Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Marc Kirchner

Eine auf Sensorfusion basierte Plattform für Personentracking in VR/AR-Umgebungen

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Marc Kirchner

Eine auf Sensorfusion basierte Plattform für Personentracking in VR/AR-Umgebungen

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung

im Studiengang Bachelor of Science Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Kai von Luck
Zweitgutachter: M.Sc. Tobias Eichler

Eingereicht am: 04.12.2017

Marc Kirchner

Thema der Arbeit
Eine auf Sensorfusion basierte Plattform für Personentracking in VR/AR-Umgebungen

Stichworte
Virtual Reality, Augmented Reality, VR, AR, Tracking, Koordinatentransformation, Kalibrierung, Plattform

Kurzzusammenfassung
Natürliche Interaktion in Virtual- und Augmented-Reality Anwendungen bedingt oftmals die Benutzung mehrerer Tracking-Systeme. Diese stellen Tracking-Daten in ihren eigenen Koordinatensystemen bereit. Diese Arbeit befasst sich mit dem Entwurf einer Plattform, welche Entwickler dabei unterstützen soll verschiedene Tracking-Systeme in ihre Virtual- und Augmented-Reality Anwendungen zu integrieren. Dazu stellt die Plattform die verschiedenen Sensordaten in einem einzelnen Koordinatensystem über eine einheitliche Schnittstelle zur Verfügung. Eine Evaluation zeigte, dass die Plattform die für die Interaktion in virtuellen Welten benötigte Latenz und Präzision einhält.
Marc Kirchner

Title of the paper
A platform based on sensorfusion for people tracking in VR/AR-environments

Keywords
virtual reality, augmented reality, VR, AR, tracking, coordinate transformation, calibration, platform

Abstract
Natural interaction in virtual- and augmented-reality applications is often bound to using multiple tracking systems. Those provide tracking data in their own coordinate systems. This thesis deals with the design of a platform which is supposed to help developers integrating different tracking systems into their virtual- and augmented-reality applications. The platform provides different sensor data in a single coordinate system over an unified interface. An evaluation has shown that the platform satisfies the requirements of latency and precision of interaction in virtual worlds.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Science-Fiction zeigte des Öfteren, wie man sich Virtual Reality (VR) vorzustellen hat. „The Matrix" ist dabei wahrscheinlich einer der bekanntesten Filme. Doch was vor wenigen Jahren noch Sci-Fi war, das nimmt immer mehr Form an und ist teilweise schon realisierbar, wenn auch in einer anderen Form als in vielen Filmen und Büchern dargestellt.

Eine große Rolle spielt hierbei die Entwicklung der Computer, der Displays und der Tracking-Systeme. Gordon Moore beobachtete 1965, dass sich die Anzahl der Transistoren, welche auf einem integrierten Schaltkreis unter gebracht werden können, mit jedem Jahr in etwa verdoppelt haben (vgl. Moore 2006a). Zehn Jahre später machte er die Vorhersage, dass sich die Rechenleistung alle zwei Jahre etwa verdoppelt (vgl. Moore 2006b). Bekannt wurde diese Vorhersage als das sogenannte Mooresche Gesetz, welches sich bis heute gut bewahrheitet hat.

Aber nicht nur die Rechenleistung hat sich verbessert, auch die Displays sind stetig anschaulicher geworden. Steigende Auflösung, Pixeldichte und Farbintensität haben es ermöglicht, dass Filme, Serien und Videos nicht nur auf dem Fernseher, sondern auch auf Tablets und Smartphones schön anzuschauen sind.

Mit diesen modernen Displays konnten erste VR-Brillen entwickelt werden. Diese VR-Brillen sind ein vor die Augen geschnalltes Display, welches das Bild dynamisch auf Kopfbewegungen angleicht. Für die Erfassung dieser Kopfbewegungen sind Sensoren sogenannter Tracking-Systeme zuständig.
Richtig reizvoll werden virtuelle Welten dann, wenn man mit ihnen interagieren kann und nicht nur passiver Betrachter ist. Die Interaktion im zweidimensionalen ist sehr etabliert und erforscht. Durchgesetzt hat sich unter anderem die sogenannte WIMP-Schnittstelle (Windows, Icons, Menus, Pointing). Für dreidimensionale Inhalte in der VR ist die WIMP-Schnittstelle jedoch nicht die beste Wahl. Besser ist es, die Interaktion an die alltägliche Interaktion anzulehnen. Aus dem Alltag wissen wir, wie Objekte mit unserem Körper manipuliert werden können (vgl. Dörner u. a. 2014, S. 15-16).

