Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Entwicklung und Evaluation eines markerbasierten Inside-Out Objekt-Trackings für VR-Brillen

Fakultät Design, Medien und Information
Studiendepartment Medientechnik

Faculty of Design, Media and Information
Department of Media technology

Entwicklung und Evaluation eines markerbasierten Inside-Out Objekt-Trackings für VR-Brillen

Masterarbeit eingereicht im Rahmen der Masterprüfung

im Studiengang Digital Reality M.Sc.
am Department Medientechnik
der Fakultät Design, Medien und Information
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Torsten Edeler
Zweitgutachter: Dr. Arthur Varkentin

Eingereicht am: 28. Juli 2023

Moritz Stoll

Thema der Arbeit
Entwicklung und Evaluation eines markerbasierten Inside-Out Objekt-Trackings für VR-Brillen

Stichworte
VR Object Tracking, Inside-Out Tracking, Stereo Vision, Kalman Filter

Kurzzusammenfassung
In dieser Arbeit wird untersucht, inwiefern es möglich ist, ein auf einer externen Stereokamera sowie reflektierenden Markersphären basierendes Inside-Out Objekt-Tracking mit sechs Freiheitsgraden auf Basis eines Unscented Kalman Filters zu entwickeln. Dieses soll es ermöglichen, reale Objekte in einer Virtual Reality-Anwendung zu verwenden. Das Tracking-System wird auf der Virtual Reality Brille montiert und ist somit mobil. Das System wird mithilfe realer Bewegungsdaten und simulierter Messdaten getestet und mit den Ergebnissen eines Systems verglichen, das die Pose eines Objektes aus den Messdaten deterministisch berechnet. Es kann nicht nachgewiesen werden, dass das auf dem Unscented Kalman Filter basierende System eine Verbesserung gegenüber dem determinstsichen System darstellt. Hinsichtlich der Bestimmung der Orientierung ist es sogar signifikant ungenauer. Zudem werden die Grenzen des Tracking-Systems bestimmt. So funktioniert es bei zunehmend ausladenden Bewegungen immer schlechter. Für minimale und statische Bewegungen, bei denen sich die Virtual Reality Brille kaum bewegt, ist das Verfahren hinsichtlich der Positionsschätzung ausreichend genau. Für komplexere Bewegungen wird die Abweichung zu groß und ist damit für die Praxis nicht einsetzbar. Die Orientierungsschätzung ist in beiden Fällen zu groß, um das System in der Praxis zu verwenden. Es kann gezeigt werden, dass die geringe Genauigkeit dem Versuch geschuldet ist, einen universellen Unscented Kalman Filter zu entwickeln, der manövrierende Objekte tracken kann, die sich durch eine Kombination verschiedener dynamischer Systeme auszeichnen. Es wird vorgeschlagen, diese Komplexität und Vielfältigkeit des zu trackenden Systems in Zukunft zu berücksichtigen, um die Genauigkeit zu verbessern.

Moritz Stoll

Title of the paper
Development and evaluation of a marker-based inside-out object tracking for VR glasses

Keywords
VR Object Tracking, Inside-Out Tracking, Stereo Vision, Kalman Filter

Abstract
In this thesis, I investigate the feasibility of developing an inside-out object tracking system with six degrees of freedom based on an external stereo camera and reflective marker spheres using an Unscented Kalman filter. This will enable real objects to be used in a virtual reality application. The tracking system is mounted on the virtual reality headset and is therefore mobile. The system is tested using real motion data and simulated measurement data and compared to the results of a system that deterministically calculates the pose of an object from the measurement data. It can not be shown that the system based on the Unscented Kalman Filter is an improvement over the deterministic system. With respect to the determination of the orientation, it is even significantly less accurate. In addition, the limitations of the tracking system are determined. Thus, it works worse for increasingly sweeping motions. For minimal and static movements, where the virtual reality headset hardly moves, the method is sufficiently accurate in terms of position estimation. For more complex movements, the deviation becomes too large and is thus not applicable for practical use. In both cases, the orientation estimation is too large to use the system in practice. It is shown that the low accuracy is due to the attempt to develop a universal Unscented Kalman Filter that can track maneuvering objects characterized by a combination of different dynamical systems. It is suggested that this complexity and diversity of the system being tracked be taken into account in the future to improve accuracy.

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einleitung

In verschiedenen Bereichen der Virtual Reality (VR) kann es nötig sein, Objekte aus der realen, physischen Welt, innerhalb einer VR-Anwendung zu verwenden. So beispielsweise im Kontext von Trainingsanwendungen in VR, in denen es hinsichtlich der Entwicklung von wahrnehmungsmotorischen Fähigkeiten nötig sein kann, mit realen Objekten zu interagieren. VR-Trainings haben den Vorteil, Situationen und Umgebungen abbilden zu können, die in der Realität gar nicht, nur unter hohem Aufwand oder verbunden mit realen Gefahren herzustellen sind. Beispielsweise die Ausführung von Wartungs- oder Reparaturarbeiten in der Höhe oder das Ausführen gewisser Tätigkeiten in Gefahrensituationen. In diesen Szenarien kann es den Trainingseffekt verstärken, wenn Nutzer*innen im Training bereits das reale, physische Werkzeug benutzen, das in der Realität verwendet werden würde.

Viele gängige VR-Headsets verfügen über ein Hand-Tracking, dass es erlaubt die Hände der Nutzer*innen, akkurat zu tracken und deren Fingerstellung auf ein dreidimensionales Modell in der VR-Anwendung zu übertragen. Wird jedoch ausschließlich das Hand-Tracking für ein Training verwendet, erfahren Nutzer*innen keine haptische Rückmeldung, wenn sie ein in der Anwendung vorhandenes Objekt berühren oder mit ihm interagieren. Werden VR-Controller statt des Hand-Trackings verwendet, können darüber Rückmeldungen in Form von Vibrationen erfolgen. Allerdings haben die Controller in der Regel eine andere Form oder ein anderes Gewicht als das Objekt, mit dem interagiert wird.

Eine vielversprechende Alternative ist daher das Tracking des Objektes selbst, dass Nutzer*innen beispielsweise in die Hand nehmen. Gepaart mit dem Hand-Tracking, das viele aktuelle VR-Headsets nativ unterstützen, ließe sich so eine realistische Trainingserfahrung erzeugen. Allerdings unterstützen die meisten gängigen VR-Headsets den Zugriff auf die unverarbeiteten Daten ihrer Sensoren nicht, was die Entwicklung eines Tracking-Systems für Objekte erschwert. Auch gibt es für VR-Headsets keine kommerziellen Lösungen, da sich diese meistens an die Verwendung mit Augmented Reality (AR)-Headsets oder Geräten richten, die in der Hand gehalten werden.

Ziel dieser Masterarbeit ist es, ein ausschließlich auf einer Stereokamera und einem Unscented Kalman Filter sowie reflektierenden Markern basierendes Tracking-Verfahren zu entwickeln und hinsichtlich seiner Qualität zu evaluieren. Es soll somit geprüft werden, ob es möglich ist, Objekte mit sechs Freiheitsgraden so präzise zu tracken, dass sie innerhalb einer VR-Anwendung als Objekte angezeigt werden können, mit denen Nutzer*innen interagieren. Die Verwendung eines Unscented Kalman Filters hebt sich dabei von anderen bekannten Arbeiten ab, die ein solches Tracking-Verfahren umzusetzen, da diese in der Regel den Extended Kalman Filter verwenden. Weiterhin soll die Qualität des auf dem Unscented Kalman Filter basierenden Verfahren im Vergleich zu einem simpleren Ansatz bewertet werden. Bei diesem Ansatz kommt kein Bayes Filter zum Einsatz. Es werden lediglich die Position und die Rotation des zu trackenden Objektes aus den Messdaten berechnet. Die grundlegenden Fragen, die diese Arbeit beantwortet, sind somit: "Lässt sich ein Objekt mit sechs Freiheitsgraden ausschließlich mithilfe einer Stereokamera und einem Unscented Kalman Filter tracken?", "Wie akkurat ist dieses Tracking im Vergleich zu einem nicht auf einem Filter basierenden, deterministischen Verfahren?", "Welche Bewegungen des Objektes verändern, inwiefern die Qualität des Trackings?".

Um die genannten Fragen zu beantworten, werden in Kapitel 2 zunächst die theoretischen Grundlagen vermittelt. Hier gehe ich in Abschnitt 2.1 auf den aktuellen Stand hinsichtlich des Trackings von Objekten ein. Dabei unterscheide ich zwischen dem industriellen und kommerziellen Stand sowie dem wissenschaftlichen Stand. Weiterhin vermittelt Absatz 2.2 einen Überblick über das theoretische Grundwissen zu Zustands-, beziehungsweise Bayes-Filtern. Hier gehe ich sowohl auf die Grundlagen des Kalman Filters, als auch den Kalman Filter selbst und schließlich den in dieser Arbeit verwendeten Unscented Kalman Filter ein.

Kapitel 3 widmet sich dem methodischen Vorgehen innerhalb dieser Arbeit. Hier wird der Aufbau des im Rahmen dieser Arbeit entwickelten Tracking-Verfahrens beschrieben. Zunächst widme ich mich dafür der Auswahl der Instrumente in Abschnitt 3.1. Anschließend erläutere ich das Konzept des Tracking-Verfahrens in Absatz 3.2. Weiterhin erläutere ich in Absatz 3.3 die Simulation einer realen Kamera in der Game Engine Unity. Darauf folgend ist die Kalibrierung dieser Kamera in Absatz 3.4 näher beschrieben. In Abschnitt 3.5 beschreibe ich einen der wichtigsten Bausteine des Algorithmus. Dabei handelt es sich um die virtuelle Filterkamera, die Messdaten erzeugt und im Tracking-Verfahren verwendet wird. In Abschnitt 3.6 wird erläutert, wie die Markerpose des Tracking-Systems aufgebaut ist. Der Absatz 3.7 beschäftigt sich mit der Erzeugung von Messdaten anhand der realen Kamera und mit der Erkennung

von reflektierenden Markern. Daran anknüpfend gehe ich in Absatz 3.8 auf die Simulation der Messdaten ein und beziehe mich auf die zuvor gewonnenen Erkenntnisse. Anschließend wird in Absatz 3.9 zunächst das deterministische Verfahren beschrieben, das Position und Rotation aus den Messdaten errechnet. Darauf folgend beschreibe ich in Absatz 3.10 das auf dem Unscented Kalman Filter basierende Verfahren. Den methodischen Teil schließt die Beschreibung des Vorgehens bei der Evaluation des Verfahrens in Abschnitt 3.11.

Die Ergebnisse der Evaluation beschreibe ich in Kapitel 4. Hier gehe ich in Absatz zunächst auf die Ergebnisse auf Basis künstlicher Bewegungsdaten ein. Anschließend widme ich mich in Absatz 4.2 den Ergebnissen einer quantitativen Evaluation der Ergebnisse, die auf Basis realer Bewegungsdaten entstanden sind. In Absatz 4.3 werden diese Ergebnisse im Rahmen einer qualitativen Evaluation tiefergehend beleichtet.

In Kapitel 5 folgt die Diskussion und Interpretation der zuvor beschriebenen Ergebnisse. Hier setze ich mich zunächst in Absatz 5.1 mit der Qualität des entwickelten Tracking-Verfahrens ein und beziehe dafür sowohl die Ergebnisse aus der qualitativen sowie quantitativen Evaluation mit ein. Auch reale und künstliche Bewegungsdaten werden hier berücksichtigt. Anschließend leite ich aus den Ergebnissen in Absatz 5.2 die Grenzen des entwickelten Tracking-Systems ab. Zuletzt setze ich mich in Absatz 5.3 mit einem spezifischen Fehler auseinander, der für den Unscented Kalman Filter auftritt.

Das Fazit der Arbeit findet sich abschließend in Kapitel 6. Hier fasse ich in Absatz 6.1 zunächst die gewonnenen Erkenntnisse zusammen. Anschließend leite ich daraus in Absatz 6.2 Empfehlungen für weitere Forschungen an dem Tracking-Verfahren ab, das ich in dieser Arbeit entwickelt und evaluiert habe.

2. Theoretischer Teil

Das in dieser Masterarbeit entwickelte und evaluierte Trackingsystem basiert auf einer Reihe theoretischer Grundlagen, die zum Verständnis der Arbeit unabdingbar sind. Dazu ist es zunächst entscheidend, die durchgeführte Arbeit und die Forschungsfrage im Kontext des aktuellen Standes der Wissenschaft einzuordnen. Diese Einordnung geschieht in Abschnitt 2.1. Hier gehe ich sowohl auf den industriellen Standard für das Tracking von Objekten als auch den wissenschaftlichen Stand ein.

Der in dieser Arbeit gewählte Ansatz für das Tracking eines linearen Systems ist der Unscented Kalman Filter. Die Begründung für diese Wahl findet sich in Abschnitt 3.1.1. Der Unscentend Kalman Filter gehört zur Familie der Zustandsfilter. Diese erkläre ich in Absatz 2.2. Dort gebe ich einen Überblick über die zugrundeliegenden Theorien und leite den Unscented Kalman Filter aus dem klassischen Kalman Filter und dessen Untauglichkeit für nicht lineare Systeme her.

Bei der hier durchgeführten Forschung steht das Tracking von Objekten im Vordergrund. Dieses kann mithilfe verschiedener Ansätze durchgeführt werden. Zudem stellen auch die zu trackenden Objekte selbst unterschiedliche Ansprüche hinsichtlich der Gestaltung eines Trackingsystems. Einen Überblick über Objekt-Tracking, die unterschiedlichen Ansätze und Ansprüche liefert Abschnitt 2.3.

2.1. State of the Art der Objekt-Tracking-Lösungen

In diesem Abschnitt sind der aktuelle Stand der Industrie und der Forschung hinsichtlich des Objekt-Trackings beschrieben. Dabei unterscheide ich zwischen zwei Bereichen, zu denen ich meine Arbeit abgrenze. Zum einen Lösungen, die für VR-Headsets bereits verwendet werden. Dabei sind diese ungeachtet ihres Ansatzes gewählt. In der Regel verwenden diese Systeme externe Sensorik und verlassen sich nicht ausschließlich auf das verwendete VR-Headset. Weiterhin betrachte ich gezielt den Stand der Forschung hinsichtlich Inside-Out-Tracking Systeme für andere Head Mounted Display (HMD). Hierbei kann zwischen mehreren Ansätzen

unterschieden werden. Manche verwenden Passermarken, andere erkennen Features in der Umgebung, wieder andere erkennen leuchtende oder reflektierende Marker.

2.1.1. Aktuelle Industrielösungen für Objekt-Tracking für Virtual Reality

Trackingverfahren können laut Gsaxner u. a. (2021) zwei Ansätzen zugeordnet werden. Beim Inside-Out-Tracking sind die Sensoren, die Messdaten erzeugen, in dem zu trackenden Objekt verortet. Beim Outside-In-Tracking sind sie außerhalb des zu trackenden Objektes verortet. Diese Unterscheidung ist für diese Arbeit insofern wichtig, dass Outside-In-Tracking Systeme in der Regel auf stationäre Sensoren angewiesen sind, die einen gewissen Tracking-Bereich abstecken. Inside-Out-Tracking hingegen wird dagegen in vielen eher mobilen Anwendungen verwendet. In Abschnitt 2.4 werde ich meine eigene Unterscheidung der beiden Tracking-Ansätze etwas von der bei Gsaxner u. a. (2021) beschriebenen lösen.

Für das Tracking von Objekten gibt es bereits diverse Lösungen. So zum beispiel klassische Motion Capture-Systeme wie das OptiTrack[1], das mithilfe stationärer Kameras Objekte trackt, an denen Posen mit reflektiven Markern angebracht sind. Alternativ werden zum Zeitpunkt der Verfassung dieser Arbeit häufig Vive Tracker[2] eingesetzt. Diese sind für den Konsumentenmarkt zugänglicher und leichter aufzubauen. Sie basieren auf dem Lighthouse Tracking-System. Hierbei werden laut Niehorster u. a. (2017) Basisstationen verwendet, die horizontale und vertikale Laserstrahlen aussenden. Diese werden von den Sensoren, die in den Vive-Trackern verbaut sind, erfasst und anhand der Art des Eintreffens berechnen die Tracker ihre Position und Orientierung im Raum. Bei dem Vive-Tracker handelt es sich um einen 70,9 mm × 79,0 mm × 44,1 mm großen Tracker, der an Gegenständen oder auch dem Körper befestigt werden kann. Zwar fällt dieser Tracking-Ansatz auch unter das Inside-Out-Tracking System, dennoch braucht er externe Geräte, die einen Raum definieren, innerhalb dessen das Tracking funktioniert. Somit unterscheidet sich dieser Ansatz von dem in dieser Arbeit entwickelten.

Nach Beginn dieser Masterarbeit, hat HTC laut dem Artikel von Lang (2023) einen "Self-Tracking Tracker" angekündigt, der den Einsatz der bisher notwendigen Basisstationen überflüssig macht und sich somit genau dem Problem zuwendet, das die hier beschriebene Arbeit versucht zu lösen. Laut HTC verwendet dieser Tracker zwei Kameras mit einem hohen Field of View (FOV) sowie einen auf dem Tracker laufenden Algorithmus für das Tracking der Position und der Orientierung. Das Ziel des Trackers ist es, das Tracking des eigenen Körpers

[1] `https://optitrack.com/applications/virtual-reality/`, Stand: 20.07.2023
[2] `https://www.vive.com/de/accessory/tracker3/`, Stand: 20.07.2023

zu vereinfachen, um es beispielsweise auf dreidimensionale Avatare zu übertragen. Der Tracker soll kompatibel zu OpenXR[3] sein und damit nicht nur auf HTC-Headsets funktionieren.

2.1.2. Aktuelle Objekt-Tracking Lösungen für Augmented Reality

Für das Tracking von Objekten gibt es insbesondere im Bereich AR einige populäre, kommerzielle Lösungen. Die bekanntesten Lösungen sind VisionLib[4] und Wikitude[5]. Beide Lösungen setzen auf die Daten von CAD-Modellen der zu trackenden Objekte und ermitteln diese dann mittels verschiedener Bilderkennungsverfahren. Sie sind allerdings nicht für VR-Headsets verfügbar und damit für das Tracking von Objekten, um diese in einer VR-Szene zu bringen, nicht direkt geeignet.

2.1.3. Passermarken basierte Inside-Out Tracking-Verfahren

In ihrer Arbeit zeigen Wu u. a. (2017) ein 6 Degree of Freedom (6DoF) Tracking eines Stiftes mit einer einzelnen monokularen Kamera. Das Verfahren basiert auf der Verwendung eines Dodekaeders, auf dessen Seiten unterschiedliche binäre Passermaker angebracht wurden. Der Stift kann laut Wu u. a. (2017) mittels des modellbasierten Trackings sowie einer Kombination aus einer annäherungsweisen Schätzung der Pose sowie einer Dichte Registrierung mit einer Abweichung im Submillimeterbereich getrackt werden. Allerdings verwendet der Algorithmus keinen Bayes-Filter, wie er in dieser Arbeit verwendet wird.

Brand u. a. (2020) beschreiben in ihrer Arbeit weitere Möglichkeiten des Inside-Out-Objekt-Trackings durch HMDs mittels Marker auf einem Würfel. Die Arbeit konnte zeigen, dass bei dieser Art des Trackings, die Genauigkeit im Zentimeter-Bereich liegt. Abhängig ist die Genauigkeit vor allem von dem Tracking-Algorithmus des SDK ARToolkit in der Version 5.4[6] sowie von der Genauigkeit des Inside-Out-Trackings der verwendeten HoloLens.

Einen 6DoF Tracking-Ansatz, der ein mobiles Gerät, wie ein Smartphone, tracken kann, präsentieren Faion u. a. (2016) in ihrer Arbeit. Sie setzen auf eine Extended Kalman Filter. Die hier verwendeten Messungen bestehen nicht in der Erkennung von reflektierenden Markern, sondern zum einen in den Werten einer Inertial Measurement Unit des Smartphones sowie einzigartigen Marker und mehrdeutigen Markern. Erstere sind durch im Raum angebrachte

[3] `https://www.khronos.org/openxr/`, Stand: 25.07.2023
[4] `https://visometry.com/products/visionlib/`, Stand: 20.07.2023
[5] `https://www.wikitude.com/augmented-reality-object-scene-recognition/#3DModelSection`, Stand: 20.07.2023
[6] `https://github.com/artoolkit/ARToolKit5`, Stand: 20.07.2023

QR-Codes dargestellt und enthalten exakte 3D-Positionen des Raumes. Letztere halten weniger Informationen, sind dafür aber leichter zu finden.

Gsaxner u. a. (2021) führen eine Reihe von Arbeiten auf, in denen die HoloLens als HMD genutzt wurde, um chirurgische Instrumente mithilfe von Passermarken zu tracken. Dazu gehören die Arbeit von Carbone u. a. (2018), Gao u. a. (2019), Kriechling u. a. (2020), Leuze u. a. (2018), Müller u. a. (2020), Qian u. a. (2018) und Qian u. a. (2019). Alle vorgestellten Arbeiten eint, dass sie sowohl Passermarken als auch die spezifisch die HoloLens und die dort vorgestellte Sensorik verwenden. In Abgrenzung dazu, basiert der in dieser Arbeit erforschte Tracking-Ansatz auf dem Einsatz einer nicht notwendigerweise in dem HMD verbauten Sensorik, die somit austauschbar ist.

2.1.4. Weitere Inside-Out Tracking-Verfahren

Hattori und Hirai (2020) zeigen ein Inside-Out-Tracking-Verfahren, das dem Inside-Out-Tracking, der in Absatz 2.1.1 beschriebenen Vive-Tracker nahekommt. Hier wird ein Smartphone als 6DoF-Controller für ein HMD genutzt. Inside-Out bezieht sich in diesem Fall auf das Smartphone und nicht auf das verwendete HMD. So verwendet es Bilderkennungsalgorithmen, um die eigene Pose mit sechs Freiheitsgraden im Raum zu ermitteln und diese an das HMD zu senden. Zwar ist der Ansatz unterschiedlich zu dem in dieser Arbeit vorgestellten. Dennoch ist auch hier das Tracking des Objektes, dargestellt durch das Smartphone, unabhängig von dem HMD und dessen verbauter Sensorik.

Eine für diese Arbeit wichtige Veröffentlichung stellt die Arbeit von Gsaxner u. a. (2021) dar. Dort wurde ein Tracking-Verfahren für ein HMD entwickelt. Konkret handelt es sich dabei um die HoloLens, mit der chirurgische Instrumente mit 6DoF getrackt werden sollten. Gsaxner u. a. (2021) haben dabei auf die Kameras der HoloLens zugegriffen und einen Single Constraint At A Time (SCAAT) Extended Kalman Filter (EKF) verwendet, um die Instrumente zu erfassen. Ich benutze keinen EKF, sondern einen Unscented Kalman Filter (UKF). Auch einige weitere der dort getroffenen Annahmen habe ich für meine Arbeit verwendet. Welche Teile dieser Veröffentlichung für meine Arbeit relevant sind, ist im Abschnitt 3 genauer erläutert.

Die Arbeit von Gsaxner u. a. (2021) beruht wiederum auf den Arbeiten von Welch und Bishop (1997) und Steinicke u. a. (2007). Erstere präsentieren den SCAAT-Filter als Lösung zum Tracking von Systemen, deren Messungen in unregelmäßigen Abständen eintreffen. Letztere beschreiben das optimale Vorgehen für die Erzeugung von Markerbasierten Festkörpern für

die Entwicklung optischer Tracking-Systeme. Sowohl bei dem von Gsaxner u. a. (2021) präsentierten, als auch bei dem in dieser Arbeit gezeigten Verfahren handelt es sich um optische Tracking-Systeme.

In meiner Recherche habe ich bis auf die Arbeit von Gsaxner u. a. (2021) keinen vergleichbaren Tracking-Ansatz gefunden. Entweder geht es bei vergleichbaren Arbeiten darum, stationäre Objekte zu tracken oder gemäß einem Inside-Out-Tracking das Gerät mit der Sensorik zu tracken. Weiterhin liegt der Fokus hier in der Regel auf AR-Geräten und Anwendungen. In meinem Fall stellt das HMD ein Outside-In-Tracking System dar, dass aber anders als bei den aktuellen Industrie-Standards nicht stationär, sondern mobil ist. Die einzige weitere Veröffentlichung, die diesen Ansatz verfolgt, ist die genannte von Gsaxner u. a. (2021). Diese verwenden allerdings eine AR-Brille, die einen Zugriff auf die ungefilterten Sensor-Daten ermöglicht. Ich hingegen verwende eine VR-Brille. Die meisten aktuellen Brillen, wie die Oculus Quest 1, 2 und Pro erlauben einen Zugriff auf diese Sensorik nicht. Zudem grenzt sich meine Arbeit durch den verwendeten Kalman-Filter ab. Statt eines EKFs verwende ich einen UKF, was neue Fragstellungen und Möglichkeiten mit sich bringt.

Ein weiterer Unterschied zu den meisten der hier genannten Inside-Out-Tracking-Verfahren besteht darin, dass das von mir entwickelte Tracking-Verfahren nicht abhängig von dem jeweiligen HMD ist. Es lässt sich mit jeder VR-Brille kombinieren und ist nicht auf die Verwendung der HoloLens angewiesen. Lässt ein HMD allerdings den Zugriff auf seine Sensorik zu, kann es ebenso darauf angepasst werden.

2.1.5. Sensordaten von Virtual Reality-Brillen

Viele gängige VR-Brillen und HMDs ermöglichen es nicht, auf die Tracking-Sensoren der Headsets zuzugreifen. So werden beispielsweise bei der Meta Quest 2[7], der Meta Quest Pro[8] und der Pico Neo3[9] sowie Pico 4[10] die Daten der Tracking-Sensoren für Entwickler unzugänglich auf Betriebssystemebene verarbeitet. Das macht es unmöglich, Tracking-Verfahren für diese HMDs zu entwickeln, die über die im Betriebssystem vorgesehenen Tracking-Möglichkeiten hinausgehen. Aus diesem Grund habe ich mich in dieser Arbeit dazu entschieden, ein allgemeines Tracking-Verfahren zu entwickeln, das auf einer externen Hardware läuft, die mit den Anwendungen auf den HMDs kommuniziert. Allerdings ergibt sich auch dann ein Mehrwert

[7] `https://www.meta.com/de/quest/products/quest-2/`, Stand: 23.06.2023
[8] `https://www.meta.com/de/quest/quest-pro/`, Stand: 20.07.2023
[9] `https://www.picoxr.com/de/products/neo3-link`, Stand: 23.06.2023
[10] `https://www.picoxr.com/de/products/pico4`, Stand: 23.06.2023

aus dem Ansatz dieser Arbeit, wenn ein bestimmtes HMD den Zugriff auf die unverarbeiteten Sensordaten erlaubt. So lässt ich das Verfahren dann auf die dort verbauten Kameras anwenden.

2.2. Zustandsfilter

Die Basis dieser Arbeit stellen Bayes-Filter dar. In diesem Abschnitt der Arbeit soll das theoretische Hintergrundwissen vermittelt werden, das zum Verständnis dieser Klasse von Filtern nötig ist. Dabei gehe ich in Absatz 2.2.1 zunächst auf die grundlegende Idee eines Bayes-Filters ein. In Abschnitt 2.2.2 gehe ich dann weiterhin auf das Bayes-Theorem ein, dass die Grundlage für die meisten dieser Filter ist. Anschließend erläutere ich die Funktionsweise der Bayes-Filter anhand des einfachen Kalman Filters, der auf lineare Systeme ausgelegt ist in 2.2.3. Im Anschluss wird in Absatz 2.2.4 erläutert, weshalb sich Kalman Filter in der Regel nicht auf nicht linearen Systemen anwenden lassen und welche Alternativen es hierfür gibt. In Abschnitt 2.2.5 stelle ich den Unscented Kalman Filter als eine solche Alternative detaillierter vor, da er die Grundlage des Tracking-Verfahrens darstellt, das ich im Rahmen dieser Arbeit entwickelt habe. Abschließend beziehe ich mich auf die Beobachtbarkeit von linearen und nicht linearen Systemen.

2.2.1. Grundkonzept von Bayes-Filtern

Laut Labbe (2022a) können Bayes-Filter so betrachtet werden, dass sie ein ungefähres Wissen über den Zustand eines Objektes oder Systems mit dem ungefähren Wissen aus einer Messung zu einer soliden Schätzung vereinen. Diese Vorstellung gibt einen Einblick über das Aufgabenfeld der Bayes-Filter. Mit ihnen lassen sich anhand von Messungen und Wissen über das Verhalten eines Systems begründete Aussagen darüber treffen, wie der Zustand des Systems aussieht und wie sicher diese Aussage ist. In dem Buch “Fundamentals of Object Tracking” (Challa u. a., 2011, S. 22) heißt es übersetzt “Der optimale Bayes’sche Filter ist ein Verfahren zur rekursiven Berechnung der posterioren Dichte des Objektzustands. Schätzprobleme im Bayes’schen Rahmen werden durch die a priori Dichte und die Likelihood-Funktion des betreffenden Parameters definiert. Im Zusammenhang mit dem Objekt-Tracking wird die a priori Dichte durch die dynamische Gleichung des Objekts bestimmt. Die Likelihood-Funktion lässt sich aus der Messgleichung ableiten.”

In den folgenden Absätzen werden die genannten Begriffe im Kontext der Bayes-Filter ausführlicher erklärt. Vereinfacht kann das Konzept allerdings damit beschrieben werden, dass Bayes-Filter zum einen ein gewisses Wissen über ein System haben und zum anderen Mes-

sungen dieses Systems vornehmen können. Anhand des Wissens über das System können sie nun eine Vorhersage darüber treffen, welchen Zustand es zu einem gewissen Zeitpunkt haben dürfte. Anhand der Messung aus dem System können sie dann prüfen, wie gut die Vorhersage ist und diese gegebenenfalls anpassen. Dabei sind weder die Vorhersage, noch die Messung als definitiv korrekt zu sehen. Beide sind mit einer Unsicherheit behaftet. So könnte das Wissen über das System nicht vollständig sein und die Vorhersage somit falsch. Auch das Messgerät könnte einen Fehler haben und die Messungen könnten somit von der Realität abweichen. Die Bayes-Filter können diese Unsicherheiten mit in ihre Schätzungen einbeziehen. Ist zum Beispiel bekannt, dass das Wissen über das System sehr ungenau, aber die Messgeräte dafür sehr genau sind, dann korrigieren sie ihre Vorhersage zugunsten der Messung.

Ein Beispiel im Kontext des Objekt-Trackings, wäre beispielsweise der Versuch, die Höhe eines Flugzeuges zu tracken. Das Flugzeug verfügt über ein Altimeter, das die aktuelle Höhe in Metern über Normalnull angibt. Die Flugeigenschaften eines Flugzeuges lassen sich zudem mithilfe physikalischer Gleichungen gut beschreiben. Die Höhe des Flugzeugs beim Start ist bekannt und beträgt 0 m. Nun lässt sich anhand der physikalischen Gleichungen eine Aussage darüber treffen, wie viel Höhe das Flugzeug nach einer Sekunde gewonnen hat. Beispielsweise ergibt sich hier eine Höhe von 5 m. Bei dieser Schätzung handelt es sich um eine a priori Schätzung. Damit ist eine Schätzung vor dem Einbeziehen der Messdaten gemeint. Nun wird die Messung des Altimeters abgelesen, das eine Höhe von nur 4,6 m angibt. Der Bayes-Filter berechnet nun die a posteriori Schätzung aus diesen beiden Informationen. Dafür ist entscheidend, was über die Genauigkeiten beider Informationen bekannt ist. Die physikalischen Gleichungen beispielsweise lassen spontane Windböen außer Acht, die Einfluss auf den Flug des Flugzeuges nehmen. Das Altimeter ist laut Herstellerangabe bis auf 10 cm genau. Angenommen, die a posteriori Schätzung geht zugunsten der Messung des Altimeters und landet bei 4,7 m. Nun beginnt der Durchgang von vorne. Es wird erneut eine Schätzung anhand der Gleichungen durchgeführt, die mithilfe der Messungen korrigiert wird. Auf diese Art, näher sich der Filter rekursiv der tatsächlichen Höhe des Flugzeugs.

2.2.2. Das Bayes-Theorem im Zusammenhang mit Bayes-Filtern

Das Bayes-Theorem liefert die theoretische Grundlage, das in Absatz 2.2.1 Beschriebene. In “Fundamentals of Object Tracking” (Challa u. a., 2011, S. 8-12) wird das Bayes-Theorem näher erläutert. Formal ausgedrückt lautet es:

$$p(\mathbf{x}|\mathbf{z}) = \frac{p(\mathbf{z}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{z})} \tag{2.1}$$

Wobei $p(\mathbf{x}|\mathbf{z})$ als die Wahrscheinlichkeit gelesen werden kann, dass ein Ereignis $\mathbf{x}$ eintritt, wenn zuvor Ereignis $\mathbf{z}$ eingetreten ist. Die Gleichung 2.1 lässt sich im Kontext der Bayes Filter laut Challa u. a. (2011) und Labbe (2022a) so interpretieren, dass $\mathbf{x}$ den Zustand eines Systems und $\mathbf{z}$ eine Messung aus diesem System darstellt. Bei der Funktion $p()$ handelt es sich um eine Wahrscheinlichkeitsdichtefunktion (Challa u. a., 2011, S.8-9), die ausdrückt, wie wahrscheinlich ein Ereignis wie $\mathbf{x}$ oder $\mathbf{z}$ ist. In Bayes Filtern werden die Zustandsvariablen und Messungen nicht als absolute Werte, sondern als Normalverteilungen gesehen. Diese sind laut Labbe (2022a) wie folgt definiert:

$$f(x, \mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right] \tag{2.2}$$

Dementsprechend lässt sich $p(\mathbf{z}|\mathbf{x})$ als die Wahrscheinlichkeit interpretieren, dass aus dem Zustand $\mathbf{x}$ die Messung $\mathbf{z}$ erzeugt wird. In Gleichung 2.1 beschreibt $p(\mathbf{x})$ die a priori Schätzung eines Filters. Bei $p(\mathbf{z})$ handelt es sich laut Labbe (2022a) um eine Normalisierungskonstante. $p(\mathbf{z}|\mathbf{x})$ ist die a posteriori Schätzung. Anhand des Bayes-Theorems lässt sich also erkennen, wie in Bayes Filtern die Messung oder die Wahrscheinlichkeit für $p(\mathbf{z}|\mathbf{x})$ dazu genutzt wird, die a priori Schätzung $p(\mathbf{x})$ zu aktualisieren, um die posteriore Schätzung $p(\mathbf{x}|\mathbf{z})$ zu erhalten.

2.2.3. Der Kalman Filter

Der klassische Kalman Filter funktioniert ausschließlich mit linearen Systemen. Dabei besteht er laut Labbe (2022a) im ersten Schritt aus den folgenden Schritten:

$$\begin{aligned} \bar{\mathbf{x}} &= \mathbf{F}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \bar{\mathbf{P}} &= \mathbf{F}\mathbf{P}\mathbf{F}^T + \mathbf{Q} \end{aligned} \tag{2.3}$$

Bei den Gleichungen 2.3 handelt es sich um den Vorhersageschritt des Kalman Filters. Hierbei wird mittels des Prozessmodells, das in Form der Matrix $\mathbf{F}$ vorliegt, der a priori Zustand $\bar{\mathbf{x}}$ berechnet. $\mathbf{B}$ beschreibt hierbei die Matrix, die etwaige Eingaben $\mathbf{u}$, wie beispielsweise die Beschleunigung des Flugzeuges, miteinbezieht. $\mathbf{P}$ drückt das Vertrauen des Kalman Filters in den derzeitigen Zustand $\mathbf{x}$ aus. Mittels $\mathbf{F}$ wird dieses Vertrauen an den a priori Zustand $\bar{\mathbf{x}}$ angepasst.

