Einblick in die Geschichte der „Motion Capture“, der Aufzeichnung von Bewegungen. Motion Capture wurde mit dem Fortschritt der Technologien in den Bereichen wie etwa der Überwachungstechnik, Medizin, Unterhaltungsmedien und vielen mehr immer wichtiger und heutzutage vereinfacht es eine große Menge an Prozessen, welche vor vielen Jahren mühsam von Hand erledigt wurden. Als Beispiel nennt sie in etwa den Zeichentrickfilm *Schneewittchen* von Walt Disney aus dem Jahre 1937. Hier wurden mittels Rotoskopie-Technik Schauspieler gefilmt und zum erstellen des Films Bild für Bild abgepaust. Heute wäre es undenkbar, sich eine solche Mühe zu machen, denn in einer Zeit von computeranimierten Drachen, Aliens und Co. existieren sogenannte Motion-Capturing-Systeme mit denen man die Gestik, die Mimik und die Bewegungen eines Menschen erfassen und mittels Computerprogrammen wiedergeben kann. (Keil, 2012)

Bewegungen können also mittels Motion Capture Technik digitalisiert werden. Der Grundstein hierfür liegt im sogenannten Body-Tracking. Wie vorher erwähnt, ist das nicht nur im Bereich der Unterhaltungsmedien ein Begriff der mit der Zeit an Relevanz gewonnen hat, sondern auch in anderen Bereichen. Einer dieser Bereiche ist logischerweise die Forschung rund um die Interaktion zwischen Mensch und Computer. Wie die Arbeit von Pardos et al. (2022) zeigt, kann man Body-Tracking im Bereich Gesundheit und Fitness wiederfinden. In dieser Arbeit beleuchten die Autoren den Aspekt der Nutzung von Body-Tracking zur Analyse und Verbesserung der Körperhaltung. Hier werden Daten in Echtzeit analysiert und verwertet. Interessant ist auch die Möglichkeit, Zeichensprache durch Body-Tracking, genauer noch „Hand-Tracking“, zu Übersetzen. Das Paper von Rodriguez-Moreno et al. (2021) gibt einen Einblick auf ein Forschungsgebiet, das die Kommunikation für Menschen mit vermindertem oder nicht-vorhandenem Hörvermögen erleichtern soll. Bei der Recherche Rund um MediaPipe BlazePose stößt man auf verschiedenste Forschungsfelder mit weitreichenden Anwendungsgebieten. Diese können so einfach sein, wie eine

Smartphone-App zur Verbesserung der Körperhaltung. Auffallend häufig wird die Verarbeitung in Echtzeit genannt. Die Problematik, die in dieser Arbeit behandelt wird beschäftigt sich jedoch mit der Aufzeichnung von Body-Tracking Daten. Es ist aus verschiedenen Gründen nicht gewollt, die gewünschten Daten in Echtzeit zu analysieren. Seit der Veröffentlichung von MediaPipe BlazePose (V. Bazarevsky et al., 2020) sind jedoch keine nennenswerten Lösungen veröffentlicht worden. In der Modellbeschreibung wird dies auch von den Entwicklern ausgesagt: „This model is optimized for real-time performance on a wide variety of mobile devices, [...]“. MediaPipe wurde zur Nutzung in Echtzeit optimiert. Die Folgenden Kapitel öffnen den Blick auf eine Methode, die MediaPipe BlazePose nutzt um 3D Body-Tracking Daten zur späteren Verwendung aufzuzeichnen.

3 Grundlagen

Um ein besseres Verständnis dafür zu schaffen, worum es sich hierbei handelt, beginnt dieses Kapitel mit der Erklärung was 3D Body-Tracking im allgemeinen ist, gefolgt von Definitionen zu wichtigen Begriffen welche im Laufe der Arbeit genannt werden.

3.1 3D Body-Tracking

3D Body Tracking bezieht sich auf die Verwendung von Technologie, um die Bewegungen und Haltung eines menschlichen Körpers im dreidimensionalen Raum zu erfassen und zu verfolgen. Es kann verwendet werden, um virtuelle Charaktere oder Objekte in einer computergenerierten Umgebung zu steuern, oder auch als Eingabe für andere Anwendungen im Bereich Gaming oder Fitness-Tracking genutzt werden. Es gibt verschiedenste Technologien und Methoden, die zu diesem Zweck genutzt werden können, diese umfassen unter Anderem Tiefenkameras, Infrarot-Sensoren, sowie Marker-basierte und markerlose Methoden. Tiefenkameras verwenden Lichtprojektion und optische Sensoren, um die Entfernung von Objekten im Sichtfeld zu messen. Infrarot-Sensoren verwenden Infrarotlicht, um die Bewegungen des Körpers zu verfolgen. Marker-basierte Methoden verwenden spezielle Marker, die an bestimmten Stellen des Körpers befestigt werden, um die Bewegungen zu verfolgen, während markerlose Methoden die Bewegungen des Körpers verfolgen, ohne dass spezielle Marker verwendet werden müssen.

Einige Anwendungsgebiete von 3D Body Tracking sind Virtual Reality, Augmented Reality, Motion Capture für Animationen in Filmen beispielsweise, Mensch-Computer Interaktion, medizinische Diagnose und Therapie, Sport- und Fitness-Training, und viele mehr. (Liu, 2022a)

Heutzutage ist 3D Body Tracking ein weit verbreitetes Werkzeug in der Forschung und durch kontinuierliche Verbesserung dieser Technologie werden ständig Fortschritte in Bezug auf die Genauigkeit, die Geschwindigkeit und Zuverlässigkeit erzielt.

Nach der Betrachtung der gebotenen Möglichkeiten, nahm ich für mich eine grobe Unterteilung in zwei Kategorien vor: Hardwaregestütz und Softwaregestützt. Da man für beides jeweils mindestens eine Kamera, bzw. Optischen Sensor und einen Computer mit lauffähiger Software benötigt, ist diese Unterteilung zu Beginn verwirrend, aber sinnvoll. Hardware-gestütztes 3D Body-Tracking nutzt zur Tiefenerkennung eine speziell für diese Anwendung entwickelte Sensorik wie Stereokameras und im Hintergrund ein künstliches neuronales Netz zur Erkennung von Körpern. Software-gestütztes 3D Body-Tracking hingegen kann aber auch mit „herkömmlichen“ Kameras arbeiten, da die Tiefenerkennung im Hintergrund durch die Software durchgeführt

wird. Diese Methode wird auch „Monocular 3D Pose Estimation“ genannt. (Liu, 2022b)

Der allgemeine Ablauf aller 3D Body-Tracking Methoden ist in einer Hinsicht aber nahezu identisch. Zuerst müssen die Hardwarebedingungen erfüllt sein, das heißt man benötigt einen Computer auf dem die Softwareprozesse ablaufen können, einen optischen Sensor dessen Output von der Software verarbeitet wird und je nach gewünschtem Ergebnis entweder ein Peripherie-Gerät um die Aufnahme des optischen Sensors darzustellen oder ein Speichermedium zum aufzeichnen der Daten.

3.2 Begriffe und Definitionen

Ambient Displays

Als Ambient Displays bezeichnet man oft interaktive Großbildschirme welche im (halb-) öffentlichen Raum auffindbar sind. Schneider (2011) erklärt in seiner Definition, dass sich Ambient Displays im Alltag integrieren. Das heißt, sie würden nicht direkt wahrgenommen werden, denn im Mittelpunkt stünden die Informationen welche sie darstellen, vielmehr als die einzelnen Bildschirme als solche.

SDK

Ein SDK, „Software Development Kit“, besteht aus mehreren Tools, die (typischerweise) vom Hersteller einer Hardwareplattform, eines Betriebssystems oder einer Programmiersprache bereitgestellt werden.

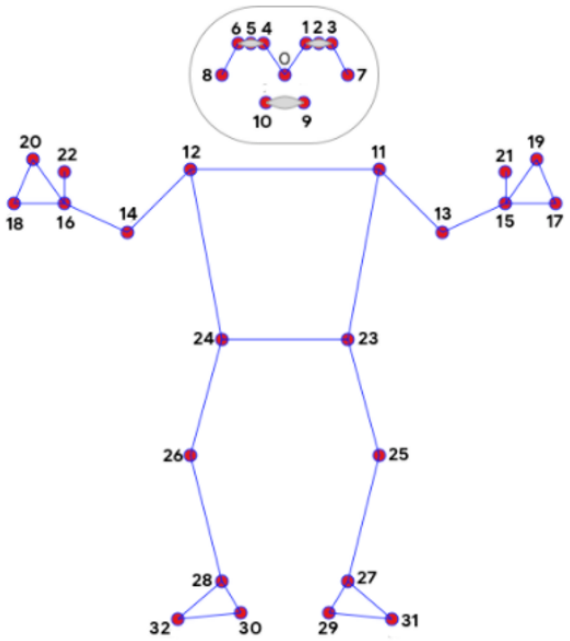
Convolutional Neural Network

Ist in dieser Arbeit von einem künstlichen Neuronalen Netz die Rede, dann ist diese besondere Art der künstlichen Neuronalen Netze gemeint. In der Forschung gibt es für die verschiedensten Arten von Problemen auch die verschiedensten Lösungen. Einige dieser Lösungen basieren auf künstlichen Neuronalen Netzwerken. Ein „gefaltetes Neuronales Netz“ (auf englisch „Convolutional Neural Network“, ist eine besondere Form des künstlichen Neuronalen Netzwerks, da es mehrere Faltungsschichten besitzt. Diese Eigenschaft macht es zu einer wertvollen Ressource für maschinelles Lernen sowie in Anwendungsgebieten mit künstlicher Intelligenz im Bereich der Bild- und Spracherkennung. (Dipl.-Ing. Luber & Litzel, 2019)