Die Interaktion mit unseren Händen ist dabei ein wesentlicher Bestandteil. Die Übertragung der Hände in die virtuelle Welt macht diese optisch realitätsnäher und bietet die Möglichkeit intuitiv zu interagieren. Dafür müssen die Hände von einem Tracking-System getrackt werden, sodass die realen Bewegungen korrekt in virtuelle umgesetzt werden. Problematisch ist, dass die verschiedenen Tracking-Systeme ihre eigenständigen Koordinatensysteme haben, weshalb diese erst in das Koordinatensystem der virtuellen Welt überführt werden müssen.
Die Berechnung der virtuellen Welten und die Übertragung von realen Bewegungen in diese Welten sorgen für hohen Rechenaufwand, welcher nur aufgrund der stetig gestiegenen Rechenleistung zu verarbeiten ist.
Insgesamt ist die Virtual Reality ein Gebiet, das von vielen neuen Technologien und Konzepten getragen wird. Neue Technologien und Konzepte bringen aber auch immer neue Probleme mit sich. Deshalb wäre es von Vorteil, wenn aufgetretene Herausforderungen abstrahiert und generische Lösungsmöglichkeiten entwickelt werden würden, um die Entwicklung von VR-Anwendungen zu beschleunigen.

1.1 Zielsetzung

Das Ziel dieser Arbeit ist es, Entwickler im Bereich Virtual- und Augmented Reality den Entwicklungsprozess zu erleichtern. Dazu soll ein System beziehungsweise eine Plattform entworfen werden, die das Arbeiten mit multiplen Tracking-Systemen vereinfacht. Die Entwickler sollen sich nicht mit konkreten Schnittstellen der einzelnen Sensoren auseinandersetzen müssen, sondern auf alle Sensordaten über eine einheitliche Schnittstelle zugreifen können. Außerdem sollen die Sensordaten der verschiedenen Tracking-Systeme bereits vorverarbeitet werden.

Da jeder Sensor sein eigenes Koordinatensystem hat, muss der Entwickler in der Regel die Daten jedes einzelnen Sensors in ein Zielkoordinatensystem umrechnen. In der Vorverarbeitung soll genau dieser Rechenschritt erledigt werden und eine Analyse der gemessenen Koordinaten stattfinden. Die Analyse soll zusammengehörige Koordinaten, also jene, die durch das Tracking der gleichen Person ausgelöst wurden, identifizieren und als Information für den Entwickler bereitstellen. Dadurch entsteht ein Tracking, welches nicht nur Informationen darüber liefert, wo sich Personen beziehungsweise bestimmte Körperteile befinden, sondern diesen auch noch Informationen beilegt, um welche Person es sich handelt. Dabei sollen Personen mit eindeutigen Identifikationsnummern (IDs) versehen werden. Es soll außerdem möglich sein, den identifizierten Personen zusätzliche Metadaten beizufügen. So könnten Informationen über die Person, wie zum Beispiel Name, Alter, Geschlecht oder Informationen über die Aufenthaltsdauer als Schlüssel/Wert-Paare hinzugefügt werden.

Mit einer solchen Plattform wird es für den Entwickler einfacher natürliche Interaktion in die VR- und AR-Anwendungen zu integrieren. Außerdem können mit den bereitgestellten Koordinaten Personen als Avatare in die virtuelle Welt geladen werden. Dies macht die Plattform besonders für Mehrbenutzer-Anwendungen nützlich, da die Interaktion der Anwender hier meist im Fokus steht und die Visualisierung der Benutzer notwendig ist, damit die Interaktion stattfinden kann.
Letztendlich soll die Plattform für zukünftige Projekte im Creative Space of Technical Innovation (vgl. Abschnitt 2.5) der HAW Hamburg bereitstehen, sodass weitere Projekte, Bachelor- oder Masterarbeiten darauf aufbauen können.

1.2 Gliederung

Im Folgenden wird eine Übersicht über die fünf Kapitel dieser Arbeit gegeben. Das erste Kapitel besteht aus der Einleitung, Zielsetzung und endet mit dieser Gliederung.

Das Kapitel 2 beginnt mit einer Einführung in Virtual- und Augmented Reality. Ein besonderer Fokus wird dabei auf die Interaktion in virtuellen Welten und den Zusammenhang mit Tracking-Systemen gelegt. Es folgt die Motivation für diese Arbeit und eine genauere Beschreibung der Aufgaben, welche die Plattform erfüllen soll. Anschließend wird das Creative Space for Technical Innovations und dort umgesetzte Projekte, welche von der Plattform profitieren könnten vorgestellt. Abschließend erfolgt eine detaillierte Analyse der Anforderungen an das System und eine Zusammenfassung des ganzen Kapitels.

In Kapitel 3 wird das Design der Plattform vorgestellt. Zu Beginn wird eine Übersicht über die Komponenten des Systems gegeben. Dann werden für aufgetretene Probleme speziell entwickelte Lösungsansätze beschrieben. Auf die Beschreibung der Abläufe innerhalb des Systems erfolgt eine Erläuterung der einzelnen Komponenten und die von der Plattform angebotenen Schnittstellen.