Man nennt $\bar{\mathbf{P}}$ auch die a priori Kovarianz des Filters. Weiterhin ist der Aktualisierungsschritt des Kalman Filters definiert durch folgende Gleichungen:

$$\begin{aligned} \mathbf{y} &= \mathbf{z} - \mathbf{H}\bar{\mathbf{x}} \\ \mathbf{K} &= \bar{\mathbf{P}}\mathbf{H}^\mathsf{T}(\mathbf{H}\bar{\mathbf{P}}\mathbf{H}^\mathsf{T} + \mathbf{R})^{-1} \\ \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{K}\mathbf{y} \\ \mathbf{P} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\bar{\mathbf{P}} \end{aligned} \tag{2.4}$$

Hier wird zunächst das Residuum $\mathbf{y}$ aus den Messdaten und einer aus dem Filterzustand erstellten Messung berechnet. Diese wird mittels der Beobachtungsmatrix $\mathbf{H}$ erzeugt. Mithilfe der Matrix $\bar{\mathbf{P}}$ sowie $\mathbf{H}$ und $\mathbf{R}$ wird anschließend das Kalman Gain $\mathbf{K}$ berechnet. Dieses wird dazu genutzt, unter Einbeziehung des Residuums und des a priori Zustandes den a posteriori Zustand $\mathbf{x}$ sowie die Matrix $\mathbf{P}$ zu berechnen.

2.2.4. Kalman Filter und nicht lineare Systeme

Der klassische Kalman Filter ist auf die Verwendung mit linearen Systemen ausgelegt. Wie in Absatz 2.2.2 erläutert, handelt es sich bei den in Kalman Filtern verwendeten Werten nicht um absolute Zustandswerte, sondern um Normalverteilungen. Diese drücken die Ungenauigkeit der Werte aus. Damit die Kalman Filter funktionieren, muss laut Labbe (2022a) gegeben sein, dass eine Normalverteilung nach Anwendung einer Funktion immer noch eine Normalverteilung darstellt. Betrachtet man den Vorhersage- und den Aktualisierungsschritt des Kalman Filters als Funktion, bringt das die Einschränkung mit sich, dass es sich dabei um lineare Funktionen handeln muss. Nur dann ist laut Labbe (2022a) die Wahrung der Normalverteilung gewährleistet. Für nicht lineare Systeme funktionieren klassische Kalman Filter also nicht.

Challa u. a. (2011) und Labbe (2022a) beschreiben eine Reihe von Alternativen zum klassischen Kalman Filter, die auch die Filterung von nicht linearen vornehmen können. Dazu gehören der EKF, der Partikelfilter (PF) und der UKF. In dieser Arbeit ist der UKF zum Einsatz gekommen, weswegen dieser in Absatz 2.2.5 detaillierter besprochen wird. Der EKF begegnet der Nichtlinearität eines Systems mit der punktuellen Linearisierung des Systems. Hierfür werden zum Zeitpunkt der Schätzung die partiellen Ableitungen des Vorhersage- und Aktualisierungsschritt gebildet und das System somit linearisiert. Dafür werden die Jakobi-Matrizen der nicht linearen Funktionen $f()$ und $h()$ berechnet und als $\mathbf{F}$ und $\mathbf{H}$ im linearen Kalman Filter verwendet. $f()$ beschreibt dabei die Zustandstransferfunktion und erfüllt eine ähnliche Aufgabe wie die Zustandstransfermatrix $\mathbf{F}$. Auf dieselbe Art stellt $h()$ ein Äquivalent zu $\mathbf{H}$ dar.

Sowohl Labbe (2022a) als auch (Challa u. a., 2011, S.36-37) sagen allerdings, dass der UKF in den meisten Fällen bessere Ergebnisse bei der Filterung nicht linearer Systeme liefert, als der EKF. Zudem ist die Berechnung der partiellen Ableitungen in vielen Fällen nicht trivial und braucht laut Labbe (2022a) vergleichbar viel Berechnungsschritte wie der UKF, wenn sie nicht auf analytischem Wegen gefunden werden können.

Eine weitere Alternative stellt der PF dar. Dieser kann im Gegensatz zum UKF und EKF auch mit nicht gaußschen Wahrscheinlichkeitsverteilungen umgehen. Hier ist also die Wahrung einer Normalverteilung nicht entscheidend. Dabei werden laut Labbe (2022a) mehrere tausende oder Hunderttausende Partikel erstellt und gleichförmig um den eigentlichen Zustand verteilt. Die Partikel selbst bestehen jeweils aus einem Zustandsvektor. Mithilfe der Funktion $f()$ werden sie entsprechend der zuvor definierten Systemdynamik fortgeführt. Anschließend werden die realen Messungen mit den aus $h()$ gewonnenen Messungen der einzelnen Partikeln vergleichen. Die Partikel werden dann daran angepasst, wie nahe die Messungen eines Partikels der realen Messungen sind und entsprechend neu verteilt. So bildet sich eine Wahrscheinlichkeitsverteilung um den tatsächlichen Systemzustand. Zwar ist der PF somit sehr zuverlässig und gut im Umgang mit nicht linearen Systemen, allerdings ist der Rechenaufwand hier aufgrund der hohen Anzahl von Partikeln sehr groß.

2.2.5. Der Unscented Kalman Filter

Der UKF ist laut Labbe (2022a) die zu bevorzugende Herangehensweise, um nicht lineare Systeme zu filtern. Anders als der EKF verwendet der UKF keine partiellen Ableitungen für den Zustandstransfer und die Messfunktion. Auch benötigt der UKF keine tausenden Partikel, um den Mittelwert einer Normalverteilung nach der Anwendung einer nicht linearen Funktion zu errechnen. Stattdessen nutzt der UKF Sigma-Punkte und den Unscented Transform (UT), um mit der nicht Linearität eines Systems umzugehen.

Im UT werden die Zustandstransferfunktion $f()$ und die Messfunktion $h()$ des UKF nicht auf den einen Zustand angewendet, sondern auf die Sigma-Punkte. Bei diesen handelt es sich um eine Reihe von Punkten, die um den tatsächlichen Zustand verstreut sind. Somit bilden sie die Normalverteilung, sowie deren Mittelwert ab. Nach der Anwendung einer nicht linearen Funktion lässt sich aus den Sigma-Punkten der Mittelwert erneut berechnen. Gibt man lediglich den Mittelwert durch eine nicht lineare Funktion, scheitert der Kalman Filter, wie in Absatz 2.2.4 beschrieben. Denn eine Normalverteilung bleibt nicht notwendigerweise eine Normalverteilung, nachdem auf sie eine nicht lineare Funktion angewendet wurde. Diese

Problematik können die Sigma-Punkte um UT laut Labbe (2022a) beheben.

In den Gleichungen unter 2.5 ist der gesamte Vorhersageschritt des UKFs beschrieben.

$$\begin{aligned} \boldsymbol{\mathcal{Y}} &= f(\boldsymbol{\chi}) \\ \boldsymbol{\mu_x} = \bar{\mathbf{x}} &= \sum w^m \boldsymbol{\mathcal{Y}} \\ \bar{\mathbf{P}} &= \sum w^c (\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})^\mathsf{T} + \mathbf{Q} \end{aligned} \tag{2.5}$$

Hierbei wird im UT zunächst die Funktion $f()$ auf alle Sigma-Punkte $\boldsymbol{\chi}$ angewendet. Anschließend wird aus diesen in der Zeit fortgeführten Sigma-Punkten der gewichtete Mittelwert berechnet. Dieser stellt die a priori Schätzung des Filters $\bar{\mathbf{x}}$ dar. Im Unscented Transform wird weiterhin die Kovarianzmatrix $\bar{\mathbf{P}}$ berechnet. Diese errechnet sich aus den Sigma-Punkten $\boldsymbol{\mathcal{Y}}$ und $\bar{\mathbf{x}}$.

In den Gleichungen unter 2.6 ist der Aktualisierungsschritt des UKFs beschrieben.

$$\begin{aligned} \boldsymbol{\mathcal{Z}} &= h(\boldsymbol{\mathcal{Y}}) \\ \boldsymbol{\mu}_z &= \sum w^m \boldsymbol{\mathcal{Z}} \\ \mathbf{y} &= \mathbf{z} - \boldsymbol{\mu}_z \\ \mathbf{P}_z &= \sum w^c (\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)(\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)^\mathsf{T} + \mathbf{R} \\ \mathbf{K} &= \left[\sum w^c (\boldsymbol{\mathcal{Y}} - \bar{\mathbf{x}})(\boldsymbol{\mathcal{Z}} - \boldsymbol{\mu}_z)^\mathsf{T}\right] \mathbf{P}_z^{-1} \\ \mathbf{x} &= \bar{\mathbf{x}} + \mathbf{K}\mathbf{y} \\ \mathbf{P} &= \bar{\mathbf{P}} - \mathbf{K}\mathbf{P}_\mathbf{z}\mathbf{K}^\mathsf{T} \end{aligned} \tag{2.6}$$

Hier kommt ebenfalls der UT zu Einsatz. Zunächst wird aus den Sigma-Punkten $\boldsymbol{\mathcal{Y}}$ mittels der Funktion $h()$ eine Messung $\boldsymbol{\mathcal{Z}}$ berechnet. Aus den so transformierten Sigma-Punkten wird für die Messung das gewichtete Mittel $\boldsymbol{\mu}_z$ berechnet. Dieses wird mit der realen Messung $\mathbf{z}$ dazu verwendet, das Residuum zwischen dem jeweiligen Sigma-Punkt und den realen Messwerten zu berechnen. Weiterhin wird mittels $\boldsymbol{\mathcal{Z}}$ und $\boldsymbol{\mu}_z$ die Kovarianz berechnet. Aus der Kreuzkovarianz zwischen den Messdaten und den a priori Zustandsdaten wird anschließend das Kalman Gain $\mathbf{K}$ berechnet. Mit dessen Hilfe wird nun wie beim klassischen Kalman Filter die a posteriori Schätzung des Filters berechnet. Zuletzt wird es verwendet, um aus der a priori und der Kovarianz $\mathbf{P}_z$ die a posteriori Kovarianz $\mathbf{P}$ zu berechnen.

Berechnung der Sigma-Punkte

Laut Labbe (2022a) ist die Berechnung der Sigma-Punkte ein zentraler Aspekt bei der Entwicklung eines UKFs. Hierfür wird in der Regel der Algorithmus von Van Der Merwe und Wan (2004) verwendet. Laut Labbe (2022a) geschieht die Berechnung der Sigma-Punkte mittels des Wertes λ:

$$\lambda = \alpha^2(n + \kappa) - n \tag{2.7}$$

Wobei n den Dimensionen des Zustandsvektors $\mathbf{x}$ entspricht. Die Punkte berechnen sich aus Lambda mittels folgender Gleichung:

$$\boldsymbol{\chi}_i = \begin{cases} \mu + \left[\sqrt{(n+\lambda)\Sigma}\right]_i & i = 1..n \\ \mu - \left[\sqrt{(n+\lambda)\Sigma}\right]_{i-n} & i = (n+1)..2n \end{cases} \tag{2.8}$$

Wobei Σ hier die Kovarianz des Filters $\mathbf{P}$ beschreibt. Somit wird also die Varianz des Filters bei dem Algorithmus von Van Der Merwe und Wan (2004) dazu verwendet, die Sigma-Punkte des Filters zu berechnen. Die Anzahl der Punkte beträgt $2n + 1$, wobei n den Dimensionen des Zustandes $\mathbf{x}$ entspricht.

Jeder Sigma-Punkt verfügt über ein Gewicht, mit dessen Hilfe das gewichtete Mittel aus den Sigma-Punkten gebildet werden kann. Auch dieses wird auf Basis von λ berechnet. Hierbei wird laut Labbe (2022a) zwischen zwei Arten von Gewichten unterschieden. Die Gewichte w^m für den Mittelwert der Sigma-Punkte und die Gewichte w^c für die Berechnung der Kovarianzen. Diese können, aber müssen nicht gleich sein. Für die Gewichte gilt die Einschränkung, dass ihre Summe eins sein muss, also:

$$\begin{aligned} 1 &= \sum_i w_i^m \\ 1 &= \sum_i w_i^c \end{aligned} \tag{2.9}$$

Die Berechnung der Sigma-Punkte und ihrer Gewichtung geschieht unter Einfluss des Parameters λ, der wiederum durch die Parameter α und κ aus Gleichung 2.7 berechnet wird. Zudem wird ein weiterer Parameter β verwendet. Da der erste Sigma-Punkt als $\boldsymbol{\chi}_0 = \mu$ definiert ist, weicht die Berechnung von w_0^m von den übrigen Gewichten ab:

$$w_0^m = \frac{\lambda}{n + \lambda} \tag{2.10}$$

Das Gewicht der Kovarianz von $\boldsymbol{\chi}_0$ wird folgendermaßen berechnet:

$$w_0^c = \frac{\lambda}{n+\lambda} + 1 - \alpha^2 + \beta \tag{2.11}$$

Die Gewichte der übrigens Sigma-Punkte werden folgendermaßen errechnet:

$$w_i^m = w_i^c = \frac{1}{2(n+\lambda)} \quad i = 1..2n \tag{2.12}$$

Die Parameter α, β und κ können bei der Entwicklung des UKFs für ein konkretes System eigenhändig bestimmt werden. Hier gibt Labbe (2022a) Hinweise für sinnvolle Parameter. So sei für gaußverteilte Systeme $\beta = 2$ ein geeigneter Wert. Weiterhin sei κ sinnvollerweise durch $\kappa = 3 - n$ zu bestimmen, wobei n den Dimensionen von $\mathbf{x}$ entspricht. Für α wird der Wertebereich $0 \leq \alpha \leq 1$ als geeignet angegeben. Hier beschreibt Labbe (2022a), dass ein größerer Wert für α für eine größere Streuung der Punkte um den Mittelwert μ sorgt. Sigma-Punkte, die sich näher am Mittelwert einer Zustandsvariable befinden, ermöglichen es, sehr kleine Unterschiede und Effekte abzubilden. Laut Labbe (2022a) eignen sie sich damit für die Verwendung bei nicht linearen Systemen. Sind die Punkte hingegen weiter um den Mittelwert gestreut, bilden sie größere Effekte und nicht normalverteilte Wahrscheinlichkeiten besser ab.

2.2.6. Beobachtbarkeit von Systemen

In ihrer Arbeit treffen Southall u. a. (1998) übersetzt die folgende Aussage: “Während unkontrollierbare Systeme manchmal für die Kalman-Filterung wünschenswert sind, funktioniert ein Kalman-Filter, der auf einem System mit nicht beobachtbaren Zuständen aufgebaut ist, schlichtweg nicht.”
Dementsprechend ist es essenziell, die Beobachtbarkeit des eines Systems zu gewährleisten. In der Arbeit von Southall u. a. (1998) wird sie formal hergeleitet. So ist ein System dann als beobachtbar anzusehen, wenn es für einen Zustand $\mathbf{x}$ zu einem Zeitpunkt t_0 eine endliche Anzahl Messungen des Systems $\mathbf{z}$ gibt, aus denen sich der Zustand $\mathbf{x}$ eindeutig ermitteln lässt. Hierbei dürfen die Messungen auch an mehreren, aber endlichen Zeitschritten $t_1 > t_0$ erhoben werden. Ein nicht beobachtbares System definiert sich dementsprechend dadurch, dass der Zustand $\mathbf{x}$ nicht eindeutig aus den Messungen $\mathbf{z}$ ermittelt werden kann, unabhängig davon, wie viele Zeitschritte miteinbezogen werden.

Während es für lineare Systeme einfache Tests gibt, um die Beobachtbarkeit eines Systems zu

beweisen, existieren diese laut Southall u. a. (1998) nicht. Allerdings kann die nicht vorhandene Beobachtbarkeit durch einen Gegenbeweis ermittelt werden. So zum Beispiel, indem gezeigt wird, dass zwei unterschiedliche Zustände zum selben Zeitpunkt dieselben Messwerte erzeugen können. In diesem Fall ist eine eindeutige Ermittlung von $\mathbf{x}$ aus $\mathbf{z}$ nicht gewährleistet. Dieser Ansatz wird in Absatz 3.10.3 für das im Rahmen dieser Arbeit entwickelte Tracking-System demonstriert.

2.3. Objekt-Tracking

Das Tracking von Objekten wird in "Fundamentals of Object Tracking" von Challa u. a. (2011) ausführlich behandelt und als das Problem definiert, bei dem Messungen von Sensoren dazu benutzt werden, den Ort, den Weg und Charakteristika eines Objektes zu bestimmten. Dabei können verschiedene Sensoren zum Einsatz kommen. So auch Kameras, wie sie in dieser Arbeit verwendet werden. Hierbei kann zwischen verschiedenen Arten des Objekt-Trackings unterschieden werden, die sich durch verschiedene Ansprüche und Probleme unterschieden, die sie mit sich bringen. So kann beispielsweise nach der Dynamik des jeweiligen Objektes unterschieden werden. Laut Challa u. a. (2011) handelt es sich bei Objekten, deren dynamisches Verhalten sich mitunter sprunghaft verändert, um manövrierende Objekte. Das Tracken manövrierender Objekte, muss dementsprechend mit der Herausforderung umgehen, die diese Änderung des dynamischen Verhaltens mit sich bringt. Bei den Objekten, die das Tracking-Verfahren dieser Arbeit tracken sollen, handelt es sich um manövrierende Objekte.

Wie in Absatz 2.2 beschrieben, ist laut Labbe (2022a) das Prozessrauschen $\mathbf{Q}$ ein Weg, mit der Ungenauigkeit eines Prozessmodells umzugehen. Diesen Weg haben auch Gsaxner u. a. (2021) gewählt. Auch in "Fundamentals of Object Tracking" (Challa u. a., 2011, S. 62) werden eine Reihe von Methoden beschrieben, um manövrierende Objekte zu tracken. Sie lassen sich demnach in zwei Kategorien aufteilen. Die Ansätze der ersten Kategorie basieren im Wesentlichen ebenfalls auf der Verwendung von Rauschen. Beispielsweise kann hier zudem versucht werden, Manöver zu erkennen und das Rauschen des Filters daran anzupassen. Die Ansätze der zweiten Kategorie setzen vorwiegend auf die Verwendungen von verschiedenen Prozessmodellen. Die Tracking-Verfahren können zwischen verschiedenen Prozessmodellen, die verschiedene dynamische Verhalten beschrieben, wechseln und versuchen sich so dem tatsächlichen Verhalten anzupassen.

Objekte, die auf ein dynamisches Verhalten beschränkt sind, das sich mittels eines Prozessmo-

dells abbilden lässt, sind keine manövrierenden Objekte. Ein Beispiel hierfür wäre beispielsweise ein Auto oder ein Roboterarm mit einer festen Anzahl von Gelenken.

Auch die Anzahl der zu trackenden Objekte kann ein Unterscheidungskriterium darstellen. So kann zwischen dem Tracking einzelner Objekte und mehrerer Objekte unterschieden werden (Challa u. a., 2011, S. 3). In dieser Arbeit wird sich auf das Tracking einzelner Objekte fokussiert.

Ein weiterer wichtiger Begriff im Kontext des Objekt-Trackings ist "Clutter". Clutter beschreibt im wesentlichen die Unordentlichkeit der zur Verfügung stehenden Messungen. So kann in einem idealen Tracking-System beschrieben werden, welche Messungen ein Sensor zu jedem Zeitpunkt liefert. In der Realität kann es allerdings passieren, dass sich diese Messung zwischen Zeitschritten auf eine zufällige Art verändert (Challa u. a., 2011, S. 103). Diese Veränderung kann verschiedene Gründe haben. So können falsche Objekte gemessen werden oder aber korrekte Objekte, die eigentlich gemessen werden sollten, nicht erkannt werden. Diese Problematik kommt auch im Rahmen dieser Arbeit zum Tragen.

2.4. Definition wichtiger Begriffe

Für das weitere Verständnis dieser Arbeit sind zwei Begriffsdefinitionen entscheidend. Zum einen unterscheide ich in der folgenden Arbeit zwischen einer Orientierung und einer Rotation. Eine Orientierung beschreibt die Drehung eines Objektes zu einem Zeitpunkt t im Raum. Sie lässt sich als Quaternion, mithilfe von Euler-Winkeln oder in einer Rotationsmatrix darstellen. Eine Rotation hingegen beschreibt den Übergang von einer Orientierung in eine weitere. Sie lässt sich genauso darstellen wie die Orientierung.

Weiterhin verwende ich eine eigene Definition von dem Begriff Inside-Out-Tracking. In einem Teil der genannten Literatur beschreibt das Inside-Out-Tracking ein Tracking, bei dem die Sensoren, die für das Tracking nötig sind, innerhalb des zu trackenden Objektes platziert sind. Ich unterscheide hingegen zwischen einem Tracking, bei dem externe Sensoren um die Nutzer*innen statisch platziert werden und einem, bei dem die Sensorik mobil und am Nutzer oder am Objekt platziert ist. Technisch gesehen handelt es sich bei dem in Absatz 2.1.1 beschriebenen Vive Tracker um ein Inside-Out-Tracking. Da hier aber eine Basisstation stationär verwendet wird, die dementsprechend den Tracking-Bereich limitiert, definiere ich das Verfahren ebenfalls als Outside-In-Tracking. Streng genommen handelt es sich bei dem in

dieser Arbeit entwickelten Verfahren also um ein mobiles Outside-In-Tracking, das an einem HMD befestigt wird. Da die Vorzüge des Systems aber nahe an denen des Inside-Out-Trackings eines HMDs liegen, definiere ich es als Inside-Out-Trackings. Dabei folge ich dem Ansatz von Gsaxner u. a. (2021).

3. Methodischer Teil

In diesem Kapitel ist das methodische Vorgehen der Arbeit beschrieben. In Absatz 3.1 gehe ich hierfür zunächst auf die Auswahl der Instrumente ein, mit deren Hilfe ich die Frage der Arbeit beantworte. Anschließend ist in Absatz 3.2 das Konzept des Tracking-Verfahrens geschildert, das ich in dieser Arbeit entwickelt habe. In Abschnitt 3.3 gehe ich auf die Simulation einer realen Kamera ein, die ich statt einer physischen Kamera benutze. Daran anknüpfend ist in Absatz 3.4 die Kalibrierung eben dieser Kamera beschrieben. In Absatz 3.5 ist die virtuelle Kamera beschrieben, die für die Generierung von Messdaten sowie für das Tracking-Verfahren verwendet wird. Sie basiert auf der zuvor beschriebene simulierten Kamera. Die für das Tracking-Verfahren erzeugte Markerpose und ihr Aufbau sind in Absatz 3.6 hergeleitet und beschrieben.

Abschnitt 3.7 beschreibt weiterhin, wie die Messdaten mittels der in Unity simulierten oder der physischen Kamera erstellt werden. Somit wird hier auch der Aufbau der Messdaten beschrieben, der im weiteren Verlauf simuliert werden kann. Anschließend wird die Siulation der Messdaten aus dem gewonnenen Wissen über die realen Messdaten in Absatz 3.8 beschrieben.

In Absatz 3.9 beschreibe ich anschließend die Bestimmung der Objektpose mittels eines deterministischen Ansatzes. Daraufhin widmet sich Absatz 3.10 dem in dieser Arbeit entwickelt und UKF-basierten Tracking-Verfahren. Zuletzt ist in Absatz 3.11 die Evaluation des UKF-basierten Tracking-Systems beschrieben.

3.1. Auswahl von Instrumenten und Methoden

In diesem Absatz widme ich mich den von mir ausgewählten Instrumenten und Methoden, um die Forschungsfragen dieser Arbeit zu beantworten. Dabei beschreibe ich in Absatz 3.1.1 zunächst, weshalb ich einen UKF und keinen EKF verwende. Anschließend widme ich mich in Absatz 3.1.2 den von mir verwendeten Geräten. Allerdings kommt die dort beschriebene reale Kamera in dieser Arbeit nicht weiter zum Einsatz und wird nur als Grundlage verwendet.

3.1.1. Wahl des Filter-Typen

Sowohl laut Labbe (2022a) als auch laut Challa u. a. (2011) und Kraft (2003) ist der UKF dem EKF vorzuziehen. So ist der UKF in der Regel akkurater, als auch weniger rechenintensiv. Die Berechnung der Jacobi-Matrizen ist laut Labbe (2022a) und Kraft (2003) dann rechenaufwendig, wenn sie nicht analytisch zu ermitteln sind. Das ist in der Regel nicht der Fall. Ich habe mich auf diese Aussagen gestützt und den UKF gewählt. Weiterhin grenze ich mich damit von der Arbeit von Gsaxner u. a. (2021) ab, um die Eignung des UKFs für ein derartiges Tracking-Verfahren zu ermitteln.

3.1.2. Verwendete Geräte

Um eine Unabhängigkeit von der Zugänglichkeit unverarbeiteter Sensor- und Kameradaten eines HMDs zu gewährleisten, habe ich mich bei der Konzeption des Tracking-Verfahrens für eine externe Stereokamera entschieden. Allerdings habe ich die reale, physische Kamera nur als Grundlage für die in dieser Arbeit verwendete simulierte Kamera genutzt. Somit konnte ich die Bild- und Markererkennung anhand realer Kamerabilder testen. Weiterhin ist damit validiert, dass die Verwendung nicht simulierter Hardware für das entwickelte Tracking-System keine Einschränkung darstellt. Es ist essenziell zeigen zu können, dass die simulierten Geräte an reale Geräte angelehnt sind.

Zudem habe ich die Kommunikation zwischen der realen Kamera und der VR-Anwendung definiert und diese mit einem Prototyp getestet. Ich habe mich hier für die Verwendung eines Rasperry Pi 4[1] entschieden, der über einen Websocket mit der VR-Anwendung kommuniziert. Auf dem Rasperry Pi 4 läuft auch das Tracking-Verfahren. Da an diesen jedoch nur eine Kamera zurzeit angeschlossen werden kann, ist zusätzlich das Arducam Camarray HAT[2] erforderlich. Dieses ermöglicht es, die Signale von bis zu vier über den 15-Pin Kamera-Anschluss des Rasperry Pi 4 angeschlossene Kameras zu synchronisieren. Dem Rasperry Pi werden die beiden Bilder dann als ein einzelnes Bild angezeigt. An das Arducam Camarray HAT habe ich zwei OV9281 Pivariety Kameras angeschlossen. Diese verfügen über eine Pixelgröße von 0,003 mm mal 0,003 mm sowie eine effektive Brennweite von 2,8 mm. Das FOV der einzelnen Kameras beträgt 70° in der Horizontalen und 43,75° in der Vertikalen. Die Wahl des Rasperry Pis als Basis für die Stereokamera liegt in der Offenheit des Systems. Hier können einfach unterschiedliche Kameras mit unterschiedlichen Auflösungen, Brennweiten und Sichtfeldern angeschlossen werden. So

[1] `https://www.raspberrypi.com/products/raspberry-pi-4-model-b/`, Stand: 20.07.2023
[2] `https://tinyurl.com/arducamHAT`, Stand: 20.07.2023

lässt sich das System später an die individuellen Bedürfnisse anpassen. Ein weiterer Vorteil liegt in der Möglichkeit, zwei weitere Kameras anzuschließen. Somit bleibt die Option, das Tracking dadurch zu verbessern oder den sichtbaren Bereich zu vergrößern. Ich habe mich jedoch für die Verwendung von zwei Kameras entschieden. Teil dieser Arbeit war es, zu prüfen, wie genau ein derartiges Verfahren sein kann bei der minimalsten Hardware-Konfiguration. Hierbei spielt neben den Kosten, die dafür aufgewendet werden müssen, auch die Anzahl der Bauteile eine Rolle.

Die Markerpose, die zum Tracking an einem Objekt angebracht wird, besteht aus Kugeln, die mit einer stark reflektierenden Beschichtung überzogen sind. Der Durchmesser der einzelnen Kugeln beträgt 2 cm. Durch ein Konstrukt werden diese Kugeln an den in Abschnitt 3.6 beschriebenen Koordinaten fixiert. Werden die Kugeln direkt angestrahlt und fällt das Licht zurück in eine Kamera, liegt der entsprechende Pixelwert für die gesamte Kugel bei 255 und somit bei ihrem Maximalwert. Die Repräsentation eines Objektes mittels dieser reflektierenden Marker vereinfacht die Erkennung der entsprechenden Punkte im Bild enorm. Mittels Feature-Erkennungs-Verfahren wäre es denkbar, auf diese zu verzichten. Für diese Arbeit habe ich mich allerdings dagegen entschieden, um den Fokus auf das eigentliche Tracking-Verfahren zu legen. Als HMD wird die Meta Quest 2 verwendet. Hier lässt sich jedes mögliche HMD verwenden, da für den Tracking-Algorithmus irrelevant ist, wie das Gerät spezifisch funktioniert. Lediglich die Ausführbarkeit eines Programms, dass einen Websocket ausführen kann, muss gewährleistet sein.

Im Rahmen dieser Arbeit habe ich für die Evaluation keine reale Hardware genutzt, sondern diese innerhalb einer VR-Umgebung simuliert. So ließen sich absolut kontrollierbare Rahmenbedingungen schaffen, die ich mit physischen Geräten nicht erreichen konnte. Zudem ließen sich so alle Fehler, die die einzelnen Bestandteile des Tracking-Systems hatten, genauestens ermitteln und analysieren. Die Simulation der Kameras erfolgt innerhalb der Game Engine Unity[3] und wird in Abschnitt 3.3 genauer erläutert. Zudem habe ich die Meta Quest 2 verwendet, um in einer selbst entwickelten VR-Anwendung die Bewegungsdaten für die sich bewegende Tracking-Kamera sowie ein zu trackendes Objekt aufzuzeichnen. Diese Daten lassen sich in der Evaluation des Tracking-Verfahrens als Ground Truth-Daten verwenden. Die Erzeugung der Bewegungsdaten innerhalb einer VR-Anwendung nutzt somit das Tracking der VR-Brille als zusätzliches Tracking-System, hält aber den Versuchsaufbau sehr überschaubar. Auch die in Abschnitt 3.6 beschriebene Markerpose habe ich in der VR-Anwendung als

[3] `https://unity.com/de`, Stand: 20.07.2023

3D-Modell erstellt. So ließ sich diese mit höchster Präzision erstellen und Fehler durch eine ungenaue Bauweise konnten für die Entwicklung des Verfahrens ausgeschlossen werden.

3.1.3. Verwendete Software

Für die Entwicklung verschiedener VR-Anwendungen habe ich die Game Engine Unity sowie das XR Interaction Toolkit[4] genutzt. 3D-Modelle wurden in der 3D-Software Blender[5] in der Version 3.3 verwendet.

Das UKF-basierte Verfahren habe ich mithilfe der Bibliothek filterpy Labbe (2022b) umgesetzt, die von Labbe (2022a) entwickelt wurde. Sie ist als Quelle angeführt, da in der Dokumentation von filterpy einige Hinweise und Handlungsempfehlungen enthalten sind, die ich ebenfalls verwendet habe.

3.2. Tracking-Algorithmus Grundidee

In diesem Absatz ist das grundlegende Konzept des Tracking-Verfahrens beschrieben. Dieses Konzept habe ich in dieser Arbeit nicht gänzlich umgesetzt. So habe ich statt der physischen Hardware eine in Unity simulierte Kamera verwendet. Auch die Bilderkennung habe ich zwar implementiert und evaluiert, bei der Erstellung von Messdaten kommt sie allerdings nicht zum Einsatz. Ebenso habe ich die Zuordnung von Markern eines dreidimensionalen Festkörpers zwar umgesetzt, aber im Rahmen der Untersuchung des Tracking-Verfahrens nicht miteinbezogen. Ich habe an manchen Stellen dieses Kapitels dennoch Bezug auf Bestandteile des hier beschriebenen Tracking-Verfahrens genommen, da sie für eine weitere Forschung an dem Verfahren essenziell sind und die Erkenntnisse aus der Arbeit an ihnen somit wichtig ist.

Die Umsetzung des Verfahrens wird in den folgenden Abschnitten dieses Kapitels beschrieben. Die Beschreibung der Grundidee soll dabei einen Überblick über das Verfahren vermitteln, um die anschließend beschriebenen Details in einem detaillierteren Kontext besser verständlich zu machen. Auch soll sie deutlich machen, wie das finalisierte Verfahren aussehen soll. Alle simulierte und umgesetzten Bestandteile orientieren sich stets an diesem Konzept. Im Anhang in Abbildung A.1 ist eine Übersicht der umgesetzten Bestandteile des Verfahrens zu finden. Die grundlegende Idee für den Tracking-Algorithmus basiert auf dem Ansatz von Gsaxner

[4] `https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.3/manual/index.html`, Stand: 20.07.2023
[5] `https://www.blender.org/download/lts/3-3/`, Stand: 20.07.2023

u. a. (2021). Allerdings verwende ich anstatt eines EKFs einen UKF. Die Gründe hierfür sind in Absatz 3.1.1 näher definiert. Außerdem handelt es sich bei den von mir erhobenen Messungen um die Daten eine synchronisierten Stereokamera. Gsaxner u. a. (2021) verwenden einen SCAAT-EKF, der auf der Arbeit von Welch und Bishop (1997) basiert, da die Stereokamera der von Gsaxner u. a. (2021) verwendeten HoloLens 2[6] nicht synchronisiert ist.

Der Tracking-Filter arbeitet mit Messdaten der in Abschnitt 3.1.2 beschriebenen simulierten Stereokamera. Aus den Bildern der Stereokamera werden die reflektierenden Marker mittels einiger bildverarbeitenden Funktionen und einer Konturerkennung erkannt und ihre Position wird in Bildkoordinaten gespeichert. Dabei werden die Pixelkoordinaten anhand der Sensorgröße in Millimeter umgerechnet. Diese Bildkoordinaten stellen die Messwerte des Systems dar. Die Marker sind Teil eines zuvor definierten Festkörpers, der als Markerpose definiert ist. Die Koordinaten der Marker innerhalb des Festkörpers sind dem UKF-basierten Tracking bekannt. Mithilfe dieses Wissens kann der UKF auf Basis des Filterzustandes eine virtuelle Markerpose erstellen und diese mithilfe der durch eine Kamerakalibrierung erhaltenen Projektionsmatrizen der linken und rechten Kamera der Stereokamera in Bildkoordinaten umwandeln. Die Position der Kamera kann über die Position des HMDs ermittelt werden. Mithilfe dieser Position lassen sich zu jedem Zeitschritt die Projektionsmatrizen beider Kameras anpassen.

Der UKF macht mithilfe eines einfach Positions- und Geschwindigkeitsmodells vorhersagen über die Bewegung des Objektes und berechnet so den a priori Zustand. Der Zustand des Objektes enthält sowohl die Position, die Geschwindigkeit, sowie die inkrementelle Orientierung und die Winkelgeschwindigkeiten. Die absolute Orientierung ist nicht Teil des Zustands und wird wie bei Gsaxner u. a. (2021) und Welch und Bishop (1997) außerhalb als Quaternion gespeichert und in jedem Zeitschritt mit der inkrementellen Orientierung verrechnet. Die Begründung hierfür findet sich in Abschnitt 3.10.1.

Trifft eine Messung der realen Stereokamera ein, erzeugt der UKF eine virtuelle Messung auf Basis des Filterzustandes. Anschließend wird der so entstandene Messvektor mit dem realen Messvektor verglichen und mithilfe des so entstehenden Residuums und dem Kalman Gain die Vorhersage des Filters unter Einbeziehung der Messdaten verbessert und der a posteriori Zustand errechnet.

[6] `https://www.microsoft.com/de-de/hololens/buy`, Stand: 20.07.2023

3.3. Simulation der realen Kamera

Wie in Abschnitt 3.1.2 bereits beschrieben, habe ich die Stereokamera, die auf dem Headset angebracht werden soll, zunächst in der Game Engine Unity simuliert. Diese Vorgehensweise wird in Abschnitt 3.3.1 zunächst ausführlich beschrieben und anschließend in Absatz 3.3.2 tiefergehend begründet.

Bei der in diesem Abschnitt beschriebenen simulierten Kamera handelt es sich im Kontext des Tracking-Verfahrens um die reale Kamera. Sie ist vergleichbar mit der in Abschnitt 3.1.2 beschriebenen physischen Kamera für den Rasperry Pi 4. Sie ist weiterhin nicht zu verwechseln mit der in Absatz 3.5 beschriebenen Filterkamera. Bei dieser handelt es sich um eine weitere Abstraktionsebene, die mithilfe der Kamerakalibrierung auf Basis der realen Kamera erstellt wurde. Die Filterkamera erzeugt Messdaten und wird im Filter verwendet.