Keypoints und Körpermodell

Standards und Konventionen sind essentiell, um bestimmte Vorgänge zu vereinfachen und zu vereinheitlichen. Im 3D Body-Tracking werden hierfür bestimmte Eigenschaften definiert um möglichst konstante Ergebnisse zu erzielen. Die Begriffe „Keypoints“ oder „Körperpunkte“ werden in diesem Zusammenhang immer wieder erwähnt und haben die selbe Bedeutung. Sie

bezeichnen im Kontext des 3D Body-Trackings jene Schlüsselpositionen auf einem durch den Trackingvorgang erkannten Körper, aus denen sich ein bestimmtes Körpermodell zusammensetzt. Je nach Körpermodell variieren die unterschiedlichen Positionen der Keypoints und deren benötigte Anzahl zum vervollständigen des Körpermodells. Das Körpermodell besteht jedoch nicht ausschließlich aus diesen Punkten, da es leichter wird einen Körper zu erkennen, wenn man spezifische Punkte innerhalb der jeweiligen Modelle miteinander verbindet, sodass sogenannte „Knochen“ entstehen. Diese Knochen lassen sich nur dann darstellen, wenn seine beiden Endpunkte dargestellt werden können. In jedem Körpermodell muss auch ein Referenzpunkt bestimmt werden, nach dem sich alle anderen Punkte ausrichten. Je nach Komplexität eines Körpermodelles, kann dieses mehr oder weniger Informationen beinhalten. Zum Beispiel braucht ein Modell eine Mindestanzahl „X“ an Punkten, um einen Menschlichen Körper darstellen zu können. Diese Zahl X muss je nach gewünschtem Ergebnis bzw. nach den Anforderungen, die man an ein Körpermodell haben kann größer oder kleiner sein. Wenn man sich vornimmt eine Person in fünf Sekunden auf ein Blatt Papier zu skizzieren, kann man eventuell am Ende ein Strichmännchen sehen, mit Kopf, Armen, Torso und Beinen. Da es ohnehin schwierig ist, die Ausrichtung des Körper auf diese Weise zwei-Dimensional darzustellen braucht man mehr Charakteristika. Gibt man sich also etwas mehr Mühe, skizziert man auch Füße und vielleicht Hände mit Fingern und ein Gesicht. Sieht man das Gesicht, schaut die Person nach vorne, sieht man keines, blickt man logischerweise auf die Darstellung eines Hinterkopfes. Möchte man also im 3D Body-Tracking Informationen erfassen, die Ausschlaggebend sind für die Ausrichtung einer Person, deren Körperhaltung und die eventuelle Blickrichtung, dann muss man sich eines Körpermodells bedienen welches eine gewisse Anzahl an Merkmalen, darunter beispielsweise die Hände und Füße beinhaltet.



0. Nase
1. Linkes Auge innen
2. Linkes Auge
3. Linkes Auge außen
4. Rechtes Auge innen
5. Rechtes Auge
6. Rechtes Auge außen
7. Linkes Ohr
8. Rechtes Ohr
9. Mundwinkel links
10. Mundwinkel rechts
11. Linke Schulter
12. Rechte Schulter
13. Linker Ellenbogen
14. Rechter Ellenbogen
15. Linkes Handgelenk
16. Rechtes Handgelenk
17. Kleiner Finger links
18. Kleiner Finger rechts
19. Mittelfinger links
20. Mittelfinger rechts
21. Daumen links
22. Daumen rechts
23. Hüfte links
24. Hüfte rechts
25. Knie links
26. Knie rechts
27. Fußgelenk links
28. Fußgelenk rechts
29. Ferse links
30. Ferse rechts
31. Zehenspitzen links
32. Zehenspitzen rechts

Abbildung 1: Beispiel eines von Google-LLC (2023b) entwickelten Körpermodells. Auf der rechten Seite, sinngemäß von mir übersetzt, steht eine Beschreibung der jeweiligen Keypoints mit deren Nummerierung.

Dargestellt wird das in MediaPipe genutzte Körpermodell, seine genaue Bezeichnung ist „Pose Landmark Model“ oder „BlazePose GHUM 3D“ *(Generative 3D Human Shape)*. Was genau MediaPipe ist und wieso speziell dieses Modell im weiteren Verlauf relevant ist, erkläre ich im Abschnitt 3.4 über MediaPipe. Auf der Abbildung 1 kann man durchnummeriert von 0 bis 32 die Körperpunkte des Modells ausmachen. Verbindet man diese (dargestellt durch die Blauen Linien) erhält man die Knochen. Der Referenzpunkt, an dem sich die restlichen Punkte ausrichten sollen ist in diesem Fall nicht grafisch dargestellt. Es gilt hier, dass man als Referenz die Mitte des Hüftknochens verwendet. Des weiteren ist ersichtlich, dass das Modell Aufschluss über die Ausrichtung des Körpers der Person liefern kann. Auch wenn es Körpermodelle gibt die detailreicher sind und mehr Körperpunkte umfassen wie zum Beispiel das MediaPipe Holistic Model von Google (2023b), ermöglicht es einem diese Abbildung sich eine genauere Vorstellung zu machen, wovon bei dem Begriff Körpermodell die Rede ist.

Die Körperpunkte eines Körpermodells verändern sich nicht. Möglich wäre es, das kNN neu zu trainieren um neue Punkte hinzuzufügen, jedoch spielen weder Hardware noch Software eine Rolle dabei, wie ein Körpermodell aussieht. Es wird von vorne rein festgelegt und ändert sich

in der Regel nicht. Dieser Aspekt ist ein weiterer Grund, warum es interessant sein könnte ein neues Format mit einem anderen Körpermodell zu implementieren.

3.3 Bisheriger Ansatz: Stereolabs und die ZED 2 Kamera

Im Jahr 2015 veröffentlichte die Firma Stereolabs eine Kamera mit deren Hilfe man einen dreidimensionalen Raum aufzeichnen und rekonstruieren kann. Die Baureihe trägt den Namen ZED und verfügt im Gegensatz zu herkömmlichen Kameras über zwei optische Sensoren mit deren Hilfe man eine gute Qualität in der Tiefenaufnahme erreichen kann. Laut Stereolabs (2022a) werden diese Kameras vornehmlich auf den Gebieten der Robotik, autonomes Fahren oder in Dronen eingesetzt und seit 2015 haben schon mehr als 90.000 Entwickler mit den Kameras gearbeitet oder auch bei der Weiterentwicklung dieser mitgewirkt. In der Forschung um Ambient Displays kann man diese Kameras nutzen, um Bewegungen der Nutzer aufzuzeichnen und diese Aufzeichnungen zu analysieren. Mittlerweile gibt es das Modell ZED 2i welches im Forschungsprojekt HoPE der Universität der Bundeswehr München am Institut für Softwaretechnologie (2022) benutzt wird. Ausschlaggebend für die Wahl dieser Kamera waren mitunter die Erkenntnisse eines Projektstudiums von Keineke (2021). Entscheidende Faktoren waren dabei unter anderem die Größe, das Gewicht und die Konnektivität der zur Auswahl stehenden Kameras. Zunächst werde ich die Funktionsweise der Kamera erläutern, um danach auf das Datenformat einzugehen das benutzt wurde um Datensätze abzuspeichern.

3.3.1 ZED 2 Kamera und Body-Tracking - Funktionsweise

Der Fokus beim Body-Tracking Modul von Stereolabs liegt darauf die einzelnen Knochen zu erkennen und zu verfolgen. Ein Knochen wird, wie in 2.2.2 erläutert, anhand seiner beiden Endpunkte - den Keypoints festgelegt. Es ist mit der ZED Kamera möglich 2D und 3D Datensätze der Keypoints zu extrahieren. Weitere nützliche Informationen die erhalten werden können sind sowohl die relative Position der Keypoints im Bezug auf die Umgebung, als auch die Geschwindigkeit mit der diese sich im Raum bewegen. Sobald die Sensoren der Kamera eine Person erfasst haben und das künstliche neuronale Netzwerk des ZED SDK Moduls die Körperpunkte als solche definiert hat, kann ein finaler Datensatz mit vordefinierten gewünschten Informationen ausgegeben werden.

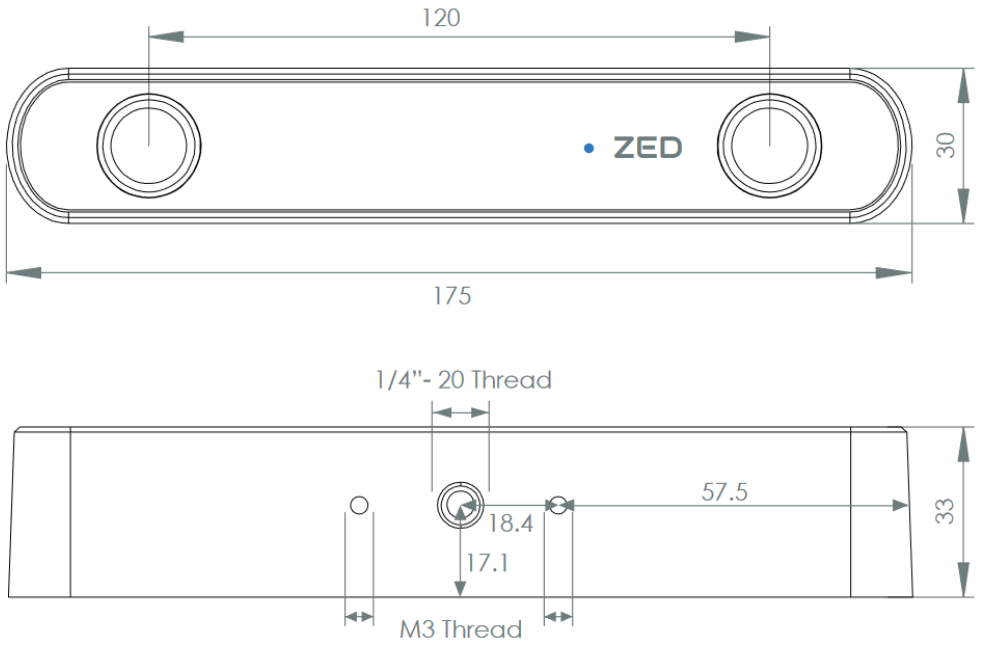


Abbildung 2: ZED 2 Kamera - Technische Zeichnung (Stereolabs, 2022a)

Der Vorteil den diese Methode im Vergleich zu anderen Methoden bietet ist die speziell für diesen Zweck entwickelte Hardware. Den Gedanken, einen zweiten optischen Sensor zur Tiefenaufnahme zu nutzen, kann man sich daraus herleiten wenn man überlegt, dass wir Menschen auch nur deshalb über ein gutes räumliches Sehvermögen verfügen, wenn wir im besten Fall zwei gesunde Augen haben. Es ist zwar so, dass wir unsere Umwelt räumlich wahrnehmen können, jedoch ist der Prozess der dafür sorgt für uns Menschen nicht wahrnehmbar. Einem Computerprogramm bzw. einem künstlichen neuronalen Netz muss diese Fähigkeit zunächst antrainiert werden, indem ihm wiederholt Daten zur selbstständigen Verarbeitung vorgelegt werden. (Fraunhofer-IKS, 2023)