Das Kapitel 4 befasst sich mit der Evaluation. Dafür wurden die im Kapitel 2 ausgearbeiteten nicht-funktionalen Anforderungen bewertet. Für die messbaren Anforderungen wurden verschiedene Szenarien getestet, in denen Messwerte aufgenommen, in Diagrammen aufgearbeitet und vorgestellt wurden. Beendet wird das Kapitel mit einem Überblick über Anknüpfungspunkte, an denen die Plattform noch erweitert werden könnte.

In Kapitel 5 erfolgt eine Zusammenfassung der Ergebnisse dieser Arbeit. Zum Schluss wird noch ein Ausblick über die Entwicklung von Virtual- und Augmented Reality gegeben.

2 Analyse

In diesem Kapitel findet eine genauere Beschreibung der Funktionen der Plattform statt. Es werden Szenarien vorgestellt, welche zur Zeit im Creative Space for Technical Innovations erarbeitet werden und in denen das System genutzt werden könnte. Anschließen werden die Anforderungen an die Plattform genauer analysiert und es findet eine Abgrenzung von Inhalten statt, die nicht Bestandteil dieser Arbeit sind.

2.1 Virtual und Augmented Reality (VR/AR)

Dieser Abschnitt dient als Einstieg in die Themen Virtual Reality und Augmented Reality. Es werden die grundlegenden Begriffe erläutert, damit ein grobes Verständnis für VR und AR erlangt wird. Das Kapitel richtet sich dabei nach dem Buch „Virtual und Augmented Reality (VR/AR)" (vgl. Dörner u. a. 2014).

2.1.1 Virtual Reality

Virtual Reality (VR) gewinnt immer mehr an Popularität (vgl. Abbildung 2.1). Doch trotz dieses Anstiegs an Interesse hat sich noch keine einheitliche Definition gebildet. Das Ziel von Virtual Reality ist es, eine virtuelle Welt zu erschaffen, die sich für den Nutzer real anfühlt. Bereits 1965, drei Jahre bevor der erste Vorgänger der heutigen Virtual Reality Brillen entwickelt wurde (vgl. Sutherland 1968), versuchte man Virtual Reality mit einer technologieorientierten Beschreibung zu erläutern:

> „The ultimate display would, of course, be a room within which the computer can control the existence of matter. A chair displayed in such a room would be good enough to sit in. Handcuffs displayed in such a room would be confining, and a bullet displayed in such a room would be fatal. With appropriate programming such a display could literally be the Wonderland into which Alice walked."
> (Sutherland 1965 zitiert in Dörner u. a. 2014, S. 12)

Im Verlaufe der letzten Jahrzehnte entwickelte sich die Hardware schnell und ist aus dem heutigen Alltag nicht mehr wegzudenken. Hardware wird von Jahr zu Jahr kleiner und schneller,

sodass es mittlerweile möglich ist sogenannte Head-Mounted Displays (HMD) herzustellen. Diese beschränken das Sichtfeld des Nutzers auf die virtuelle Welt, wodurch der Nutzer visuell von der realen Welt abgeschottet ist. Dies hat den Effekt, dass der Nutzer stärker in die virtuelle Welt eintaucht - sie wird immersiver (vgl. Dörner u. a. 2014, S. 14). Immersion ist ein wichtiger Aspekt im Bereich Virtual Reality und wurde schon 1993, lange Zeit vor marktfähigen Head-Mounted Displays, in folgender Definition für Virtual Reality genutzt:

> „Virtual Reality refers to immersive, interactive, multi-sensory, viewer-centered, three-dimensional computer generated environments and the combination of technologies required to build these environments."
> (Carolina Cruz-Neira 1993 zitiert in Dörner u. a. 2014, S. 13)

Diese Definition beschreibt die aktuelle Auffassung des modernen Begriffes 'VR' sehr treffend. Neben den HMDs gibt es noch Cave Automatic Virtual Environments (CAVE) und andere Technologien zur Darstellung der virtuellen Welten. Diese Arbeit bezieht sich jedoch überwiegend auf Virtual Reality im Zusammenhang mit den Head-Mounted Displays.