3.3.1. Aufbau der Kamera-Simulation

Die Stereo-Kamera, die das Tracking-Verfahren nutzt, ist in der Idee des Verfahrens an oder auf einem HMD montiert. Dementsprechend benötigt sie Bewegungsdaten der Kopfbewegung von Nutzer*innen. In Unity habe ich hierfür eine Szene erstellt und mittels des XR-Interaction-Toolkits ein XR-Origin-Objekt erstellt. Dieses enthält ein Kameraobjekt, das die Kamera darstellt, deren stereoskopische Bilder auf das verwendete HMD gerendert werden. Orientierung und Position dieses Kameraobjektes basieren auf den Bewegungen der verwendeten Meta Quest 2. Zusätzlich habe ich ein Objekt mit dem Namen “Stereokamera” erstellt. Dieses ist in der Hierarchie der Unity-Szene ein untergeordnetes Objekt der Hauptkamera und bewegt sich dementsprechend mit den Bewegungen des Kopfes. Somit konnte ich die Montage einer Stereokamera auf einem HMD simulieren. Das Stereokamera-Objekt wiederum enthält zwei Kameraobjekte mit den Namen “Linke Kamera” und “Rechte Kamera”. Bei beiden handelt es sich um Unity-Kameras, die als physikalische Kameras[7] betrachtet werden. Das bedeutet, für sie lassen sich eine Brennweite sowie die Größe und Verschiebung des Sensors auf der x- und y-Ebene einstellen. In Tabelle 3.1 sind die Werte beider Kameras zu sehen. Da beide Kameras der Stereokamera simuliert sind, gleichen sie sich exakt. Für eine physische Kamera könnte es zwischen beiden Kameras zu Abweichungen kommen.

[7] `https://docs.unity3d.com/Manual/PhysicalCameras.html`, Stand: 20.07.2023

Tabelle 3.1.: Werte der in Unity simulierten Kamera

Brennweite	Sensorgröße x/y	Linsenverschiebung x\y	Sichtfeld horizontal	Sichtfeld vertikal
20,78 mm	36 mm / 24 mm	0 mm / 0mm	81,79°	60°

Um Bilder der Kamera zu Unity zu exportieren und mittels OpenCV OpenCV (2022) und der in Abschnitt 3.7.2 beschriebenen Markererkennung zu verarbeiten, müssen diese aus Unity exportiert werden können. Dabei ist es entscheidend, dass sie das korrekte Seitenverhältnis vorweisen. Hinsichtlich der Bildgröße habe ich mich an der Bildgröße der Bilder der Rasperry-Pi-Kamera orientiert. Dementsprechend habe ich in Unity zwei Objekte des Typs "RenderTexture"[8] erstellt. Diese können einer Kamera hinzugefügt werden. Das Bild der Kamera wird dann auf die entsprechende Textur gerendert. Die Render-Texturen sind jeweils 1200x800 Pixel groß. Um in Unity Bilder und Videos im png- beziehungsweise mp4-Format aufzunehmen, habe ich den "Unity Recorder"[9] verwendet. Dabei handelt es sich um ein Editor-Werkzeug, mit dem Videos und Bilder in Unity aufgenommen werden können. Hier lassen sich die Anzahl der Bilder pro Sekunde sowie das Ausgabeformat einstellen. Die Render-Texturen können als Quelle hinzugefügt werden. So ließen sich synchrone Bilder beider Kameras speichern und in Python-Skripten verwenden.

3.3.2. Begründung für die simulierte Kamera

Dieser Absatz widmet sich der Frage, wieso die Kamera simuliert und nicht die reale Kamera verwendet wurde, wenn doch, wie in Absatz 3.5 beschrieben, die Messdaten nicht mittels der simulierten Kamera erzeugt werden. Der Grund hierfür liegt in der Vermeidung von Fehlern bei der Datengenerierung. Wie in Absatz 3.8 noch erläutert wird, werden die Messdaten anhand der Filterkamera und der Bewegung von Kamera und Objekt erzeugt. Indem die Filterkamera auf der in Unity simulierten Kamera basiert und nicht auf der realen Kamera, lässt sich bei der Aufzeichnung von Bewegungsdaten exakt angeben, ob und wie lange das zu trackende Objekt im Sichtfeld der Stereokamera war, da bei der Datenaufzeichnung der Bewegungsdaten das Sichtfeld als Nutzer mit dem Sichtfeld der Stereokamera gleichzusetzen war. Weiterhin ließ sich so die Position im Weltkoordinatensystem der in Unity simulierten Kamera besser definieren, da sie einfach in der VR-Anwendung gesetzt und ausgelesen werden konnte. Dasselbe für die physische Kamera durchzuführen wäre mit Ungenauigkeiten verbunden gewesen, da sie nur in Relation zu der Position des Kamera-Objektes in der VR-Anwendung vorgelegen hätte.

[8] `https://docs.unity3d.com/ScriptReference/RenderTexture.html`
[9] `https://unitytech.github.io/unity-recorder/manual/index.html`, Stand: 20.07.2023

Zudem ließen sich so die Kameraparameter der Kamera in Unity einfach anpassen, ohne dass die echten Geräte ausgetauscht werden mussten.

Ich habe die Filterkamera auch auf Basis der physischen, realen Kamera für den Rasperry Pi 4 erstellt, um zu validieren, dass durch die Simulation der Kamera in Unity keine Fehler auftreten.

3.4. Kalibrierung der Stereokamera

Ich habe die Kalibrierung sowohl mit der physischen Kamera für den Rasperry Pi 4 als auch mit der simulierten Kamera in Unity durchgeführt. Im Laufe der Arbeit wird allerdings lediglich die simulierte Kamera beschrieben. Mithilfe der simulierten Kamera lässt sich überprüfen, ob die anhand der Unity-Kamera ermittelte Projektion der Filterkamera korrekt sind und ob die in Absatz 3.5 beschriebene Filterkamera somit korrekt funktioniert.

Mittels der Kamerakalibrierung lassen sich die intrinsischen und extrinsischen Kameraparameter ermitteln. Zu Ersteren gehören die Brennweite f und die Koordinaten des optischen Zentrums der Kamera c. Die extrinsischen Parameter bestehen aus der Orientierung sowie der Position der Kamera im Raum und sind definiert durch die Rotationsmatrix $\boldsymbol{R}$ und den Translationsvektor $\boldsymbol{T}$. Für die Kalibrierung der Kamera gehe ich von dem Modell einer Lochkamera aus, wie es in Semeniuta (2016) definiert ist. Für die Kalibrierung der Kameras habe ich die Funktionen von OpenCV (2022) verwendet.

Zur Ermittlung der intrinsischen Parameter muss die Kameramatrix $\boldsymbol{M}$ mit der einfachen Kamerakalibrierung für die jeweilige Kamera errechnet werden. Dabei werden im Falle der Stereokamera die Matrizen $\boldsymbol{M}_l$ und $\boldsymbol{M}_r$ für die linke und rechte Kamera zunächst unabhängig voneinander berechnet. Die Kameramatrix ist definiert als:

$$\boldsymbol{M} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.1}$$

Durchgeführt wurde die Kalibrierung der Kamera mittels eines 5x8 Schachbrettmusters. Die Ecken des Schachbretts stellen die Punkte dar, die mithilfe von OpenCV (2022) im Bild der jeweiligen Kamera erkannt werden und somit im Bildkoordinatensystem vorliegen. Das Welt-Koordinatensystem ist als das Koordinatensystem des Schachbrettmusters definiert. Dabei ist

die z-Komponente der Positions-Vektoren immer gleich null. Die Koordinaten der Eckpunkte habe ich im Vorfeld definiert. Hierbei ist es wichtig, dass diese korrekt skaliert sind. Das also die Kante an der Position $p = [1, 1]$ mit der realen Kantenlänge multipliziert wird. Diese betrug 21 mm. Dieser Wert ist abhängig von dem verwendeten Schachbrettmuster. Für die Kalibrierung der in Unity simulierten Stereokamera habe ich das Schachbrettmuster in der 3D-Software Blender in der korrekten Größe modelliert. Diese habe ich in Unity importiert und dort Bilder mittels der simulierten Stereokamera aufgenommen.

Hierbei wurden pro Kamera insgesamt 20 Bilder des Schachbrettmusters gemacht, die Eckpunkte der Quadrate im Bild erkannt und ihre Pixelkoordinaten gespeichert. Abbildung 3.1 zeigt ein Bild aus der Unity-Simulation, auf dem die erkannten Ecken des Schachbrettmusters eingezeichnet wurden. Es liegen somit zwei Mengen vor E_w, in der die Eckpunkte in Weltkoordinaten gespeichert sind und E_i in der die korrespondierenden Punkte in Bildkoordinaten vorliegen. OpenCV verfügt über eine Funktion[10], die die entsprechenden Kanten findet und in der korrekten Reihenfolge zurückgibt.

Mittels E_w und E_i berechne ich nun die Verzerrungskoeffizienten k_1, k_2, k_3 für die radiale Verzerrung sowie p_1 und p_2 für die tangentiale Verzerrung. Außerdem werden die intrinsischen Kameramatrizen $\boldsymbol{M}_l$ sowie $\boldsymbol{M}_r$ für die linke und rechte Kamera der Stereokamera berechnet. Hierfür nutze ich die Funktion “calibrateCamera” aus OpenCV[11] Mithilfe dieser Werte lässt sich der Fehler der Kalibrierung bestimmen, indem die Punkte aus E_w auf Basis der ermittelten Werte erneut projiziert werden und anschließend die Abweichung zu den Punkten in E_i ermittelt wird. Der durchschnittliche Fehler für beide Kameras liegt bei 0.04 Pixeln, ist also sehr gering.
Neben den intrinsischen Parametern werden zudem die extrinsischen Parameter der Stereokamera benötigt. Diese habe ich mithilfe der Stereokalibrierung[12] aus OpenCV berechnet. Hierfür werden ebenfalls die Punkte E_w sowie die Bildpunkte der erkannten Ecken aus der rechten und linken Kamera E_{i_r} und E_{i_l} benötigt.

[10] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga93efa9b0aa890de240ca32b11253dd4a, Stand: 20.07.2023
[11] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga3207604e4b1a1758aa66acb6ed5aa65d, Stand: 20.07.2023
[12] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga9d2539c1ebcda647487a616bdf0fc716, Stand: 20.07.2023

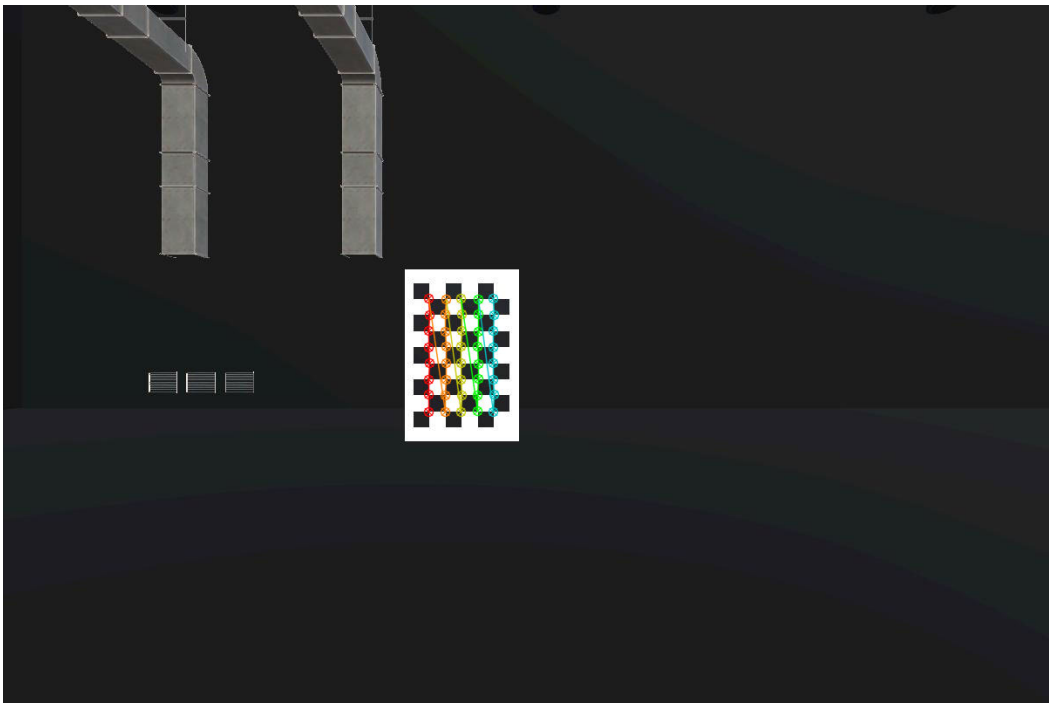


Abbildung 3.1.: Bild des Schachbrettmusters mit erkannten Eckpunkten aus der Unity-Simulation

Gesucht sind die Orientierung der Kamera $\boldsymbol{R}$ und die Position $\boldsymbol{T}$. Der Einfachheit halber wird während der Kalibrierung davon ausgegangen, dass sich die linke Kamera im Nullpunkt des Weltkoordinatensystems befindet und sie nicht rotiert ist:

$$\boldsymbol{RT}_l = \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{array}\right] \tag{3.2}$$

Damit werden $\boldsymbol{R}_r$ und $\boldsymbol{T}_r$ immer in Bezug auf die linke Kamera ausgedrückt:

$$\boldsymbol{RT}_r = \left[\begin{array}{ccc|c} 0.999 & 0.001 & 0.005 & 0.073 \\ -0.001 & 0.999 & -0.002 & -0.001 \\ -0.001 & 0.002 & 0.999 & -0.011 \end{array}\right] \tag{3.3}$$

Anhand der extrinsischen Parameter lässt sich ablesen, dass das Ergebnis der Kalibrierung Sinn ergibt. Denn dem Aufbau der Stereokamera nach liegen beide Kameras auf derselben Ebene und schauen in dieselbe Richtung. Wie $\boldsymbol{RT}_r$ zeigt, ist $\boldsymbol{R} \approx \boldsymbol{I}$, also nicht rotiert. $\boldsymbol{T}$ zeigt eine

Verschiebung auf der x-Achse um etwa 7 cm. Auch das entspricht annäherungsweise dem Bau der Kamera. Die simulierte Unity-Kamera ist auf der x-Achse allerdings 8 cm auseinander. Der sich dadurch ergebene Fehler ist in Absatz 3.5 beschrieben. Dort wird auch erläutert, weshalb er für die in dieser Arbeit durchgeführte Untersuchung irrelevant ist.

Da sich die Kamera später bewegen kann, und sich somit ihre Position und Orientierung verändert, kann das Wissen über das Verhältnis zwischen der linken und rechten Kamera dazu genutzt werden, um $\boldsymbol{R}_r$ und $\boldsymbol{T}_r$ neu zu berechnen, sobald sich die Position des HMDs ändert. Hierfür ist lediglich das Wissen über die Position der linken Kamera im Koordinatensystem des HMDs nötig. Die Bewegung der Filterkamera sowie ihr Aufbau sind in Absatz 3.5 genauer beschrieben. Für die Stereokalibrierung werden nun die zuvor ermittelten Bildpunkte beider Kameras E_{i_l} und E_{i_r} sowie die Objektpunkte E_w, $\boldsymbol{M}_l$, $\boldsymbol{M}_r$ und die Verzerrungskoeffizienten verwendet. Die entsprechende Funktion[13] aus OpenCV liefert anschließend $\boldsymbol{R}_r$ und $\boldsymbol{T}_r$. Die Projektionsmatrizen für die linke und rechte Kamera $\boldsymbol{P}_l$ und $\boldsymbol{P}_r$ werden nun wie folgt berechnet:

$$\boldsymbol{P} = \boldsymbol{M} \cdot \boldsymbol{RT} \tag{3.4}$$

3.5. Aufbau der Filterkamera

Unabhängig von der Frage, ob eine echte Kamera oder eine in Unity simulierte Kamera verwendet wird. Der Filter und, wie in Absatz 3.7 beschrieben, auch die Generierung der Messdaten zur Evaluation des Filters und dessen Qualität, brauchen Zugriff auf eine simulierte Kamera. Hierbei handelt es sich nicht um die in Unity simulierten Kamera. Um beide besser voneinander unterscheidbar zu machen, nenne ich diese Kamera die “Filterkamera”, da sie unter anderem Verwendung in der Messfunktion des Filters findet. Die Filterkamera definiert sich durch die Eigenschaft, dass mit ihrer Hilfe Objektpunkte aus dem Weltkoordinatensystem in das Bildkoordinatensystem der linken und rechten Kamera I_l und I_r überführt werden können. Hierbei ist es sowohl für die in Abschnitt 3.8 beschriebene Messdatensimulation sowie die in Abschnitt 3.10.2 definierte Messfunktion entscheidend, dass sich die Filterkamera so ähnlich zu der realen Kamera verhält, wie möglich. Mit realer Kamera ist in diesem Fall die in Unity simulierte Kamera gemeint. Ich habe mich für Verwendung der Filterkamera zur Messdatenerzeugung entschieden, da die Messdaten so direkt aus den dreidimensionalen Weltkoordinaten der Marker einer Markerpose berechnet werden können. Damit ist es nicht notwendig, die

[13] https://docs.opencv.org/4.x/d9/d0c/group__calib3d.html#ga9d2539c1ebcda647487a616bdf0fc716

Markererkennung für die Evaluation des UKF-basierten Tracking zu verwenden, was es nötig machen würde, ein Verfahren zu entwickeln, um die Einhaltung einer festen Sortierung der Messvektoren einzuhalten. Dieses Verfahren könnte ebenso zu Fehlern führen, die das Ergebnis weiter verfälschen und den Fokus auf die Untersuchung des Tracking-Filters verringern.

Die Kamera des Tracking-Systems ist als bewegliche Kamera angelegt, da sie auf einem mit 6DoF beweglichen HMD montiert sein soll. Dementsprechend ändern sich zu jedem Zeitpunkt die Projektionsmatrizen der Stereokamera. Hierfür greift die Filterkamera zu jedem verfügbaren Zeitschritt auf die Position und Orientierung des HMDs zu. Diese liegen als dreidimensionaler Vektor $\mathbf{t}_{cam} \in \mathbb{R}^3$ und als Quaternion $\mathbf{q}_{cam} \in \mathbb{H}$ vor:

$$\mathbf{t}_{cam} = [x, y, z], \mathbf{q}_{cam} = [q_x, q_y, q_z, q_w] \tag{3.5}$$

Die Projektionsmatrix setzt sich wie in Abschnitt 3.4 beschrieben aus den intrinsischen und extrinsischen Parametern der Kamera zusammen. Die intrinsischen Parameter liegen dank der Kamerakalibrierung als Matrix $\boldsymbol{M}$ vor. Nach der Kalibrierung liegt die Kamera im Nullpunkt des Welt-Koordinatensystems und ist nicht rotiert. Nun können $\mathbf{t}_{cam}$ und $\mathbf{q}_{cam}$ dazu verwendet werden, die extrinsischen Parameter der Kamera zu erneuern und die Projektion somit von einem anderen Punkt im Weltkoordinatensystem aus durchzuführen. Hierfür muss die Pose der Kamera allerdings invertiert werden. Innerhalb der Projektionsmatrix $\boldsymbol{P}$ ist $\boldsymbol{RT}$ als die Matrix zu verstehen, die die Weltkoordinaten eines Punktes $\mathbf{p}_W = [x, y, z]$ in das Koordinatensystem der Kamera überführt.

Anschließend stellt $\boldsymbol{M}$ die Überführung von dem Punkt im Kamerakoordinatensystem der Kamera $\mathbf{p}_{cam}$ in das Bildkoordinatensystem $\mathbf{p}_i$ dar. Ist also eine Pose bestehend aus $\mathbf{t}_{cam}$ und $\mathbf{q}_{cam}$ gegeben, berechnet sich die angepasste Projektionsmatrix $\boldsymbol{P}$ wie folgt:

$$\boldsymbol{P} = \Big[\mathbf{M}\Big] \left[\begin{array}{c|c} \mathbf{R}^T & -\mathbf{R}^T\mathbf{T} \end{array}\right] \tag{3.6}$$

Diese Rechnung wird zunächst für $\boldsymbol{P}_l$ vorgenommen. $\boldsymbol{P}_r$ lässt sich nun mittels der extrinsischen Parameter, die für die rechte Kamera in Absatz 3.4 bereits ermittelt wurden, einfach berechnen. Mithilfe dieser Überlegung lässt sich die Bewegung der Kamera im Raum abbil-

den. Die Projektion eines Bildpunktes geschieht nun mittels $\boldsymbol{P}_l$ und $\boldsymbol{P}_r$. Hierbei lautet die Gleichung für die Projektion eines Punktes im Raum $\mathbf{p}_W = [x, y, z]$ folgendermaßen.

$$\mathbf{p}'_{img} = \begin{bmatrix} x_i \\ y_i \\ w \end{bmatrix} = \Big[\mathbf{M}\Big] \left[\begin{array}{c|c} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{T} \end{array} \right] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \boldsymbol{P} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.7}$$

Nach der Projektion liegt p_{img} in homogenen Koordinaten vor. Um den Bildpunkt in kartesische Koordinaten umzurechnen, muss der folgende Schritt durchgeführt werden:

$$\boldsymbol{p}_{img} = \begin{bmatrix} \frac{p'_{img_x}}{w} \\ \frac{p'_{img_y}}{w} \end{bmatrix} \tag{3.8}$$

Hierbei sei zusätzlich erwähnt, dass die obige Invertierung der Kamerapose ausschließlich für die linke Kamera durchgeführt wird. Für die rechte Kamera liegen wie in Abschnitt 3.4 beschrieben $\boldsymbol{R}_r$ und $\boldsymbol{T}_r$ vor. Diese beschreiben die Überführung von Objektpunkten im Koordinatensystem der linken Kamera in das der rechten Kamera. Mittels folgender Gleichung werden $\boldsymbol{R}_r$ und $\boldsymbol{T}_r$ aus der mit $\mathbf{t}_{cam}$ und $\mathbf{q}_{cam}$ aktualisierten Pose $\boldsymbol{R}_l$ und $\boldsymbol{T}_l$ errechnet:

$$\begin{aligned} \boldsymbol{R}_{r_{new}} &= \boldsymbol{R}_r \boldsymbol{R}_l \\ \boldsymbol{T}_{r_{new}} &= \boldsymbol{R}_r \boldsymbol{T}_l + \boldsymbol{T}_r \end{aligned} \tag{3.9}$$

Für die Umsetzung der Kameraprojektion kommen zusätzlich die Verzerrungskoeffizienten der Kameraprojektion zum Einsatz. Auch diese werden bei der Kamerakalibrierung in Absatz 3.4 ermittelt. Ich verwende für die Projektion von Punkten die in OpenCV (2022) definierte Funktion "projectPoints"[14]. Diese führt die Kameraprojektion, anhand der zuvor ermittelten intrinsischen und extrinsischen Kameraparametern sowie den Verzerrungskoeffizienten durch. Die radiale und die tangentiale Verzerrung ist dabei laut OpenCV (2023) folgendermaßen definiert:

$$\begin{aligned} x'_i &= x_i(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y'_i &= y_i(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ x''_i &= x'_i + [2p_1 x'_i y'_i + p_2(r^2 + 2x'^2_i)] \\ y''_i &= y'_i + [p1(r2 + 2y'_i 2) + 2p2x'_i y'_i] \end{aligned} \tag{3.10}$$

[14] https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html#ga1019495a2c8d1743ed5cc23fa0daff8c, Stand: 20.07.2023

In Abbildung 3.2 ist eine beispielhafte Projektion der Markerpose mittels der Filterkamera zu erkennen.

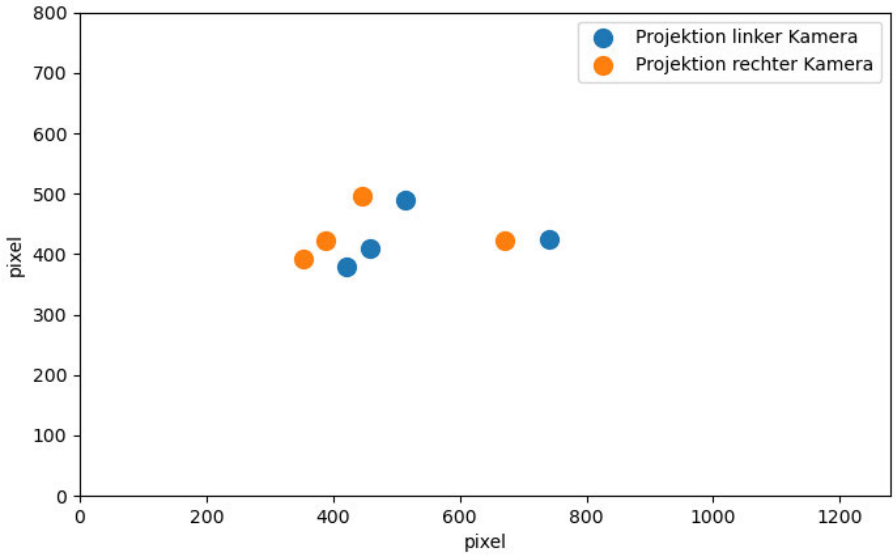


Abbildung 3.2.: Projektion einer Markerpose beider Kameras

Bevor die Filterkamera Verwendung in der Messdatengenerierung und in der Messfunktion des UKFs findet, muss geprüft werden, wie gut sie funktioniert. Hierbei sei erwähnt, dass eine Kamerakalibrierung in der Regel einen gewissen Fehler aufweist. Abbildung 3.3 zeigt den Fehler der Filterkamera.

Die dort grün eingezeichneten Kreuze sind die ermittelten Koordinaten der Marker, die mithilfe der Markererkennung in dem tatsächlichen Bild gefunden wurden. Das Bild wurde mittels der in Unity simulierten Kamera aufgenommen. Die roten Kreuze stellen das Ergebnis einer Messdatensimulation unter Verwendung der Filterkamera dar. Die Markerpose hatte sowohl für die in Unity simulierte Kamera als auch bei der Messdatensimulation dieselbe Position und Orientierung. Hier ist zu sehen, dass die Projektion grundsätzlich sinnvoll ist und stimmt. Allerdings ist ein leichter Versatz in beiden Bildern zu sehen. Dieser geht auf den Fehler in der Kamerakalibrierung zurück.

Es ließe sich argumentieren, dass der beschriebene Fehler das Ergebnis des Filters verfälscht. Dem ist allerdings nicht so. Wie in Abschnitt 3.8.2 beschrieben, kommt für die Generierung der Messdaten nicht die in Unity simulierte Kamera, die in Abbildung 3.3 durch die grünen Marker dargestellt ist, sondern die Filterkamera zum Einsatz. Für das Gelingen des Filters ist es entscheidend, dass die Kameraprojektion in der Messfunktion (siehe 3.10.2) und die Kameraprojektion, die Messdaten generiert, ähnlich sind. Das ist somit gegeben, da beide dieselbe Projektion verwenden. Die in Unity simulierte Kamera, als Basis für die Filterkamera zu verwenden hat den Zweck, dass der in Abbildung 3.3 gezeigte Fehler besser untersucht werden konnte. Auch ist hier die Änderung der Kameraparameter flexibler. So war es einfach, die Brennweite oder die Sensorgröße zu erhöhen.

(a) Projektion der Markerpose mit linker Filterkamera

(b) Projektion der Markerpose mit rechter Filterkamera

Abbildung 3.3.: Projektion stereo Filterkamera - rote Punkte entsprechen der Filterkamera, grüne Punkte entsprechen der tatsächlichen Projektion

3.6. Aufbau der Markerpose

Die Markerpose besteht aus vier reflektierenden, sphärischen Markern $\mathbf{m}_i$, die wie folgt angeordnet sind:

$$\begin{aligned} \mathbf{m}_0 &= [0, 0] \\ \mathbf{m}_1 &= [3.125, 3.903] \\ \mathbf{m}_2 &= [-1.939, -1.6194] \\ \mathbf{m}_3 &= [15, 0] \end{aligned} \tag{3.11}$$

Wobei i der jeweiligen Marker-ID entspricht. Der Aufbau der Markerpose ist so gewählt, dass die Abstände zwischen den Markern $\mathbf{m}_i$ möglichst verschieden sind. Dieser Ansatz folgt der Arbeit von Steinicke u. a. (2007).

Diesem Ansatz nach habe ich zunächst die Menge aller Abstände zwischen den Markern einer Markerpose betrachtet. Diese Menge ist für die Markerpose M_r definiert als D_{M_r}. Die Menge aller möglichen Abstandsmengen D_M ist D.

Ich beschränke mich in dieser Arbeit auf lediglich eine Markerpose für das Trackingsystem. Damit ist für das in dieser Arbeit entwickelte Tracking-System die Menge D definiert als $D = \{D_{M_r}\}$. Diese Entscheidung ist auch durch die Größe der reflektierenden Marker begünstigt. Diese verringert die Menge aller unterscheidbaren Distanzen, die für die Verwendung in einem Trackingsystem infrage kommen. Die Markergröße nimmt Einfluss auf die Granularität des Systems. Die Granularität g beschreibt die kleinste messbare Distanz zwischen zwei Markern. Die verwendeten Marker haben einen Durchmesser von 2cm. Ich habe daher $g = 12,5$ mm definiert. Denn Steinicke u. a. (2007) schreiben hierzu übersetzt: “Wenn ein Punkt P mit einer Genauigkeit getrackt wird, die mittels g bestimmt wird, kann sicher behauptet werden, dass sich der Punkt P in einer Sphäre mit Radius g um diesen Punkt befindet. Deshalb sind zwei Distanzen zwischen Marker dann gut definiert, wenn sie sich um mindestens $2g$ unterscheiden”. Dieser Zusammenhang wird in Gleichung 3.12 deutlich.

Die maximale Größe der Markerpose ist für das im Rahmen dieser Arbeit entwickelte System als $y = 200$ mm definiert und limitiert damit den größten Eintrag aus C dar. Die Menge C stellt die Konfiguration der Markerpose dar. In C sind alle unterscheidbaren Distanzen definiert, die für die Verwendung in einem Trackingsystem infrage kommen und den durch y und g definierten Kriterien entsprechen. Für die Menge der tatsächlich verwendeten Distanzen gilt $D \in C$. Die Konfiguration ist laut Steinicke u. a. (2007) wie folgt definiert:

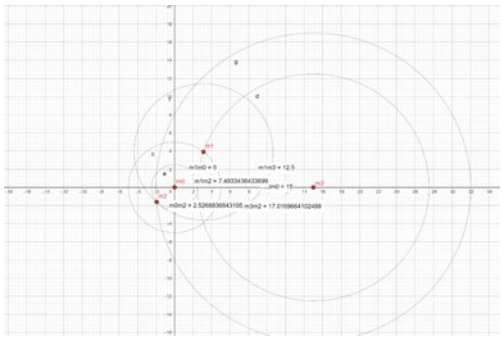
$$C = \{d_i | d_i := 2 \cdot i \cdot g; i = 1, ..., \lfloor \frac{y}{2g} \rfloor\} \tag{3.12}$$

Wie bereits beschrieben, habe ich für diese Arbeit lediglich eine Markerpose erstellt und verwendet. Theoretisch lassen sich mit dieser Methode weitere erstellen. Praktisch sind die dafür verwendeten Marker etwas zu groß. Ich habe die Größe von C für verschiedene y unter der Annahme getestet, dass eine kleinere Markerpose handlicher und somit zu bevorzugen ist. Für $y = 200$ mm beinhaltet C lediglich sieben Einträge. Die Markergröße bei Steinicke u. a. (2007) beträgt gerade einmal 4 mm. Hierbei sind entsprechend mehr Markerposen möglich. Allerdings ist eine Markerpose für die in dieser Arbeit angestellten Untersuchung mehr als ausreichend.

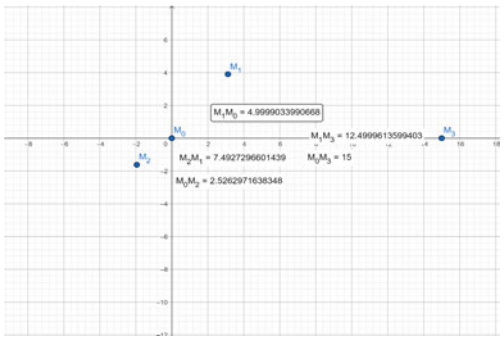
Die einzelnen Marker müssen so platziert werden, dass alle Abstände zwischen allen Markern aus C stammen. Es wäre theoretisch möglich, diese händisch zu platzieren. Allerdings wird das bei vier Markern, wie ich sie verwende, beinahe unmöglich. Daher habe ich ein Gleichungssystem aufgestellt und dieses näherungsweise numerisch gelöst:

$$\begin{aligned}
d(m_0m_1)^2 &= (m_{0x} - m_{1x})^2 + (m_{0y} - m_{1y})^2 \\
d(m_0m_2)^2 &= (m_{0x} - m_{2x})^2 + (m_{0y} - m_{2y})^2 \\
d(m_0m_3)^2 &= (m_{0x} - m_{3x})^2 + (m_{0y} - m_{3y})^2 \\
d(m_1m_2)^2 &= (m_{1x} - m_{2x})^2 + (m_{1y} - m_{2y})^2 \\
d(m_1m_3)^2 &= (m_{1x} - m_{3x})^2 + (m_{1y} - m_{3y})^2 \\
d(m_2m_3)^2 &= (m_{2x} - m_{3x})^2 + (m_{2y} - m_{3y})^2
\end{aligned} \tag{3.13}$$

Hierfür habe ich die SciPy-Funktion “fsolve()”[15] genutzt. Deren Genauigkeit hängt jedoch von einer ungefähren Schätzung ab, von der ausgehend die Variablen des Gleichungssystems berechnet werden können. Mittels der Geometrie-Software GeoGebra[16] habe ich also eine ungefähre Annäherung an die Markerkoordinaten konstruiert und die so entstandenen Koordinaten als Variablen des Gleichungssystems an die fsolve-Funktion gegeben. Die Konstruktion, sowie das Ergebnis sind in Abbildung 3.4 zu sehen.



(a) Händsche Konstruktion der Markerpose



(b) Lösung des Gleichungssystems 3.13

Abbildung 3.4.: Konstruktionsskizze und Ergebnis der Markerpose

[15] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.optimize.fsolve.html`, Stand: 20.07.2023

[16] `https://www.geogebra.org/?lang=de`, Stand: 20.07.2023

3.6.1. Die eindeutige Erkennbarkeit der Markerpose

Ich habe in dieser Arbeit mit Simulationsdaten gearbeitet und damit das Problem der Sortierung des Messvektors umgangen. Allerdings wäre es in der Praxis entscheidend, Marker, die erkannt wurden, eindeutig einander zuzuordnen. Diese Notwendigkeit begründet das zuvor beschriebene Verfahren. Wie in Absatz 3.9 und 3.10.9 noch beschrieben wird, nutzt das UKF-basierte Verfahren einen deterministischen Ansatz, um den Filter zu initialisieren. Auch wird die Berechnung der Tiefe der Pose im Koordinatensystem immer wieder mittels der deterministischen Bestimmung der Pose berechnet (siehe 3.10.3). Hierfür müssen nach der dreidimensionalen Rekonstruktion der Marker mittels Triangulation die Marker der rekonstruierten Pose den Markern der zuvor definierten Pose eindeutig zugeordnet werden können. Dafür verwenden Steinicke u. a. (2007) die beschriebenen internen Distanzen. Ich habe dieses Verfahren zwar nicht mit in dieser Arbeit aufgenommen, aber dennoch getestet, ob es mit der erstellen Pose möglich ist, die Zuordnung der Marker durchzuführen. Eine zuvor definierte Markerpose, die nicht nach dem in diesem Absatz beschriebenen Verfahren erstellt wurde, konnte schon bei leichtem Rauschen nicht mehr eindeutig zugeordnet werden.

Hier haben sich die internen Distanzen durch das Rauschen der Messung zu stark geändert, sodass die Kantenlängen zwischen Markern aus der Rekonstruktion und der Vorlage nicht mehr eindeutig zugeordnet werden konnten. Dank der für C definierten Granularität g konnte ich dieses Problem beheben. Zwar handelt es sich hierbei um keine absolute Gewissheit, dass die Marker immer korrekt zugeordnet werden können, aber das Problem trat selten bis nie auf. Auch wenn es für die Evaluation dieser Arbeit keine Rolle spielt, liefert der Aufbau der Markerpose dennoch eine Grundlage für weitere Forschung.