Grundlegend hierfür müssen zwei verschiedene Bilder der selben Szene aus zwei unterschiedlichen Blickwinkeln aufgenommen werden können. Diese werden dann von der KI miteinander Verglichen und anhand der Unterschiede in den Winkeln und der Distanz zu einem erfassten Punkt ist es möglich eine ungefähre räumliche Tiefe zu berechnen. Möchte man also 3D Body-Tracking mit einem Kameramodell der ZED Reihe betreiben können, ist die nächste Voraussetzung nur noch über die von Stereolabs bereitgestellte Software zu verfügen und diese in das eigene Programm zu implementieren. Betrachtet man das nachstehende von mir übersetzte Diagramm, wird das Konzept noch einmal deutlicher vor Augen geführt:

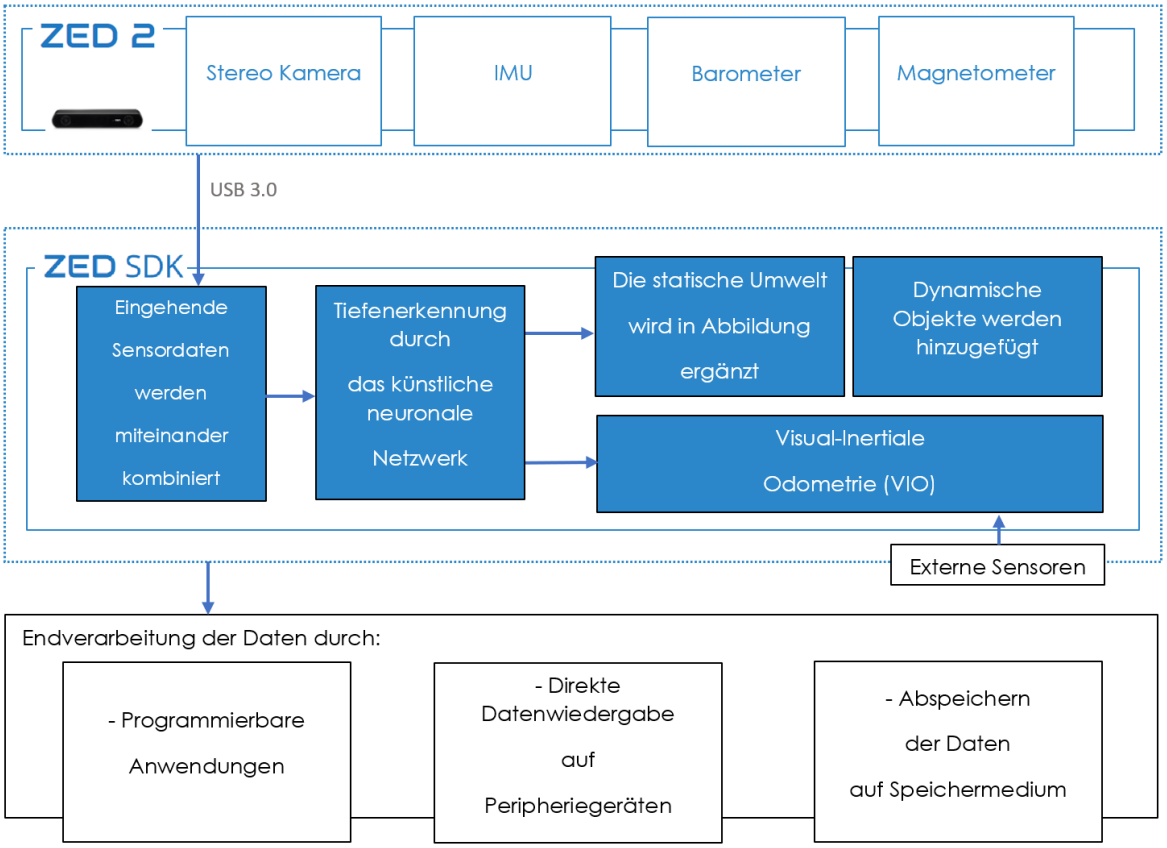


Abbildung 3: Funktionales SDK Diagramm, Nachempfunden wurde dieses dem Original von Stereolabs (2022b). Hier werden die Zusammenhänge zwischen der Sensorik und der Body-Tracking Software durch das deutliche Abgrenzen beider Bereiche dargestellt. Die Endverarbeitung der Daten ist genauso abgegrenzt, da diese je nach Zweck des Programms angepasst werden kann.

Das essentielle Hardwaremodul, bestehend aus der Stereokamera samt zusätzlicher Sensorik, wird hier als ZED 2 zusammengefasst. Die Datenübertragung erfolgt über eine USB 3.0 Schnittstelle. Die Datenverarbeitung erfolgt über die Software welche als ZED SDK bezeichnet wird: Die eingehenden Sensordaten werden kombiniert, das künstliche neuronale Netzwerk führt die Tiefenerkennung durch und die restlichen Bilddaten werden ergänzt. Zum Schluss kann man den erhaltenen Datensatz seiner finalen Verwendung zuführen. Wie so ein Datensatz aussieht, hängt davon ab, welche Daten beim Ausführen der Software generiert werden sollen und ist deshalb variabel und abhängig vom Programmierer der den Softwarecode schreibt.

Output

Der bisherige Stand an den mit diesem Projekt angeknüpft wird ist oben technisch beschrieben. Ein mögliches Ergebnis kann man auf der nachfolgenden Grafik betrachten. Die Abbildung enthält zwei Screenshots, die während der Ausführung des lauffähigen Programms von Julian Fietkau entstanden sind. War ein Durchlauf erfolgreich, erhält man eine 3D Animation in Länge der Aufnahme welche mittels ZED Kamera erfolgte. Während dieser Aufnahme wurden alle irrelevanten Umweltinformationen zwar sensorisch erfasst, jedoch nicht aufgezeichnet, sodass man am Ende im 3D Koordinatensystem nur die aufgezeichneten Bewegungen der im Vorgang verfolgten Person computeranimiert wiedergeben kann.

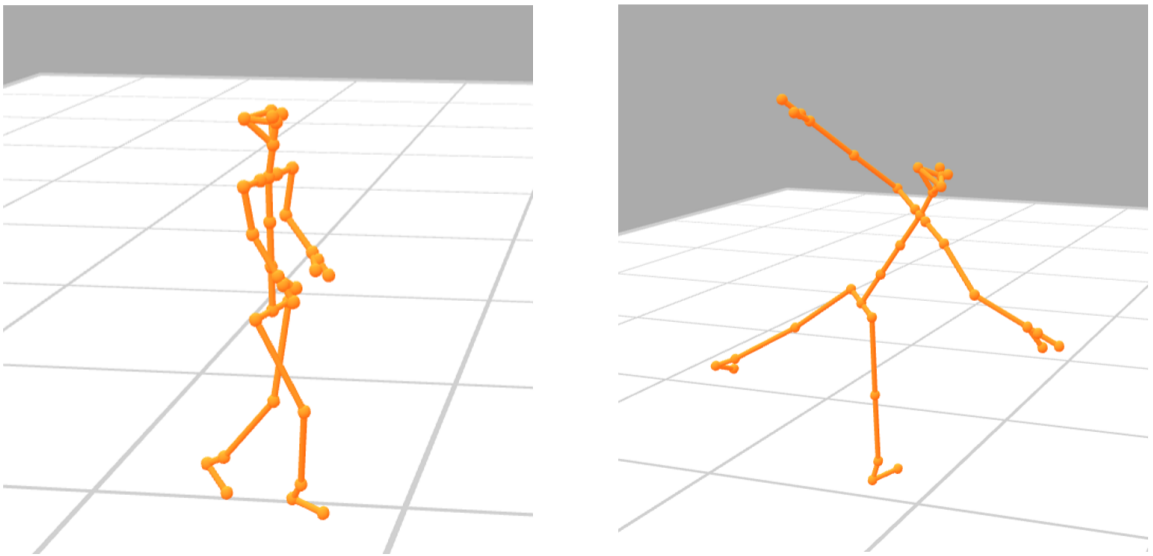


Abbildung 4: ZED Output - Animation der Aufgenommenen Datensätze in PoseViz dargestellt

Was genau wird abgebildet? Die Orangefarbenen Punkte und Verbindungselemente sind jeweils Körperpunkte und die Knochen die sie verbindet. Die weißen „Kacheln“ stellen den Boden dar: im 3D-Koordinatensystem (x, y, z) wäre dies die Ebene (x, y, 0). Der graue Hintergrund ist leerer Raum und für die Abbildung nicht relevant, dieser könnte jede andere Farbe haben oder sogar eine virtuell erstellte Umgebung widerspiegeln. Es wäre auch möglich, die am Aufnahmeort befindliche Umgebung darzustellen - jedoch ist diese Information irrelevant für den Ablauf. Die humanoide Figur ist eine 3D-Animation der Person, die aufgezeichnet wurde. Die Originale Aufzeichnung dauerte 40 Sekunden und umfasst ca. 53.000 Zeilen in denen die jeweiligen Körperpunkte mitsamt deren zum Zeitpunkt ihrer Entstehung aufgezeichneten Zeitstempel gespeichert sind. Für jeden der 34 Körperpunkte wird hier eine separate Zeile verwendet, plus eine Zeile mit deren Zeitstempel und eine weitere Zeile welche die Zusatzinformationen bezüglich

Referenzpunkt für das Modell und Orientierung speichert: man benötigt 36 Zeilen pro Frame. Um diese Erklärung zu verdeutlichen ist nachfolgend ein Ausschnitt aus einer solchen Datei zu sehen.

```
p 0 0.056523,0.531211,2.210657
cf 100
tst OK
ast IDLE
gro -0.020828,0.988744,-0.080947,-0.124096
v -0.003836,0.00896,0.05385
dim 0.597661,1.319372,0.597662
kg 0 0.025211,0.498701,2.219008
kg 1 0.034087,0.355879,2.241445
kg 2 0.042962,0.213057,2.263882
kg 3 0.052851,0.070277,2.286164
kg 4 0.088021,0.071378,2.277612
kg 5 0.202174,0.074952,2.249852
kg 6 0.222517,0.287998,2.255474
kg 7 0.242633,0.492723,2.26315
kg 8 0.246657,0.533668,2.264685
(...)
kg 28 0.078514,-0.131626,2.299824
kg 29 0.133291,-0.103901,2.359276
kg 30 0.036192,-0.131564,2.309676
kg 31 0.013378,-0.103725,2.387188
kg 32 0.145102,1.216362,2.154867
kg 33 0.004279,1.207192,2.144626
```

Listing 1: Dargestellt wird hier der Datenstrom, der alle relevanten Rohdaten zur Wiedergabe einer 3D Animation enthält. Jedoch würde man mit nur diesem Ausschnitt genau einen Frame erhalten, also die Reproduktion einer Momentaufnahme der Kamera.