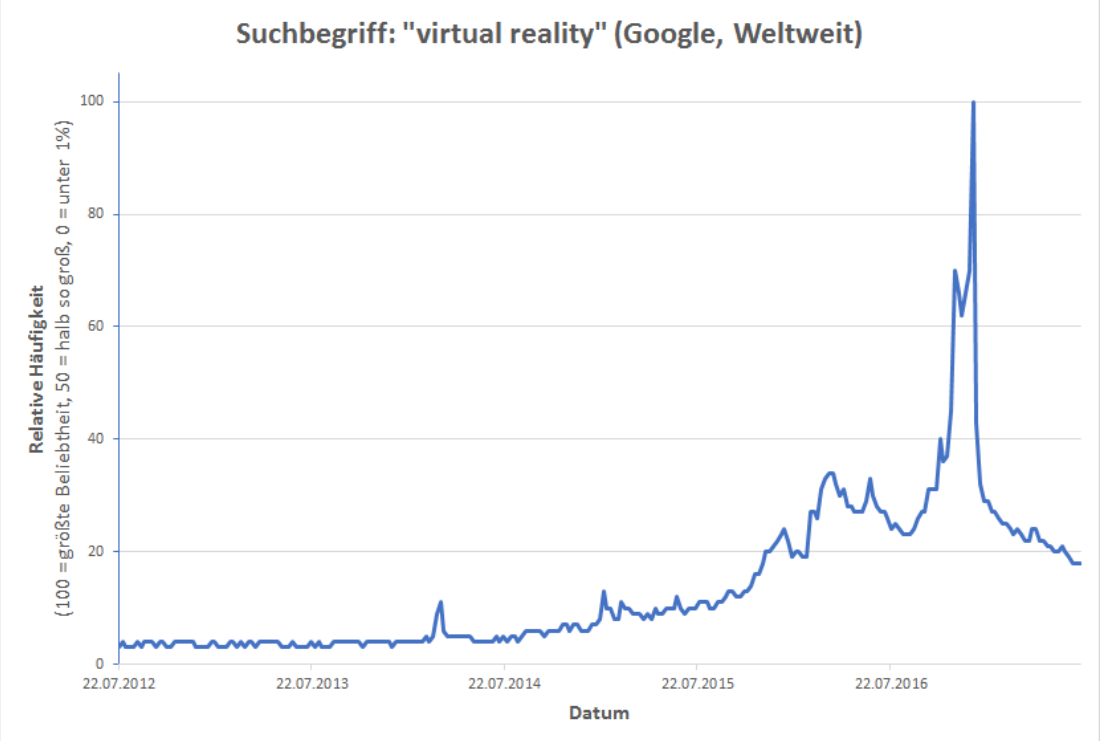

Abbildung 2.1: Suchhäufigkeit des Begriffes „virtual reality" (Google Inc.)

2.1.2 Augmented Reality

Augmented Reality (AR) gewann, neben Virtual Reality, mit dem Erfolg des AR-Spiels „Pokémon GO!" an Aufmerksamkeit. Trotzdem ist der Begriff oftmals noch unklar und wird synonym mit dem Begriff der „Mixed Reality" verwendet (vgl. Dörner u. a. 2014, S. 246). Auch für Augmented Reality gibt es bisher noch keinen globalen Standard. AR lässt sich aber -unter anderem- wie folgt definieren:

> Augmentierte Realität ist eine (unmittelbare, interaktive und echtzeitfähige) Erweiterung der Wahrnehmung der realen Umgebung um virtuelle Inhalte (für beliebige Sinne), welche sich in ihrer Ausprägung und Anmutung soweit wie möglich an der Realität orientieren, so dass im Extremfall (so das gewollt ist) eine Unterscheidung zwischen realen und virtuellen (Sinnes-) Eindrücken nicht mehr möglich ist.
> (Dörner u. a. 2014, S. 246)

Dementsprechend versucht Augmented Reality im Vergleich zur VR nicht den Benutzer von der realen Welt abzuschotten, sondern die reale Welt gezielt zu erweitern (z.B. durch Informationen). Leichter verständlich wird der Unterschied zwischen VR und AR durch das sogenannte „Mixed Reality (MR) Kontinuum" (nach Milgram et al. zitiert in Dörner u. a. 2014, S. 246). Mixed Reality ist demnach ein Kontinuum zwischen der Realität und der Virtualität. Ist der Anteil der Realität höher als der Anteil der Virtualität, so spricht man von Augmented Reality, andersherum spricht man von Augmented Virtuality. Ist der Realitätsanteil gleich Null, so spricht man von Virtual Reality. Das MR Kontinuum ist in der nachfolgenden Grafik verdeutlicht.