3.6.2. Erstellung einer virtuellen Markerpose

Sowohl die Messdatensimulation in Abschnitt 3.8.2, als auch die Messfunktion des UKFs in Abschnitt 3.10.2 benötigen die Möglichkeit, eine virtuelle Markerpose aus dem Zustand des zu trackenden Objektes zu generieren. Hierbei handelt es sich um eine virtuelle Repräsentation der in Absatz 3.6 definierten Markerpose bestehend aus den Markern $\mathbf{m}_0$, $\mathbf{m}_1$, $\mathbf{m}_2$ und $\mathbf{m}_3$. Diese ist so definiert, dass das zu trackende Objekt im Ursprung des Weltkoordinatensystems liegt und nicht rotiert wurde. Dementsprechend müssen für die Erstellung der virtuellen Markerpose die Orientierung $\mathbf{q}_w$ und die Translation $\mathbf{t}_{pose}$ des Objektzustandes auf diese Marker angewandt werden.

Das Objekt liegt, wie in Absatz 3.10.1 noch näher erläutert als ein Zustand $\mathbf{x}$, bestehend aus der Position und der inkrementellen Orientierung vor sowie deren ersten zeitlichen Ableitungen, vor. Die inkrementelle Orientierung beschreibt die Rotation des Objektes seit dem letzten Zeitschritt. Zusätzliche ist die globale Orientierung des Objektes in $\mathbf{q}_w \in \mathbb{H}$ als Quaternion gespeichert.

Für die Erstellung der virtuellen Markerpose wird zunächst die inkrementelle Orientierung bestehend aus ϕ, θ und ψ auf die globale Orientierung $\mathbf{q}_w$ angewandt, um so die korrekte Orientierung des Objektes zu erhalten (siehe 3.10.1). Anschließend werden mittels der Position $\mathbf{t}_{pose} = [x, y, z]$ aus $\mathbf{x}$ und $\mathbf{q}_w$ die Positionen der Marker aus der Markerpose berechnet. Um $\mathbf{q}_w$ auf einen Marker $\mathbf{m}_i \in \mathbb{R}^3$ anzuwenden, kann $\mathbf{q}_w$ zunächst in eine Rotationsmatrix umgerechnet werden. Das ist nicht nötig, da sich der Quaternion auch anderweitig auf einen Vektor aus $\mathbb{R}^3$ anwenden lässt, stellt aber eine valide Möglichkeit dar. Für die meisten Operationen mit Quaternionen habe ich das Spatial-Transform-Paket[17] der Bibliothek SciPy verwendet. Insbesondere Rotationen sind aufgrund vieler einzelner Operationen sehr fehleranfällig. In der folgenden Gleichung definiere ich die Anwendung von $\mathbf{q}_w$ auf einen Marker $\mathbf{m}_i$ mittels der aus $\mathbf{q}_w$ gewonnen Rotationsmatrix $\boldsymbol{R}_{qw}$. Es wird also für jeden Marker $\mathbf{m}_i$ der Markerpose folgende Gleichung angewandt:

$$\mathbf{m}_i' = \boldsymbol{R}_{qw} \cdot \mathbf{m}_i + \mathbf{t}_{pose} \tag{3.14}$$

3.7. Erzeugung der realen Messdaten

Da der Filter, wie in Abschnitt 3.11 beschrieben, auf Basis der Simulationsdaten evaluiert und hinsichtlich seiner Genauigkeit überprüft wird, ist es entscheidend, dass nicht nur der Aufbau, sondern auch die Entstehung der Messdaten genau verstanden werden. Die Informationen über Rauschgrößen sowie den Ausfall von beispielsweise einzelnen Markern der Markerpose im Bild muss nachvollziehbar sein, um diese Phänomene anschließend realitätsnah simulieren zu können. In diesem Abschnitt beschreibe ich zunächst die Erzeugung der Messdaten aus Bildern einer Kamera. Hierbei kann es sich um eine reale, physische Kamera handeln. Aus Gründen der Überprüfbarkeit beziehe ich mich auch hier auf die in Abschnitt 3.3 beschriebene in Unity simulierte Kamera. Die folgenden Schritte zur Erhebung von Messdaten wurden sowohl für die in Unity simulierte Kamera als auch die physische Kamera für den Rasperry Pi 4 getestet. In Abschnitt 3.7.1 sind zunächst die Sortierung und der Aufbau des Messvektors beschrieben.

[17] `https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.transform.Rotation.html`, Stand: 20.07.2023

Anschließend gehe ich in 3.7.2 auf die Erkennung von Markern im Bild der Kamera ein. Zwar verwende ich diese Markererkennung schlussendlich für die Evaluation des Filters nicht, doch ist es wichtig sie zu definieren. Aus der Markererkennung ergeben sich nämlich diverse Fehlerquellen, die bei der Simulation von Messdaten beachtet werden müssen. Auf diese Fehlerquellen und den Umgang des Tracking-Verfahrens damit gehe ich in Abschnitt 3.7.3 näher ein.

3.7.1. Sortierung des Messvektors

Damit im UKF zwischen der durchschnittlichen virtuellen Messungen $\mu_{\mathbf{z}}$ der einzelnen Sigma-Punkte $\mathcal{X}$ und der realen Messung $\mathbf{z}$ der Sensoren ein Residuum $\mathbf{y}$ berechnet werden kann, müssen die Messvektoren beider Messungen einer definierten Form entsprechen. Der im Filter verwendete Messvektor entsteht aus der von der Kamera erzeugten Messung. Dieser beinhaltet die Bildkoordinaten aller Marker der Markerpose. Die Einträge des Vektors sind im UKF eindimensional und nicht mehr zweidimensional, woraus sich folgender Messvektor ergibt:

$$\mathbf{z} = [x_{1_l}, y_{1_l}, x_{2_l}, y_{2_l}, x_{3_l}, y_{3_l}, x_{4_l}, y_{4_l}, x_{1_r}, y_{1_r}, x_{2_r}, y_{2_r}, x_{3_r}, y_{3_r}, x_{4_r}, y_{4_r}] \tag{3.15}$$

Dabei stehen die ersten acht Werte in $\mathbf{z}$ für die erkannten Marker in der linken Kamera und die folgenden acht Einträge für die erkannten Marker in der rechten Kamera. Die x- und y-Koordinate stehen jeweils nebeneinander im Vektor. Die Nummerierung der Einträge orientiert sich dabei an den IDs der Markerpose. Diese ist im Vorfeld definiert und besteht aus den Markern mit den in 3.6 beschriebenen Koordinaten. Wie bereits erläutert ist es entscheidend, dass die Messvektoren korrekt sortiert sind. Also, dass zum Beispiel die erste beiden Einträge aus $\mathbf{z}$ den Koordinaten des ersten Markers $\mathbf{m}_0$ aus I_l entsprechen. Bei der Simulation der Daten, ist diese Einhaltung einfach zu gewährleisten, da die Marker einfach immer in derselben Reihenfolge erzeugt werden können. Anders sieht das bei der Verwendung von Realdaten aus. Hierfür bräuchte es ein zusätzliches System, dass die Sortierung des Messvektors gewährleistet. Dieses habe ich im Rahmen dieser Arbeit nicht umgesetzt.

Marker, die sich außerhalb des Bildes befinden, werden auf den Wert $\mathbf{m}_i = [0, 0]$ gesetzt. Das ist zulässig, da von der Kamera real erfasste Marker niemals diesen Wert annehmen können. Damit das möglich ist, muss sich das Zentrum eines der Marker der Pose genau in der Ecke des Bildes befinden. Falls dem allerdings so ist, sind drei Viertel des projizierten Kreises nicht im Bild zu sehen. Die in Absatz 3.7.2 beschriebene Erkennung der Marker, würde

die so entstehende Kontur nicht als Marker erkennen. Dementsprechend ist es zulässig, die Koordinaten $[0, 0]$ für die Kennzeichnung von nicht gefundenen Markern zu verwenden. Die Notwendigkeit einer solchen Kennzeichnung besteht in der beizubehaltenden Länge eines Messvektors. Dieser muss immer die x- und y-Koordinaten aller vier Marker aus I_l und I_r beinhalten.

3.7.2. Markererkennung und Bildverarbeitung

Die in diesem Abschnitt beschriebene Markererkennung und Bildverarbeitung bezieht sich zunächst auf die Markererkennung im Bild der physischen Kamera. Aus dieser ergeben sich praktische Gegebenheiten, die für die Simulation der Messdaten von Bedeutung sind. Sie wurde ebenso für die Bilder der in Unity simulierten Kamera getestet.

Da die Synchronität der physischen Stereokamera dadurch erreicht wird, dass beide Bilder von dem Arducam Camarray HAT zu einem Bild zusammengefasst werden, muss dieses große Bild zunächst in zwei einzelne Bilder mit zwei eigenen Bildkoordinatensystemen I_l und I_r geteilt werden. Dabei können die erkannten Marker $\mathbf{m}_{i_l} = [x_{i_l}, y_{i_l}]$ und $\mathbf{m}_{i_r} = [x_{i_r}, y_{i_r}]$ aus I_l und I_r nur innerhalb der Grenzen des Bildes liegen. Jedes Bild hat eine maximale Auflösung von 1200 x 800 Pixeln. Die Bildgröße gilt sowohl für die physische als auch die in Unity simulierte Kamera. Letztere liefert allerdings zwei separate, synchronisierte Bilder, womit die Zerteilung des Gesamtbildes entfällt. Für die x- und y-Koordinaten der erkannten Marker $\mathbf{m}$ gilt also:

$$\begin{aligned} 0 \leq x_{i_l} \leq 1200 \forall x_{i_l} \in I_l \wedge 0 \leq x_{i_r} \leq 1200 \forall x_{i_r} \in I_r \\ 0 \leq y_{i_l} \leq 800 \forall y_{i_l} \in I_l \wedge 0 \leq y_{i_r} \leq 800 \forall y_{i_r} \in I_r \end{aligned} \tag{3.16}$$

Der zeitliche Abstand zwischen zwei Messungen ist schwankend, beträgt für die physische Kamera im Mittel aber etwa 0,25 Sekunden. Dieser Wert ergibt sich aus der Geschwindigkeit der Markererkennung. Die in Unity simulierte Kamera läuft mit einem festen Zeitschritt von 60 Bildern pro Sekunde. Im weiteren Verlauf der Arbeit ist dies der Zeitschritt für alle weiteren Bereiche. Die physische Stereokamera verfügt zudem über eine Lichtquelle, die den zu trackenden Bereich anstrahlt. Das Licht wird von den reflektierenden Markern reflektiert und in die Kamera zurückgeworfen. Diesen Ansatz verfolgen auch Gsaxner u. a. (2021).

Das so entstehende Bild wird zunächst mit einem Mittelwertrauschen[18] belegt. Weiterhin wird eine Threshold-Funktion[19] auf das Kamerabild angewendet, um die reflektierenden Marker, die sehr hell leuchten, von dem Hintergrund des Bildes zu trennen. Weiterhin werden mit einer OpenCV-Funktion[20] die Konturen der Marker im Bild gesucht. Dieser Schritt wird durch die vorherige Segmentierung erleichtert. Die gefundenen Konturen werden gespeichert und auf ihre Kreisigkeit überprüft. So sollen die Objekte, die nach der Segmentierung des Bildes mittels Threshold-Funktion fälschlicherweise noch sichtbar sind, aussortiert werden.

Jede erfasste Kontur im Bild besteht aus einer Menge von Punkten K für die gilt $K \subset I$. Somit gelten die Einschränkungen aus Gleichung 3.16 für alle $\mathbf{k} \in K$. Die Elemente $\mathbf{k}$ aus K liegen allesamt auf dem Umriss der jeweiligen Kontur. Mittels OpenCV lassen sich für jede Kontur K ihr Umfang[21] u sowie ihr Mittelpunkt[22] $\mathbf{c}$ berechnen. Beide Werte lassen sich zur Überprüfung der Kreisigkeit der jeweiligen Kontur verwenden. Für jedes erkannte K wird nun ein Sample K_{rand} erstellt. Hierfür werden zufällig 30 Punkte dieser Kontur gewählt. Für jedes $\mathbf{k}_i \in K_{rand}$ wird nun der zugehörige Radius r_i berechnet.

$$r_i = ||\mathbf{c} - \mathbf{k}_i|| \tag{3.17}$$

Anschließend werden die so ermittelten Radien verwendet, um einen Umfang u_{guess} mittels der Formel $u = 2r\pi$ zu berechnen. Bei u_{guess} handelt es sich nicht notwendigerweise um den tatsächlichen Umfang u, der zu K gehört. Hier habe ich die Annahme getroffen, dass die Differenz zwischen der Schätzung und dem tatsächlichen Umfang kleiner wird, je runder K ist. Ergeben die Punkte $\mathbf{k}_i$ aus K einen idealen Kreis, dann gilt für jeden Punkt $\mathbf{k}_i \in K_{rand}$ und damit für den zugehörigen Radius r_i die folgende Gleichung:

$$u = 2r_i\pi = u_{guess} \tag{3.18}$$

Zwar ließe sich diese Prüfung auch für jeden der Punkte $\mathbf{k}$ untersuchen, allerdings habe ich mich aus zwei Gründe dagegen entschieden.

[18] https://docs.opencv.org/3.4/d4/d86/group__imgproc__filter.html#ga564869aa33e58769b4469101aac458f9, Stand: 20.07.2023
[19] https://docs.opencv.org/4.x/d7/d1b/group__imgproc__misc.html#ggaa9e58d2860d4afa658ef70a9b1115576a147222a96556ebc1d948b372bcd7ac59, Stand: 20.07.2023
[20] https://docs.opencv.org/3.4/d3/dc0/group__imgproc__shape.html#ga17ed9f5d79ae97bd4c7cf18403e1689a, Stand: 20.07.2023
[21] https://docs.opencv.org/4.x/d3/dc0/group__imgproc__shape.html#ga8d26483c636be6b35c3ec6335798a47c, Stand: 20.07.2023
[22] https://docs.opencv.org/4.x/d8/d23/classcv_1_1Moments.html, Stand: 20.07.2023

Zum einen geschieht diese Prüfung zur Laufzeit und auf jedem von der Stereokamera erfassten Bild. Somit ist es hier vonnöten auf die Laufzeitgeschwindigkeit zu achten und die Zahl der Operationen so gering wie möglich zu halten. Insbesondere, da die Berechnungen auf einem Rasperry Pi 4 laufen, der eine begrenzte Rechengeschwindigkeit vorweist. Zum anderen ist die Prüfung aller Punkte der Kontur nicht nötig, da nicht für jeden Punkt gegeben sein muss, dass dieser auf dem Umfang eines perfekten Kreises liegt. Denn durch den Bau der Markerpose und die Anbringung an einem Objekt, kann es zu Verdeckung kommen, die die Kontur eines Kreises beeinflussen.

Daher habe ich im nächsten Schritt für alle so berechneten u_{guess_i} der Punkte k_i die entsprechende Differenz d_i gemäß folgender Gleichung berechnet:

$$d = |u_{guess} - u| \tag{3.19}$$

Anschließend habe ich hiervon den Mittelwert gebildet und die prozentuale Abweichung von dem realen Umfang berechnet. Über einen Schwellwert lässt sich nun bestimmen, ab wann ein Marker noch mit einbezogen wird und wann er als falsche Kontur erkannt und nicht als Marker interpretiert wird. Dieser Schwellwert kann beispielsweise an die Lichtverhältnisse angepasst werden. Ist die Umgebung sehr hell oder gibt es andere Objekte, die Licht stark reflektieren, dann sollte der Schwellwert geringer sein. Gibt es dagegen nur wenig äußeres Licht und einen neutralen Hintergrund, dann kann der Schwellwert größer sein, da es unwahrscheinlich ist, falsche Marker zu erkennen. In letzterem Fall wäre die Erkennung zudem robuster gegenüber Verdeckungen durch die Pose, eine Hand oder das zu trackende Objekt selbst, da die Abweichung vom idealen Umfang größer sein dürften. Wird für einen Marker eine gewisse Kreisigkeit festgestellt, werden die Koordinaten seines Zentrums als erkannten Marker gespeichert.

3.7.3. Umgang mit Ungenauigkeiten

Die Messungen der Stereokamera folgen einem idealisierten Format. Ziel ist es, dass der Filter Messungen in der immer selben Form erhält. Da die Markerpose, wie in Abschnitt 3.6 erläutert, aus vier Markern besteht, wird für beide Kameras $\mathbf{P}_l$ und $\mathbf{P}_r$ ein Vektor mit vier Einträgen erstellt. Die Einträge enthalten die vermutlichen Koordinaten der jeweiligen Marker $\mathbf{m}_i$ der Markerpose aus I_l, beziehungsweise I_r. Die reale Markererkennung ist nicht perfekt. So gibt es verschiedene Ungenauigkeiten, die beachtet werden müssen, um die Messdaten basierend auf der Markererkennung zu simulieren.

Ich habe mit der physischen Kamera zwei mögliche Fehlerquellen identifiziert. Entweder werden Marker fälschlicherweise nicht erkannt oder es werden Objekte im Bild fälschlicherweise als Marker erkannt. Beide Probleme gilt es, zu simulieren und anschließend mit ihnen umzugehen. Hier vermute ich den entscheidenden Vorteil meines Tracking-Algorithmus gegenüber deterministischen Verfahren. Denn durch die Verwendung eines UKFs lässt sich zumindest in der Theorie mit derartigen Ungenauigkeiten umgehen.

Umgang mit falsch-negativen Werten

Falsch-negative Werte können mehrere Ursachen haben. Entweder wurde ein Marker gänzlich oder so stark verdeckt, dass er nicht als solcher erkannt werden kann. Alternativ kann der Kontrast zwischen anderen Lichtquellen in der Umgebung und dem Marker zu gering sein. Somit würde bei der Segmentierung des Bildes der Marker beispielsweise dem Hintergrund zugeordnet werden und könnte gar nicht als Kontur erkannt werden. Auch ist es möglich, dass durch eine nicht ausreichende Anstrahlung des Markers nur ein Teil der reflektierenden Oberfläche genügend Licht zurückwirft.

Enthält der Messvektor nicht mehr vier Einträge, wird zunächst versucht, den Wert aus den vorherigen Zeitschritten zu konstruieren. Hierfür wird geprüft, ob zum Zeitpunkt $t-1$ Koordinaten für den Marker $\mathbf{m}_i$ vorlagen. Ist das der Fall, werden diese für den fehlenden Marker an derselben Stelle im Messvektor eingesetzt. Falls das nicht der Fall ist, wird so lange in der Zeit zurückgegangen, bis ein Zeitschritt gefunden wird, an dem Messwerte vorhanden waren. Sollte es zu keinem Zeitpunkt Messwerte für den entsprechenden Marker gegeben haben, wird ein Mittelwert aus den Markerkoordinaten im Bildkoordinatensystem zum Zeitpunkt t gebildet und für $\mathbf{m}_i$ eingesetzt.

Umgang mit falsch-positiven Werten

Falsch-positive Werte können durch Lichtquellen oder Reflexionen in der Umgebung entstehen. Die in Abschnitt 3.7.2 beschriebene Überprüfung der Kreisigkeit ist ein Mechanismus, der verhindern soll, dass so entstehende Konturen im segmentierten Bild aussortiert werden. Falls allerdings deren Kreisigkeit über dem beschriebenen Schwellwert liegt, müssen weitere Maßnahmen eintreten. Hier gibt es zwei denkbare Methoden.

Zum einen könnte versucht werden, die Bauweise der Markerpose miteinzubeziehen, um zu prüfen, ob ein Punkt an der Stelle im Bild sein dürfte. Da zum Zeitpunkt der Messung aber nicht bekannt ist, wo sich die Pose befindet und vor allem wie sie im Raum orientiert ist, ist das nur schwer umzusetzen. Eine einfachere Variante, für die ich mich entschieden habe, liegt ebenfalls im Versuch, unrealistische Werte auszusortieren. Hierbei werden, falls mehr als vier Marker vorhanden sind, die Distanzen d_i, j zwischen dem i-ten und dem j-ten Marker aus der Messung berechnet und in einer Matrix $\mathbf{D}$ gespeichert. Hierbei stellt die i-te Zeile den i-ten Marker und die j-te Spalte den j-ten Marker dar und die Einträge entsprechen der jeweiligen Distanz zwischen beiden. Anschließend wird für jede Zeile ein Mittelwert gebildet. Dieser enthält nun die mittlere Distanz des Markers zu allen anderen Markern aus der Messung. Nun werden diese Einträge sortiert. Je nach Anzahl der Falsch-positiven Marker werden nun ausgehend von der höchsten mittleren Distanz Marker entfernt.

Dieser Ansatz basiert auf der Annahme, dass falsche Werte idealerweise keine Verschlechterung der Aktualisierung der UKF-Schätzung erzeugen. Dabei ist die Überlegung die Folgende: Wird ein falsch-positiver Wert gelöscht, verbessert es das Ergebnis. Wird ein korrekter Marker gelöscht, verschlechtert sich das Ergebnis. Allerdings kann davon ausgegangen werden, dass die Falsch-positiven Werte sich nicht unmittelbar an der Markerpose befinden, sondern im Hintergrund des Bildes. Sollte dem nicht so sein, bedeutet das, dass die Marker nah beieinander und die Pose somit weiter entfernt ist. In diesem Fall ist es denkbar, dass ein korrekter Marker gelöscht wird. Allerdings ist das ähnlich zu der Mittelwert-Bildung bei falsch-negativen Werten. Die Marker sind in so einem Fall ausreichend nah beieinander, um die Schätzung des Filters nicht in eine gänzliche andere Richtung zu verändern.

3.8. Messdatensimulation aus dem Objektzustand

Die Messdatensimulation nutzt das in Abschnitt 3.7 gewonnen Wissen über die realen Messdaten, um diese zu simulieren. Die Simulation wird deshalb verwendet, da sich so die Ungenauigkeit der Messdaten anhand von Parametern definieren lassen. Sie erfolgt in mehreren Schritten. Zunächst werden Bewegungsdaten für die Kamera sowie für die Markerpose benötigt. Diese werden mithilfe eines HMDs in einer Unity-Anwendung aufgezeichnet. Die Aufzeichnung der Bewegungsdaten ist in Absatz 3.8.1 näher erläutert. Die einzelnen Bewegungen der Kamera sowie der Markerpose werden daraufhin als zwei einzelne unabhängige Datensätze gespeichert. Aus diesen werden anschließend die simulierten Messdaten generiert. Diese Messdaten liegen ebenfalls als unabhängige Datensätze vor.

Die Generierung der simulierten Messdaten ist in Absatz 3.8.2 genauer beschrieben. Die Zwischenspeicherung der verschiedenen Datensätze dient einer differenzierteren Evaluation, da so gezielt einzelne Datensätze genauer betrachtet werden können. Die Bewegungsdaten des Objektes entsprechen dabei den Ground Truth-Daten, die später zur Evaluation des Tracking-Verfahrens verwendet werden können.

3.8.1. Aufzeichnung realer Bewegungsdaten

Die Bewegungsdaten sollten möglichst natürlich sein. Gleichzeitig ist es nötig, Ground Truth-Daten zu erzeugen, mit denen die Filterschätzung später verglichen werden kann. Ich habe deshalb die Daten mithilfe des HMDs aufgenommen. Hierfür habe ich eine Unity-Anwendung entwickelt, die sowohl die Position und Rotation des HMDs als auch eines Objektes in der Unity-Szene erfasst. Diese Daten wurden anschließend als CSV-Dateien gespeichert. Somit liegen also realistische Bewegungsdaten für den Kopf sowie für ein zu trackendes Objekt vor. Letzteres ließ sich in der Unity-Anwendung mittels der Eingabe des Controllers der Meta Quest 2 in die Hand nehmen und bewegen.

3.8.2. Simulation der Messung mit Ungenauigkeiten

Für die Messdatensimulation kommt die in Abschnitt 3.5 genauer erläuterte Filterkamera zum Einsatz. Dabei handelt es sich nicht um die in Unity simulierte Kamera, sondern um eine virtuelle Kamera auf Basis dieser simulierten Kamera. Mithilfe der Filterkamera werden die einzelnen Marker der Markerpose $\mathbf{m}_W$ in Weltkoordinaten zu Bildpunkten der rechten $\mathbf{m}_{I_r}$ oder linken Kamera $\mathbf{m}_{I_l}$ umgewandelt. Hierfür werden zwei Komponenten benötigt. Zum einen verwendet die Filterkamera die Bewegungsdaten der Kamera, um die Projektion der Kamera anzupassen. Die Anpassung der Kameraprojektion ist in Absatz 3.5 beschrieben. Weiterhin liegt für die Markerpose ein Datensatz vor, der die Position sowie die Orientierung des zu trackenden Objektes enthält. Auf Basis dessen muss eine virtuelle Markerpose erzeugt werden, die anschließend von der Filterkamera in I_l und I_r projiziert werden kann. Die Erzeugung einer virtuellen Markerpose auf Basis des Zustandes des zu trackenden Objektes ist in Abschnitt 3.6.2 erläutert.

Die Messdatensimulation versucht neben der Projektion der Raumkoordinaten der Marker auch die Ungenauigkeiten der Messungen, bestehend aus dem Wegfall oder Hinzukommen von Messwerten, zu simulieren. Wie mit diesen umgegangen wird, ist in Abschnitt 3.7.3 erläutert.

So ist es sehr wahrscheinlich, dass bei der Handhabung eines zu trackenden Objektes, einzelne Marker der Markerpose durch beispielsweise den zu trackenden Gegenstand verdeckt werden. Weiterhin kann es passieren, dass ein Marker nicht genügend Licht in die Kamera zurückwirft und nicht als Marker erkannt wird. Auch können sich Marker gegenseitig verdecken oder so nah beieinander liegen, dass sie von der in Abschnitt 3.7.2 beschriebenen Markererkennung als ein einzelner Marker erkannt werden. Diese Fälle stellen die falsch-negativen Messungen dar. Zudem kann es zu falsch-positiven Messungen kommen. Hierbei handelt es sich um weitere Lichtquellen oder reflektierende Flächen im Raum, die von der Markererkennung auf dem Bild fälschlicherweise als Marker erkannt werden.

Das Auftreten der beschriebenen Fälle habe ich über Werte abgebildet, die angeben, wie wahrscheinlich das Auftreten der beschriebenen Fehlertypen aufgrund von verschiedenen Ereignissen sind. Hierbei gibt es, wie in Absatz 3.11.4 beschrieben, insgesamt fünf Varianten dieser Wahrscheinlichkeiten. Wobei in jeder Kategorie die Wahrscheinlichkeit für die Verdeckung von mehreren Markern oder das Auftreten von falsch-positiven Markern mit dem Aufsteigen der Variante zunimmt. Zudem werden in der Messdatensimulation zwei Arten von Rauschen hinzugefügt. Eine Übersicht über die Wahrscheinlichkeiten für das Eintreten der beschriebenen Fehler während der Messdatensimulation ist in den Tabellen 3.2, 3.3 und 3.4 zu finden. Dort sind jeweils die Wahrscheinlichkeiten für den Wegfall von null bis vier Markern für alle Messdatenvarianten aufgelistet.

Eine zuverlässige und realitätsnahe Messdatensimulation ist von enormer Wichtigkeit. Schließlich sind die Messdaten für die Korrektur einer Filtervorhersage verantwortlich. Um eine Aussage über die Qualität des Tracking-Verfahrens treffen zu können, müssen die simulierten Daten nahe an der Realität sein.

Falsch-negative Marker durch Verdeckung

Die Wahrscheinlichkeiten für das Wegfallen von Markern aufgrund von Verdeckung durch andere Objekte oder die Hand der Nutzer*innen sind in Tabelle 3.2 abgebildet. Die Wahrscheinlichkeit dafür, dass der Wegfall im Bild beider Kameras der Stereokamera auftritt, liegt bei 90%.

Tabelle 3.2.: Wahrscheinlichkeiten für das Wegfallen von Markern aufgrund von Verdeckung

Variante	p 0 Marker	p 1 Marker	p 2 Marker	p 3 Marker	p 4 Marker
0	90,00%	5,00%	2,50%	2,00%	0,50%
1	80,00%	15,00%	2,50%	2,00%	0,50%
2	70,00%	15,00%	12,50%	2,00%	0,50%
3	60,00%	15,00%	12,50%	12,00%	0,50%
4	50,00%	15,00%	12,50%	12,00%	10,50%

Falsch-negative Marker durch Licht

Die Wahrscheinlichkeiten für das Wegfallen von Markern auf Grund von schlechter Beleuchtung der Marker sind in Tabelle 3.3 abgebildet. Die Wahrscheinlichkeit dafür, dass der Wegfall im Bild beider Kameras der Stereokamera auftritt, liegt bei 98%.

Tabelle 3.3.: Wahrscheinlichkeiten für das Wegfallen von Markern aufgrund von mangelnder Beleuchtung

Variante	p 0 Marker	p 1 Marker	p 2 Marker	p 3 Marker	p 4 Marker
0	90,00%	5,00%	2,50%	2,00%	0,50%
1	80,00%	15,00%	2,50%	2,00%	0,50%
2	70,00%	15,00%	12,50%	2,00%	0,50%
3	60,00%	15,00%	12,50%	12,00%	0,50%
4	50,00%	15,00%	12,50%	12,00%	10,50%

Falsch-positive Marker durch Lichtquellen

Die Wahrscheinlichkeiten für die fälschliche Erkennung von Lichtquellen oder Reflexionen als Marker sind in Tabelle 3.4 abgebildet. Die Wahrscheinlichkeit, dass der falsch erkannte Marker in beide Bildern auftritt, liegt bei 50%.

Tabelle 3.4.: Wahrscheinlichkeiten für das Auftreten von falsch erkannten Markern

Variante	p 0 Marker	p 1 Marker	p 2 Marker	p 3 Marker	p 4 Marker
0	95,00%	2,50%	2,50%	0,00%	0,00%
1	90,00%	5,00%	2,50%	2,50%	0,00%
2	85,00%	5,00%	5,00%	2,50%	2,50%
3	80,00%	7,50%	5,00%	5,00%	2,50%
4	75,00%	10,00%	7,50%	5,00%	2,50%

Anwendung von Rauschen auf die Messdaten

Bei dem auf die Messdaten angewendeten Rauschen handelt es sich um zwei verschiedene Varianten. Bei der ersten handelt es sich um einen Fehler, der die Bildkoordinaten aller erkannten Marker in dieselbe Richtung verschiebt. Dieses Rauschen soll einen allgemeinen Fehler in der Projektion darstellen, der sich auf alle erkannten Marker gleichermaßen auswirkt. Er wird wie folgt berechnet:

$$\mathbf{m}'_{i_I} = \mathbf{m}_{i_I} + \mathcal{N}(0,\, 1px) \tag{3.20}$$

Die zweite Variante versieht die Markerkoordinaten mit einem gerichteten Fehler. Hierfür werden, wie in der folgenden Gleichung zu sehen, zwei Werte aus einer Normalverteilung ermittelt und auf die x- und y-Komponente jedes Markers gerechnet.

$$\mathbf{m}'_{i_I} = \mathbf{m}_{i_I} + \begin{bmatrix} \mathcal{N}(0,\, 0.12px) \\ \mathcal{N}(0,\, 0.12px) \end{bmatrix} \tag{3.21}$$

3.9. Deterministische Bestimmung der Pose

Das in dieser Arbeit entwickelte UKF-basierte Tracking-Verfahren basiert auf einem Bayes-Filter. Die vermuteten Vorteile dieses Verfahrens liegen in der Toleranz gegenüber ungenauen Messwerten. Dennoch ist es im Rahmen der Ermittlung der Qualität des Verfahrens interessant, zu ermitteln, ob ein derartiges Verfahren einen Vorteil gegenüber einer deterministischen Methode bietet und wie dieser Vor- oder Nachteil aussieht. Hierfür habe ich eine Methode umgesetzt, die die Pose anhand von Triangulation und dem Kabsch-Umeyama-Algorithmus Umeyama (1991) bestimmt. Dieses Verfahren wird außerdem auch in dem in Absatz 3.10 beschriebenen UKF verwendet.

3.9.1. Markerzuordnung zwischen den Stereokameras

Um das deterministische Verfahren einzusetzen, müssen zwei Grundlagen erfüllt sein. Zum einen müssen in den Bildern beider Kameras mindestens drei Marker erkannt worden sein und zum anderen muss es sich dabei um dieselben Marker handeln. So muss die Projektion von $\mathbf{m}_{0_W}$ der linken Kamera $\mathbf{m}_{0_{I_l}}$ der Projektion der rechten Kamera $\mathbf{m}_{0_{I_r}}$ zugeordnet sein. Diese Einschränkung muss für alle erkannten Marker erfüllt sein.

Da ich in dieser Arbeit simulierte Messdaten verwende, lässt sich diese Einschränkung leicht erfüllen. In einem anderen Kontext käme hier ein mitunter fehleranfälliger Zuordnungsalgorithmus zum Einsatz, der die Marker in beiden Bildern eindeutig einander zuzuordnen.

Es ist allerdings valide, diese Zuordnung nicht in die Evaluation miteinzubeziehen. Denn sowohl der Filter, als auch das deterministische Verfahren verschlechtern sich, je häufiger, die Zuordnung der Marker aus beiden Kamerabildern falsch ist. Denn auch der Filter benötigt eine feste Sortierung des Messvektors, bei der die Koordinaten der Marker $\mathbf{m}_0$ bis $\mathbf{m}_3$ immer an derselben Stelle stehen.

3.9.2. Triangulation

Sind in beiden Bildern vier Marker erkannt worden, wird eine Triangulation der Punkte durchgeführt. Wie in Absatz 3.9.1 bereits erklärt, ist es hierfür wichtig, dass die einzelnen Bildpunkte einander zugeordnet sind. Die Triangulation verwendet die Projektionsmatrizen der Filterkamera. Deren Berechnung wird in Absatz 3.5 näher beschrieben. Nach der Triangulation liegt für die erkannten Marker $\mathbf{m}_{i_{I_l}}$ und $\mathbf{m}_{i_{I_r}}$ jeweils der rekonstruierte Marker $\mathbf{m}_{i_r} \in \mathbb{R}^3$ vor. Für die gesamte Pose liegen somit alle vier Marker in der Menge M im euklidischen Raum vor.

3.9.3. Ermittlung der Pose

Die in Abschnitt 3.6 beschriebene Markerpose kann als Menge von Punkten $Y = \mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_2, \mathbf{m}_3$ beschrieben werden. Y enthält also alle Marker der Markerpose mit ihren Koordinaten. Bei Y handelt es sich um die nicht rotierte oder verschobene Markerpose im Ursprung des Weltkoordinatensystems handelt. Die mittels der Triangulation rekonstruierte Pose ist die Menge aller rekonstruierten Marker M. Mithilfe des Kabsch-Umeyama-Algorithmus Umeyama (1991) können die nötige Rotation $\boldsymbol{R}$, die Translation $\boldsymbol{T}$ und die Skalierung s berechnet werden, für die gilt:

$$M = \boldsymbol{T} + \boldsymbol{R}Y \cdot s \tag{3.22}$$

Auch hierbei gibt es eine Einschränkung, die es zu beachten gilt. Denn bevor der Kabsch-Umeyama-Algorithmus angewandt werden kann, müssen die einzelnen Punkte aus den Festkörpern M und Y einander zugeordnet werden. Hierfür lässt sich auf die Arbeit von Steinicke u. a. (2007) zurückgreifen. Auf dieser baut bereits das in Abschnitt 3.6 beschriebene Design der Markerpose auf. Die Zuordnung der starren Körper findet hier über die einzigartigen Abstände der einzelnen Marker innerhalb einer Markerpose statt. Steinicke u. a. (2007) setzen dabei auf die Zuordnung der einzelnen Kanten mittels Minimum-Weight-Matching. Ich habe hier allerdings die Kanten einander dadurch zugeordnet, dass ich sie der Länge nach sortiert habe. Dieses Vorgehen hat in meinen Versuchen ebenso zuverlässige Resultate erzeugt. Es ist aber denkbar, dass bei einem System, dass mehrere Markerposen tracken kann, die Sortierung nicht

mehr genau genug ist, da die internen Abstände nicht verschieden genug sind. Aufgrund der Datensimulation ist es allerdings auch hier nicht nötig, die Zuordnung einzubauen.