Die mit `p` beginnende Zeile beschreibt welche der erfassten Personen durch die darauf folgenden Körperpunkte dargestellt wird und enthält die Koordinaten vom Referenz-Keypoint.

Darauf folgen dann Zeilen, die Auskunft über die Ausrichtung des Körpers und der Tracking- und Detection-Confidence geben.

Der wichtigste Teil besteht aber aus den mit `kg` beginnenden Zeilen: hier werden die Keypoints mit ihrer zugehörigen Nummer erfasst.

Dateiformat der Ausgabe

Ausgehend davon, dass der Setup vollständig und lauffähig ist, kann man in einfacheren Worten erklären: Befindet sich eine Person im Aufnahmebereich des optischen Sensors und wird erfasst, gibt das dem Programm die Möglichkeit diese zu erkennen. Ist die Person als solche erkannt worden, wird das Körpermodell auf besagte Person projiziert. Dadurch werden zu einem bestimmten Zeitpunkt im 3D-Raum bestimmte Koordinaten generiert. Das sind die für

den weiteren Verlauf wichtigen Rohdaten. Diese Rohdaten müssen durch das Aufnahmetool in der Reihenfolge ihrer Entstehung zusammen mit ihrem Zeitstempel abgespeichert werden. Sind diese Daten abgespeichert, kann man sie im Nachhinein mit einer anderen Software entweder weiterverarbeiten oder abspielen. Das Abspielen geschieht durch einfaches positionieren der Koordinaten in einem 3D-Koordinatensystem im Falle einer Momentaufnahme. Möchte man ein ganzes Video wiedergeben, muss man mehrere konsekutive Momentaufnahmen nacheinander darstellen.

Nüchtern betrachtet gibt es also kein spezielles Dateiformat: Das was hier verwendet wird ist eine Textdatei, angepasst an die Anforderungen von PoseViz[1], die Software mit der man besagte Aufnahmen abspielen möchte. Deshalb ist es technisch machbar, das Programm in dem Sinne zu erweitern, dass nicht nur Aufnahmen mit der ZED Kamera abgespielt werden können, sondern auch andere Aufnahmen, vorausgesetzt dass gewisse Anforderungen im Format erfüllt sind. Diese Anforderungen werden durch das Programm welches die extrahierten Datensätze umwandeln soll vorgegeben und können jederzeit angepasst werden.

3.4 Warum MediaPipe BlazePose? Was ist das?

Ein Team von Entwicklern der Firma Google stellte am 22. Juni 2021 MediaPipe BlazePose GHUM 3D vor. (Bazarevsky, 2022)

BlazePose ist sogenanntes Pose-Detection-Modell, charakterisiert durch 33 Körperpunkte welche auf einen erkannten Körper projiziert werden können. Im wesentlichen unterscheidet es sich also in dieser Hinsicht nicht von den beiden Stereolabs Körpermodellen. Dennoch ist die Kompatibilität ein wichtiger Faktor, den man bei der Entwicklung von Software nicht außer Acht lassen darf, wenn man die eigene Software zugänglicher gestalten möchte. Im Vergleich : Die von Stereolabs entwickelte ZED Kamera bedient sich einer Software, welche direkt mit der ZED Kamera kompatibel ist. Dies kann zwar einschränkend wirken, da eine handelsübliche Webcam oder die in einem Laptop fest verbaute Kamera (im Gegensatz zur ZED Kamera) lediglich über einen optischen Sensor verfügt und daher nicht zum 3D Body-Tracking mit Stereolabs genutzt werden kann. Die Nutzung eines zweiten optischen Sensors wirkt sich positiv auf die Qualität der Aufnahmen aus, weil der Input beider Sensoren im Hintergrund der Aufnahme verrechnet wird, um so die Tiefe des Bildes bestmöglich zu erfassen. Der gebotene Vorteil an diesem Punkt ist also nicht nur, dass die Software direkt kompatibel mit der erforderlichen Hardware ist, sondern auch die Verfügbarkeit eines weiteren Sensors, wodurch Fehler beim Tracking reduziert werden können. Mit dem Gedanken, diesen Vorteil auszugleichen und die Erforderlichkeit eines zweiten optischen Sensors zu umgehen, bedient sich MediaPipe BlazePose eines künstlichen neuronalen Netzes, welches im Hintergrund dann einen Ausgleich schafft. Dies hat zur Folge, dass die Tiefenerkennung, trotz Nutzung herkömmlicher Kameras welche nur über

[1] PoseViz wurde von Julian Fietkau entwickelt um 3D Animationen von 3D Body-Tracking Datensätzen abspielen zu können `https://poseviz.com`

einen einzelnen optischen Sensor verfügen, keine Hürde mehr bei der Körpererkennung darstellen muss. Es kann es also als logische Konsequenz gesehen werden, dass die Implementierung von BlazePose eine erhöhte Flexibilität der Software zur Folge hat.

3.4.1 MediaPipe BlazePose und Body-Tracking - Funktionsweise

BlazePose vereint die Topologie von drei verschiedenen, bereits bekannten Formaten des Body-Trackings: BlazeFace, BlazePalm und COCO. Laut Entwicklern ist BlazePose dank der leichten CNN-Architektur besonders gut für Echtzeitaufnahmen geeignet. (V. Bazarevsky et al., 2020)

Das Grundkonzept beim Body-Tracking mit MediaPipe bleibt das gleiche wie bei Stereolabs mit der ZED Kamera. Der Unterschied liegt hier in der Software im Hintergrund, die zusätzliche Berechnungen ausführt um den fehlenden Tiefensensor auszugleichen. An dieser Stelle ist ein wichtiges Werkzeug das Machine Learning Kit von Google.

Google's ML Kit Pose Detection API

„Machine Learning“ (zu deutsch maschinelles Lernen) kann dabei helfen, komplexe Probleme ganz einfach zu lösen : mit Hilfe der Pose Detection API zum Beispiel kann in Echtzeit die Position einer Person aus einem Video oder auf einem statischen Bild erkannt werden. Erfasst diese API eine Person auf einem statischen Bild, wird ein Ganzkörperskelett mit 33 Punkten erzeugt. Diese Struktur bildet das Körpermodell und umfasst sowohl die Gesichtsmerkmale wie Ohren, Augen, Nase und den Mund, als auch Hände und Füße einer Person. Laut Entwicklern ist keine spezielle Ausrüstung oder ML-Kenntnis erforderlich, um mit der Software arbeiten zu können und obwohl das beste Ergebnis erzielt wird, wenn die Kamera einen vollständigen Körper aufnehmen kann, reicht es auch aus, wenn zunächst nur das Gesicht erkannt wird, um eine Pose zu erstellen. (Google-LLC, 2023a)

Interessant ist es auch an dieser Stelle anzumerken, dass das Problem des Monokularen 3D Body-Trackings schon vor weit mehr als einem Jahrzent erforscht wurde und erst mit den technologischen Fortschritten auf dem Gebiet der künstlichen Intelligenz und Machine Learning zufriedenstellende Ergebnisse erzielt werden konnten. (Plagemann, 2010)

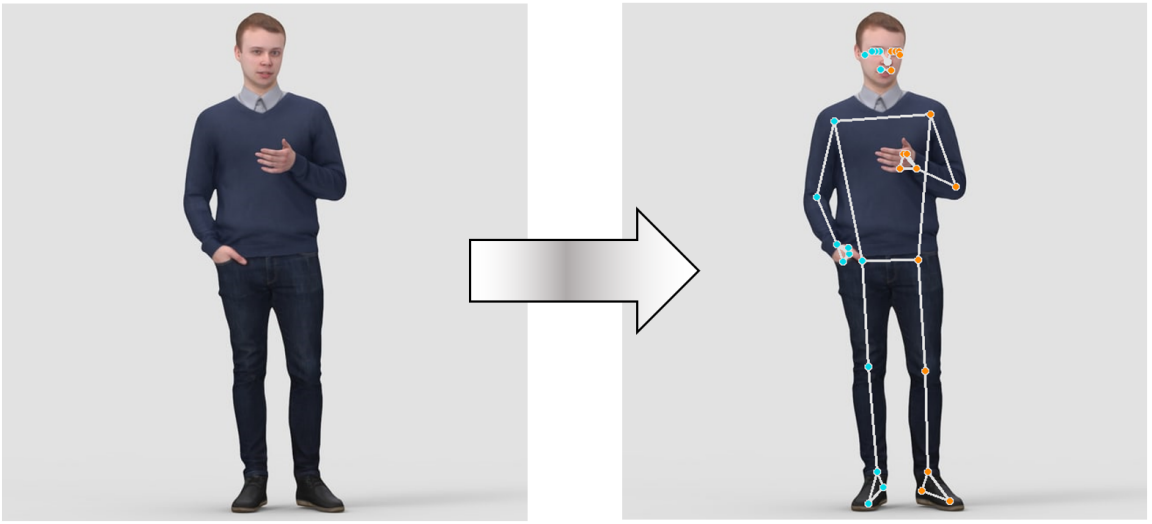


Abbildung 5: Bildverarbeitung mit MediaPipe BlazePose

Hier zu sehen : Als Input wurde dieses Bild einer Person eingespeist, der Output ist das zweite oben aufgeführte Bild. Die Software hat die Person als solche erkannt und als Folge dessen eine Kopie erstellt, auf die eine Pose, das Körpermodell projiziert wurde.