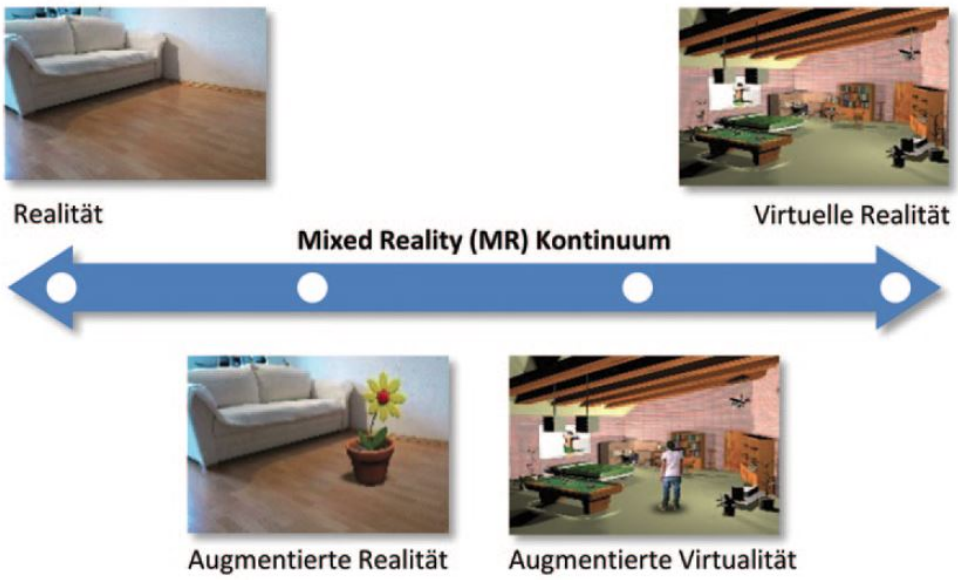

Abbildung 2.2: Mixed Reality Kontinuum (Dörner u. a. 2014, S. 246)

Um die virtuellen Inhalte abhängig von der Position und Ausrichtung des Benutzers korrekt darzustellen, sind Sensoren unerlässlich. Das sogenannte Head-Tracking ist dafür verantwortlich die Position und Ausrichtung des Kopfes zu bestimmen, sodass - im Bereich der VR - der korrekte Ausschnitt der virtuellen Welt gezeigt werden kann.

> „If I turn my head and nothing happens, it ain?t VR!" (Steve Bryson)
> (zitiert nach Dörner u. a., 2014, Seite 13)

2.1.3 Interaktion in VR/AR

Im Abschnitt 2.1.1 wurde Virtual Reality von einer sehr technischen Sicht betrachtet. Oft wird VR auch als Schnittstelle zwischen Mensch und Maschine charakterisiert. Im Bereich der Human-Computer Interaction (HCI) beschäftigt man sich damit, wie man solche Schnittstellen möglichst gebrauchstauglich gestalten kann. Eine grundlegende Frage bei der Wahl der Interaktionsart ist, ob diese auf eine natürliche oder magische Weise geschehen soll. Natürliche Interaktionen sollen möglichst nah am Verhalten in der realen Welt angelehnt sein. Dementsprechend soll Bewegung möglichst durch echtes Laufen umgesetzt werden, während magische Interaktion Bewegung durch z.B. Teleportation gestattet (vgl. Dörner u. a. 2014, S. 158-159).

Wenn die virtuelle Welt größer als die physische VR-Umgebung ist, dann ist das natürliche Fortbewegen nicht ohne Weiteres möglich. Das sogenannte „Redirected Walking" ist eine Technik, die Nutzer in virtuellen Welten umlenkt. Dadurch ist es möglich eine Person im Kreis gehen zu lassen, ihr aber in der virtuellen Welt vorzutäuschen, dass sie geradeaus läuft. Dabei wird ausgenutzt, dass die visuelle Wahrnehmung bei Diskrepanzen mit dem Gleichgewichtssinn für das Gehirn ausschlaggebender ist. Somit können Nutzer große virtuelle Welten erkunden, während sie sich in einem relativ kleinen physischen Bereich aufhalten (vgl. Steinicke u. a. 2010).

Für die verschiedenen Interaktionsarten können unterschiedliche Interaktionsgeräte genutzt werden. Für die natürliche Interaktion wird im Allgemeinen der Körper des Nutzers von Tracking-Systemen (siehe Abschnitt 2.2) getrackt. Ist die Anwendung eine Mehrbenutzer Anwendung, bei der die Anwender miteinander über das System miteinander agieren sollen, so können die Positionsdaten der Tracking-Systeme auch für die Darstellung der Benutzer in der virtuellen Welt genutzt werden. Tracking-Systeme sind damit ein nützliches Mittel für natürliche Interaktion im Bereich VR/AR.

2.2 Tracking-Systeme

Tracking-Systeme spielen in den folgenden Kapiteln eine wichtige Rolle, weil sie die Basis für die in Abschnitt 2.3.2 vorgestellte Plattform bilden. Deswegen werden grundlegende Informationen aufbereitet, um einen Überblick zu erhalten, der es ermöglicht das Thema in den Gesamtkontext dieser Arbeit einzuordnen.