3.10. Das UKF-basierte Tracking-Verfahren

Das in dieser Arbeit entstandene Tracking-Verfahren basiert auf einem UKF. In diesem Abschnitt ist das Design dieses UKFs beschrieben. Weiterhin werden Begründungen für die jeweilige Wahl bestimmter Verfahren oder Werte geliefert. Hierbei gehe ich in Abschnitt 3.10.1 auf das verwendete Prozessmodell und die Vorhersagefunktion ein. Abschnitt 3.10.2 beschäftigt sich mit dem Aufbau der Messfunktion. Hierbei ist zudem die Beobachtbarkeit des Systems relevant. Auf diese wird in Abschnitt 3.10.3 eingegangen. Aus dem Zustand des Filters ergeben sich für die Berechnungen innerhalb des UKFs einige Besonderheiten. Konkret geht es dabei um die Mittelwertbildung der Sigma-Punkte des Zustandes sowie die Berechnung des Residuums zwischen zwei Zustandsvektoren. Der Aufbau und die Berechnung in diesen Funktionen sind in Abschnitt 3.10.4 und 3.10.5 genauer beschrieben. In Absatz 3.10.6 und 3.10.7 sind das Prozessrauschen sowie das Messrauschen beschrieben, die der UKF verwendet. Die Initialisierung des Filters ist in Absatz 3.10.9 beschrieben. In Abschnitt 3.10.11 ist der Umgang mit einem Fehler beschrieben, der der starken Nichtlinearität des Prozessmodells geschuldet ist.

3.10.1. Prozessmodell und Zustandstransferfunktion

Der Zustand des Systems beinhaltet die Größen, die der zu entwickelnde Filter tracken soll. Bei dem Tracking-System, das im Rahmen dieser Masterarbeit entwickelt wurde, handelt es sich um eines mit sechs Freiheitsgraden. Das bedeutet, die Position und Orientierung des zu trackenden Objektes werden in $\mathbb{R}^3$ verfolgt. Ich habe mich dabei an der Arbeit von Gsaxner u. a. (2021) orientiert. Hier befindet sich die Orientierung des zu trackenden Objektes nicht im Zustand des Systems. Diese wird außerhalb des Zustands gespeichert. Der Zustand ist hier wie auch bei Welch und Bishop (1997) folgendermaßen aufgebaut.

$$\mathbf{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}, \phi, \theta, \psi, \dot{\phi}, \dot{\theta}, \dot{\psi}] \tag{3.23}$$

Die globale Orientierung ist daneben als externer Quaternion gespeichert.

$$\mathbf{q}_w = [q_x, q_y, q_z, q_w] \tag{3.24}$$

Bei dem verwendeten Prozessmodell handelt es sich wie auch bei Gsaxner u. a. (2021) um ein einfaches Positions-Geschwindigkeits-Modell. Hierbei wird die Geschwindigkeit des Systems als konstant angenommen. Das Positions-Geschwindigkeits-Modell wird von Gsaxner u. a. (2021) und Welch und Bishop (1997) als ausreichend angesehen. Laut Labbe (2022a) verschlechtern weitere zeitliche Ableitungen wie die Beschleunigung oder der Ruck die Filterschätzung.

Das Modell basiert auf der allgemein bekannten Newtonschen Bewegungsgleichung. In zeitdiskretisierter Form ergeben sich für das Prozessmodell folgende Gleichungen:

$$
\begin{aligned}
x_{t+1} &= x_t + \dot{x}_t \Delta t \\
y_{t+1} &= y_t + \dot{y}_t \Delta t \\
z_{t+1} &= z_t + \dot{z}_t \Delta t \\
\dot{x}_{t+1} &= \dot{x}_t \\
\dot{y}_{t+1} &= \dot{y}_t \\
\dot{z}_{t+1} &= \dot{z}_t \\
\phi_{t+1} &= \phi_t + \dot{\phi}_t \Delta t \\
\theta_{t+1} &= \theta_t + \dot{\theta}_t \Delta t \\
\psi_{t+1} &= \psi_t + \dot{\psi}_t \Delta t \\
\dot{\phi}_{t+1} &= \dot{\phi}_t \\
\dot{\theta}_{t+1} &= \dot{\theta}_t \\
\dot{\psi}_{t+1} &= \dot{\psi}_t
\end{aligned}
\tag{3.25}
$$

Diese Gleichungen lassen sich mittels einer Matrix zusammenfassen:

$$\mathbf{x}_{t+1} = \mathbf{F}\mathbf{x}_t = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \mathbf{x}_t \tag{3.26}$$

Das Prozessmodell weicht stark von der Dynamik des tatsächlichen Systems ab. Bei den zu trackenden Objekten handelt es sich um hochgradig nicht lineare und manövrierende Objekte. Diese sind dazu in der Lage, ihre Geschwindigkeit und Beschleunigung binnen kürzester Zeit zu verändern. Eine akkurate Beschreibung dieser manövrierenden Bewegung mittels eines Modells ist nahezu unmöglich. Die theoretischen Hintergründe zu dieser Problematik sind im theoretischen Teil dieser Arbeit in Absatz 2.3 genauer beschrieben. Als Umgang mit dieser Diskrepanz zwischen Prozessmodell und tatsächlicher Bewegung schlagen Challa u. a. (2011) zwei Ansätze vor. Ich habe den Ansatz gewählt, der auch in der Arbeit von Gsaxner u. a. (2021) verwendet wurde. Somit wird über das Prozessrauschen $\mathbf{Q}$ die Ungenauigkeit des Prozessmodells abgebildet. Wie dieses erstellt wird, ist in Absatz 3.10.6 näher beschrieben.

Nicht Bestandteil des Prozessmodells ist der Einfluss der Winkelgeschwindigkeiten $\dot{\phi}$,$\dot{\theta}$ und $\dot{\psi}$ auf die globale Orientierung $\mathbf{q}$. Die Winkelgeschwindigkeiten wirken nur auf die inkrementelle Orientierungen bestehend aus ϕ, θ und ψ ein. Diese inkrementellen Orientierungen beschreiben somit lediglich die Rotation auf der x-, y- und z-Achse seit dem letzten Zeitschritt. Nach jeder Iteration des UKF werden sie zur globalen Orientierung $\mathbf{q}$ gerechnet und gleich null gesetzt. Durch die Speicherung der Orientierung als Quaternion $\mathbf{q} \in \mathbb{H}$ wird laut Challis (2020) das Problem einer kardanischen Lagerung der drei Rotationsachsen bei einer Euler-Darstellung vermieden. Die Anwendung der inkrementellen Orientierung auf die globale Orientierung wird von Welch und Bishop (1997) definiert und für den in dieser Arbeit entwickelten UKF

übernommen. Hierbei wird zunächst die inkrementelle Rotation von der Euler-Darstellung in einen Quaternion umgewandelt.

$$\begin{aligned}
\Delta q_x &= \sin(\phi/2)\cos(\theta/2)\cos(\psi/2) - \cos(\phi/2)\sin(\theta/2)\sin(\psi/2) \\
\Delta q_y &= \cos(\phi/2)\sin(\theta/2)\cos(\psi/2) + \sin(\phi/2)\cos(\theta/2)\sin(\psi/2) \\
\Delta q_z &= \cos(\phi/2)\cos(\theta/2)\sin(\psi/2) - \sin(\phi/2)\sin(\theta/2)\cos(\psi/2) \\
\Delta q_w &= \cos(\phi/2)\cos(\theta/2)\cos(\psi/2) + \sin(\phi/2)\sin(\theta/2)\sin(\psi/2) \\
\Delta \mathbf{q} &= [q_x, q_y, q_z, q_w]
\end{aligned} \tag{3.27}$$

Dieser wird anschließend mit der globalen Orientierung $\mathbf{q}$ multipliziert. Wobei es sich hierbei um die nicht kommutative Quaternionenmultiplikation handelt.

$$\mathbf{q}_{t+1} = \mathbf{q}_t \otimes \Delta\mathbf{q} \tag{3.28}$$

3.10.2. Messfunktion

Die Messfunktion des Filters liefert auf Basis eines Zustandsvektors $\mathbf{x}$ die dazugehörige Messung. Damit ist sie mit der Simulation der Messdaten eng verwandt. Für die korrekte Funktion der Messfunktion ist es entscheidend zu wissen, ob sich der Zustand hinter oder vor der Kamera beziehungsweise der Filterkamera befindet. Der Messvektor verfügt wie in Abschnitt 3.7.1 beschrieben über 16 Einträge bestehend aus den x- und y-Koordinaten der erkannten Marker im Bild. Der Messvektor aus Gleichung 3.15 ist allerdings nicht vollständig. Neben den Bildkoordinaten verfügt er über einen 17ten Eintrag dz_{cam}. Hier ist die Differenz der z-Position des Zustandes zur Kamera im Koordinatensystem der Kamera angegeben. Diese gibt Auskunft darüber, ob sich die Pose im Weltkoordinatensystem hinter oder vor der Stereokamera befindet und wie weit entfernt sie dabei ist. Die Begründung für dieses Vorgehen wird in Abschnitt 3.10.3 näher erläutert.

Zur Berechnung von dz_{cam} wird zunächst die Position $\mathbf{t}_W$, die im Zustand $\mathbf{x}$ enthalten ist, in das Koordinatensystem der beiden Kameras überführt. Dafür lässt sich die invertierte Pose der Kamera verwenden, die in Absatz 3.5 bereits berechnet wurde. Diese wird im Tracking-Verfahren

für jeden Zeitschritt auf Basis der Pose der Kamera berechnet und überführt Koordinaten aus dem Weltkoordinatensystem in das Kamerakoordinatensystem.

$$\mathbf{t}_C = \left[\begin{array}{c|c} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{T} \end{array} \right] \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{3.29}$$

Hierfür ist es nötig $\mathbf{t}_W$ in homogene Koordinaten umzuwandeln, indem eine 1 angehängt wird. Dieses Vorgehen ist bereits aus der Kameraprojektion bekannt. Anschließend wird dz_{cam} aus $\mathbf{t}_{C_l}$ und $\mathbf{t}_{C_r}$ wie folgt berechnet.

$$dz_{cam} = \frac{(\mathbf{t}_{C_{l_z}} + \mathbf{t}_{C_{r_z}})}{2} \tag{3.30}$$

Die Abbildung 3.5 zeigt beispielhaft die Ergebnisse der beschriebenen Prüfung für die Fälle, dass sich die Markerpose vor und hinter der Kamera befindet. Die Kamera ist dabei um 40 Grad um die y-Achse gedreht.

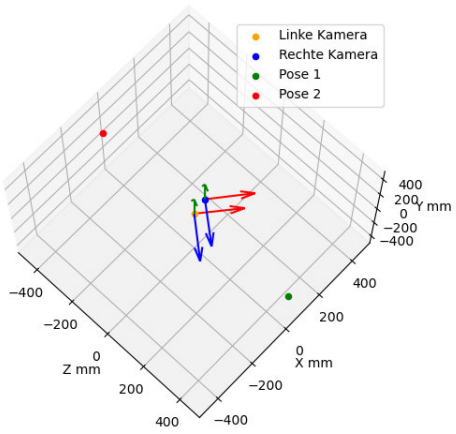


Abbildung 3.5.: Beispielhafte Darstellung für die Berechnung von dz_{cam}. Für Pose 1 ist $dz_{cam} = 375,29$ mm, für Pose 2 ist $dz_{cam} = -386,42$ mm.

Nach der Berechnung von dz_{cam} wird die eigentliche Messung vorgenommen. Zunächst wird auf Basis von $\mathbf{x}$ und der globalen Orientierung $\mathbf{q}$ eine virtuelle Markerpose erstellt. Dieses Vorgehen ist in Abschnitt 3.6.2 ausführlich erläutert. Anschließend, werden die Marker $\mathbf{m}'_{i_W}$ mittels der Kameramatrizen $\boldsymbol{P}_l$ und $\boldsymbol{P}_r$ in Bildkoordinaten $\mathbf{m}'_{i_I}$ umgewandelt.

Hierfür ist es nötig, alle $\mathbf{m}'_{i_W}$ zunächst in homogene Koordinaten und anschließend zurück in kartesische Koordinaten umzuwandeln. Weiterhin werden die im Rahmen der Kamerakalibrierung ermittelten Verzerrungskoeffizienten auf die Bildkoordinaten angewendet. Diese Kameraprojektion geschieht mittels der Filterkamera und ist dementsprechend in Absatz 3.5 ausführlich definiert.

3.10.3. Umgang mit der mangelnden Beobachtbarkeit des Systems

Wie im theoretischen Teil dieser Arbeit in Abschnitt 2.2.6 bereits erläutert, kann ein Kalman Filter, der auf einem nicht beobachtbaren System basiert, nicht funktionieren. An dieser Stelle sei der Begriff nur kurz in einfachen Worten wiederholt. So gilt ein System nur dann als beobachtbar, wenn es für jeden Zustand eine eindeutige und einzigartige Messung gibt und jede Messung aus nur einem Zustand entstehen kann. Diese Bedingung ist die Begründung für die Verwendung der Variable dz_{cam}. Ohne diese ist das beschriebene System eindeutig nicht beobachtbar. Hierfür lässt sich ein Beispiel anführen, bei dem die Erzeugung eines Messvektors $\mathbf{z}$ aus zwei Zuständen $\mathbf{x}_1$ und $\mathbf{x}_2$ möglich ist. Die Kameraprojektion, definiert als f_p, lässt sich als Abbildung definieren, die den $\mathbb{R}^3$ auf den $\mathbb{R}^2$ abbildet:

$$\begin{aligned} f :&\mathbb{R}^3 \to \mathbb{R}^2 \\ &\mathbf{t}_W \to f_p(\mathbf{t}_W) \end{aligned} \tag{3.31}$$

Diese Abbildung ist surjektiv, aber nicht injektiv. Damit das System beobachtbar ist, müsste sie allerdings bijektiv sein. Nimmt man beispielsweise die folgenden Raumpositionen

$$\begin{aligned} \mathbf{t}_{W_1} &= \begin{bmatrix} 100 \\ 100 \\ 100 \end{bmatrix} \\ \mathbf{t}_{W_2} &= \begin{bmatrix} -100 \\ -100 \\ -100 \end{bmatrix} \end{aligned} \tag{3.32}$$

und wendet auf beide f_p an, erhält man $\mathbf{t}_{I_1} = [1352, 1142] = \mathbf{t}_{I_2}$. Dieser Fall tritt dann ein, wenn eine der Positionen, das exakte Inverse der anderen ist. Für eine Markerpose, die aus mehreren Markern besteht, ist dieser Fall sehr unwahrscheinlich, aber dennoch möglich. So müssten zwei Zustandsvektoren so aufgebaut sein, dass die daraus erzeugte Markerpose invers zueinander sind, wie in Abbildung 3.6 zu sehen. In diesem Fall sind die jeweiligen Markerpaare der beiden Posen invers zueinander und somit erzeugt f_p für alle $\mathbf{m}_{i_W}$ der beiden Posen die-

selbe Projektion. Da sich so zwei identische Messvektoren $\mathbf{z}_1 = \mathbf{z}_2$ aus zwei unterschiedlichen Zustandsvektoren $\mathbf{x}_1 \neq \mathbf{x_2}$ ergeben, ist die Beobachtbarkeit des Systems nicht gegeben. Mittels des Wertes dz_{cam} kann sichergestellt werden, dass $\mathbf{z}_1 \neq \mathbf{z}_2$, wenn $\mathbf{x}_1 \neq \mathbf{x_2}$ gilt.

Hierfür ließe sich der Wert für dz_{cam} jedoch auch auf einen statischen Wert setzen. So könnte immer -10 für Posen hinter und 10 für Posen vor der Kamera verwendet werden. Allerdings führt das dazu, dass die Kovarianzmatrix $\mathbf{P}$ des UKF häufiger nicht positiv definit wird und damit keine echte Kovarianzmatrix mehr ist (siehe 3.10.11). Ein sinnvoller Wert, der sich aus dem Zustand des Systems ergibt, liefert hier ein zuverlässigeres Tracking sowie bessere Ergebnisse.

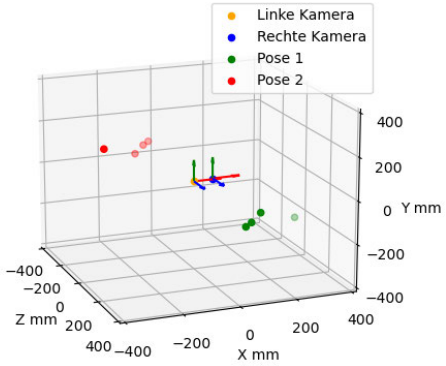


Abbildung 3.6.: Darstellung zwei inverser Markerposen, die denselben Messvektor ergeben

Das beschriebene Problem der mangelnden Beobachtbarkeit aufgrund der Kameraprojektion wird bei Gsaxner u. a. (2021) nicht behandelt. Die Begründung dafür liegt in dem verwendeten EKF und der Einschränkung des Trackings. So limitieren Gsaxner u. a. (2021) die Entfernung des Trackings auf etwa Armeslänge und damit auf 20-80 cm in Blickrichtung des HMDs. Bei der Evaluation wird das Instrument die ganze Zeit vor die Kamera gehalten. Außerdem geht der EKF anders mit der Nichtlinearität des Systems um. Hier wird das System immer an einer Stelle linearisiert. Der UKF hingegen erzeugt, wie in Absatz 2.2.5 beschrieben, Sigma-Punkte um den aktuellen Filterzustand, um so den Mittelwert nach der Anwendung einer nicht linearen Vorhersagefunktion aus diesen wiederherstellen zu können. Dabei kann es passieren, dass einer dieser Sigma-Punkte räumlich hinter der Kamera platziert wird. Wie bereits beschrieben, kann die Messfunktion aufgrund der Beschaffenheit der Kameraprojektion f_p auch für diesen Punkt plausible Messwerte erzeugen. Ohne dz_{cam} kann es passieren, dass der entstehende

virtuelle Messvektor $\mathbf{z}_p$ näher an dem realen Messvektor $\mathbf{z}$ liegt und das Residuum $\mathbf{y}$ damit kleiner wird. Somit kann es passieren, dass der Zustand des Filters sich hinter die Kamera bewegt und sich dort festsetzt.

Der Wert dz_{cam} muss neben $\mathbf{z}_p$ auch für $\mathbf{z}$ berechnet werden, damit zwischen $\mathbf{z}$ und $\mathbf{z}_p$ das Residuum berechnet werden kann. Hierfür wird die in Abschnitt 3.9 beschriebene Bestimmung der Pose mittels Triangulation und dem Kabsch-Umeyama-Algorithmus Umeyama (1991) berechnet. Aus der so errechneten Pose lässt sich dz_{cam} ermitteln. Dieser Wert kann jedoch nur dann erneuert werden, wenn in den Bildern beider Kameras genügend Marker erkannt und einander zugeordnet wurden. Für das hier beschriebene Tracking-Verfahren müssen das alle vier Marker sein. Da das nicht zu jedem Zeitschritt möglich ist, handelt es sich um eine annehmbare Näherung für dz_{cam} und keinen exakten Wert.

3.10.4. Mittelwert der Zustandsvektoren

Der Zustand des Systems beinhaltet kartesische Koordinaten und Winkel sowie deren ersten zeitlichen Ableitungen. Wie in Abschnitt 2.2.5 beschrieben, werden bei einem Vorhersageschritt des UKFs eine Reihe von Sigma-Punkten $\boldsymbol{\chi}$ mittels der Vorhersage Funktion $f(\boldsymbol{\chi}) = \boldsymbol{\mathcal{Y}}$ in der Zeit fortgeführt. Das geschieht im UT des UKFs. Teil des UTs ist die Berechnung eines neuen Mittelwerts. Hierbei wird in der Bibliothek filterpy vonLabbe (2022b) der gewichtete Mittelwert verwendet. Dabei wird jedem $\boldsymbol{\chi}$ ein Gewicht w_m zugeordnet.

$$\boldsymbol{\mu}_x = \bar{\mathbf{x}} = \sum_{i=0} w_i^m f(\boldsymbol{\chi_i}) = \sum_{i=0} w_i^m \boldsymbol{\mathcal{Y}} \tag{3.33}$$

Hierbei gilt jedoch die folgende Einschränkung:

$$1 = \sum_i w_i^m \tag{3.34}$$

Diese Mittelwertberechnung lässt sich für die ersten sechs Einträge x, y, z, $\dot{x}$, $\dot{y}$, $\dot{z}$ des Zustandsvektors $\mathbf{x}$ anwenden. Lediglich die letzten sechs Einträge bestehend aus ϕ, θ und ψ sowie $\dot{\phi}$, $\dot{\theta}$ und $\dot{\psi}$ lassen sich nicht so einfach berechnen. Das arithmetische Mittel entspricht hier nicht zwangsweise dem realen Mittel, da sich laut Lee (2010) die Topologie einer geraden und die eines Kreises unterscheiden. Für die Mittelwertberechnung der inkrementellen Orientierung $o = [\phi, \theta, \psi]$ sowie der Winkelgeschwindigkeiten $\omega = \dot{o} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$ verwende ich daher den zirkulären Mittelwert in Anlehnung an die Empfehlung von Labbe (2022b).
Die Definition des zirkulären Mittelwerts findet sich in der Arbeit von Jammalamadaka und

SenGupta (2001). Ich habe die dort verwendete Berechnung allerdings an den gewichteten Mittelwert angepasst:

$$\mu_o = arctan2(\sum_{i=0} \sin(o) w_i^m, \sum_{i=0} \cos(o) w_i^m) \tag{3.35}$$

Hierbei werden die Winkel jeweils in kartesische Koordinaten umgewandelt und der Mittelwert dieser Koordinaten wird gebildet. Der Arcus Tangens 2 gibt dann den Winkel des Vektor der gemittelten Koordinaten an.

3.10.5. Residuumsberechnung der Zustandsvektoren

Ein ähnliches Problem wie in Absatz 3.10.4 ergibt sich außerdem für die Berechnung des Residuums zwischen zwei Zustandsvektoren. Wie in Kapitel 2 in Absatz 2.2.5 ausführlich beschrieben, wird im UT zur Berechnung der Kovarianzmatrix $\mathbf{P}$ nach dem Vorhersageschritt des UKFs ein Residuum $\mathbf{y}$ zwischen zwei Zustandsvektoren berechnet, um die Differenz zwischen dem Mittelwert und den in der Zeit fortgeführten Sigma-Punkten zu berechnen $\bar{\mathbf{x}} - \boldsymbol{\mathcal{Y}}$. Im Falle der kartesischen Koordinaten x, y und z sowie deren ersten zeitlichen Ableitung, kann hier einfach subtrahiert werden. Für o und ω allerdings gilt erneut die Topologie des Kreises. So soll gelten $|359 - 1| = 2 \neq 358$, wobei 358 arithmetisch gesehen korrekt wäre. Folgende Gleichung beschreibt daher die Subtraktion zwischen zwei Winkeln in der Residuumsfunktion des Zustandes im UKF:

$$s(a, b) = \begin{cases} (a - b) - 2\pi, & \text{wenn } (a - b) \geq \pi \\ (a - b) + 2\pi, & \text{wenn } (a - b) \leq -\pi \\ (a - b), & \text{ansonsten} \end{cases} \tag{3.36}$$

3.10.6. Prozessrauschen

Das Prozessrauschen $\mathbf{Q}$ ist eine $12x12$ Matrix, deren Aufgabe es ist, die Ungenauigkeit des Prozessmodells abzubilden. Da es sich bei der tatsächlichen Dynamik des Systems um sehr stark manövrierende Bewegungen handelt, ist es nahezu unmöglich ein Prozessmodell zu definieren, das diese Bewegungen ganzheitlich abbildet. Wie in Abschnitt 3.10.1 bereits erläutert, habe ich mich daher in Anlehnung an Gsaxner u. a. (2021) für ein Positions-Geschwindigkeits-Modell entschieden. Dieses ist allerdings sehr ungenau und wird der tatsächlichen Bewegung nicht gerecht. Die Erzeugung der Matrix $\mathbf{Q}$ erfolgte im Wesentlichen durch das Ausprobieren verschiedener Werte. Dieses Vorgehen basiert auf dem Buch von Labbe (2022a).

Dort heißt es übersetzt: “Wenn $\mathbf{Q}$ zu klein ist, wird der Filter in seinem Vorhersagemodell zu sicher und weicht von der tatsächlichen Lösung ab. Ist $\mathbf{Q}$ zu groß, wird der Filter durch das Rauschen in den Messungen übermäßig beeinflusst und arbeitet suboptimal. In der Praxis verbringen wir viel Zeit damit, Simulationen durchzuführen und die gesammelten Daten auszuwerten, um einen geeigneten Wert für $\mathbf{Q}$ zu finden.”

$\mathbf{Q}$ wird mittels der Funktion “Q_discrete_white_noise”[23] der Software-Bibliothek filterpy von Labbe (2022b) berechnet. Diese erzeugt eine Matrix $\mathbf{Q}$ auf Basis des diskretisierten, konstanten, weißen Rauschens. Ich habe mich für die diskretisierte Form des weißen Rauschens entschieden, um die plötzlichen und starken Änderungen in der Beschleunigung des zu trackenden Objektes abzubilden, da für das diskrete, weiße Rauschen angenommen wird, dass beispielsweise die Zustandswerte innerhalb eines Zeitschrittes zwar konstant, aber zwischen den Zeitschritten gänzlich unabhängig sind Labbe (2022a). Sowohl für die Position, als auch die Orientierung innerhalb des Zustandes mussten der filterpy-Funktion die Varianzen übergeben werden. Nach einigen Experimenten habe ich hier für die Position eine Standardabweichung von $std_{pos} = 1500$ mm und die Orientierung eine Standardabweichung von $std_{rot} = 0,1$ rad als beste Werte identifiziert. Der für die Erzeugung dieser Matrix verwendete Zeitschritt ist $\Delta t = \frac{1}{60}$ s

3.10.7. Messrauschen

Das Messrauschen $\mathbf{R}$ lässt sich aus den verfügbaren Messdaten berechnen. Hierfür habe ich mit der Messdatensimulation aus Abschnitt 3.8.2 einen Satz Messdaten generiert und die Kovarianz der Messdaten berechnet. Die Matrix $\mathbf{R}$ wird auf Basis der Messdaten erzeugt, die entstehen, wenn sich weder die Kamera, noch das zu trackende Objekt bewegen. Ideale Messinstrumente würden hier einen Datensatz ohne Varianz erzeugen. Da die Messdatensimulation allerdings ein Rauschen auf die Messdaten rechnet und zudem falsch-negative sowie falsch-positive Werte erzeugt, entsteht hier ein Fehler. Zusätzlich habe ich die Matrix $\mathbf{R}$ mit dem Faktor 1000 skaliert, da ansonsten der Filter zu viel Vertrauen in die Messdaten setzt und die Varianz der Zustandsvariablen zu gering ist.

3.10.8. Berechnung der Sigma-Punkte

Für die Berechnung der Sigma-Punkte verwende ich den Algorithmus von Van Der Merwe und Wan (2004) verwendet. Dieser ist laut Labbe (2022a) seit etwa 2005 der Standard für die

[23] `https://filterpy.readthedocs.io/en/latest/_modules/filterpy/common/discretization.html#Q_discrete_white_noise`, Stand: 19.07.2023

Berechnung von Sigma-Punkten. Der Algorithmus berechnet die Sigma-Punkte anhand der übergebenen Parameter β, κ und α. Ich habe folgende Parameter gewählt:

$$\begin{aligned} \alpha &= 0,01 \\ \beta &= 2 \\ \kappa &= 0 \end{aligned} \tag{3.37}$$

Der Wert α bestimmt bei der Berechnung der Sigma-Punkte die Streuung um den Mittelwert. In Absatz 2.2.5 ist näher beschrieben, inwiefern die Größe von α Einfluss auf den Umgang mit nicht linearen und nicht gaußverteiltem Verhalten nimmt. Laut Labbe (2022a) ist ein kleineres α dann zu bevorzugen, wenn sich das System besonders nicht linear verhält. Zwar ist die Zustandsübergangsfunktion an sich sehr linear, die eigentliche Bewegung des Systems ist allerdings sehr nicht linear, da es sich um manövrierende Objekte handelt, die abrupt zwischen mehreren dynamischen Modellen wechseln können. Die Messfunktion selbst ist zudem nicht linear. So ist $h(\mathbf{x_1}) + h(\mathbf{x_2}) \neq h(\mathbf{x_1} + \mathbf{x_2})$. Aufgrund der hohen Nichtlinearität des Systems und der Messfunktion habe ich $\alpha = 0,01$ gesetzt. Dieser Wert sorgt auch dafür, dass der in Absatz 3.10.11 beschriebene Fehler seltener auftritt.

3.10.9. Initialisierung

Das auf dem UKF basierende Tracking-Verfahren braucht einen anfangs angenommenen Zustand, von dem aus die Zustandsschätzung des UKF fortfahren kann. Um diese zu ermitteln, wird die deterministische Bestimmung der Pose verwendet, wie sie in Abschnitt 3.9 erläutert ist. Hierfür muss allerdings die Bedingung erfüllt sein, dass in beiden Kamerabildern I_l und I_r alle vier Marker erkannt werden.

3.10.10. Abfangen zu hoher Varianzen

Der UKF gibt in der Matrix $\mathbf{P}$ die vom Filter selbst angenommene Abweichungen für einzelne Zustandsvariablen an. Wird diese zu groß, ist ab einer gewissen Abweichung davon auszugehen, dass die Vorhersage des Filters nicht mehr brauchbar ist und dieser einen Ausreißer erzeugt. Um sie abzufangen, greift im UKF-basierten Verfahren ein Kontrollmechanismus. Dieser versucht, einen Ausreißer zu erkennen und falls dieser Auftritt auf die Schätzung des deterministischen Verfahrens umzuschwenken, die parallel zu dem UKF-basierten Verfahren läuft. Hierfür wird geprüft, ob die Abweichung in $\mathbf{P}$ für die x-, y- oder z-Position größer als 500 mm und die Standardabweichung der Orientierungsschätzung auf den einzelnen Rotationsachsen größer

als $\frac{\pi}{2}$ wird. Geschieht dies insgesamt zweimal, wird auf die Schätzung des deterministischen Verfahrens gesetzt.

3.10.11. Fehler der Kovarianzmatrix und Neustart des Filters

Der beschriebene Filter bringt auf manchen Datensätzen nach wenigen Iterationen eine nicht Positiv semi definit (PSD) Matrix $\mathbf{P}$ hervor. Bei $\mathbf{P}$ handelt es sich um die Kovarianz-Matrix des Filters, die der Definition nach immer PSD sein muss. Der UKF kann mit einer nicht PSD Matrix nicht funktionieren. Im UT des Filters wird aus der Matrix die Wurzel gezogen. Hier wird die Cholesky-Zerlegung verwendet, die nach Labbe (2022a) empfohlen ist. Diese eignet sich hinsichtlich der Laufzeitgeschwindigkeit am besten für die Zerlegung der Matrix $\mathbf{P}$ und ist laut Labbe (2022a) heute Standard für den UKF.

Das Auftreten des Fehlers ließ sich in keiner Konfiguration zuverlässig vermeiden. Ein technischer Fehler konnte durch ein ausgiebiges Testen aller Funktionen ausgeschlossen werden. In einem GitHub-Beitrag beschreibt Roger Labbe zudem, dass dieses Problem ein erwartbares Beiprodukt ist, wenn versucht wird, nicht lineare Systeme mittels eines UKFs zu linearisieren[24]. Zwar handelt es sich hierbei nicht um eine wissenschaftliche Quelle, dennoch unterstützt sie die praktische Lösung, die auch ich gewählt habe. So fange ich den entsprechenden Fehler ab und initialisiere den Filter danach neu. Für den Ausfall des UKF greift das Tracking-Verfahren auf den aktuellen Zustand des deterministischen Verfahrens zurück und setzt den Filterzustand $\mathbf{x}$ auf dessen Schätzung $\mathbf{x}_{determ}$. Die Kovarianzmatrix $\mathbf{P}$ wird auf ihren Initialwert gesetzt. Die Neustarts des Filters werden zukünftig als β bezeichnet und zur Evaluation gespeichert.

3.11. Evaluation des Tracking-Verfahrens

In diesem Abschnitt ist das Vorgehen bei der Evaluation des UKF-basierten Tracking-Verfahrens beschrieben. Hierbei wird in Absatz 3.11.1 zunächst beschrieben, welches Qualitätsmaß ich gewählt habe und wie dieses berechnet wird. Anschließend gehe ich in Absatz 3.11.2 auf die Berechnung des zeitlichen Verlaufs des Fehlers bei der Schätzung einzelner Zustandsvariablen ein. Dieser Verlauf ist nötig, um Tracking-Durchläufe, deren Qualität dem Qualitätsmaß nach schlecht sind, näher zu untersuchen und auf Plausibilität zu überprüfen. Aus demselben Grund speichere ich den zeitlichen Verlauf der Varianzen des UKF für die einzelnen Zustandsvariablen. Deren Speicherung und Nutzen sind in Abschnitt 3.11.3 ebenfalls näher beschrieben.

[24] `https://github.com/rlabbe/filterpy/issues/31`, Stand: 19.07.2023

Abschließend erläutere ich in Absatz 3.11.6 das Vorgehen bei der quantitativen Analyse auf Basis der Simulationsdaten und in Absatz 3.11.7 das Vorgehen bei der qualitativen Analyse.

3.11.1. Definition eines Qualitätsmaßes

Die Evaluation des Filters erfolgt mittels des Root Mean Squared Error (RMSE). Hierbei habe ich mich für ein differenzierteres Bild dazu entschlossen, den RMSE nicht für den gesamten Zustand, sondern individuell für die Position und die globale Orientierung zu berechnen. Diese Entscheidung ist darin begründet, dass bei den zu trackenden Bewegungen keine Abhängigkeit zwischen der Position und Orientierung besteht. Ich habe also angenommen, dass die Positionsvorhersage sehr gut sein kann, während die Vorhersage der Orientierung schlecht sein kann. Die Richtung, in die ein zu trackendes Objekt beschleunigt, ist zudem unabhängig von dessen Orientierung. Diese Tatsache lässt sich anhand des Prozessmodells ablesen. Dieses Vorgehen findet sich so auch in der Arbeit von Gsaxner u. a. (2021). Es werden also der $RMSE_p$ für die Abweichung der Positionsschätzung und der $RMSE_o$ für die Abweichung der Orientierungsschätzung berechnet. Den RMSE habe ich statt des Mean Squared Error gewählt, da es sich bei dem RMSE nicht um quadrierte Werte handelt, die sich somit leichter interpretieren lassen.