Aufgrund der Beschaffenheit dieser API, kann diese auch Plattform-übergreifende Lösungen für Android und iOS bieten, das bedeutet, dass sie vielseitig Einsetzbar ist und eine erhöhte Kompatibilität im Vergleich zu anderen Programmen aufzeigt. Somit wird die Frage „Warum BlazePose?“ damit beantwortet, dass obwohl die Funktionsweise des Body-Trackings mittels ZED 2 Kamera im Hintergrund ähnlich funktioniert und die Kamera einen Vorteil in der Hardware aufweist, ist man bei der Herangehensweise mit MediaPipe BlazePose nicht darauf angewiesen, eine spezielle Kamera zu verwenden - es reicht zum Beispiel auch eine Handelsübliche Webcam oder die Frontkamera eines Smartphones aus um zufriedenstellende Ergebnisse bei der Körpererkennung zu erzielen. (Google, 2022)

3.4.2 MediaPipe BlazePose in der Praxis

Zusätzlich zur Einführung in die Welt von MediaPipe und BlazePose ist zu erwähnen, dass es Dank der vielseitigen Anwendungsmöglichkeiten von MediaPipe mit wenig Aufwand möglich ist, verschiedene Arten von Programmen zu erstellen und diese mit der selben Hardware zu nutzen. Es ist zum Beispiel möglich ein Programm zu schreiben, welches die Körperpose von Personen auf statischen Bildern erzeugt. Verändert man einige Parameter, kann man auch vorhandene Videos von Personen einspielen und analysieren indem die Pose über einer im Video erkannten Person erzeugt wird, während man dieses abspielt. Der erzeugte Output beinhaltet 33 vordefinierte Körperpunkte mit 3D Koordinaten (x, y und z) und deren Sichtbarkeit auf

einer Skala zwischen [0.0 und 1.0]. Hierbei wird im Gegensatz zu anderen Methoden etwa nur die minimal benötigte Anzahl an Körperpunkten zum Tracking und zur Abschätzung der Körperausrichtung verwendet. Implementiert man den Beispielcode der „Python Solution API“, wird ein in Echtzeit über die Kamera aufgenommener Stream doppelt ausgegeben: Zum einen die unbearbeitete Aufnahme der Kamera als Video-Stream und daneben die gleiche Aufnahme mit dem Zusatz, dass erkannte Körperpunkte über eine Spiegelung des Eingangs-Streams projiziert werden. (O'Connor, 2022)

Der Fokus in dieser Arbeit liegt letztendlich darin, einen Ansatz zu finden, unabhängig von der genutzten Hardware oder Software, Personen die sich vor der Kamera befinden zu erkennen und aufzunehmen. Aus Datenschutzgründen sollen diese Aufnahmen aber weder die Person als solche, noch die Aspekte der Umwelt wiedergeben: lediglich das während der Aufnahme erzeugte Pose-Modell wird aufgezeichnet. Genauer noch liegt die Begründung hierfür darin, dass es der Zustimmung einer jeden Person bedarf, wenn man diese per Video aufzeichnen will (Hintergrundinformationen erklärt der §4 des Bundesdatenschutzgesetzes in der DS-GVO (ICS-AG, 2019)). Weiterhin ist es wichtig, diese Aufnahmen in einem einheitlichen Format zu speichern um die Auswertung zu vereinfachen. Daher muss das Datenformat welches über MediaPipe generiert wird an das der initialen Lösung angeglichen werden.

Standards und Konventionen rund um MediaPipe BlazePose

Entwickelt wurde das System auf der Basis, 3D Body-Tracking in Echtzeit betreiben zu können. In Echtzeit heißt, dass es nicht Ziel ist, abspielbare Aufnahmen zu erstellen. Die Daten werden im Moment ihrer Entstehung direkt verarbeitet und zumeist einfach visuell dargestellt. Bei der Recherche im Bezug auf MediaPipe BlazePose und vorgeschlagener Lösungen zur Nutzung, stellen die Entwickler also keine einheitliche Lösung vor dies umzusetzen.

Demnach ist einer der Schritte zur Implementierung dieses neuen Formats, das Extrahieren der Echtzeit-Rohdaten die während der Aufnahme entstehen und ein weiterer Schritt besteht aus dem Abspeichern gemäß der Konventionen, die bereits im Projekt etabliert sind.

4 Anforderungsanalyse

Die Anforderungen werden hier in zwei größere Kategorien unterteilt werden: Die Anforderungen an die zu entwickelnde Software und die Anforderungen der Software an ihre Umgebung. Beginnend mit den Anforderungen an die Software wird zunächst geklärt, was die wesentliche Leistung vom Programm erfüllt werden soll. Anschließend werden die Rahmenbedingungen festgelegt unter denen die Software entwickelt und ausgeführt wird.

4.1 Funktionalität

Nachdem der technische Stand erklärt ist und man weiß welche Abläufe im Projekt gewünscht sind werden nun die Anforderungen für den Implementierungs-Abschnitt klar definiert. Im „Ist-Zustand“ des Projektes ist es möglich 3D Body-Tracking Aufnahmen zu erstellen und wieder abzuspielen. Gebunden ist man dabei jedoch an die verwendete ZED Kamera. Diese spezielle Hardware ist mit Kosten verbunden, die vermieden werden können, wenn ein bestimmter Teil der Prozedur (wie in Kapitel 3 erklärt, handelt es sich um die Tiefenerkennung) durch die Software erledigt werden würde.

Dadurch, dass es sich bei der Arbeit um einen sogenannten „Artefakt-Beitrag“ handelt, kann man zwei Anforderungen an die Software klar hervorheben: Die erste Anforderung ist, eine 3D Body-Tracking Aufnahme erstellen zu können ohne sich einer speziellen Kamera oder Sensorik zu bedienen. Daraus folgt die zweite Anforderung: Diese neuen Aufnahmen sollen mit dem selben Programm abgespielt werden können, das zum abspielen der Aufnahmen der ZED Kamera verwendet wird: PoseViz. Die beabsichtigte Leistung ist also eine Unabhängigkeit zur Hardware zu schaffen.

Bei Ausführung des Programms wird die angeschlossene Kamera gestartet und beginnt mit der Aufnahme. Während der Aufnahme, die mit bis zu 30 Bildern pro Sekunde läuft, entstehen in Echtzeit besagte 3D Koordinaten der Körperpunkten für eine einzelne Person und bilden für die gesamte Dauer das Pose-Modell nach BlazePose auf einem erkannten Körper ab. Wie im Beispiel der **Python Solution API**[1] könnte man nun einen Videostream ausgeben lassen, der die beschriebene Szene abspielt - was jedoch nicht Zielführend ist, wenn man eine im Nachhinein abspielbare Datei erzeugen möchte. Die gewünschte Datei lässt sich erzeugen, indem die während der laufenden Aufnahme vom Pose-Modell gebildeten Keypoint-Koordinaten der jeweiligen Körperpunkte zeitgleich mit dem Zeitstempel ihrer Erzeugung in einem serialisierten Byte-Strom

[1] Die Python Solution API ist die von Google vorgeschlagene Musterlösung zum testen von MediaPipe BlazePose

abgelegt werden. Wie das Ganze aussehen kann ist vorher schon im Codeausschnitt 1 ersichtlich gewesen und auf diese Weise sollen auch die mit MediaPipe erzeugten BlazePose-Rohdaten abgespeichert werden.

Software-seitig bestand für eine erfolgreiche Einbindung von MediaPipe BlazePose in das Projekt, die essentielle Bedingung eines Rohdaten-Exports, da diese in PoseViz verarbeitet werden müssen.

4.2 Hardwarekapazität und Betriebsumgebung

Um nachvollziehen zu können, mit welchen Hardwarebedingungen ich bei meinem Vorhaben gestartet habe nenne ich nachfolgend einige Eckdaten der von mir genutzten Hardware. Dies soll veranschaulichen, dass keine besonderen Geräte genutzt wurden. Zur Umsetzung, Implementierung und Durchführung meines Codes nutzte ich:

- Zwei 27"Monitore
- Einen Windows-PC mit 16 GB RAM, 6GB Grafikspeicher, 512GB SSD Speicher mit dem Betriebssystem Windows 11
- Sowohl eine High-End Logitech BRIO 4K Webcam für 259€, als auch eine vergleichsweise sehr günstige Logitech C270 für 39€

An dieser Stelle kann ich anmerken, dass ein einzelner Bildschirm auch ausgereicht hätte, dies hätte den Entwicklungsprozess aber erheblich verlangsamt. Vom heutigen Technikstand gesehen liegt die Rechenleistung des von mir genutzten PC-Systems im Mittelfeld, das heißt, es gibt erheblich bessere aber auch erheblich schlechtere. Voraussetzung für die Arbeit mit dem Code-Editor, sind laut Hersteller mindestens 4GB Arbeitsspeicher (empfohlen werden hier 8GB). Wie Anforderungen an einen PC aussehen müssten, um ein Programm mit der von mir Implementierten Software ausführen zu können ist derzeit noch nicht abschätzbar, dennoch werden die 8GB Arbeitsspeicher voraussichtlich nicht überschritten. Die von mir genutzten Kameras, vom gleichen Hersteller Logitech, verwenden keine besondere Software zur Nutzung. Diese werden beide über eine USB3.0 Schnittstelle mit dem PC verbunden und sind nach einer Treiberinstallation funktionsfähig. Ich verwendete hier zwei unterschiedliche Modelle, um die „Hardwareunabhängigkeit“ testen zu können. Die BRIO 4K Kamera kann hochauflösende 4K Aufnahmen erstellen und mit einer Geschwindigkeit von bis zu 60 FPS[2] aufnehmen, die zweite Kamera mit Modellbezeichnung C270 hingegen hat eine maximale Auflösung von 720P und nimmt mit 30 FPS auf. Inwiefern sich die Unterschiede beider Modelle auf das gewünschte Ergebnis auswirken, erläutere ich in Kapitel 6. Eine Anforderung, die meine Hardware erfüllen sollte und die durch beide Kameras erfüllt wurde, ist die Fähigkeit mit mindestens 30 FPS aufnehmen zu können.