2.2.1 Allgemeines

Tracking-Systeme haben die Funktion Positionen von Personen oder Objekten zu bestimmen. Eines der wohl bekanntesten Tracking-Systeme ist das Global Positioning System (GPS). GPS wird genutzt um Positionen auf der ganzen Erde zu bestimmen. Im Rahmen dieser Arbeit wird jedoch eine andere Art von Tracking-Systemen behandelt. Im Folgenden sind mit Tracking-Systemen solche gemeint, die Positionen in einem lokalen Bereich bestimmen. Das Tracking von Personen bzw. Objekten, am Körper von Personen, ist im Bereich von VR/AR wichtig, da die Personen in Relation zur virtuellen Welt gesetzt werden müssen. Die unterschiedlichen Systeme arbeiten mit verschiedensten Techniken, jedoch basieren viele dieser Systeme auf Infrarot-Sensoren. Nutzt man mehrere Tracking-Systeme mit gleicher Technik, so muss darauf geachtet werden, dass keine Interferenzen entstehen.
Um die Unterschiede zwischen den Systemen zu verdeutlichen werden im Folgenden die verwendeten Systeme kurz aufgezählt und erläutert.

2.2.2 Beispiele

Advanced-Realtime-Tracking (AR-Tracking)

Das Advanced-Realtime-Tracking arbeitet mit Infrarot-Kameras und sogenannten „Targets". Targets bestehen wiederum aus „Markern". Man unterscheidet zwischen aktiven und passiven Markern. Aktive Marker emittieren Infrarotlicht, während passive Marker Strahlung in Richtung ihrer Quelle zurück reflektieren. Im Rahmen dieser Arbeit wurden lediglich passive Targets benutzt, also Targets, welche aus mindestens 4 passiven Markern bestehen. Aufgrund der einzigartigen Konstellation der Marker auf den einzelnen Targets, ist es dem AR-Tracking möglich die Targets eindeutig zu identifizieren und neben der Position auch eine Richtung zu bestimmen (vgl. Advanced Realtime Tracking GmbH a).

Abbildung 2.3: AR-Tracking - passives Clawtarget (Advanced Realtime Tracking GmbH b)

Kinect V2

Die Kinect V2 ist eine Kamera, die neben einem HD-Bild auch Informationen über Skelette liefert. Ein Skelett besteht dabei aus 25 „Joints", welche Knochen und Gelenke repräsentieren. Somit ist die Kinect sehr gut dafür geeignet ganze Personen zu tracken, sodass diese als Avatar in VR- oder AR-Anwendungen dargestellt werden können (vgl. Microsoft Corporation).

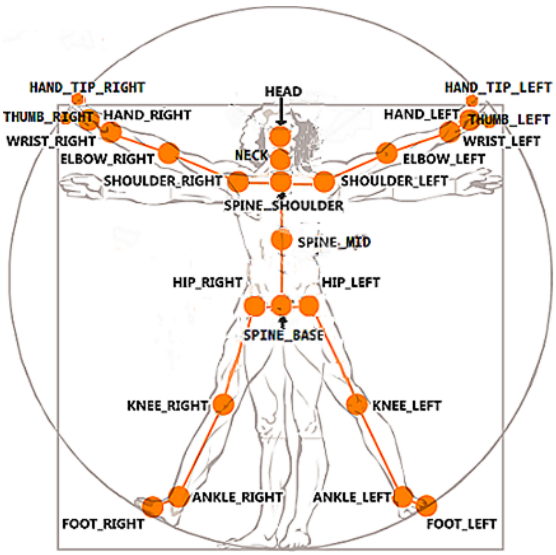

Abbildung 2.4: Kinect V2 - Die 25 Joints (Microsoft)

Leap Motion

Die Leap Motion ist ein etwa zwei Finger großer Sensor, welcher dafür ausgelegt ist Hände zu tracken. Damit hat die Leap Motion einen Vorteil gegenüber klassischen Interaktionsgeräten, wie z.B. dem Controller, wenn es darum geht natürliche Interaktion (vgl. Abschnitt 2.1.3) zu realisieren. Sie wird in den meisten Fällen auf das Head-Mounted Display aufgeklebt, da die Leap Motion so in der Nähe der Hände bleibt und eine relative Position der Hände zum Kopf hergestellt werden kann (vgl. Leap Motion, Inc a).

Abbildung 2.5: Leap Motion und Halterung für HMDs (Leap Motion, Inc b)

2.3 Die Plattform für VR/AR-Projekte

2.3.1 Motivation

Das Erstellen von VR/AR-Projekten ist häufig nicht einfach, da die Gebiete noch sehr jung sind und zur Zeit wenige einheitliche Schnittstellen für Interaktionsgeräte und Sensoren vorhanden sind. Verschiedene Sensoren haben unterschiedliche Stärken und Schwächen, so ist die Leap Motion sehr gut dafür geeignet Hände zu erkennen, aber die Daten sind nur unter Benutzung bestimmter Game Engines leicht verfügbar (vgl. Leap Motion, Inc a). Um die Arbeit von Entwicklern zu vereinfachen wäre es praktisch von konkreten Sensoren zu abstrahieren und die Sensordaten in einer einheitlichen Schnittstelle verfügbar zu machen. Dies hätte den Vorteil, das ein Entwickler sich nicht mit mehreren Schnittstellen verschiedenster Sensoren auseinandersetzen muss. Außerdem wird der produzierte Quelltext übersichtlicher, da die Sensordaten über die gleichen Methodenaufrufe ausgelesen werden.