Die allgemeine Formel für die Berechnung des RMSEs lautet wie folgt:

$$RMSE = \sqrt{\sum_{i=0}^{N} (gt_i - est_i)^2 \frac{1}{N}} \tag{3.38}$$

Wobei gt_i die Ground Trut-Daten und est_i die Filter-Schätzung zum Zeitpunkt i beschreibt. Für die Position des zu trackenden Objektes wird der $RMSE_p$ folgendermaßen berechnet:

$$RMSE_p = \sqrt{\sum_{i=0}^{N} ((x_{gt_i} - x_{est_i})^2 + (y_{gt_i} - y_{est_i})^2 + (z_{gt_i} - z_{est_i})^2) \frac{1}{N}} \tag{3.39}$$

Die Berechnung des $RMSE_o$ basiert auf der externen, globalen Orientierung, die außerhalb des Zustandes als Quaternion gespeichert wurde. Um die Abweichung zwischen zwei Quaternionen zu berechnen, lassen sich nicht die einzelnen imaginären und reellen Komponenten des Quaternions subtrahieren. Das Ergebnis dieser Subtraktion ist nicht zwangsweise ein Normquaternion und muss damit keine Orientierung darstellen. Gesucht ist die Differenz zwischen der Orientierung der Ground Truth-Daten $\mathbf{q}_{gt}$ und der Orientierung, die der Filter

zu dem entsprechenden Zeitpunkt berechnet hat $\mathbf{q}_{est}$. Diese Differenz lässt sich als die Rotation $\mathbf{e} \in \mathbb{H}$ beschrieben, die die Ground Truth-Orientierung in die vom Filter berechnete Orientierung überführt. Weiterhin ist es für die Evaluation der Filterperformance einfacher, einen Skalar zu nutzen, der die Abweichung zwischen den Orientierungen angibt. Dieser ist einfacher zu interpretieren als ein Quaternion. Somit verwende ich lediglich den Winkel der Rotation und nicht dessen Achse als Angabe dafür, wie weit zwei Orientierungen voneinander abweichen. Die Berechnung des Differenzquaternions basiert auf der Arbeit von Kraft (2003). Das Differenzquaternion $\mathbf{e} \in \mathbb{H}$ berechnet sich folgendermaßen:

$$\mathbf{q}_{est} = \mathbf{q}_e \mathbf{q}_{gt} \tag{3.40}$$

$\mathbf{q}_e$ beschreibt also die Rotation, die die Orientierung der Ground Truth-Daten in die der Schätzung überführt. Diese Rotation lässt sich mittels des Inversen von $\mathbf{q}_{gt}$ errechnen:

$$\mathbf{q}_e = \mathbf{q}_{est} \mathbf{q}_{gt}^{-1} \tag{3.41}$$

Um die so errechnete Differenz auf einen einfach zu interpretierenden Skalar zu reduzieren, wird der Winkel θ_{q_e} der Rotation berechnet. Dieser kann mithilfe des Skalarteils des Quaternions ermittelt werden:

$$\theta_{q_e} = 2 \cdot arccos(q_{e_w}) \tag{3.42}$$

Weiterhin muss der Mittelwert aller Differenzen θ_{q_e} gefunden werden. Hierfür gibt es zwei denkbare Ansätze. Zum einen lässt sich der Mittelwert aus den Differenzquaternionen berechnen. Aus diesem kann dann anschließend wie in 3.42 beschrieben der Winkel berechnet werden. Alternativ lässt sich auch der Mittelwert aus allen bereits errechneten Winkeln bilden. Ich habe mich für die zweite Variante entschieden. Wie Kraft (2003) beschreibt, ist die Mittelwertbildung mehrerer Quaternionen aufwendiger. Dafür muss der Fehler zwischen dem realen Mittelwert und dem angenommenen Mittelwert in einem iterativen Prozess verringert werden. Für die Mittelwertberechnung der einzelnen Winkel kann auf den zirkulären Mittelwert zurückgegriffen werden. Dieser ist sehr ähnlich zu dem in Abschnitt 3.10.4 beschriebenen gewichteten zirkulären Mittelwert. Im Falle der Evaluation ist dieser jedoch nicht gewichtet, beziehungsweise haben alle Winkel dasselbe Gewicht.

3.11.2. Zeitlicher Verlauf des Fehlers

Während der RMSE die gesamte Filterperformance in einem einzigen gut interpretierbaren Wert liefert, verbirgt er gleichzeitig einige wichtige Informationen. Um ein detaillierteres

Bild der Genauigkeit eines Filterdurchlaufs zu erhalten, habe ich für die Evaluation der Filterperformance ebenso den zeitlichen Verlauf der Fehler für die einzelnen Zustandsvariablen miteinbezogen. Es wird also für jeden Durchlauf des Filters für jeden Zeitschritt ein Fehlervektor gebildet, der den absoluten Fehler der jeweiligen Zustandsvariable enthält. Zusätzlich enthält der Fehlervektor die Abweichung der globalen Orientierung. Wie in Abschnitt 3.11.1 beschrieben, handelt es sich hierbei um den Winkel, den eine Rotation vollziehen muss, um die Ground Truth-Orientierung in die Orientierung zu überführen, die der Filter vorhergesagt hat. Der Fehlervektor ist folgendermaßen definiert:

$$\mathbf{e} = [e_x, e_y, e_z, e_{\dot{x}}, e_{\dot{y}}, e_{\dot{z}}, e_\phi, e_\theta, e_\psi, e_{\dot{\phi}}, e_{\dot{\theta}}, e_{\dot{\psi}}, e_o] \tag{3.43}$$

Die Abweichung für x, y, z sowie $\dot{x}, \dot{y}, \dot{z}$ zum Zeitpunkt t berechnen sich wie folgt:

$$\mathbf{e}_t = |\mathbf{gt}_t - \mathbf{est}_t| \tag{3.44}$$

$\mathbf{e}_t$ beschriebt also die absolute Abweichung der Schätzung von den Ground Truth-Daten zum Teitpunkt t. Die Richtung des Fehlers ist für das Ermitteln der Genauigkeit irrelevant.

Die Fehlerberechnung der inkrementellen Orientierung ϕ, θ, ψ sowie deren Winkelbeschleunigung wird ähnlich wie in Abschnitt 3.10.5 beschrieben unter Berücksichtigung der Subtraktion bei Winkeln durchgeführt. Er ist aber für die Evaluation nicht entscheidend, da es hier auf den Fehler bei der Berechnung der globalen Orientierung ankommt. Für die Berechnung dieses Fehlers kommen für jeden Zeitschritt t die Gleichungen 3.41 und 3.42 zum Einsatz. Auch hier wird der absolute Wert des Winkels genommen. Die Richtung des Winkels ist für die Ermittlung der Abweichung irrelevant.

3.11.3. Zeitlicher Verlauf der Filter-Varianzen

Neben dem zeitlichen Verlauf der Fehler der einzelnen Zustandsvariablen sind außerdem ihre Varianzen entscheidend. Hierbei handelt es sich um die Angabe des UKFs darüber, für wie sicher er seine Schätzung des aktuellen Systemzustandes hält. Diese Werte werden deshalb gespeichert, da sie bei der Analyse der Ergebnisse Aufschlüsse über die Qualität des Tracking-Verfahrens geben können. Die Fehler des Filters sollten in den Grenzen der angegebenen Genauigkeit des UKFs liegen. Ist das der Fall, kann davon ausgegangen werden, dass der Aufbau des Filters und die Informationen über das zu trackende System in Form von $\mathbf{Q}$ und $\mathbf{R}$ korrekt sind.

3.11.4. Reale Bewegungsdaten zur Evaluation

Wie in Abschnitt 3.8 und beschrieben, werden die Messdaten auf Basis von Bewegungsdaten für die Kamera und das zu trackende Objekt erstellt. Diese habe ich wie in 3.8.1 beschrieben mittels einer Meta Quest 2 in einer auf Unity basierenden VR-Anwendung aufgezeichnet. Für die Datenevaluation habe ich insgesamt neun unterschiedliche Datensätze in drei Kategorien erzeugt. Diese unterscheiden sich durch unterschiedliche Arten von Bewegungen und die Sichtbarkeit des Objektes. Der Aufbau dieser Messdaten ist in den folgen Abschnitten näher beschrieben. In Tabelle 3.5 befindet sich eine Übersicht der erzeugten Bewegungsdatensätze für Kamera und Objekt. Das Ergebnis des UKF-basierten und des deterministischen Verfahrens zählt für die quantitative und qualitative Evaluation als Untersuchungsobjekt. Die Untersuchungsobjekte unterschieden sich nach den Messdaten, auf deren Basis beide Verfahren ihre Zustandsschätzungen erzeugen.

Aus den neun zusammengehörigen Datensätzen der Kamerabewegung und der Objektbewegung habe ich mit der Messdatensimulation insgesamt 45 Messdatensätze erzeugt. Hierbei habe ich für jedes Paar aus Kamera- und Objektbewegung vier Varianten der Messdaten erzeugt. Die erste Variante enthält dabei am wenigsten Fehler und Ungenauigkeiten. Die Ungenauigkeiten der Messdaten nehmen mit den Varianten zu, sodass die vierte Variante die meisten Fehler enthält. Die Berechnung der Messdaten ist in Abschnitt 3.8.2 näher beschrieben. Dort ist auch eine Tabelle zu finden, die die Wahrscheinlichkeiten für die definierten Fehlertypen der jeweiligen Variante angibt. Die zunehmende Verschlechterung der Messdaten ermöglicht es, das auf dem UKF basierende Tracking-Verfahren differenzierter zu evaluieren und dem deterministischen Verfahren gegenüberzustellen. Jeder Datensatz beinhaltet Bewegungsdaten aus einem Zeitintervall von ungefähr 60 Sekunden, die mit einem Zeitschritt von $\Delta t = \frac{1}{60}$ erzeugt wurden.

Für die quantitative und qualitative Evaluation werden die Ergebnisse der Filterung und die Ground Truth-Daten miteinander verglichen. Dabei werden zunächst für die Position und die Orientierung des Zustandes der RMSE, wie in Abschnitt 3.11.1 beschrieben, berechnet. Für jeden Durchgang liegen also der $RMSE_p$ und der $RMSE_o$ für die Ergebnisse beider Verfahren vor, wobei p hier für Position und o für Orientierung stehen.

Zur Berechnung von $RMSE_p$ und $RMSE_o$ werden lediglich die Schätzungen miteinbezogen, für die ein Messwert vorhanden war, die Markerpose also nicht außerhalb des Sichtfeldes der Kamera lag. Zusätzlich wird die Anzahl der Neustarts β (siehe Absatz 3.10.11) sowie die prozentuale Sichtbarkeit der Pose und die jeweilige Variante der Messdaten gespeichert.

Für jeden Durchgang des UKF-basierten und des deterministischen Verfahrens liegen somit folgende Werte vor:

- Variante der Messdaten (0-4)
- Anzahl der Neustarts nach Absatz 3.10.11
- Prozentuale Sichtbarkeit
- $RMSE_p$
- $RMSE_o$

Kategorie 1 - statisches Objekt

Eine mögliche Aufgabe, die das Tracking-System erfüllen kann, ist das Tracking eines nicht beweglichen Gegenstandes. Hierfür habe ich einen Datensatz erzeugt, in dem die Kamera einen nicht beweglichen Gegenstand durchgängig fixiert. Weiterhin habe ich einen Datensatz aufgezeichnet, bei dem sich der Gegenstand nicht bewegt, aber die Kamera. Zuletzt habe ich einen Datensatz aufgezeichnet, bei dem die Kamera das statische Objekt immer wieder aus dem Blickfeld verliert.

Kategorie 2 - kleine langsame Bewegungen des Objekts

Weiterhin habe ich eine Reihe Datensätze erzeugt, bei denen sich das Objekt nur minimal bewegt. Hierbei habe ich zunächst einen Datensatz erzeugt, bei dem sich das Objekt minimal im Blickfeld einer sich ebenfalls kaum bewegenden Kamera bewegt. Anschließend habe ich einen Datensatz erzeugt, bei dem sich das Objekt ebenfalls nur in kleinen langsamen Bewegungen bewegt, während die Kamera ausladendere Bewegungen macht, sich das Objekt allerdings den Großteil der Zeit im Blickfeld der Kamera befindet. Zuletzt habe ich einen Datensatz erzeugt, bei dem sich das Objekt erneut minimal bewegt, die Kamera es aber regelmäßig aus den Augen verliert.

Kategorie 3 - große ausladende Bewegungen des Objekts

Zuletzt habe ich Datensätze mit großen und ausladenden Bewegungen des Objektes erstellt. Hierbei habe ich das Objekt in der VR in die Hand genommen und vor meinem Blickfeld damit ausladende Bewegungen durchgeführt. Bei diesen Bewegungen ließ sich der Anteil der Sichtbarkeit des Objektes schlechter kontrollieren, daher habe ich diese anders als bei den übrigen Datensätzen nicht explizit als Kriterium zur Datenerstellung genutzt. Ich habe aus

Tabelle 3.5.: Übersicht der zur qualitativen Analyse erzeugten Datensätze

Kategorie	Name	Beschreibung
1	**static-fixiated-min-move**	Objekt unbewegt Kamera minimal bewegt Objekt durchgängig im Blick
	static-fixiated-max-move	Objekt unbewegt Kamera bewegt Objekt durchgängig im Blick
	static-lost-fixiation-max-move	Objekt unbewegt Kamera bewegt Verliert Objekt aus dem Blick
2	**min-move-pose-min-move-cam-fixiation**	Objekt minimal bewegt Kamera minimal bewegt Objekt durchgängig im Blick
	min-move-pose-max-move-cam-fixiation	Objekt minimal bewegt Kamera ausladend bewegt Objekt durchgängig im Blick
	min-move-pose-max-move-cam-lost-fixiation	Objekt minimal bewegt Kamera ausladend bewegt Verliert Objekt aus dem Blick
3	**max-move-constant-slow**	Objekt konstant langsam, ausladend bewegt
	max-move-constant-fast	Objekt konstant schnell, ausladend bewegt
	max-move-changing-speed	Objekt mit stark wechselnder Geschwindigkeit ausladend bewegt

dieser Kategorie drei Datensätze erzeugt. Bei einem bleibt die Geschwindigkeit konstant eher gering. Bei dem zweiten bleibt sie konstant hoch und bei dem dritten variiert sie stark.

3.11.5. Künstliche Bewegungsdaten zur Evaluation

Die eigentliche Evaluation des Filters findet wie in Absatz 3.11.4 beschrieben mit realen Bewegungsdatensätzen statt, die mittels eines HMDs aufgezeichnet wurden. Zusätzlich habe ich jedoch künstliche Bewegungsdaten erstellt. Diese folgen demselben Prinzip wie die realen Datensätze. Es wird jeweils für die Bewegung des zu trackenden Objektes und der Kamera ein Datensatz erstellt. Anschließend werden diese miteinander kombiniert. Auf Basis der Kombinationen wird dann mittels der in Abschnitt 3.8 beschriebenen Messdatensimulation ein Messdatensatz erzeugt. Auch für diese wird eine in Abschnitten 3.11.6 beschriebenen quantitative Analysen durchgeführt, bei der die Ergebnisse des deterministischen Verfahrens mit denen des UKF-basierten Verfahrens verglichen werden. Einzelne Ergebnisse werden zudem näher betrachtet. Ziel der Evaluation anhand künstlicher Bewegungsdaten besteht darin, die grundsätzliche Funktion des entwickelten Tracking-Verfahrens, zu validieren.

Die Datensätze wurden mittels des in Absatz 3.10.1 beschriebenen Prozessmodells erzeugt. Dabei variiert die Distanz, die ein zu trackendes Objekt und die Kamera zurücklegen. Weiterhin wird das Modell hier künstlich verfälscht, sodass sich die Beschleunigung in x-, y- oder z-Richtung künstlich abrupt erhöht oder verringert. Ziel ist es, manövrierende Bewe-

gungen nachzuahmen. Der Zeitschritt der so erstellen Bewegungsdatensätze beträgt ebenfalls $\Delta t = \frac{1}{60}s$. Hierbei werden Bewegungsdaten für 600 Zeitschritte aufgezeichnet, sodass das Tracking insgesamt 10 Sekunden dauert. Insgesamt liegen 605 Messdatensätze vor, sodass für jedes der beiden Verfahren insgesamt 605 Ergebnisse vorliegen.

3.11.6. Quantitative Evaluation

Ziel der quantitativen Evaluation war es festzustellen, ob das UKF-basierte Verfahren eine signifikante Verbesserung gegenüber dem deterministischen Verfahren darstellt. Um das anhand statistischer Signifikanztests zu ermitteln, habe ich die folgenden Hypothesen aufgestellt:

Hypothese 1 $RMSE_{p_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{p_{determ}}$ des deterministischen Verfahrens.

Hypothese 2 $RMSE_{o_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{o_{determ}}$ des deterministischen Verfahrens.

Hypothese 3 $RMSE_{p_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{p_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 0.

Hypothese 4 $RMSE_{p_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{p_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 1.

Hypothese 5 $RMSE_{p_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{p_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 2.

Hypothese 6 $RMSE_{p_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{p_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 3.

Hypothese 7 $RMSE_{p_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{p_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 4.

Hypothese 8 $RMSE_{o_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{o_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 0.

Hypothese 9 $RMSE_{o_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{o_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 1.

Hypothese 10 $RMSE_{o_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{o_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 2.

Hypothese 11 $RMSE_{o_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{o_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 3.

Hypothese 12 $RMSE_{o_{ukf}}$ des UKF-basierten Verfahrens ist kleiner als $RMSE_{o_{determ}}$ des deterministischen Verfahrens bei Messdaten der Variante 4.

Diese Hypothesen werden mittels eines statistischen Signifikanztests überprüft. Hypothese 1 und Hypothese 2 beziehen sich auf die Gesamtheit aller Durchläufe. Hier wurden also alle Varianten der Messdaten miteinbezogen, womit insgesamt 45 Untersuchungsobjekte vorliegen. Die übrigen Hypothesen beziehen sich jeweils auf den signifikanten Unterschied von $RMSE_p$ und $RMSE_o$ innerhalb einer Messdatenvariante. Diese machen ein Fünftel der gesamten Stichprobe aus, womit jeweils neun Untersuchungsobjekte vorliegen.

Für keine der Daten $RMSE_{o_{ukf}}$, $RMSE_{p_{ukf}}$,$RMSE_{o_{determ}}$ und $RMSE_{p_{determ}}$ liegt eine Normalverteilung vor. Da es sich bei dem RMSE um Werte einer metrischen Skala handelt und zwei unabhängige Gruppen miteinander vergleichen werden, kommt ein Mann-Whitney-U-Test, wie er in Nachar (2008) beschrieben wird, zum Einsatz, um die Hypothesen eins bis zwölf zu überprüfen. Da die Effektgröße als groß anzusehen ist, sind die minimal 9 Stichproben ausreichend.

Weiterhin habe ich den Pearson Korrelationskoeffizienten r verwendet, um Korrelationen zwischen Informationen und Variablen zu testen. Dieser ist für zwei Variablen a und b laut Benesty u. a. (2009) wie folgt definiert:

$$r(a,b) = \frac{E(a,b)}{\sigma_a, \sigma_b} \tag{3.45}$$

Wobei $E(a,b)$ die Kreuzkorrelation von a und b angibt.

3.11.7. Qualitative Evaluation mit echten Bewegungsdaten

Neben einer quantitativen Betrachtung, die zudem den Vergleich zu dem deterministischen Verfahren beinhaltet, habe ich das auf dem UKF basierende Verfahren zudem qualitativ untersucht. Diese Untersuchung sollte feststellen, unter welchen Bedingungen das Verfahren mit welcher

Qualität funktioniert. Hierbei habe ich ebenfalls die in Absatz 3.11.4 erzeugten Bewegungsdaten verwendet. Untersucht wurde hier, wie sich die Qualität des UKF-basierten Verfahrens bei den Datensätzen der verschiedenen Kategorien verhält und bei welchen Bewegungsabläufen das Tracking wie genau war. Weiterhin habe ich mit der qualitativen Untersuchung auch Ergebnisse tiefergehend untersucht, die eine geringe Genauigkeit aufwiesen. Dafür habe ich auch auf den $RMSE_p$ und den $RMSE_o$ zurückgegriffen. Zusätzlich kommen für die qualitative Untersuchung auch die Fehlervektoren und Varianzen des Filters aus Absatz 3.11.2 und 3.11.3 zum Einsatz.

4. Resultate

In diesem Kapitel werden die Ergebnisse der in Abschnitt 3.11 beschriebenen Evaluationen beschrieben. Die Interpretation dieser Ergebnisse ist in Kapitel 5 zu finden. Hierfür gehe ich in Absatz 4.1 auf die Ergebnisse der Evaluation mit künstlichen Bewegungsdaten ein. In Absatz 4.2 gehe ich anschließend auf die Ergebnisse der quantitativen Auswertung ein. Im Absatz 4.3 beschreibe ich weiterhin die Ergebnisse der qualitativen Auswertung. Die Ergebnisse der Durchläufe beider Verfahren auf Messdaten aus realen Bewegungsdaten sind im Anhang in Tabelle A.1 abgebildet.

4.1. Ergebnisse der Evaluation mit künstlichen Bewegungsdaten

Die in Absatz 3.11.6 beschriebenen Hypothesentests habe ich für die Ergebnisse auf Basis künstlich erstellter Bewegungsdaten ebenfalls durchgeführt. Den Aufbau der Tests werde ich in diesem Absatz nicht detailliert besprechen, da er in Absatz 4.2.1 ausreichend besprochen wird. Wie dort auch, liegt für die Ergebnisse auf Basis künstlicher Bewegungsdaten keine Normalverteilung vor, weswegen auch hier ein Mann-Whitney-U-Test zum Einsatz kommt. Mit einer Sicherheitswahrscheinlichkeit von 5% lässt sich sagen, dass das UKF-basierte Verfahren hinsichtlich der Orientierungsschätzung ungenauer ist. Für die Positionsschätzung ergibt sich kein signifikanter Unterschied. Diese Erkenntnis deckt sich mit der in Absatz 4.2.1 beschriebenen.

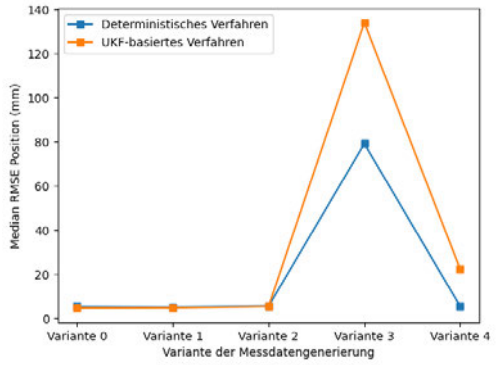
Tabelle 4.1.: Übersicht der Auswertung des UKF-basierten Verfahrens mit künstlichen Bewegungsdaten

UKF-basiert	$RMSE_p(mm)$	$RMSE_o(rad)$
Mittelwert	36,65	0,41
Median	5,85	0,30
Varianz	3064,76	0,11
Standardabweichung	55,36	0,34

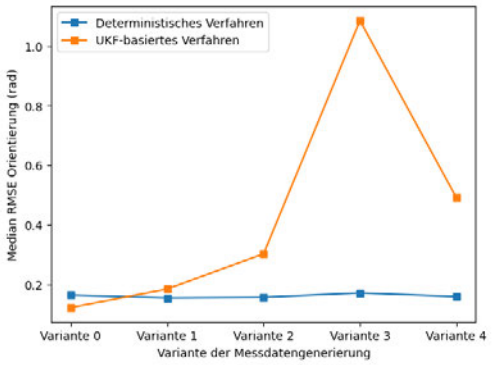
Tabelle 4.2.: Übersicht der Auswertung des deterministischen Verfahrens mit künstlichen Bewegungsdaten

Deterministisch	$RMSE_p$(mm)	$RMSE_o$(rad)
Mittelwert	24,65	0,16
Median	5,54	0,16
Varianz	1661,41	0,0003
Standardabweichung	40,76	0,02

In Tabelle 4.1 und 4.2 sind die Mittel- und Medianwerte sowie die Varianz und die Standardabweichung für den $RMSE_p$ und den $RMSE_o$ beider Verfahren zu sehen. Hier sieht man in Tabelle 4.1 und 4.2 sowie den Abbildungen 4.1 und 4.2 deutlich, dass das deterministische Tracking-Verfahren hinsichtlich der Positionsschätzung minimal besser abschneidet, während es hinsichtlich der Orientierungsschätzung deutlich schlechter abschneidet. Diese Abbildungen decken sich mit den Ergebnissen der Hypothesentests. Der RMSE der Positionsschätzung ist mit 5,5 bis 5,9 mm im Median gering. Ebenso bei der Orientierungsschätzung ist der RMSE mit 0,3 bis 0,16 im Medien gering. Insbesondere im Vergleich zu den Ergebnissen bei realen Bewegungsdaten. Die Abbildungen 4.2 und 4.1 zeigen das, was die Hypothesentests bereits ergeben haben. Hinsichtlich der Positionsschätzung ist sowohl im Median, als auch im Mittel kein signifikanter Unterschied festzustellen. Hier sind die Werte des deterministischen Verfahrens geringfügig besser. Bezogen auf die Orientierungsschätzung zeigt sich, dass sowohl der Mittel- als auch der Medianwert des $RMSE_{o_{determ}}$ für die meisten Messdatenvarianten geringer ist. Der Abstand zwischen dem $RMSE_{o_{determ}}$ und dem $RMSE_{o_{ukf}}$ vergrößert sich bei zunehmender verschlechterung der Messdaten.

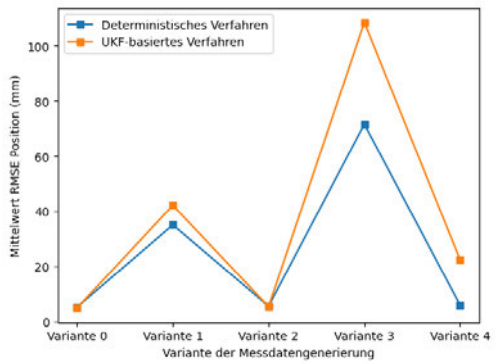


(a) $RMSE_p$ Median für beide Verfahren im Verhältnis zu den Messdatenvarianten

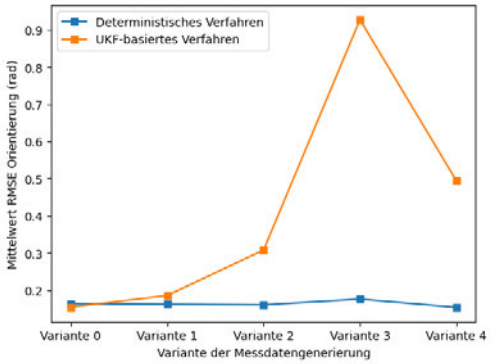


(b) $RMSE_o$ Median für beide Verfahren im Verhältnis zu den Messdatenvarianten

Abbildung 4.1.: Vergleich Median des $RMSE_p$ und des $RMSE_o$ bei verschiedenen Messdatenvarianten und künstlichen Bewegungsdaten



(a) $RMSE_p$ Mittelwert für beide Verfahren im Verhältnis zu den Messdatenvarianten



(b) $RMSE_o$ Mittelwert für beide Verfahren im Verhältnis zu den Messdatenvarianten

Abbildung 4.2.: Vergleich Mittelwert des $RMSE_p$ und des $RMSE_o$ bei verschiedenen Messdatenvarianten und künstlichen Bewegungsdaten

4.2. Ergebnisse der quantitativen Evaluation mit realen Bewegungsdaten

In den Tabellen 4.3 und 4.4 sind die wichtigsten Werte der Ergebnisse auf realen Bewegungsdaten abgebildet. Hervorzuheben sind hier die hohe Varianz und Standardabweichung des $RMSE_p$ beider Verfahren. Diese deuten auf vereinzelte Ausreißer hin. der $RMSE_{p_{ukf}}$ ist im Mittel geringer als der $RMSE_{p_{determ}}$, im Median ist er geringfügig höher. Ein eindeutiger Unterschied ist hier anhand der beschreibenden Statistik nicht festzustellen.

Der $RMSE_{o_{ukf}}$ und $RMSE_{o_{determ}}$ unterscheiden sich sowohl im Median, als auch hinsichtlich des Mittelwerts deutlich voneinander. So sind sowohl der Mittelwert als auch der Median, die Varianz und die Standardabweichung des $RMSE_{o_{determ}}$ geringer als dieselben Werte des $RMSE_{o_{ukf}}$. Generell sind die RMSE-Werte bei realen Bewegungsdaten höher als bei künstlichen. Die Verhältnisse der Median- und Mittelwerte sind aber ähnlich zu den Ergebnissen auf Basis künstlicher Bewegungsdaten.

Tabelle 4.3.: Übersicht der Auswertung des UKF-basierten Verfahrens mit realen Bewegungsdaten

UKF-basiert	$RMSE_p$(mm)	$RMSE_o$(rad)
Mittelwert	436,75	1,22
Median	245,09	0,79
Varianz	435133,75	0,99
Standardabweichung	659,65	0,99

Tabelle 4.4.: Übersicht der Auswertung des deterministischen Verfahrens mit realen Bewegungsdaten

Deterministisch	$RMSE_p$(mm)	$RMSE_o$(rad)
Mittelwert	586,75	0,27
Median	159,75	0,28
Varianz	830345,15	0,01
Standardabweichung	911,23	0,10

4.2.1. Ergebnisse der Hypothesentests

Mithilfe der erzeugten Daten habe für jede der zwölf Hypothesen im ersten Schritt ein ungerichteter Mann-Whitney-U-Test durchgeführt, um zu ermitteln, ob zwischen den beiden Verfahren ein signifikanter Unterschied besteht. Dieser wurde mit einer Verwerfungsgrenze von 5% durchgeführt.

Für Hypothese 1, 3, 4, 5, 6 und 7 konnte kein signifikanter Unterschied zwischen den beiden Verfahren nachgewiesen werden. Für die übrigen Hypothesen ließ sich bei einer Sicherheitswahrscheinlichkeit von 95% jedoch ein signifikanter Unterschied feststellen. Hypothese 1 besagt $RMSE_{p_{ukf}} < RMSE_{p_{determ}}$ für alle Messdaten und Messdatenvarianten. Die sich daraus ergebende Nullhypothese lautet $RMSE_{p_{ukf}} \geq RMSE_{p_{determ}}$. Ein ungerichteter Mann-

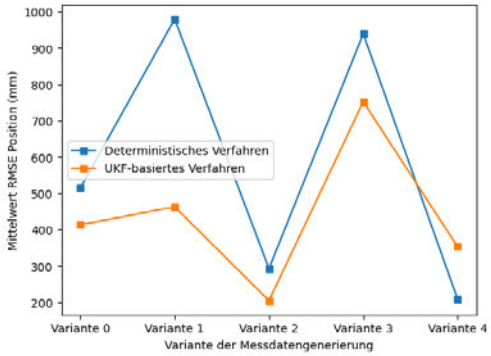
Whitney-U-Test hat einen p-Wert von 0,796 ergeben, womit die für diesen ungerichteten Test aufgestellte Nullhypothese $RMSE_{p,ukf} = RMSE_{p,determ}$ nicht abgelehnt werden kann. Hinsichtlich der Positionsschätzung konnte auch für die übrigen Hypothesen, die sich, wie in Absatz 3.11.6 beschrieben, auf die Unterschiede innerhalb einer Messdatenvariante beziehen, kein statistisch signifikanter Unterschied festgestellt werden. Womit hier kein Unterschied der Qualität nachgewiesen werden konnte. Das deckt sich mit den Ergebnissen, die in Tabelle 4.3 und 4.4 abgebildet sind. Hier ist allerdings der Mittelwert von $RMSE_{p_{ukf}}$ bedeutend kleiner, während der Median zwar größer, der Unterschied aber geringer ist. Zudem ist die Varianz der $RMSE_{p_{ukf}}$ ungefähr um die Hälfte kleiner.

Für die Hypothesen 2, 8, 9, 10, 11 und 12 konnte ein signifikanter Unterschied festgestellt werden. Hypothese 2 besagt $RMSE_{o_{ukf}} < RMSE_{o_{determ}}$. Ein linksseitiger Mann-Whitney-U-Test hat einen p-Wert von 0,999 ergeben, wodurch die Nullhypothese $RMSE_{p_{ukf}} \geq RMSE_{p_{determ}}$ nicht abgelehnt werden konnte. Ein weiterer rechtsseitiger Test hat einen p-Wert von 0,00000000297 ergeben. Dieser Wert liegt deutlich unter der Verwerfungsgrenze, womit die zu dem rechtsseitigen Test gehörende Nullhypothese $RMSE_{o_{ukf}} \leq RMSE_{o_{determ}}$ verworfen und somit $RMSE_{o_{ukf}} > RMSE_{o_{determ}}$ angenommen werden muss. Dieses Ergebnis wiederholt sich bei den Tests zu den Hypothesen 8, 9, 10, 11 und 12, die sich auf den Unterschied zwischen $RMSE_{o_{ukf}}$ und $RMSE_{o_{determ}}$ innerhalb einzelner Messdatenvarianten beziehen.

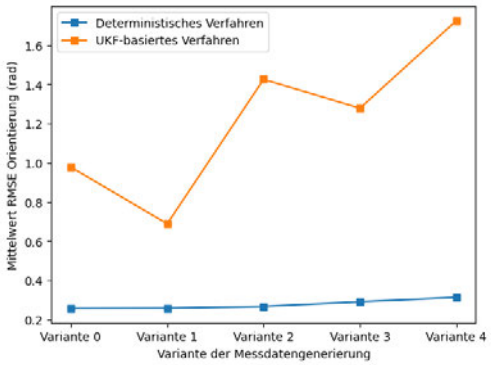
In Abbildung 4.3 sind die Mittelwerte des $RMSE_p$ und $RMSE_o$ im Verhältnis zu den Messdaten zu sehen. Der Unterschied zwischen beiden Verfahren ist hinsichtlich der Positions- und Orientierungsschätzung sehr verschieden. In den Abbildungen 4.4 ist der Median statt dem Mittelwert zu sehen. Hier ist zu erkennen, dass das deterministische Verfahren bei den Messdatenvarianten eins und vier besser abschneidet. Es lässt sich somit festhalten, dass der Mittelwert in Bezug auf die Positionsschätzung des UKF-basierten Systems niedriger ist, während der Median hier höher ist als das deterministische Verfahren. Hinsichtlich der Orientierungsschätzung ist die Unterscheidung zwischen Median und Mittelwert nicht so gravierend. In beiden Fällen ist der $RMSE_{o_{ukf}}$ bei allen Messdatenvarianten deutlich höher.

4.2.2. Korrelation zwischen Neustarts und Filtergenauigkeit

Der in Abschnitt 3.10.11 beschriebene Fehler der Kovarianz-Matrix $\mathbf{P}$ hat das Potenzial, die Schätzung des Tracking-Verfahrens zu beeinflussen. Für eine Verschlechterung aufgrund des Fehlers und der damit verbundenen Neustarts β müsste dementsprechend eine positive

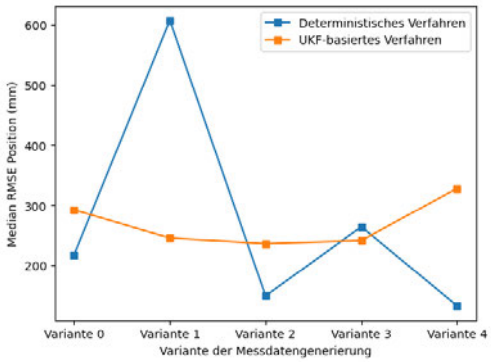


(a) $RMSE_p$ Mittelwert für beide Verfahren im Verhältnis zu den Messdatenvarianten

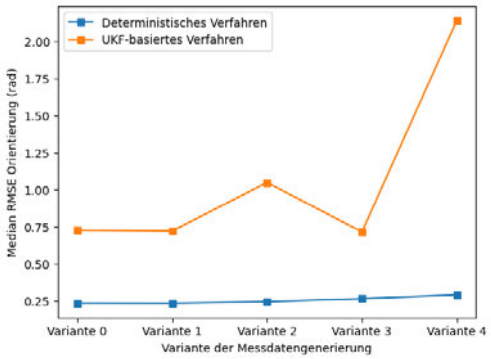


(b) $RMSE_o$ Mittelwert für beide Verfahren im Verhältnis zu den Messdatenvarianten

Abbildung 4.3.: Vergleich Mittelwert des $RMSE_p$ und des $RMSE_o$ bei verschiedenen Messdatenvarianten und realen Bewegungsdaten



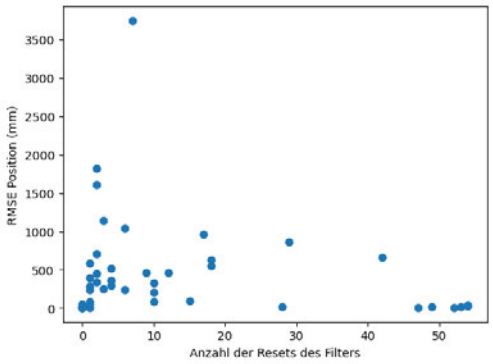
(a) $RMSE_p$ Median für beide Verfahren im Verhältnis zu den Messdatenvarianten



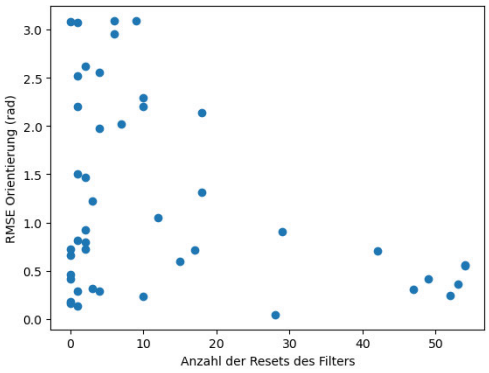
(b) $RMSE_o$ Median für beide Verfahren im Verhältnis zu den Messdatenvarianten

Abbildung 4.4.: Vergleich Median des $RMSE_p$ und des $RMSE_o$ bei verschiedenen Messdatenvarianten und realen Bewegungsdaten

Korrelation zwischen β und des $RMSE_p$ sowie zwischen den β und $RMSE_o$ bestehen. Die Korrelation wurde mithilfe des Pearson Korrelationskoeffizienten r berechnet. So ist $r(\beta, RMSE_p) = -0,166$, womit sich eine negative Korrelation erkennen lässt. Ähnlich sieht das für $r(\beta, RMSE_o) = -0,331$ aus. Hier ist ebenfalls eine negative Korrelation abzulesen. Die Korrelationen sind in Abbildung 4.5 zu erkennen.