[2]Frames Per Second

5 Umsetzung

5.1 Entwicklungsumgebung

Die Softwarekomponente in meiner Arbeit hat sich im Laufe des Projektes einmal geändert: Ich startete damit, eine Web-Anwendung zu Programmieren. Die hierfür genutzte Programmiersprache war JavaScript und die genutzte Plattform war „Webstorm 2022.3“ von JetBrains. Während meiner Recherche musste ich jedoch feststellen, dass die Quellen und die gefundene Literatur im Bezug auf MediaPipe meinen Ansatz nicht unterstützten, weshalb ich nach neuem Bewerten der Situation komplett neu Anfangen musste. Diesen neuen Ansatz verfolgte ich mit der Programmiersprache Python und dem Editor „PyCharm 2022.3“, auch von JetBrains. Für beide Ansätze existieren bereits sogenannte Solution APIs von Google: vorgefertigte Lösungsbausteine welche man in seine eigene Software implementieren kann. (Google, 2023a)

Die JavaScript Solution API kann in einem Webbrowser getestet werden: der folgende Link

`https://codepen.io/mediapipe/pen/jOMbvxw`

führt auf eine Seite mit dem Codebeispiel und einer Demonstration der Software in Echtzeit. Nach näherer Betrachtung der von Google vorgestellten Solution APIs zeigte sich, dass keiner der Programmausschnitte Eigenschaften aufwies um Rohdaten zu extrahieren und abzuspeichern. Um dieses Problem anzugehen wollte ich zunächst eine lauffähige Software erstellen, mit der man in Echtzeit 3D Body-Tracking durchführen kann. Der nächste Schritt war dann zu überprüfen, ob eine Erkennung stattfindet und ob das Körpermodell zuverlässig auf einen erkannten Körper projiziert wird. Wird das Modell auf einen erkannten Körper projiziert, bedeutet das, dass Keypoints generiert werden und somit auch deren 3D Koordinaten. Diese können somit extrahiert und abgespeichert werden.

5.2 Code im Detail

Dieser Abschnitt beinhaltet eine Aufstückelung des Quellcodes der Anwendung **mediapipe-recorder**[1]. Dieses Programm wurde auf dem Grundbaustein von MediaPipe entwickelt, um dessen Funktion zur Körpererkennung nutzen. Jeder der gezeigten Codeabschnitte ist bis auf weitere Updates der Software genau so im Originalprogramm zu finden. Nachfolgend zu jedem Ausschnitt ist eine Erklärung bezüglich seiner Funktionalität.

[1] Name des Artefaktbeitrags, Gegenstand dieser Arbeit

```
import cv2
import mediapipe as mp
import uuid
from datetime import datetime

def print_vec(vector):
return ' '.join(map(lambda c: "%0.5f" % c, vector))

mp_drawing = mp.solutions.drawing_utils
mp_drawing_styles = mp.solutions.drawing_styles
mp_pose = mp.solutions.pose
```

Listing 2: Der erste Code Abschnitt mit Imports und Initialisierung der Tracking-KI.

Der oben abgebildete Code-Ausschnitt zeigt die sogenannten Imports. Diese werden genutzt um Zugriff auf die Kamera zu erhalten und um das Grundgerüst von MediaPipe in meine Software einzubinden. Die Initialisierung von Variablen um die Lesbarkeit des restlichen Codes zu verbessern findet in den letzten drei Zeilen statt.

```
# Webcam Input
cap = cv2.VideoCapture(0) # => VideoCapture(Quelle?): 0 zum auswählen der PC-WebCam
cap.set(cv2.CAP_PROP_FPS, 30)
with mp_pose.Pose(
      min_detection_confidence=0.5,
      min_tracking_confidence=0.5,
      model_complexity=1,
      smooth_landmarks=True) as pose:
```

In diesem Abschnitt wird der Input deklariert: MediaPipe soll die Aufnahme der Webcam verarbeiten. Hierfür wird die Aufnahmegeschwindigkeit auf 30 FPS reguliert und es wird eingestellt, mit welcher Genauigkeit ein Körper erkannt und verfolgt wird. Die „model_complexity“ beschreibt, welches Körpermodell verwendet werden soll. Es stehen bei MediaPipe drei verschiedene zur Auswahl, je nach gewünschtem Ergebnis kann diese mehr oder weniger Merkmale beinhalten.

```
file = open("test.hope", "w")
start_time = datetime.utcnow()

file.write('# HoPE Tracking Data - Schema 0.3\n')
file.write('###################################################################\n')
file.write('# title: Test\n')
file.write('# owner: James Beutler\n')
file.write('# date: ' + start_time.isoformat()[:10] + '\n')
file.write('# id: ' + str(uuid.uuid4()) + '\n')
file.write('###################################################################\n')
file.write('# --- camera initialisation parameters' + '\n')
file.write('# --- camera information' + '\n')
file.write('# camera resolution: ' + str(cap.get(cv2.CAP_PROP_FRAME_WIDTH)) + 'x' +
str(cap.get(cv2.CAP_PROP_FRAME_HEIGHT)) + '\n')
file.write('# camera fps: ' + str(cap.get(cv2.CAP_PROP_FPS)) + '\n')
file.write('# --- object detection initialisation parameters' + '\n')
file.write('# --- object detection runtime parameters' + '\n')
file.write('###################################################################\n')
file.write('ts ' + start_time.isoformat()[:23] + '\n')
```

Listing 3: Speicherung der Metadaten einer abspielbaren Aufzeichnung

Das Speichern der Aufnahme wird hier initialisiert: In eine Text-Datei kann man etwas schreiben. Hier wird eine Text-Datei generiert, in welche dann die von MediaPipe generierten Rohdaten geschrieben werden können. Zunächst beschreibt dieser Codeausschnitt aber nur, welche Metadaten erfasst werden, um die spätere Analyse und die Wiedergabe der Datei zu erleichtern.

```
while cap.isOpened():
    current_time = datetime.utcnow()
    success, image = cap.read()
    if not success:
        print("Leeres Kamerabild wird ignoriert")
        continue # 'break' für Videos nutzen; 'continue' für Echtzeitaufnahmen

    # Drawing
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    results = pose.process(image)

    # Speichern des Zeitstempels und der Körperpunkte als txt. Datei
    plm = results.pose_landmarks.landmark # plm steht für pose_landmarks
```

```
            timestamp = str(round(1000 * (current_time - start_time).total_seconds()))
            ergebnis = ''

            ergebnis += 'f ' + timestamp + '\n'
            ergebnis += 'p 0 ' + print_vec([(plm[23].x + plm[24].x) / 2,
               (plm[23].y + plm[24].y) / 2, (plm[23].z + plm[24].z) / 2]) + '\n'

            for kpindex in range(len(plm)):
               x = plm[kpindex].x
               y = plm[kpindex].y
               z = plm[kpindex].z
               ergebnis += 'kg ' + str(kpindex)
               ergebnis += ' ' + print_vec([x, y, z])
               ergebnis += '' + '\n'
            file.write(ergebnis)
      file.close()
cap.release()
```

Listing 4: Letzter Codeabschnitt

In diesem finalen Abschnitt wird, solange die Aufnahme aktiv ist, geprüft ob eine Person aufgezeichnet werden kann. Wird keine Person erkannt, gibt das Programm die Zeile „Leeres Kamerabild wird ignoriert“ aus. Ist das Gegenteil der Fall, wird durch MediaPipe ein Bild generiert, welches die Körperpunkte digital auf die erkannte Person projiziert. Diese Körperpunkte erhalten den Variablennamen `plm`. Der Zeitstempel wird dann gespeichert, gefolgt von jedem soeben generierten Keypoint, welcher aus seinen drei Koordinaten x, y und z besteht. Diese ganzen Daten werden in einem Array `ergebnis` aneinander gereiht und mit `file.write` in der Datei abgespeichert. Beendet man das Programm, wird die Datei mit `file.close` geschlossen und die Aufnahme mit `cap.release` beendet.

6 Evaluation

6.1 MediaPipe als Lösung

Die Eingangsproblematik bestand darin, ein bereits bestehendes und funktionierendes System zu Erweitern, um dessen Flexibilität zu erhöhen. Dieses System hat die Fähigkeit 3D Body-Tracking Aufnahmen zu erstellen und abzuspeichern, damit diese zu Analysezwecken als 3D Animation wiedergegeben werden können. Eine der essentiellen Eigenschaften dieses Systems ist, dass sich mehrere Personen im Aufnahmebereich der Sensorik aufhalten können und jede einzelne Person nicht nur parallel erkannt und aufgezeichnet wird, sondern diese auch ihre eigene ID erhalten. Dieser Faktor ist vor allem für spätere geplante Analysen im Verhalten der Passanten wichtig um den zu Beginn im Kapitel 1 Einleitung angesprochenen Honeypot-Effekt erforschen zu können.

Die von mir in den letzten Kapitel vorgestellte Lösung, Aufnahmen mit MediaPipe BlazePose zu erstellen, bietet diese Option nicht. Ersichtlich ist das unter anderem in der Modellbeschreibung von V. Bazarevsky et al. (2020) und nach einschlägigen Tests der Software. Daraus folgere ich, dass meine Lösung zwar kostengünstiger ist und ein gewisser Vorteil gegenüber der Anfangslösung hinsichtlich der Kompatibilität mit vielen Endgeräten besteht, da man nicht auf eine spezielle Hardware angewiesen ist, jedoch fehlt hier ein für die Untersuchung des Honeypot-Effekts wichtiger Aspekt.

Aufbauend auf meine Lösung könnte man als Entwickler an dieser Stelle Software-seitig gegensteuern, indem man die Funktion zur Erkennung mehrerer Personen implementiert, beispielsweise durch Frame-Weise erzwungenes Ausblenden bereits aufgenommener Personen.

Vergleicht man die Ergebnisse, die mit beiden Aufnahmemethoden erzielt werden, kann man auch feststellen, dass die Tiefenerkennung der Stereokamera zuverlässiger ist, als jene die durch das künstliche Neuronale Netzwerk von MediaPipe berechnet werden kann. Der Vorteil der ZED Kamera von Stereolabs hierbei ist logisch nachvollziehbar durch den zweiten optischen Sensor gegeben. Im Kapitel 3 erkläre ich diesen Fakt, durch einen Vergleich zur Natur und dem räumlichen Sehvermögen des Menschen. An dieser Stelle kann aber angemerkt werden, dass es absehbar ist, dass durch stetiges Voranschreiten der Technologien und der konstanten Verbesserungen im Bereich Machine-Learning und Artificial Intelligence, dieser Vorteil in Zukunft nicht mehr so stark ins Gewicht fallen muss: die Tiefenerkennung könnte durch weiteres Training des künstlichen Neuronalen Netzwerkes verbessert werden, sodass zukünftige Qualitätseinbußen der Tiefenerkennung geringer ausfallen könnten.