2.3.2 People Tracking Platform (PTP)

Aus der Motivation eine einheitliche Schnittstelle für Tracking-Sensoren zu entwerfen entstand eine Idee, welche mehr umfasste. Arbeitet ein Entwickler mit mehreren Sensoren, so muss er sich in der Regel dem Problem stellen, die Koordinaten der unterschiedlichen Koordinatensysteme in ein gemeinsames Koordinatensystem zu transformieren. Dieses Problem lässt sich abstrahieren und kann den Entwicklern abgenommen werden. Befasst sich der Entwickler mit einer VR-Erfahrung für eine einzelne Person, so ist er mit dieser Schnittstelle bereits sehr gut bedient. Bei mehreren Personen bleibt das Problem offen, die Positionsdaten den einzelnen Personen zuzuordnen. Es wäre also praktisch für Entwickler, wenn Personen, die einmal identifiziert wurden, langfristig identifiziert bleiben. Das bedeutet, wenn das System eine (unbekannte) Person verortet, dann soll diese Person eine Identifikationsnummer (ID) erhalten und bei der nächsten Verortung (sofern möglich) die gleiche Identifikationsnummer zugewiesen bekommen. Besteht eine solche Zuordnung, so ist es möglich den einzelnen IDs weitere Informationen (Metadaten) hinzuzufügen. Solch eine Funktion könnte nützlich sein um gegebenenfalls Erweiterungen für das System zu entwickeln. Es wäre zum Beispiel eine Gesichtserkennung denkbar, welche den IDs dann die gespeicherte Informationen hinzufügt (Name, Alter, E-Mail Adresse).
Zusammengefasst soll die People Tracking Platform (PTP) folgende Aufgaben erfüllen:

1. Transformation von Koordinaten verschiedener Koordinatensysteme in ein einheitliches Koordinatensystem

2. Zuordnung von Koordinatenmengen zu Personen bzw. IDs
3. Bereitstellen einer Schnittstelle, welche es ermöglicht weitere Informationen zu den IDs hinzuzufügen
4. Bereitstellen einer einheitlichen Schnittstelle, welche die identifizierten Personen mit ihren Koordinaten zur Verfügung stellt

2.4 Voraussetzungen

Um ein System, wie im Abschnitt 2.3.2 beschrieben, realisieren und benutzen zu können, sind einige Voraussetzungen zu erfüllen.

Die multiplen Sensordaten bilden die Basis des Systems, sie werden verarbeitet und über eine einheitliche Schnittstelle wieder zur Verfügung gestellt. Somit sind Sensoren wie das AR-Tracking, die Kinect und die Leap Motion notwendig, um das zu entwickelnde System zu testen und später zu benutzen. Die benötigte Rechenleistung ist von der Menge und Art der verwendeten Sensoren abhängig. Einige Sensoren unterstützen es nicht, dass mehrere Instanzen über den gleichen Rechner betrieben werden. Möchte man beispielsweise mehrere Leap Motions benutzen, so benötigt jede einen einzelnen Computer. Somit entsteht ein weiteres Problem: die Kommunikation der verschiedenen Rechner. Die Daten der verschiedenen Sensoren müssen in einem festgelegten Format (z.B. JSON oder XML) kommuniziert werden. Wie diese Kommunikation im Detail aussieht ist dabei irrelevant. Denkbar wäre beispielsweise die Nutzung von Protokollen wie TCP und UDP. Die identifizierten Voraussetzungen sind somit folgende:

1. Es sind ausreichend viele Sensoren passender Tracking-Systeme vorhanden.
2. Es stehen genügend viele Computer mit entsprechender Rechenleistung zur Verfügung.
3. Es gibt eine performante Messaging-Struktur (auf Grund der zeitkritischen VR-/AR-Anwendungen), welche die Kommunikation zwischen den einzelnen Rechnern ermöglicht.
4. (Es steht ein VR-/AR-System für Entwickler bereit, welche Projekte auf dem System aufbauen wollen.)

2.5 Creative Space for Technical Innovations (CSTI)

Das Creative Space for Technical Innovations (CSTI) der HAW Hamburg ist ein Ort für Experimente und interdisziplinäre Projekte. Innovative Ideen werden entwickelt und prototypisch umgesetzt. Dafür kooperiert der Informatikbereich mit anderen Wissenschaftsbereichen, wie beispielsweise der Mechatronik und dem Kunst & Design (vgl. CSTI b).