(a) Streudiagramm von β im Verhältnis zu $RMSE_p$



(b) Streudiagram von β im Verhältnis zu $RMSE_o$

Abbildung 4.5.: Verhältnis zwischen den Neustarts des UKF-basierten Verfahrens und dessen Qualität

4.3. Ergebnisse der qualitativen Auswertung

Für die qualitative Evaluation kommen dieselben Datensätze wie für die quantitative Analyse zum Einsatz. Allerdings werden diese hier differenzierter und nach den in Absatz 3.11.4 eingeführten Kategorien betrachtet. Der Vergleich zum deterministischen Verfahren ist dabei nebensächlich, da er im Rahmen der quantitativen Analyse abgedeckt wurde. In den Tabellen 4.5, 4.6 und 4.7 sind jeweils der Mittelwert, der Median sowie die Varianz und die Standardabweichung des $RMSE_p$ und $RMSE_o$ für die Kategorie 1, 2 und 3 abgebildet. In Absatz 4.3.1 schildere ich zunächst isoliert die Ergebnisse für die Datensätze aus Kategorie 1. In Absatz 4.3.2 und 4.3.3 folgen die isolierten Betrachtungen der Ergebnisse aus Kategorie 2 und 3. In Absatz 4.3.4 folgt eine abschließende Betrachtung der Ergebnisse aller Datensätze.

4.3.1. Ergebnisse für statisches Objekt - Kategorie 1

In der Tabelle 4.5 sind die Ergebnisse für das Tracking des UKF-basierten Trackingverfahrens zu sehen. In der Tabelle wird deutlich, dass sich der $RMSE_p$ und der $RMSE_o$ mit zunehmender Bewegung verschlechtern. Hier bezieht sich die Bewegung auf die Kamera, da das Objekt statisch ist. Die niedrigsten Werte sind für den Datensatz “static-fixiated-min-move” zu finden. Der $RMSE_p$ liegt hier im Mittel bei 12,19 mm, während der Median bei 7,86 mm liegt. Das Mittel des $RMSE_o$ liegt bei 0,33 rad. Der Median liegt bei 0,29 rad. Aus den Varianzen ergeben sich die Standardabweichungen von 8,55 mm und 0,14 rad.

Der Datensatz “static-fixiated-max-move” weist einen deutlich höheren Mittel- und Median-Wert für den $RMSE_p$ und den $RMSE_o$ auf. Der $RMSE_p$ liegt hier bei 193,12 mm im Mittel und 83,93 mm im Median. Hier ist zudem die Varianz und somit die Standardabweichung mit 260,34 mm deutlich größer. Für den $RMSE_o$ ergibt sich ein Mittelwert von 0,49 rad und ein Median von 0,59 rad. Die Standardabweichung liegt bei 0,3 rad.

Der Datensatz “static-lost-fixiated-max-move” beschreibt ebenfalls große Kamerabewegungen um das statische Objekt. Hier ist das Objekt allerdings über den gesamten Zeitraum des Datensatzes nur 81,576% der Zeit im Sichtfeld der Tracking-Kamera. Der Mittelwert des $RMSE_p$ liegt bei 609,43 mm. Der Median ist etwas niedriger und liegt bei 582,87 mm. Die Standardabweichung ist mit 152,84 mm ebenfalls größer. Während Median und Mittelwert größer sind als bei dem Datensatz “statisch-fixiated-max-move”, sind Varianz und Standardabweichung hier geringer. Der Mittelwert des $RMSE_o$ liegt bei 1,01 rad, während der Median bei 1,04 rad liegt. Die Standardabweichung beträgt 0,47 rad. Somit ist auch hier eine deutliche Verschlechterung gegenüber den übrigen Datensätzen zu erkennen.

Tabelle 4.5.: Übersicht der Auswertung des UKF-basierten Verfahrens bei statischem Objekt

Kategorie 1	**static-fixiated-min-move**	**static-fixiated-max-move**	**static-lost-fixiation-max-move**
Mittelwert $RMSE_p$ (mm)	12,19	193,12	609,43
Median $RMSE_p$(mm)	7,86	83,93	582,87
Varianz $RMSE_p$	73,07	67775,45	23361,37
Standardabweichung $RMSE_p$(mm)	8,55	260,34	152,84
Mittelwert $RMSE_o$(rad)	0,33	0,49	1,01
Median $RMSE_o$(rad)	0,29	0,59	1,04
Varianz $RMSE_o$(rad)	0,02	0,09	0,21
Standardabweichung $RMSE_o$(rad)	0,14	0,3	0,47

4.3.2. Ergebnisse für minimal bewegtes Objekt - Kategorie 2

Auch für die Ergebnisse der Kategorie 2 sind der $RMSE_p$ und der $RMSE_o$ im Mittel mit 16,79 mm und 0,48 rad sowie im Median mit 9,83 mm und 0,41 rad für den Datensatz “min-move-pose-min-move-cam-fixiation” am geringsten. Dieser weist die geringste Kamerabewegung der Kategorie 2 auf. Dasselbe gilt für die Standardabweichung von 13,81 mm beziehungsweise 0,16 rad.

Für den zweiten Datensatz, bei dem sich die Kamera um das Objekt bewegt, es aber durchgängig fixiert, verschlechtern sich $RMSE_p$ und $RMSE_o$ mit 20,66 mm und 0,88 rad im Mittel sowie

mit 15,97 mm und 0,46 rad im Median. Während die Verschlechterung des $RMSE_p$ sowohl für den Mittelwert als auch den Median eher gering ausfällt, verdoppelt sich der Mittelwert des $RMSE_o$ gegenüber dem vorherigen Datensatz beinahe. Das spricht für einige Durchgänge, bei denen die Orientierungsschätzung drastisch schlechter geworden ist. Die Standardabweichung liegt hier bei 16,98 mm beziehungsweise 1,26 rad.

Der dritte Datensatz vollzieht eine ähnliche Bewegung wie der zweite. Allerdings fixieren die Kameras hier das Objekt nicht durchgängig. Der $RMSE_p$ und $RMSE_o$ liegen hier im Mittel bei 837,18 mm beziehungsweise 1,26 rad und im Median bei 961,18 mm beziehungsweise 0,79 rad. Hier liegt eine deutliche Verschlechterung vor. Allerdings fällt die Verschlechterung für den $RMSE_p$ stärker aus, als für den $RMSE_o$. Die Standardabweichung hat sich hier stark erhöht und liegt bei 354930,24 mm. Das spricht ebenfalls für einige starke Abweichungen. Die Standardabweichung des $RMSE_o$ ist mit 0,92 rad etwas gesunken.

Tabelle 4.6.: Übersicht der Auswertung des UKF-basierten Verfahrens bei minimal bewegtem Objekt

Kategorie 2	**min-move-pose-min-move-cam-fixiation**	**min-move-pose-max-move-cam-fixiation**	**min-move-pose-max-move-cam-lost-fixiation**
Mittelwert $RMSE_p$ (mm)	16,79	20,66	837,18
Median $RMSE_p$(mm)	9,83	15,97	961,18
Varianz $RMSE_p$(mm)	190,58	288,33	354930,24
Standardabweichung $RMSE_p$(mm)	13,81	16,98	595,76
Mittelwert $RMSE_o$(rad)	0,48	0,88	1,26
Median $RMSE_o$(rad)	0,41	0,46	0,79
Varianz $RMSE_o$(rad)	0,03	1,57	0,84
Standardabweichung $RMSE_o$(rad)	0,16	1,26	0,92

4.3.3. Ergebnisse für maximal bewegtes Objekt - Kategorie 3

Die Ergebnisse aus Kategorie 3 weichen allesamt sehr stark von denen der anderen beiden Kategorien ab. Für den Datensatz “max-move-constant-slow” liegen der $RMSE_p$ und der $RMSE_o$ im Mittel bei 1089,43 mm sowie 2,052 rad und im Median bei 452,63 mm beziehungsweise 2,141 rad. Die Standardabweichung des $RMSE_p$ ist hier mit 1486,47 mm höher als bei den übrigen Datensätzen. Für den $RMSE_o$ liegt sie bei 0,68 rad.

Für den Datensatz “max-move-constant-fast” liegt eine Verbesserung der Positions- und eine Verschlechterung der Orientierungsschätzung vor. So liegen der $RMSE_p$ und der $RMSE_o$

hier im Mittel bei 852,70 mm beziehungsweise 2,261 rad und im Median bei 708,62 mm und 2,95 rad. Die Standardabweichung ist hier mit 616,46 mm für den $RMSE_p$ geringer. Für den $RMSE_o$ ist sie mit 1,1 rad gestiegen.

Der Datensatz "max-move-changing-speed" stellt weiterhin eine Verbesserung der Positionsschätzung dar, während für die Orientierungsschätzung weder eine eindeutige Verbesserung noch eine eindeutige Verschlechterung zu den übrigen Datensätzen der Kategorie 3 feststellen lässt. So liegen der $RMSE_p$ und der $RMSE_o$ hier im Mittel bei 299,28 mm und 2,2 rad sowie im Median bei 291,97 mm und 2,1 rad. Auffällig ist hier, dass Mittel und Median nah beieinander liegen und auch die Standardabweichungen mit 62,38 mm und 0,47 rad am geringsten sind.

Tabelle 4.7.: Übersicht der Auswertung des UKF-basierten Verfahrens bei maximal bewegtem Objekt

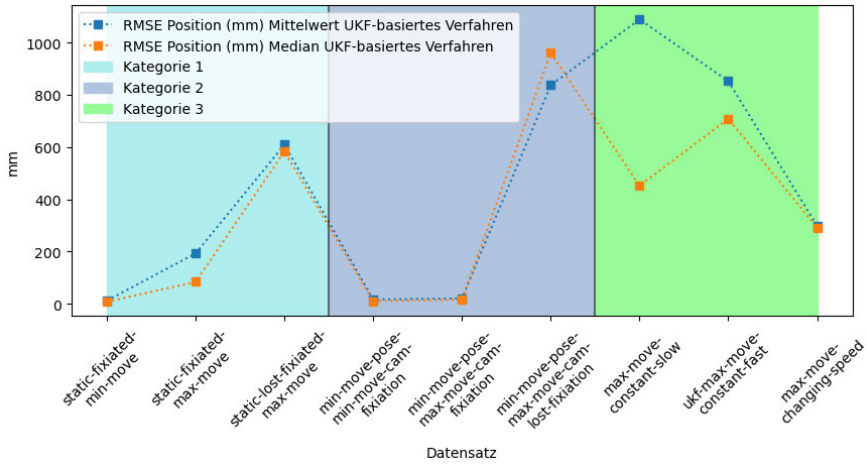
Kategorie 3	**max-move-constant-slow**	**max-move-constant-fast**	**max-move-changing-speed**
Mittelwert $RMSE_p$ (mm)	1089,43	852,70	299,28
Median $RMSE_p$(mm)	452,63	708,62	291,97
Varianz $RMSE_p$(mm)	2209604,88	379905,44	3891,13
Standardabweichung $RMSE_p$(mm)	1486,47	616,46	62,38
Mittelwert $RMSE_o$(rad)	2,052	2,261	2,20
Median $RMSE_o$(rad)	2,141	2,950	2,10
Varianz $RMSE_o$(rad)	0,462	1,20	0,48
Standardabweichung $RMSE_o$(rad)	0,68	1,10	0,47

4.3.4. Ergebnisse aller Kategorien

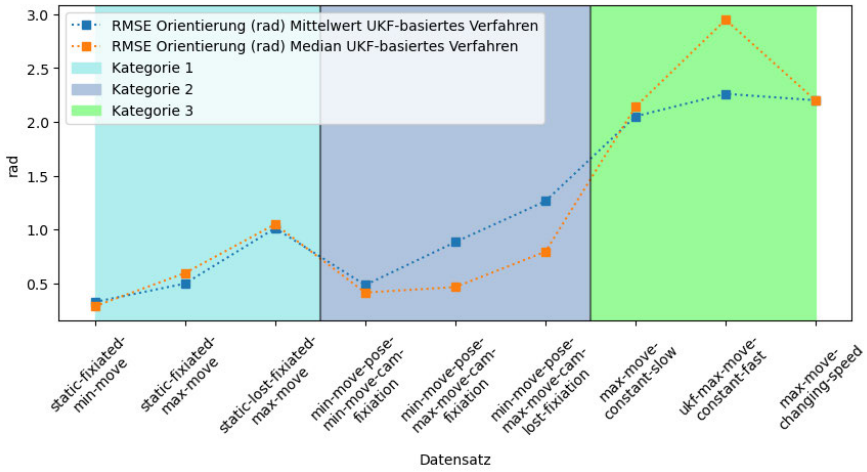
Die Ergebnisse aus den vorangegangenen Abschnitten werden in diesem Absatz kurz aufeinander bezogen. Die einzelnen Median und Mittelwerte des $RMSE_p$ und des $RMSE_o$ für das UKF-basierte Verfahren sind in den beiden Abbildungen unter 4.6 grafisch dargestellt. Die farblichen Hintergründe heben hier die drei in Tabelle 3.5 aufgeführten Kategorien hervor. Hier ist für den $RMSE_p$ in Abbildung 4.6a mit zunehmender Bewegung des zu trackenden Objektes ein Aufwärtstrend zu erkennen ist. Dabei steigt er für Datensatz "static-lost-fixiated-min-mov" stark an. Die beiden ersten Datensätze der zweiten Kategorie "min-move-pose-min-move-cam-fixiation" und "min-move-pose-max-move-cam-fixiation" weisen im Mittel und Median einen deutlich niedrigeren $RMSE_p$ auf. Hier wird der Aufwärtstrend unterbrochen.

Für Kategorie 1 und 2 wird deutlich, dass der $RMSE_p$ für den Datensatz, bei dem das Objekt aus dem Sichtfeld gerät, am höchsten ist. Für Kategorie 3 wird diese Unterscheidung nicht explizit gemacht. Allerdings sind hier die beiden letzten Datensätze "max-move-constant-fast" und "max-move-changing-speed" 93,918% und 95,744% der Zeit im Sichtfeld. Der Erste ist durchgängig sichtbar. Für die Datensätze, die in Kategorie 3 außer Sicht geraten, sind das Mittel und der Median des $RMSE_p$ nicht am höchsten. Die entsprechenden Datensätze aus Kategorie 1 und 2 sind allerdings für eine längere Zeit nicht sichtbar, als die Datensätze in Kategorie 3. So ist "statisch-lost-fixiated-max-move" nur 81,576% der Zeit sichtbar und "min-move-pose-max-move-cam-lost-fixiation" nur 80,164% der Zeit sichtbar.

Die Form der Kurven ist für den $RMSE_p$ und den $RMSE_o$ ungefähr gleich. Allerdings ist der Abfall in Kategorie 2 in Abbildung 4.6b etwas weniger stark, auch gehen hier die Mittel- und Medianwerte in Kategorie 2 stärker auseinander. Generell sind die Anstiege für den $RMSE_o$ weniger stark.



(a) $RMSE_p$ Mittelwert und Median bei verschiedenen Bewegungen



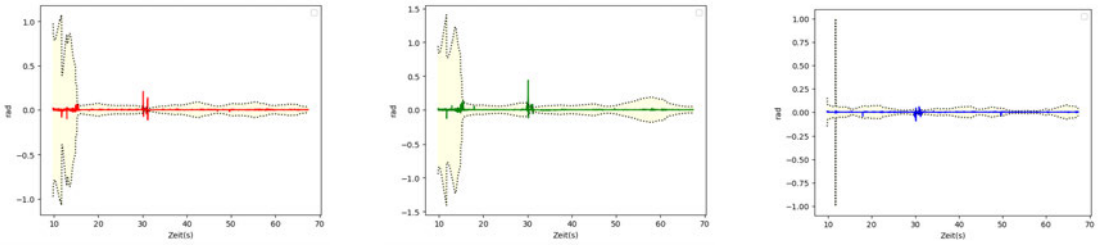
(b) $RMSE_o$ Mittelwert und Median bei verschiedenen Bewegungen

Abbildung 4.6.: Mittelwert und Median des $RMSE_p$ und des $RMSE_o$ bei verschiedenen Bewegungstypen der drei Datenkategorien aus Tabelle 3.5

4.3.5. Einhaltung der vom Filter angegebenen Abweichung

Für die Evaluation des Tracking-Verfahrens ist es erforderlich, zu ermitteln, ob die Varianz des UKF, die auf der Hauptdiagonalen der Matrix $\mathbf{P}$ gespeichert ist, sinnvoll ist. Dafür muss das Residuum zwischen der Filterschätzung und den Ground Truth-Daten zu 68% in der vom Filter angenommenen Abweichung liegen. Ist das der Fall, kann laut Labbe (2022a) davon ausgegangen werden, dass die Informationen, die ich dem Filter über das zu trackende System gegeben habe, korrekt sind.

Aufgrund der Beschaffenheit des UKFs lässt sich hierbei lediglich die Abweichung der Position sinnvoll betrachtet. Die Orientierung des Filters ist außerhalb des Zustands gespeichert. Im Zustand selbst ist nur die inkrementelle Orientierung angegeben. Auf diese bezieht sich aber die Varianz des Filters. Abbildung 4.7 zeigt das Verhältnis zwischen der Abweichung der inkrementellen Orientierung und der vom Filter angegebenen Standardabweichung. Der Fehler der inkrementellen Orientierung liegt dabei für einen Datensatz, bei dem das Objekt statisch ist und die Kamera maximal bewegt wurde, nahezu immer in den Grenzen des Filters. Da diese aber nach jedem Zeitschritt zurück auf null gesetzt wurde, ist das nicht aussagekräftig. Die Abbildung soll belegen, dass der UKF sich bei der Schätzung der inkrementellen Orientierung wie erwartet verhält. Da die Orientierung aber eigentlich außerhalb des Zustandes gespeichert wird, werden hier keine weiteren Beispiele angeführt.



(a) Abweichung und Standardabweichung des UKFs für ϕ

(b) Abweichung und Standardabweichung des UKFs für θ

(c) Abweichung und Standardabweichung des UKFs für ψ

Abbildung 4.7.: Darstellung der vom UKF angegebenen Abweichungen der Schätzung der inkrementellen Orientierung in Bezug auf die tatsächliche Abweichung bei minimaler Kamerabewegung und statischem Objekt

In den Abbildungen unter 4.9 sind die Fehler in x-Richtung in Bezug zu der vom Filter angegebenen Standardabweichungen bei verschiedenen Bewegungsarten abgebildet, die in Tabelle 3.5 definiert wurden. Der dort zu sehende zeitliche Verlauf und die Einhaltung des Fehlers der vom Filter angegebenen Grenzen sind ähnlich für die y- und z-Achse. Aus Gründen der Übersicht habe ich deswegen nur die Fehler bei der Schätzung der x-Position angegeben.

Der Abbildung 4.9a zugrundeliegende Datensatz bezieht sich auf Bewegungsdaten, bei denen sich die Kamera nur minimal und das zu trackende Objekt überhaupt nicht bewegt. Die Abbildung zeigt, dass der Fehler der Filterschätzung überwiegend innerhalb der vom Filter angegebenen Standardabweichung liegt. In Abbildung 4.9b sind der Fehler der Positionsschätzung und die Standardabweichung des Filters für einen Datensatz abgebildet, bei dem sich

das Objekt nur minimal und die Kamera ausladend um das zu trackende Objekt bewegt. Das Objekt ist hier ebenfalls die ganze Zeit über im Blickfeld. Auch hier liegen die Fehler in den Grenzen des Filters. Allerdings ist der Fehler, wie in der Abbildung zu sehen, größer als bei dem in Abbildung 4.9a dargestellten Datensatz, bei dem sich die Pose überhaupt nicht bewegt. Das gilt auch für die nicht abgebildete y- und z-Richtung. Das ist allerdings erwartbar und deckt sich mit den Erkenntnissen aus Absatz 4.3

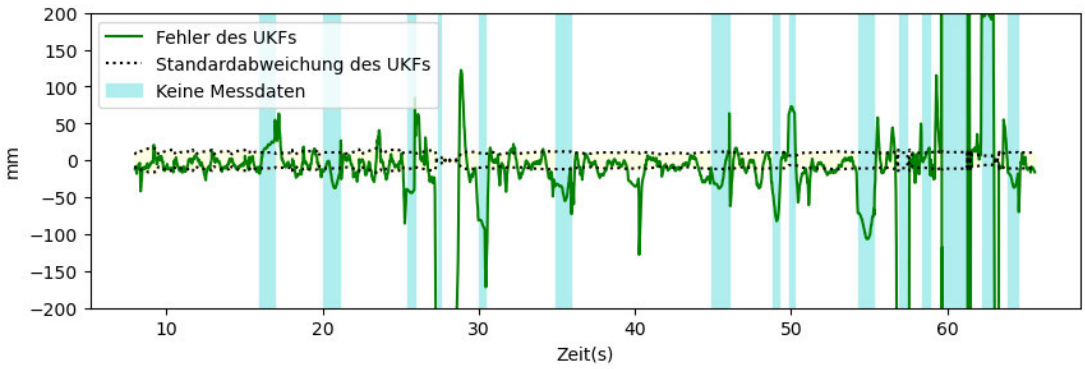
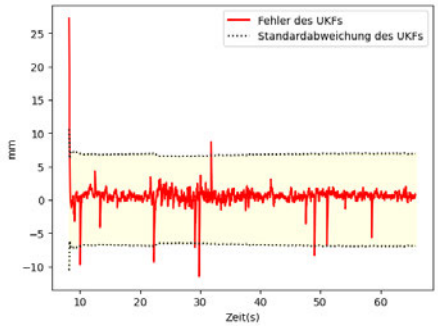


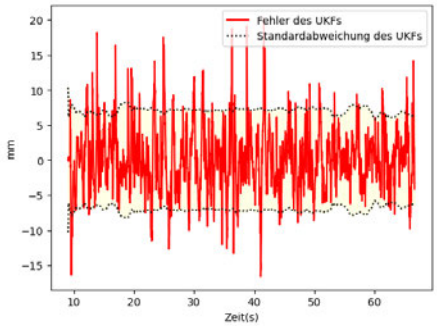
Abbildung 4.8.: Abbildung des Fehlers der Position in y-Richtung in Bezug auf die Standardabweichung des UKF, die markierten Abschnitten stellen die Intervalle dar, für die keine Messdaten vorlagen

Neben diesen idealen Beispielen finden sich unter den Ergebnissen allerdings auch Beispiele, bei denen die Abweichung der Schätzung nicht in den Grenzen des Filters liegt. Für Datensätze, bei denen das Objekt sehr ausladend bewegt wird, bleibt der Fehler der Positionsschätzung nicht in den Grenzen des Filters. Abbildung 4.9c zeigt den Fehler der Positionsschätzung in x-Richtung für einen Durchlauf, bei dem das Objekt mit einer schnellen Geschwindigkeit ausladend bewegt wurde. Hier ist der Fehler durchgängig größer als die angegebene Abweichung, so auch für die y- und z-Richtung. Für eine langsame, aber dennoch ausladende Bewegung des Objektes liegt der Fehler nur zeitweise innerhalb der Grenzen des Filters. Bei dem in Abbildung 4.9d gezeigten Durchlauf, befand sich das Objekt durchgängig im Tracking-Bereich der Stereokamera.

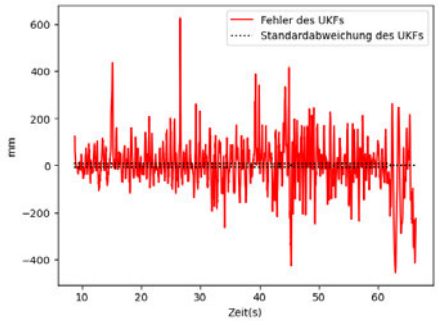
In Abbildung 4.8 ist der Fehler der Positionsschätzung in y-Richtung zu sehen. Hier sieht man klar, dass dieser grundsätzlich in den Grenzen des Filters liegt, diese aber kurzzeitig verlässt. Die türkis markierten Bereiche heben die Zeitabschnitte hervor, in denen das Objekt nicht im Sichtfeld der Kamera lag. Insbesondere zum Ende des Durchlaufs wird der Fehler sehr groß und somit sehr viel größer als die vom Filter angegebene Abweichung.



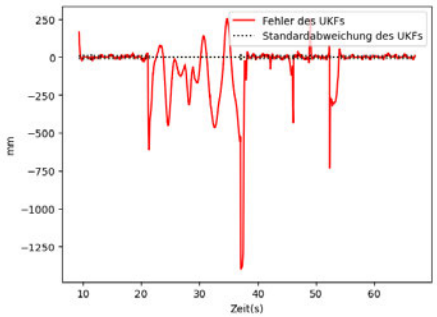
(a) Abweichung und Standardabweichung des UKFs für minimal bewegte Kamera und statisches Objekt



(b) Abweichung und Standardabweichung des UKFs für maximal bewegte Kamera und minimal bewegtes Objekt



(c) Abweichung und Standardabweichung des UKFs für ausladend und schnell bewegtes Objekt



(d) Abweichung und Standardabweichung des UKFs für ausladend und langsam bewegtes Objekt

Abbildung 4.9.: Darstellung der vom UKF angegebenen Abweichungen der Positionsschätzung in Bezug auf die tatsächliche Abweichung in x-Richtung

5. Diskussion

In diesem Kapitel werden die zuvor ermittelten Ergebnisse diskutiert und hinsichtlich ihrer Bedeutung interpretiert. Hierbei beantworte ich in Absatz 5.1 zunächst die Frage, ob das UKF-basierte Tracking-System funktioniert und ob es eine ausreichende Genauigkeit aufweist. Dabei ziehe ich zudem den Vergleich zu dem deterministischen Verfahren, um zu prüfen, ob das UKF-basierte Verfahren eine Verbesserung gegenüber dem deterministischen Verfahren darstellt. Weiterhin werden in Absatz 5.2 auch die Grenzen des UKF-basierten Verfahrens besprochen. Unter welchen Konditionen funktioniert es gut? Wieso funktioniert es in bestimmten Fällen nicht gut?

5.1. Die Qualität des UKF-basierten Tracking-Verfahrens

Die Ergebnisse auf Basis der künstlichen und realen Bewegungsdaten decken sich in Bezug auf den Vergleich des deterministischen und des UKF-basierten Tracking-Verfahrens. Bei beiden Datensätzen ist zu sehen, dass es hinsichtlich der Positionsschätzung keinen signifikanten Unterschied zwischen dem UKF-basierten und deterministischen Verfahren gibt. Das bedeutet, dass hier keinem der beiden Verfahren der Vorzug zu gewähren ist. Allerdings schneidet das deterministische Verfahren hinsichtlich der Orientierungsschätzung signifikant besser ab. Somit ließe sich argumentieren, dass es zu bevorzugen ist. Die Ergebnisse der künstlichen Daten sind für die abschließende Bewertung des Verfahrens nicht interessant. So sind diese Daten nicht zwingend realistisch und belegen vielmehr, dass das Verfahren grundsätzlich funktioniert. Sie spiegeln die Ergebnisse der quantitativen und qualitativen Evaluation wider.

Generell lässt sich anhand des Medians und des Mittelwerts für $RMSE_p$ und $RMSE_o$ sagen, dass das Verfahren für einen Einsatz in VR-Anwendung noch nicht genau genug ist. So ist eine Abweichung von ungefähr 45° im Median sehr gravierend, wenn es um das Tracking von beispielsweise Werkzeugen geht. Zum Vergleich liegt die bei Gsaxner u. a. (2021) angegebene Abweichung bei rund 1,11°. Wobei hier erwähnt werden sollte, dass bei Gsaxner u. a. (2021) für die Evaluation des Trackings eines bewegten Objektes lediglich fünf Durchläufe miteinbezogen wurden. Zudem ist hier die Distanz zwischen der Tracking-Kamera und dem Objekt limitiert.

Weiterhin werden keine Durchläufe miteinbezogen, bei denen das Objekt aus der Sicht der Tracking-Kamera verschwindet. Insofern macht hier ein direkter Vergleich wenig Sinn, da das Einsatzgebiet dort viel stärker begrenzt ist. Nicht zuletzt auch durch die eher minimalen Bewegungen eines chirurgischen Instruments, das nicht ausladend oder sehr schnell bewegt wird.

Auch der Median des $RMSE_p$ von 24,51 cm deutet auf eine nicht ausreichende Zuverlässigkeit hin. Allerdings muss sowohl bei der Positions- als auch bei der Orientierungsschätzung hinsichtlich der in Absatz 4.3 aufgeschlüsselten Art der Bewegung differenziert werden. So ist das Tracking meines Erachtens für statische und minimal bewegte Objekte ausreichend akkurat. Denkbar wäre hier das Tracking von statischen Objekten in der Umgebung, die nicht von Nutzer*innen einer VR-Anwendung bewegt werden. Für stark bewegte Objekte ist die Qualität des Trackings, wie in Tabelle 4.3.3 zu sehen, definitiv zu gering. Die Abweichung beträgt hier im Mittel ungefähr 29 cm bis 70 cm. Spannend ist jedoch, dass eine Varianz in der Geschwindigkeit des bewegten Objektes zu einer drastischen Verbesserung um circa 20 cm bis 50 cm führt.

Weiterhin lässt sich feststellen, dass für die Datensätze, bei denen der $RMSE_p$ eher hoch ist, auch die Standardabweichung eher hoch ist. Das spricht dafür, dass hinsichtlich der Positionsschätzung eine schlechte Qualität überwiegend durch einzelne Ausreißer begründet ist. Exemplarisch ist das in Abbildung 5.1 dargestellt. Daraus lässt sich ableiten, dass das Verfahren grundsätzlich funktioniert und zeitweise ausreichend gute Resultate liefert.

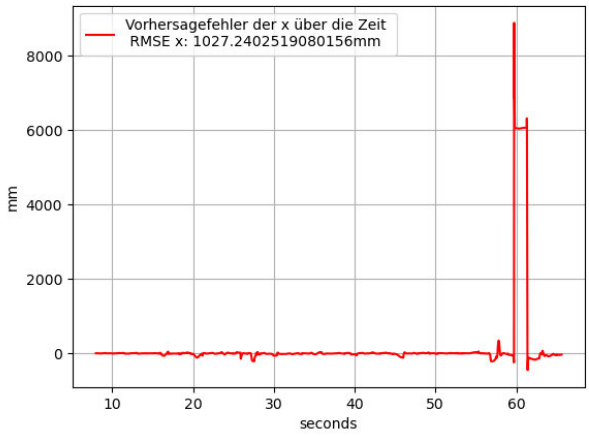


Abbildung 5.1.: Darstellung einer spontanen, starken Abweichung bei einem Sichtverlust

5.2. Grenzen des UKF-basierten Tracking-Verfahrens

Die in Absatz 5.1 aufgeführten Qualitätsunterschiede des UKF-basierten Trackings zwischen den Arten der Bewegung geben einen ersten Eindruck für die Grenzen des Trackings. In diesem Absatz werden diese Grenzen ausführlicher untersucht und besprochen.

5.2.1. UKF-basiertes Verfahren bei Blickverlust des Objektes

So lässt sich sagen, dass die Qualität des Trackings unter dem Austritt eines Objektes aus dem Sichtfeld der Stereokamera stark leidet. In Abbildung 5.2 ist die dreidimensionale Darstellung eines Durchlaufs des UKF-basierten Trackings zu sehen, bei dem das Objekt nur 80,14% der Zeit im Blickfeld zu der Kamera war. Hier sind immer wieder Ausreißer der Positionsschätzung zu erkennen.

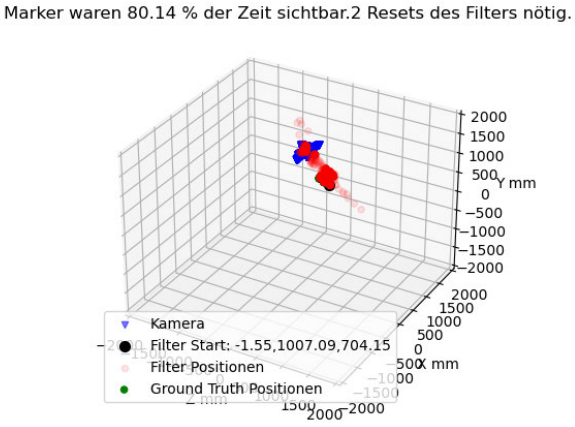


Abbildung 5.2.: Darstellung der Tracking-Ergebnisse für ein minimal bewegtes Objekt, das 80,14% der Zeit im Sichtfeld der Tracking-Kamera war

Auch die in Absatz 4.3.5 gezeigte Abbildung 4.8 zeigt, dass das Ausfallen der Messdaten negativen Einfluss auf die Positionsschätzung hat. Zwar wurden bei der Berechnung des $RMSE_p$ und des $RMSE_o$ nur die Zeitschritte miteinbezogen, für die Messdaten vorhanden waren, dennoch braucht es einige Zeitschritte, bis die Schätzung des UKF-basierten Trackings wieder bei ihrer eigentlichen Qualität angekommen ist. Bei Abbildung 5.3 handelt es sich um einen kurzen Ausschnitt aus Abbildung 5.2. Hier ist klar zu sehen, wie der Fehler während des Ausfalls der Messdaten ansteigt. In diesem Intervall führt das UKF-basierte Tracking-Verfahren

keine Vorhersage mehr durch. Für die Berechnung des $RMSE_p$ spielen die Zeitschritte in dem türkis eingefärbten Intervall keine Rolle. Danach braucht der Filter ungefähr 0,3 Sekunden, um wieder innerhalb der angegebenen Standardabweichung zu liegen.

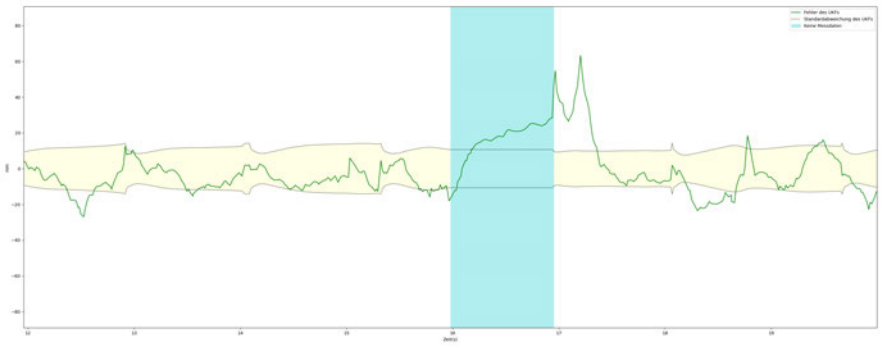


Abbildung 5.3.: Fehler in den Grenzen des UKFs bei minimal bewegtem Objekt, das 80,14% der Zeit im Sichtfeld der Tracking-Kamera war

Es lässt sich also festhalten, dass der Austritt aus dem Sichtfeld negativ auf die Qualität der Schätzung auswirkt. Es ist anzunehmen, dass dies auch für die Orientierungsschätzung gilt, alllerdings kann diese aufgrund der Beschaffenheit des Filters nicht so detailliert untersucht werden.

5.2.2. UKF-basiertes Verfahren bei unterschiedlichen Bewegungen

Weiterhin geben die Ergebnisse Aufschluss darüber, für welche Arten von Bewegung das UKF-basierte Verfahren gut funktioniert und für welche nicht. So ist die generelle Genauigkeit des UKF-basierten Verfahrens für ausladende Bewegungen, wie bereits in Absatz 4.3.3 beschrieben, sehr gering. Die Abbildung 4.9c zeigt zudem, dass der Fehler bei schnellen und ausladenden Bewegungen nicht in den Grenzen des Filters liegt. Für diese Art Bewegung ist das Verfahren somit nicht geeignet. Anders sieht das bei minimalen Bewegungen aus. Hier sind die $RMSE_p$ und $RMSE_o$ Werte erkennbar niedriger. Auch die Abweichungen der Positionsschätzung liegen größtenteils in den Grenzen des Filters. Für diese Art Bewegung ist das Verfahren somit grundsätzlich geeignet.