An dieser Stelle scheint es wichtig zu erwähnen, dass die Tiefenerkennung mit MediaPipe rein auf der „Erfahrung“ des künstlichen neuronalen Netzes basiert. Diese ist laut Entwicklern auf eine Anzahl von ca. 85 000 Bildern zum Training und 30 000 Bilder zur Auswertung begrenzt. Die Tatsache, dass die Auswertung von Videos und Videoaufnahmen in Echtzeit nicht berücksichtigt wurde, zeigt auf, dass hier Verbesserungspotential besteht. Außerdem wird im Modelsheet von Bazarevsky (2022) darauf hingewiesen, dass ein Teil des auszuwertenden Materials aus Bildern bestand, auf denen der menschliche Körper nicht vollständig abgebildet war.

6.2 Datenformat und -ausgabe

Das Datenformat mit dem beide Systeme operieren kann flexibel angepasst werden und schränkt den Nutzer hinsichtlich des Aufnahmeformats in keiner Weise ein. Zum Beispiel kann die Anzahl der in einer 3D Animation angezeigten Körperpunkte pro Modell und Aufnahme nach Belieben modifiziert werden. Des weiteren ist es durch die flexible Softwarestruktur möglich, weitere Systeme zur Aufnahme der gewünschten Daten zu implementieren. Dieser Faktor kann interessant sein um verschiedene Werkzeuge im Bereich 3D Body-Tracking zu vergleichen und gegebenenfalls eine dritte, bessere Lösung für die weitere Forschung im Projekt zu etablieren.

Während des Entwicklungsprozesses wurden verschiedene Aufnahmen generiert. Durch das Testen von unterschiedlichen Speichermöglichkeiten, entstanden Bilder, GIFs und Videodateien. Im Kapitel 4 wurde die geplante Nutzung von zwei unterschiedlichen Kameramodellen erwähnt: eines mit der Möglichkeit Aufnahmen in einer hohen Auflösung zu erstellen und ein etwas erschwinglicheres Modell mit einer geringeren Auflösung. Hier konnte festgestellt werden, dass die Auflösung der Kamera in einem Bereich über 720p keine tragende Rolle bei der Qualität der 3D Body-Tracking Rohdaten spielt. Im nachfolgenden Teil der Arbeit stehen Bilder zur Veranschaulichung der erzielten Ergebnisse zur Verfügung.

Datenausgabe

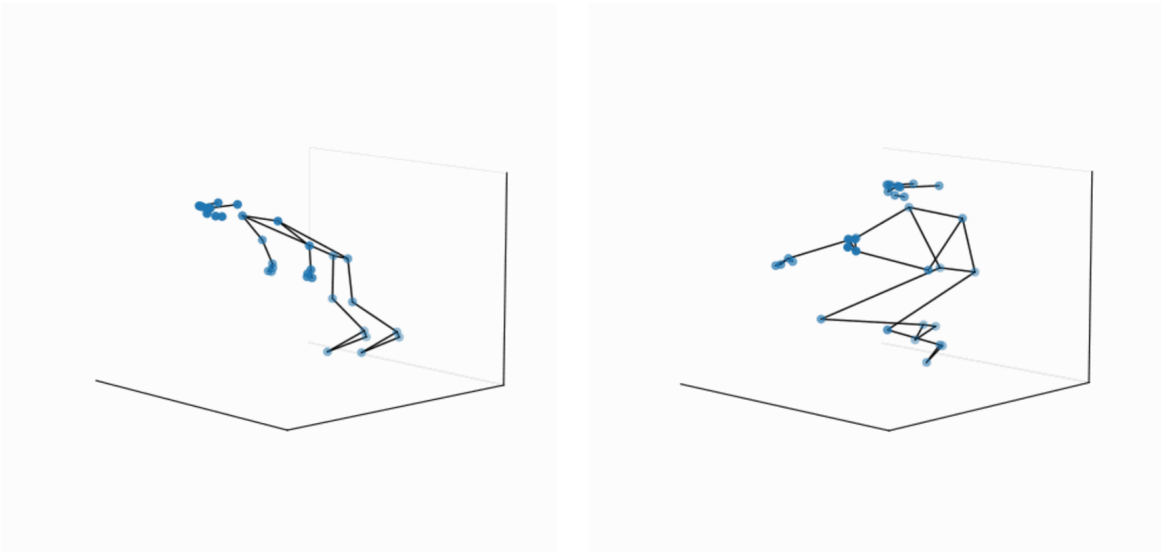


Abbildung 6: Screenshots eigener Aufzeichnungen mit MediaPipe. Die linke Grafik entstand, während die erkannte Person auf dem Schreibtisch aufgestützt stand und die Beine nicht erfasst werden konnten. Die gleiche Situation ist bei der Rechten Grafik gegeben: Die Beine konnten von der Kamera nicht erfasst werden, daher wird die Position der erforderlichen Körperpunkte „geschätzt“.

Nachfolgende Abbildungen sind Screenshots die entstanden sind, als aufgezeichnete Datensätze über PoseViz abgespielt wurden. Sie dienen dem Vergleich zu den Ergebnissen, welche die Stereokamera erzielt hat. Diese grafische Darstellung der Erzielten Ergebnisse verdeutlicht die erwähnten Mängel und Abzüge, die bei der Nutzung von MediaPipe feststellbar sind.

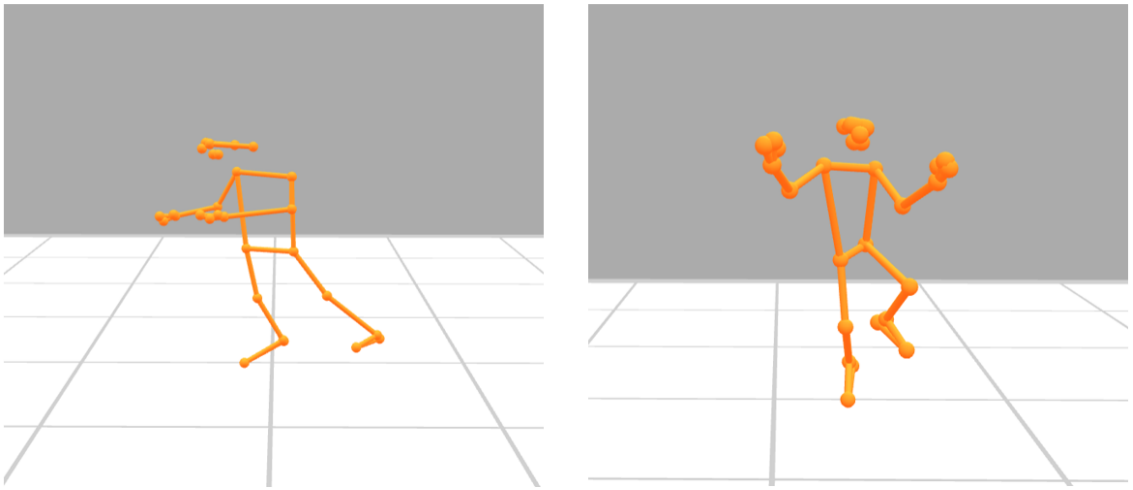


Abbildung 7: Frontale Aufnahmen aus einer Distanz von zwei Metern.

Leichte Abzüge bei der Tiefenerkennung sind hier schon sichtbar. Zusätzlich zeigt sich, dass sich einige der dargestellten Punkte Teilweise überlappen - Idealerweise sollte das nicht passieren. Vermutlich spielt hier die Distanz zu Kamera eine Rolle.

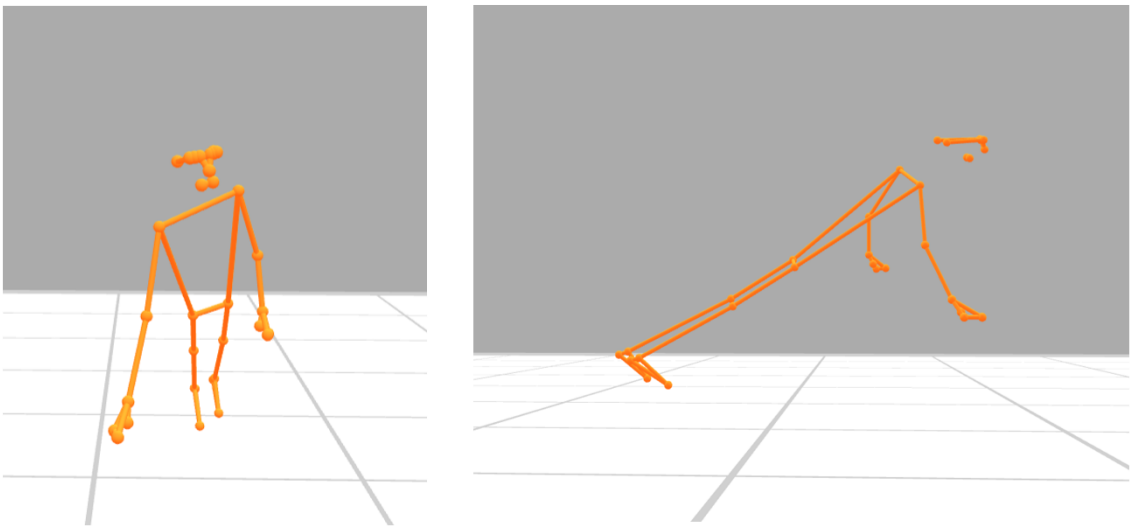


Abbildung 8: Eine weitere frontale Aufnahme, diesmal auf einen halben Meter Distanz. Durch Betrachtung der Seitenansicht fällt auf: die Tiefenerkennung ist nicht optimal.

Auf dieser Abbildung ist deutlich erkennbar, welche Schwachstelle die Tiefenerkennung von

MediaPipe hat, sobald nicht der gesamte Körper einer Person aufgenommen werden kann. Links sieht man die 3D Projektion der Person in der Vorderansicht. Das Bild daneben stellt die Seitenansicht von rechts der selben Szene dar. Während der Aufnahme stand die von MediaPipe erkannte Person auf dem Fußboden. Die linke Seite der Abbildung lässt vermuten, dass dies auch auf der Animation angezeigt würde, jedoch zeigt die rechte Seite, dass die Beine durch MediaPipe nach hinten gestreckt dargestellt werden. Durch diesen Side-by-Side Vergleich ist leichter feststellbar, dass das „schätzen“ der Tiefenposition und der Allgemeinposition nicht erfasster Körperpunkte keine optimalen Ergebnisse liefert.

Diese Ergebnisse wurden durch Tests erzielt, bei denen die Position der Kamera nicht verändert wurde. Denkbar ist es, durch unterschiedliche Positionen der Kamera oder der räumlichen Gegebenheiten auch andere, möglicherweise bessere Ergebnisse zu erzielen. Diese Vermutung kann im weiteren Verlauf getestet werden, sollte die Weiterentwicklung der Software fortgesetzt werden.

7 Fazit und Ausblick

Durch die Einarbeitung in das Themengebiet MediaPipe BlazePose rund um die Machine-Learning Technologie von Google (2022) erkannte ich, dass ich auf ein Werkzeug gestoßen bin, das viele Möglichkeiten hinsichtlich der Erforschung von Mensch-Computer-Interaktionen bietet. Durch weitere Recherchen rund um MediaPipe konnte ich auf zahlreiche Anwendungsgebiete stoßen, welche unter anderem Gesichts- und Gestikerkennung beinhalten. Ein interessanter Aspekt hierbei ist, dass jede dieser Möglichkeiten auf dem selben Grundbaustein basieren und es nur einige wenige Modifikationen und Codezeilen in meinem Programm ausreichen können um weitere Funktionen zu implementieren - ferner noch ist es denkbar diese Funktionen miteinander zu kombinieren.

Durch das Sichten und Aussortieren von wissenschaftlichen Arbeiten und Anwendungen von MediaPipe BlazePose wurde zu Beginn, noch bevor ich mit der Implementierung anfangen konnte, sehr schnell ersichtlich, dass ein wichtiger Aspekt selten oder bisweilen gar nicht in Betracht gezogen wurde: Das extrahieren der Rohdaten zum speichern und abspielen einer 3D Animation des aufgezeichneten Körpermodells. Diese Anforderung setzte ich, mit Hilfe der gefundenen Dokumentation der Entwickler von MediaPipe, in dieser Arbeit erfolgreich um.

Als ich mit der Bearbeitung der hier vorgestellten Thematik begonnen hatte, besaß ich nach eigener Einschätzung nur sehr wenig bis gar kein Vorwissen im Bereich 3D Body-Tracking, dessen Umsetzung und der gegebenen Möglichkeiten. Nach Fertigstellung meiner Software bemerkte ich, dass diese Einschätzung korrekt war und auch immer noch ist. Dennoch konnte ich alle Fragen, die ich Anfangs gestellt habe, zufriedenstellend beantworten und zusammenfassend stelle ich fest, dass die geforderte Hauptfunktion umgesetzt wurde. Hinsichtlich einiger gewünschter Funktionen im 3D Body-Tracking ist meine Lösung jedoch eingeschränkt. Die Eingangsproblematik wurde somit zwar erfolgreich gelöst, die Lösung ist aber nicht ideal für das präsentierte Anwendungsgebiet im Forschungsprojekt. Trotzdem ist nach der fertigen Implementierung eine stark erhöhte Erweiterbarkeit der Software gegeben und es ist ersichtlich, dass nicht erfüllte Aspekte mit dieser Technologie durch zukünftige Updates auch ausgebessert werden können.

Ausblick

Abschließend ist festzuhalten, dass der Einblick in die Thematik 3D Body-Tracking, im Einklang mit den in meiner Arbeit gewonnenen Erkenntnissen über MediaPipe BlazePose zwar keine neue Tür im Forschungsprojekt HoPE öffnet, dafür aber ein Fenster mit Aussicht auf viele

faszinierende Methoden, Technologien und deren Anwendungen im allgemeinen Bereich der Mensch-Computer Interaktionen. Die Einbindung des Systems in den Tracking-Server des Projekts kann für weitere Updates der Software förderlich sein und begünstigt das zukünftige Implementieren zusätzlicher Funktionen, wie zum Beispiel der Erkennung mehrerer Personen mit MediaPipe. Außerdem ist es denkbar, zu Analysezwecken einen direkten Vergleich beider Methoden des 3D Body-Trackings, also mit der ZED 2 Kamera und mit MediaPipe gleichzeitig, zu erstellen. Ein Vergleich dieser Rohdaten kann dann Informationen zur quantitativen Analyse der Genauigkeit der MediaPipe-Körpererkennung im Vergleich zur ZED 2 Kamera liefern.

Denkbar ist auch, die Steuerung der Ambient Displays durch eine Gestikerkennung zu ersetzen. In Pandemie-Zeiten war es oft wichtig, nicht in Kontakt Oberflächen zu kommen, die viele Menschen berührt haben könnten. Daher könnte es sinnvoll erscheinen, bestimmte Aspekte im Alltag auf diese Weise zu verändern. Eine Gestiksteuerung muss in diese Fall auch nicht den gesamten Körper erfassen, ausreichend hierfür wären dann nur die Arme und Hände im Detail.

Literatur

Bazarevsky. (2022). GHUM 3D Pose Model Card [abgerufen am 18. Dezember 2022]. https://developers.google.com/static/ml-kit/images/vision/pose-detection/pose_model_card.pdf

Bazarevsky, V., Grishchenko, I., Raveendran, K., Zhu, T., Zhang, F., & Grundmann, M. (2020). BlazePose: On-device Real-time Body Pose tracking. https://doi.org/10.48550/ARXIV.2006.10204

Dipl.-Ing. Luber, S., & Litzel, N. (2019). Was ist ein Convolutional Neural Network [abgerufen am 07. November 2022]. https://www.bigdata-insider.de/was-ist-ein-convolutional-neural-network-a-801246/

Fraunhofer-IKS. (2023). Neuronale Netze und Deep Learning - eine Definition [abgerufen am 29. Januar 2023]. https://www.iks.fraunhofer.de/de/themen/kuenstliche-intelligenz.html

für Softwaretechnologie, I. (2022). Honeypot-Effekt an interaktiven Ambient Displays [abgerufen am 08. November 2022]. https://www.unibw.de/inf2/forschung/projekte/hope

Google. (2022). MediaPipe BlazePose GHUM 3D [abgerufen am 18. Dezember 2022]. https://developers.google.com/ml-kit/vision/pose-detection#under_the_hood

Google. (2023a). MediaPipe BlazePose [abgerufen am 20. Januar 2023]. https://google.github.io/mediapipe/solutions/pose.html#pose-landmark-model-blazepose-ghum-3d

Google. (2023b). MediaPipe Holistic [abgerufen am 20. Januar 2023]. https://google.github.io/mediapipe/solutions/holistic.html

Google-LLC. (2023a). ML Kit [abgerufen am 20. Januar 2023]. https://developers.google.com/ml-kit/guides

Google-LLC. (2023b). Solutions - Pose [abgerufen am 16. Januar 2023]. https://google.github.io/mediapipe/solutions/pose.html#segmentation_mask

ICS-AG. (2019). Videoüberwachung öffentlich zugänglicher Räume [abgerufen am 20. Januar 2023]. https://dsgvo-gesetz.de/bdsg/4-bdsg/

Keil, M. (2012). Body Tracking- Echtzeitbewegung von 3D Charakteren durch Bewegungssteuerung am Beispiel von Kinect [abgerufen am 28. Januar 2023]. https://monami.hs-mittweida.de/frontdoor/deliver/index/docId/2730/file/Diplomarbeit_BodyTracking.pdf

Keineke, S. (2021). Hardware Vergleich und Auswahl [abgerufen am 16. Januar 2023]. https://wiki.unibw.de/display/MCI/Hardware+Vergleich+und+Auswahl

Liu, W. (2022a). Recent Advances of Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective [abgerufen am 20. Januar 2023]. https://dl.acm.org/doi/10.1145/3524497#sec-ref

Liu, W. (2022b). Recent Advances of Monocular 2D and 3D Human Pose Estimation: A Deep Learning Perspective [abgerufen am 20. Januar 2023]. https://dl.acm.org/doi/10.1145/3524497#sec-ref

O'Connor, R. (2022). MediaPipe for Dummies [abgerufen am 18. Dezember 2022]. https://www.assemblyai.com/blog/mediapipe-for-dummies/

Pardos, A., Tziomaka, M., Menychtas, A., & Maglogiannis, I. (2022). Automated Posture Analysis for the Assessment of Sports Exercises. ISBN: 9781450395977. https://doi.org/10.1145/3549737.3549784

Plagemann, C. (2010). Real Time Motion Capture Using a Single Time-of-Flight Camera. http://robotics.stanford.edu/~koller/Papers/Ganapathi+al:CVPR10.pdf

Rodriguez-Moreno, I., Martinez-Otzeta, J. M., Goienetxea, I., & Sierra, B. (2021). Sign Language Recognition by Means of Common Spatial Patterns. ISBN: 9781450387613. https://doi.org/10.1145/3453800.3453818

Schneider, F. (2011). Mobile Interaktion mit Public Displays im öffentlichen Verkehr [abgerufen am 17. Januar 2023]. https://tu-dresden.de/ing/informatik/smt/seus/ressourcen/dateien/lehre/files_closed/2011/GB/files/PDF_2011-GB-02?lang=de

Stereolabs. (2022a). Built for Developers. Trusted by Entreprises. [abgerufen am 08. November 2022]. stereolabs.com

Stereolabs. (2022b). Datasheet [abgerufen am 16. Januar 2023]. https://cdn.stereolabs.com/assets/datasheets/zed-2i-datasheet-feb2022.pdf

Hiermit versichere ich, dass ich die vorliegende Arbeit selbständig verfasst habe, dass keine anderen als die angegebenen Quellen und Hilfsmittel benutzt wurden und dass alle Zitate ordnungsgemäß gekennzeichnet worden sind.

Ferner habe ich vom Merkblatt über die Verwendung der Bachelorarbeit Kenntnis genommen und räume der Universität der Bundeswehr München das einfache Nutzungsrecht an meiner Bachelorarbeit ein.

Neubiberg, den 02.02.2023

.....

James Daniel Beutler