Für langsame und ausladende Bewegungen hingegen liegt der Fehler zeitweise nicht in den Grenzen des Filters. Die in Abbildung 4.9d gezeigten Abweichungen lassen sich allerdings nicht auf das Wegfallen von Messdaten zurückführen. Hier steigt lediglich der Fehler zeitweise stark an, ohne dass der UKF die Standardabweichung daran anpasst.

Die Unterschiede in der Qualität sind durchaus nachvollziehbar. So widmet sich "Fundamentals of Object-Tracking"(Challa u. a., 2011, S. 62) dem Tracking manövrierender Objekte. Hier werden diverse Ansätze vorgeschlagen, die entweder verschiedene Prozessmodelle verwenden oder das Prozessrauschen der tatsächlichen Bewegung eines Objektes anpassen. Ich habe dem UKF-basierten Verfahren in Form von dem Prozessrauschen und dem Messrauschen Informationen über ein System gegeben. Zwar habe ich das Messrauschen an die Messdatenvariante angepasst, das Prozessrauschen bleibt aber durchgängig gleich. Anschließend wende ich das Verfahren auf verschiedene Arten von Bewegungen an. Hierbei verschlechtern sich die Ergebnisse je stärker das zu trackende Objekt manövriert. Das ist eine logische Konsequenz, da hier die Differenz zwischen dem beschriebenen und dem tatsächlichen System größer wird. Somit gibt es nachvollziehbare Gründe für ein ungenaues Tracking bei gewissen Arten von Bewegungen. Damit sind die Grenzen des Filters klar definiert. Je stärker das System durch manövrierende Bewegungen geprägt ist, desto geringer ist die Genauigkeit des Trackings.

5.3. P-Matrix Fehler

Der UKF ist insofern nicht stabil, als die Matrix $\mathbf{P}$ immer wieder nicht PSD wird. Die Ursache des Problems kann an vielen verschiedenen Stellen liegen. Meine Vermutung besteht jedoch darin, dass der Fehler aufgrund der hohen Nichtlinearität des eigentlichen Systems auftritt. So versuche ich mit dem UKF-basierte Verfahren viele verschiedene Systeme mit einem sehr simplen Prozessmodell zu tracken. Dabei ist die Differenz zwischen dem, was der UKF über das System weiß und dem eigentlichen System stellenweise hoch. Wie in Absatz 3.10.11 bereits erwähnt, handelt es sich bei dem Fehler um einen gängigen und üblichen Fehler, der auftritt, wenn mit dem UKF nicht lineare Systeme linearisiert werden. Interessanterweise ist die in Abschnitt 4.2.2 gezeigte Korrelation zwischen den Neustarts des Filters aufgrund des Fehlers und der Genauigkeit der Positions- und Orientierungsschätzung negativ.

Der Fehler tritt somit seltener auf, wenn die Genauigkeit des Filters geringer ist. Für nicht stark manövrierende Bewegungen ist der Filter jedoch genauer. Der Korrelation nach zu Urteilen tritt für diese Datensätze der Fehler somit häufiger auf. Ich habe das Prozessrauschen $\mathbf{Q}$ des

Filters testweise auf sehr kleine Werte geändert. Für diese Durchläufe hat die Anzahl der Fehler abgenommen. Ich begründe das damit, dass dann das Prozessrauschen für diese Durchläufe näher an der Wirklichkeit ist. Es scheint also so zu sein, dass der Fehler dann auftritt, wenn das Prozessrauschen weit von dem eigentlichen System entfernt ist. Das ist durchaus erwartbar, da das Manövrieren des Objektes lediglich über $\mathbf{Q}$ abgebildet wird. Die Diskrepanz zwischen dem modellierten System und dem tatsächlichen System ist wie bereits beschrieben auch hinsichtlich der Genauigkeit problematisch.

Eine weitere Möglichkeit für den Fehler könnte darin liegen, dass die Orientierung nicht Bestandteil des Zustandes ist. Innerhalb der Messfunktion wird sie allerdings verwendet und wirkt sich so indirekt auch auf die Berechnung von $\mathbf{P}$ aus. Um diese Fehlerquelle auszuschließen, habe ich eine Variante des Filters implementiert, bei der die absolute Orientierung des zu trackenden Objektes als Quaternion im Filterzustand enthalten ist. Doch auch hier ist der Fehler aufgetreten.

6. Fazit und Ausblick

In diesem Kapitel sind die gewonnenen Erkenntnisse dieser Arbeit abschließend zusammengefasst. Dabei werde ich im ersten Abschnitt 6.1 die Erkenntnisse der Arbeit zusammenfassen. Mithilfe dieser gewonnenen Erkenntnisse gebe ich in Abschnitt 6.2 Anregungen für die weitere Forschung und die Entwicklung des Verfahrens.

6.1. Fazit

Ein voll funktionierendes Inside-Out-Tracking-System für VR-Brillen ausschließlich anhand visueller Messungen mittels günstiger und verfügbarer Hardware zu entwickeln, ist nicht abschließend gelungen. Das entstandene Tracking-System funktioniert grundsätzlich und ist in der Lage unter gewissen Einschränkungen ein annehmbares Tracking zu erzeugen, allerdings haben die Untersuchungen gezeigt, dass die verwendeten Methoden nicht ausreichen, um ein in jedem Fall einsatzfähiges System zu entwickeln. Allerdings konnte ich in dieser Arbeit einige wertvolle Erkenntnisse gewinnen, die die weitere Entwicklung eines solchen Tracking-Systems voranbringen könnten.

Das entwickelte Verfahren ist in der Lage, die Position eines statischen oder nur minimal bewegten Objektes mit einer Genauigkeit von 1 bis 2 cm vorherzusagen. Die Trackingqualität ist somit nicht ausreichend genug, um damit in Echtzeit Gegenstände zu tracken, mit denen Nutzer*innen interagieren. Innerhalb des Trackings kommt es immer wieder zu Ausreißern, die die Trackingqualität verschlechtern. Zwischen diesen Ausreißern ist das Tracking häufig genau genug.

Zwischen dem deterministischen und UKF-basierten Verfahren gibt es hinsichtlich der Positionsschätzung keinen signifikanten Unterschied in der Qualität. Für die Orientierungsschätzung allerdings gibt es diesen schon. Hier schneidet das deterministische Verfahren immer besser ab. Das Tracking-System ist nicht ausreichend genau für ausladende Bewegungen und kann somit für diese in der jetzigen Form nicht in Betracht gezogen werden. Gerät das zu trackende

Objekt aus dem Sichtfeld der Kamera, verschlechtert sich das Tracking, da keine Messdaten mehr vorliegen.

6.2. Empfehlungen für zukünftige Forschung

Aus den zuvor gewonnen Erkenntnissen lassen sich einige Empfehlungen für zukünftige Forschung an dem UKF-basierten Tracking-Verfahren ableiten. So könnte getestet werden, wie sich die Gesamtqualität des Filters verhält, wenn der Filter nur noch die Positions- und das deterministische Verfahren die Orientierungsschätzung vornimmt.

Das dem UKF zugrundeliegende System deckt die verschiedenen Bewegungsarten, die in einer VR-Anwendung, in der Nutzer*innen mit den zu trackenden Objekten interagieren, möglich sind, nicht ab. Dementsprechend sollte zukünftig versucht werden, entweder den von Challa u. a. (2011) gezeigten Ansatz von der Verwendung mehrerer Prozessmodelle umzusetzen. Alternativ kann auch versucht werden, das Prozessrauschen an die aktuelle Bewegung anzupassen. Hierfür ließe sich die Varianz des Filters verwenden. Überschreitet diese einen gewissen Wert, verändert sich das Prozessrauschen, um die höhere Diskrepanz zwischen Prozessmodell und tatsächlichem System abzubilden.

Weiterhin ist es mitunter sinnvoll zu testen, ob ein Prozessmodell höherer Ordnung, das die Beschleunigung in x-, y- und z-Richtung sowie die Winkelbeschleunigung der inkrementellen Orientierung miteinbezieht, bessere Resultate hervorbringt.

Dazu passend ist es auch denkbar, dass weitere Sensoren das Ergebnis stark verbessern. Hier könnten zum einen mehr Kameras verwendet werden. Diese könnten die Schätzung in z-Richtung verbessern. Weiterhin ließe sich so ein höheres Sichtfeld abdecken, wodurch das Objekt seltener aus dem Sichtfeld der Tracking-Kameras geraten würde. Ein Beschleunigungssensor und ein Gyroskop, die an dem zu trackenden Objekt angebracht werden, könnten zudem Informationen über die Beschleunigung, Ausrichtung und Winkelbeschleunigung liefern. Das hätte das Potenzial, das Ergebnis aufgrund von mehr Daten zu verbessern. Weiterhin wären so auch dann noch Daten verfügbar, wenn sich das Objekt nicht im Sichtfeld der Kameras befinden.

Auch habe ich in dieser Arbeit die Vorbereitung der Messdaten stark simplifiziert, da diese simuliert sind. In der Zukunft ist es meiner Meinung nach notwendig, hier einen Mechanismus einzubauen, der die Marker im Bild der Tracking-Kameras den Markern in der Pose zuordnet.

Hierfür ließe sich der entsprechende Algorithmus aus der Arbeit von Steinicke u. a. (2007) verwenden, der auf der Markerzuordnung anhand der Zuordnung der Kantenlängen zwischen den Markern beruht. Weiterhin könnten die Markerpositionen im Bild zwischen den einzelnen Zeitschritten fortgeführt werden. Zuletzt sollte das Verfahren, sobald es unter den in dieser Arbeit beschriebenen Bedingungen eine ausreichende Qualität erreicht, nicht nur mit realen Bewegungsdaten, sondern auch mit realen Messdaten getestet werden.

Abschließend sollte auch die Schätzung der Orientierung genauer betrachtet werden. Da die globale Orientierung des Objektes im Zustand nicht enthalten ist, habe ich die Orientierungsschätzung nicht so tiefgehend untersuchen können, wie die Positionsschätzung. Ein UKF, bei dem die Orientierung in Form eines Quaternions im Zustand enthalten ist, wäre hier denkbar.

A. Anhang

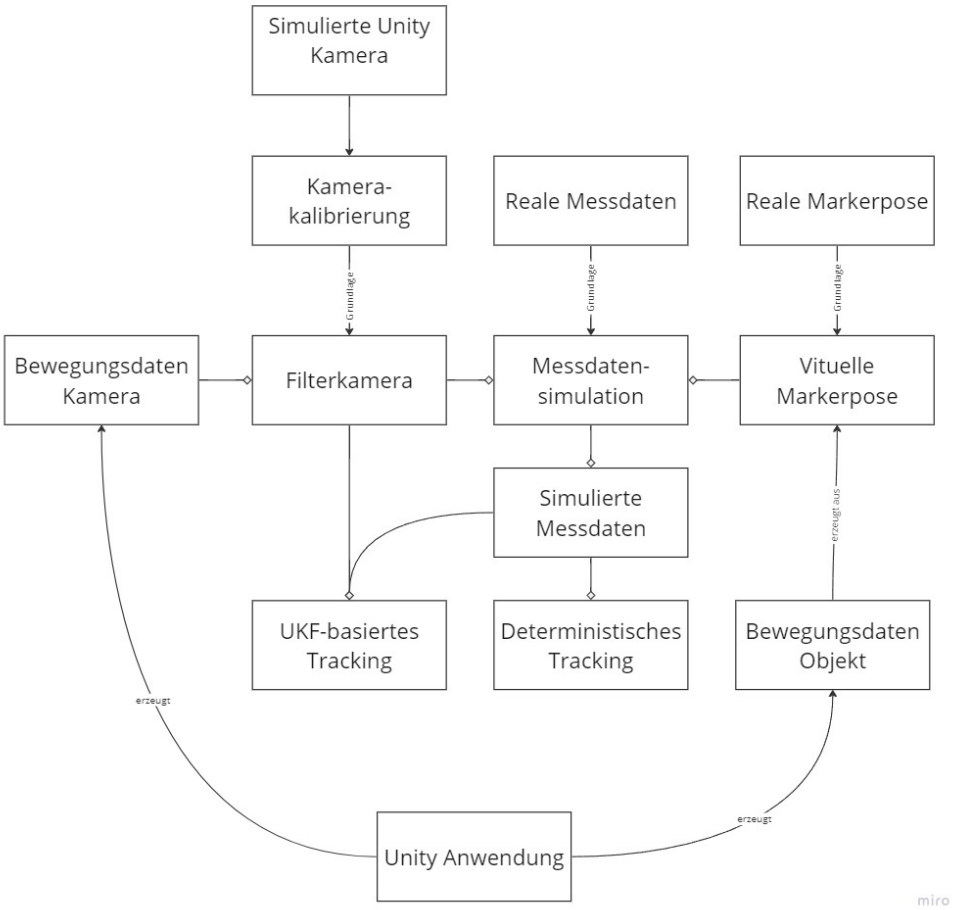


Abbildung A.1.: Übersicht der Bestandteile dieser Arbeit

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	Sichtbarkeit	Neustarts	Variante	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	Typ
max-move-changing-speed	95.78	0	0	701.98	0.21	determ
max-move-changing-speed	95.72	0	1	3006.10	0.21	determ
max-move-changing-speed	95.75	0	2	1105.90	0.23	determ
max-move-changing-speed	95.72	0	3	263.92	0.27	determ
max-move-changing-speed	95.75	0	4	211.31	0.29	determ
max-move-constant-fast	93.96	0	0	1787.03	0.33	determ
max-move-constant-fast	93.84	0	1	3230.73	0.36	determ
max-move-constant-fast	93.87	0	2	159.12	0.39	determ
max-move-constant-fast	93.99	0	3	1699.95	0.48	determ
max-move-constant-fast	93.93	0	4	211.80	0.58	determ
max-move-constant-slow	100.0	0	0	450.19	0.21	determ
max-move-constant-slow	100.0	0	1	607.02	0.21	determ
max-move-constant-slow	100.0	0	2	862.39	0.23	determ
max-move-constant-slow	100.0	0	3	3730.74	0.23	determ
max-move-constant-slow	100.0	0	4	351.62	0.27	determ
min-move-pose-max-move-cam-						

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	Sichtbarkeit	Neustarts	Variante	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	Typ
fixiation min-move-pose-max-move-cam-	100.0	0	0	5.19	0.16	determ
fixiation min-move-pose-max-move-cam-	100.0	0	1	5.64	0.16	determ
fixiation min-move-pose-max-move-cam-	100.0	0	2	7.72	0.16	determ
fixiation min-move-pose-max-move-cam-	100.0	0	3	21.83	0.16	determ
fixiation min-move-pose-max-move-cam-	100.0	0	4	15.25	0.18	determ
lost-fixiation min-move-pose-max-move-cam-	80.14	0	0	1361.87	0.23	determ
lost-fixiation min-move-pose-max-move-cam-	80.14	0	1	1094.98	0.23	determ
lost-fixiation min-move-pose-max-move-cam-	80.23	0	2	148.88	0.24	determ
lost-fixiation min-move-pose-max-move-cam-	80.17	0	3	2110.11	0.26	determ
lost-fixiation min-move-pose-max-move-cam-	80.14	0	4	132.26	0.27	determ
fixiation min-move-pose-min-move-cam-	100.0	0	0	6.94	0.16	determ

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	Sichtbarkeit	Neustarts	Variante	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	Typ
min-move-cam-fixiation	100.0	0	1	6.86	0.15	determ
min-move-pose-min-move-cam-fixiation	100.0	0	2	4.99	0.16	determ
min-move-pose-min-move-cam-fixiation	100.0	0	3	5.87	0.16	determ
min-move-pose-min-move-cam-fixiation	100.0	0	4	11.98	0.16	determ
static-fixiated-max-move	100.0	0	0	91.25	0.32	determ
static-fixiated-max-move	100.0	0	1	100.13	0.31	determ
static-fixiated-max-move	100.0	0	2	94.60	0.31	determ
static-fixiated-max-move	100.0	0	3	84.42	0.32	determ
static-fixiated-max-move	100.0	0	4	113.12	0.33	determ
static-fixiated-min-move	100.0	0	0	9.32	0.31	determ
static-fixiated-min-move	100.0	0	1	9.25	0.30	determ
static-fixiated-min-move	100.0	0	2	9.21	0.30	determ
static-fixiated-min-move	100.0	0	3	9.83	0.30	determ
static-fixiated-min-move	100.0	0	4	9.23	0.31	determ
static-lost-						

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	Sichtbarkeit	Neustarts	Variante	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	Typ
fixiation-max-move	81.58	0	0	217.46	0.38	determ
static-lost-fixiation-max-move	81.56	0	1	752.38	0.38	determ
static-lost-fixiation-max-move	81.58	0	2	240.00	0.37	determ
static-lost-fixiation-max-move	81.58	0	3	521.96	0.43	determ
static-lost-fixiation-max-move	81.58	0	4	821.38	0.43	determ
max-move-changing-speed	95.78	4	0	291.97	1.97	ukf
max-move-changing-speed	95.72	3	1	245.09	1.22	ukf
max-move-changing-speed	95.75	1	2	390.65	2.52	ukf
max-move-changing-speed	95.72	6	3	240.87	3.09	ukf
max-move-changing-speed	95.75	10	4	327.82	2.20	ukf
max-move-constant-fast	93.96	2	0	708.62	0.73	ukf
max-move-constant-fast	93.84	2	1	1817.68	1.47	ukf
max-move-constant-fast	93.87	1	2	235.77	3.07	ukf
max-move-constant-fast	93.99	6	3	1039.07	2.95	ukf

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	Sichtbarkeit	Neustarts	Variante	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	Typ
max-move-constant-fast	93.93	9	4	462.38	3.09	ukf
max-move-constant-slow	100.0	2	0	452.63	2.62	ukf
max-move-constant-slow	100.0	2	1	342.78	0.92	ukf
max-move-constant-slow	100.0	4	2	355.68	2.55	ukf
max-move-constant-slow	100.0	7	3	3744.24	2.02	ukf
max-move-constant-slow	100.0	18	4	551.80	2.14	ukf
min-move-pose-max-move-cam-fixiation	100.0	0	0	9.78	0.46	ukf
min-move-pose-max-move-cam-fixiation	100.0	0	1	7.20	0.16	ukf
min-move-pose-max-move-cam-fixiation	100.0	0	2	15.97	0.66	ukf
min-move-pose-max-move-cam-fixiation	100.0	28	3	20.85	0.04	ukf
min-move-pose-max-move-cam-fixiation	100.0	0	4	49.50	3.08	ukf
min-move-pose-max-move-cam-lost-fixiation	80.14	2	0	1609.84	0.79	ukf
min-move-pose-max-move-cam-						

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	**Sichtbarkeit**	**Neustarts**	**Variante**	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	**Typ**
lost-fixiation min-move-pose-max-move-cam-	80.14	3	1	1136.43	0.31	ukf
lost-fixiation min-move-pose-max-move-cam-	80.23	1	2	277.32	2.20	ukf
lost-fixiation min-move-pose-max-move-cam-	80.17	17	3	961.03	0.72	ukf
lost-fixiation min-move-pose-min-move-cam-	80.14	10	4	201.27	2.29	ukf
fixiation min-move-pose-min-move-cam-	100.0	0	0	9.42	0.41	ukf
fixiation min-move-pose-min-move-cam-	100.0	0	1	9.83	0.72	ukf
fixiation min-move-pose-min-move-cam-	100.0	47	2	7.67	0.30	ukf
fixiation min-move-pose-min-move-cam-	100.0	49	3	16.23	0.41	ukf
fixiation min-move-pose-min-move-cam-	100.0	54	4	40.79	0.56	ukf
static-fixiated-max-move	100.0	1	0	45.06	0.13	ukf
static-fixiated-max-move	100.0	1	1	80.74	0.82	ukf
static-fixiated-max-move	100.0	10	2	83.93	0.23	ukf
static-fixiated-						

Tabelle A.1.: Ergebnisse der Durchläufe des UKF-basierten und deterministischen Verfahrens auf den für die qualitative und quantitative Evaluation erzeugten Messdatensätzen

Datensatz	Sichtbarkeit	Neustarts	Variante	$\mathbf{RMSE_p}$	$\mathbf{RMSE_o}$	Typ
max-move	100.0	15	3	98.39	0.59	ukf
static-fixiated-max-move	100.0	42	4	657.50	0.71	ukf
static-fixiated-min-move	100.0	0	0	5.44	0.18	ukf
static-fixiated-min-move	100.0	1	1	6.09	0.28	ukf
static-fixiated-min-move	100.0	52	2	7.86	0.24	ukf
static-fixiated-min-move	100.0	53	3	16.18	0.36	ukf
static-fixiated-min-move	100.0	54	4	25.41	0.55	ukf
static-lost-fixiation-max-move	81.58	1	0	582.86	1.50	ukf
static-lost-fixiation-max-move	81.56	4	1	517.11	0.29	ukf
static-lost-fixiation-max-move	81.58	12	2	460.79	1.05	ukf
static-lost-fixiation-max-move	81.58	18	3	628.41	1.32	ukf
static-lost-fixiation-max-move	81.58	29	4	857.95	0.90	ukf

Literaturverzeichnis

[OpenCV 2022] *Camera Calibration and 3D Reconstruction.* Dezember 2022. – URL `https://docs.opencv.org/4.7.0/d9/d0c/group__calib3d.html`. – Stand: 08.06.2023

[Benesty u. a. 2009] Benesty, Jacob ; Chen, Jingdong ; Huang, Yiteng ; Cohen, Israel: Pearson Correlation Coefficient. In: *Noise Reduction in Speech Processing.* Springer Berlin Heidelberg, 2009

[Brand u. a. 2020] Brand, Michael ; Wulff, Lukas A. ; Hamdani, Yogi ; Schüppstuhl, Thorsten: Accuracy of Marker Tracking on an Optical See-Through Head Mounted Display. In: *Annals of Scientific Society for Assembly, Handling and Industrial Robotics.* Springer Berlin Heidelberg, 2020, S. 21–31

[Carbone u. a. 2018] Carbone, Marina ; Condino, Sara ; Cutolo, Fabrizio ; Viglialoro, Rosanna M. ; Kaschke, Oliver ; Thomale, Ulrich W. ; Ferrari, Vincenzo: Proof of Concept: Wearable Augmented Reality Video See-Through Display for Neuro-Endoscopy. In: *Lecture Notes in Computer Science.* Springer International Publishing, 2018, S. 95–104

[Challa u. a. 2011] Challa, Subhash ; Morelande, Mark R. ; Mušicki, Darko ; Evans, Robin J.: *Fundamentals of Object Tracking.* Cambridge University Press, jul 2011

[Challis 2020] Challis, John H.: Quaternions as a solution to determining the angular kinematics of human movement. In: *BMC Biomedical Engineering* 2 (2020), mar, Nr. 1

[Faion u. a. 2016] Faion, Florian ; Zea, Antonio ; Noack, Benjamin ; Steinbring, Jannik ; Hanebeck, Uwe D.: Camera- and IMU-based pose tracking for augmented reality. In: *2016 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, IEEE, sep 2016

[Gao u. a. 2019] Gao, Yuan ; Lin, Li ; Chai, Gang ; Xie, Le: A feasibility study of a new method to enhance the augmented reality navigation effect in mandibular angle split osteotomy. In: *Journal of Cranio-Maxillofacial Surgery* 47 (2019), aug, Nr. 8, S. 1242–1248

[Gsaxner u. a. 2021] Gsaxner, Christina ; Li, Jianning ; Pepe, Antonio ; Schmalstieg, Dieter ; Egger, Jan: Inside-Out Instrument Tracking for Surgical Navigation in Augmented Reality. In: *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, ACM, dec 2021

[Hattori und Hirai 2020] Hattori, Keisuke ; Hirai, Tatsunori: Inside-out Tracking Controller for VR/AR HMD using Image Recognition with Smartphones. In: *ACM SIGGRAPH 2020 Posters*, ACM, aug 2020

[Jammalamadaka und SenGupta 2001] Jammalamadaka, S R. ; SenGupta, Ashis: *Topics in Circular Statistics*. WORLD SCIENTIFIC, apr 2001

[Kraft 2003] Kraft, E.: A quaternion-based unscented Kalman filter for orientation tracking. In: *Sixth International Conference of Information Fusion, 2003. Proceedings of the*, IEEE, 2003

[Kriechling u. a. 2020] Kriechling, Philipp ; Roner, Simon ; Liebmann, Florentin ; Casari, Fabio ; Fürnstahl, Philipp ; Wieser, Karl: Augmented reality for base plate component placement in reverse total shoulder arthroplasty: a feasibility study. In: *Archives of Orthopaedic and Trauma Surgery* 141 (2020), jul, Nr. 9, S. 1447–1453

[Labbe 2022a] Labbe, Roger ; Labbe, Roger (Hrsg.): *Kalman and Bayesian Filters in Python*. Roger Labbe, 2022. – URL `https://github.com/rlabbe/Kalman-and-Bayesian-Filters-in-Python`. – Stand: 25.07.2023

[Labbe 2022b] Labbe, Roger: Software Bibliothek "filterpy". (2022). – URL `https://github.com/rlabbe/filterpy/tree/master/filterpy`. – Stand: 14.06.2023

[Lang 2023] Lang, Ben: *HTC Announces Inside-out Tracker for VR Accessories amp; Body Tracking*. Mar 2023. – URL `https://www.roadtovr.com/htc-vive-tracker-standalone-inside-out-camera-tracking/`. – Stand: 19.07.2023

[Lee 2010] Lee, Alan: Circular data. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2 (2010), jul, Nr. 4, S. 477–486

[Leuze u. a. 2018] Leuze, Christoph ; Yang, Grant ; Hargreaves, Brian ; Daniel, Bruce ; McNab, Jennifer A.: Mixed-Reality Guidance for Brain Stimulation Treatment of Depression. In: *2018 IEEE International Symposium on Mixed and Augmented Reality Adjunct (ISMAR-Adjunct)*, IEEE, oct 2018

[Müller u. a. 2020] Müller, Fabio ; Roner, Simon ; Liebmann, Florentin ; Spirig, José M. ; Fürnstahl, Philipp ; Farshad, Mazda: Augmented reality navigation for spinal pedicle screw instrumentation using intraoperative 3D imaging. In: *The Spine Journal* 20 (2020), apr, Nr. 4, S. 621–628

[Nachar 2008] Nachar, Nadim: The Mann-Whitney U: A Test for Assessing Whether Two Independent Samples Come from the Same Distribution. In: *Tutorials in Quantitative Methods for Psychology* 4 (2008), mar, Nr. 1, S. 13–20

[Niehorster u. a. 2017] Niehorster, Diederick C. ; Li, Li ; Lappe, Markus: The Accuracy and Precision of Position and Orientation Tracking in the HTC Vive Virtual Reality System for Scientific Research. In: *i-Perception* 8 (2017), may, Nr. 3, S. 204166951770820

[OpenCV 2023] OpenCV: OpenCV Camera Calibration. (2023). – URL `https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html`. – Stand: 19.07.2023

[Qian u. a. 2018] Qian, Long ; Deguet, Anton ; Kazanzides, Peter: ARssist: augmented reality on a head-mounted display for the first assistant in robotic surgery. In: *Healthcare Technology Letters* 5 (2018), sep, Nr. 5, S. 194–200

[Qian u. a. 2019] Qian, Long ; Zhang, Xiran ; Deguet, Anton ; Kazanzides, Peter: ARAMIS: Augmented Reality Assistance for Minimally Invasive Surgery Using a Head-Mounted Display. In: *Lecture Notes in Computer Science*. Springer International Publishing, 2019, S. 74–82

[Semeniuta 2016] Semeniuta, Oleksandr: Analysis of Camera Calibration with Respect to Measurement Accuracy. In: *Procedia CIRP* 41 (2016), S. 765–770

[Southall u. a. 1998] Southall, B. ; Buxton, B. F. ; Marchant, J. A.: Controllability and Observability: Tools for Kalman Filter Design. In: *Procedings of the British Machine Vision Conference 1998*, British Machine Vision Association, 1998

[Steinicke u. a. 2007] Steinicke, Frank ; Jansen, Christian ; Hinrichs, Klaus ; Vahrenhold, Jan ; Schwald, Bernd: GENERATING OPTIMIZED MARKER-BASED RIGID BODIES FOR OPTICAL TRACKING SYSTEMS. In: *Proceedings of the Second International Conference on Computer Vision Theory and Applications - Volume 2: VISAPP,* INSTICC (Veranst.), SciTePress, 2007, S. 387–395. – ISBN 978-972-8865-74-0

[Umeyama 1991] Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 13 (1991), apr, Nr. 4, S. 376–380

[Van Der Merwe und Wan 2004] Van Der Merwe, Rudolph ; Wan, Eric A.: *Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space Models*, Dissertation, 2004. – AAI3129163

[Welch und Bishop 1997] Welch, Greg ; Bishop, Gary: SCAAT. In: *Proceedings of the 24th annual conference on Computer graphics and interactive techniques - SIGGRAPH '97*, ACM Press, 1997

[Wu u. a. 2017] Wu, Po-Chen ; Wang, Robert ; Kin, Kenrick ; Twigg, Christopher ; Han, Shangchen ; Yang, Ming-Hsuan ; Chien, Shao-Yi: DodecaPen. In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, ACM, oct 2017

Symbolverzeichnis

Evaluation

β	Anzahl der Neustarts eines Filters aufgrund einer nicht positiv semi-definiten Matrix $\mathbf{P}$
$\mathbf{e}$	Fehlervektor an einem Zeitschritt beinhaltet die Abweichung aller Zustandsvariablen und der globalen Orientierung
$\mathbf{q}_{est}$	Vom UKF geschätzte globale Orientierung
$\mathbf{q}_e$	Differenzquaternion zwischen $\mathbf{q}_{est}$ und $\mathbf{q}_{gt}$
$\mathbf{q}_{gt}$	Tatsächliche globale Orientierung
r	Pearson Korrelationskoeffizient
$RMSE_o$	Root Mean Squared Error für die Orientierung
$RMSE_p$	Root Mean Squared Error für die Position

Generell

$\boldsymbol{M}$	Intrinsische Kameramatrix einer Kamera
$\boldsymbol{P}$	Projektionsmatrix einer Kamera
$\boldsymbol{R}$	Rotationsmatrix einer Kamera
$\boldsymbol{T}$	Translationsvektor einer Kamera
$\mathbf{m}_{i_I}$	Der i-te Marker im Bildkoordinatensystem I
$\mathbf{m}_{i_r}$	Rekonstruktion des i-ten Markers durch Triangulation
$\mathbf{q}_{cam}$	Orientierung der realen Kamera, die an die Filterkamera übergeben wird
$\mathbf{t}_{cam}$	Position der realen Kamera, die an die Filterkamera übergeben wird

$\mathbf{t}_C$	Translationsvektor im Kamerakootrdinatensystem
$\mathbf{t}_W$	Translationsvektor im Weltkootrdinatensystem
c	Optisches Zentrum der Kamera
E_i	Menge aller Eckpunkte eines Schachbrettmusters im Bildkoordinatensystem bei der Kamerakalibrierung
E_w	Menge aller Eckpunkte eines Schachbrettmusters im Weltkoordinatensystem bei der Kamerakalibrierung
f	Brennweite der Kamera
I_l	Bildkoordinatensystem der linken Kamera der Stereokamera
I_r	Bildkoordinatensystem der rechten Kamera der Stereokamera
$m_{i_{I_l}}$	Die Projektion des i-ten Markers durch die linke Kamera im Bildkoordinatensystem
$m_{i_{I_r}}$	Die Projektion des i-ten Markers durch die linke Kamera im Bildkoordinatensystem
W	Das Weltkoordinatensystem

Kamera und Kamerakalibrierung

$\bar{\mathbf{P}}$	A priori Kovarianzmatrix Kalman Filters
$\bar{\mathbf{x}}$	A priori Zustandsvektor eines Kalman Filters
$\boldsymbol{\chi}$	Sigma-Punkt eines Unscented Kalman Filters
$\boldsymbol{\mathcal{Y}}$	Sigma-Punkt, auf den $f()$ angewendet wurde
$\boldsymbol{\mathcal{Z}}$	Sigma-Punkt, auf den $h()$ angewendet wurde
$\mathbf{B}$	Kontrollmatrix eines Kalman Filters
$\mathbf{F}$	Zustandstransfermatrix eines Kalman Filters
$\mathbf{H}$	Beobachtungsmatrix eines Kalman Filters
$\mathbf{K}$	Kalman Gain eines Kalman Filters
$\mathbf{P_z}$	Kovarianzmatrix der Messung eines Unscented Kalman Filters

$\mathbf{P}$	A posteriori Kovarianzmatrix eines Kalman Filters
$\mathbf{Q}$	Prozessrauschen eines Kalman Filters
$\mathbf{R}$	Messrauschen eines Kalman Filters
$\mathbf{u}$	Kontrolleingaben eines Kalman Filters
$\mathbf{y}$	Residuum zwischen zwei Messvektoren
$\mathbf{z}$	Messvektor der realen Messung
$f()$	Zustandstransferfunktion eines nicht linearen Kalman Filters
$h()$	Mess- oder Beobachtungsfunktion eines nicht linearen Kalman Filters
w_i^c	Gewichte der Sigma-Punkte zur Berechnung der Kovarianz in einem Unscented Kalman Filter
w_i^m	Gewichte der Sigma-Punkte zur Berechnung des Mittelwerts in einem Unscented Kalman Filter

Markerpose und Markererkennung

$\mathbf{c}$	Mittelpunkt einer von OpenCV erkannten Kontur
$\mathbf{D}$	Matrix in deren Einträgen die Abstände zwischen Koordinaten aus einer Liste stehen (beispielsweise Distanzen zwischen Markern einer Pose
$\mathbf{m}_i$	Der i-te Marker der Markerpose
C	Die Konfiguration eines Tracking-Systems definiert als die Menge aller möglichen Abstände zwischen Markern
D	Die Menge aller möglichen Mengen D_M für alle möglichen Markerposen M eines Trackng-Systems
D_{M_r}	Die Menge aller Abstände zwischen den Markern innerhalb der Markerpose M_r, die eine Teilmenge von C darstellt
g	Granularität eines Tracking-Systems definiert als die kleines Messbare Distanz zwischen zwei Markern
K	Menge aller Bildpunkte einer von OpenCV erkannten Kontur

M_r Die Markerpose, entspricht der Menge aller Marker

u Umfang einer von OpenCV erkannten Kontur

Y Die nicht rotierte oder verschobene Markerpose im Ursprung des Weltkoordinatensystems

y Maximale Distanz zwischen zwei Markern einer Markerpose in einem Trackingsystem

UKF-basiertes Verfahren

$\mathbf{q}$ Globale Orientierung des zu trackenden Objektes als Quaternion

$\mathbf{z_p}$ Messvektor, der aus einem Sigma-Punkt erstellt wurde

$d_{z_{cam}}$ Wert der angibt, wie weit ein Objekt entlang der z-Achse im Koordinatensystem der Kamera vor oder hinter der Kamera liegt

Akronyme

6DoF 6 Degree of Freedom. 6, 7

AR Augmented Reality. 1, 6

EKF Extended Kalman Filter. 7, 8, 12, 13, 20, 21, 24, 56

FOV Field of View. 5, 21

HMD Head Mounted Display. 4, 6–9, 19, 21, 22, 24, 25, 30, 31, 44, 45, 56, 67

PF Partikelfilter. 12, 13

PSD Positiv semi definit. 61

RMSE Root Mean Squared Error. 62, 63, 65, 69, 74

SCAAT Single Constraint At A Time. 7, 24

UKF Unscented Kalman Filter. viii, ix, 7, 8, 12–16, 20, 21, 23, 24, 31, 33, 37, 39, 43, 44, 48, 50, 52, 56–58, 60, 61, 64–71, 74, 75, 77–80, 82–86, 88–90, 92–94

UT Unscented Transform. 13, 14, 57, 58, 61

VR Virtual Reality. 1, 2, 4, 6, 8, 21–23, 26, 65, 66

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 28. Juli 2023