Das CSTI ist ein etwa $250\,\mathrm{m}^2$ großes Labor, welches diverse Arbeitsplätze, 3D-Drucker, einen 3D-Scanner, eine Lötstation, einen Show-Room und vieles mehr beinhaltet. Das Herz des CSTI wird durch einen etwa $25\,\mathrm{m}^2$ großen Traversenbereich gebildet. Die Traversen sind mit diversen Lautsprechern und Sensoren ausgestattet, sodass dort unterschiedlichste VR-/AR- und andere Projekte aufgebaut werden können (vgl. CSTI a). Neben der Hardware werden auch Know-How und Softwarestrukturen, wie beispielsweise die Middleware (vgl. Abschnitt 2.5.1), zur Verfügung gestellt.

Damit erfüllt das CSTI alle gestellten Voraussetzungen. Außerdem lässt sich das gesetzte Ziel hervorragend in den Aufgabenbereich des CSTI einordnen. Die fertige PTP wäre außerdem für zukünftige VR-/AR-Projekte im CSTI nützlich, da diese darauf aufbauen könnten.

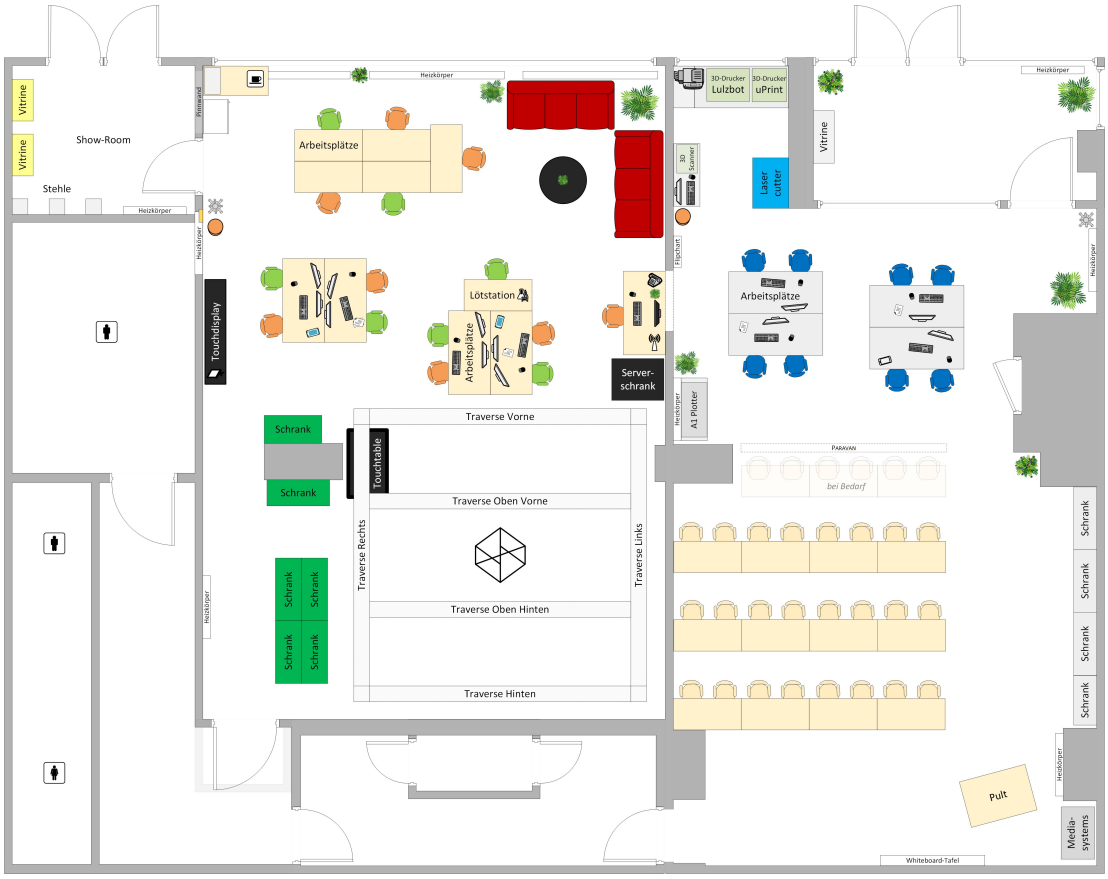

Abbildung 2.6: Raumplan des CSTI (Jeworutzki)

2.5.1 CSTI Plattform

Die CSTI Plattform ist die Middleware des CSTI. Sie basiert auf einer Publish-Subscribe-Architektur, welche es Agenten ermöglicht über Gruppen miteinander zu kommunizieren. Alle Services werden von Agenten implementiert, welche Nachrichten an Gruppen schicken und aus abonnierten Gruppen empfangen können. Empfangene Nachrichten werden von den Agenten sequenziell und unabhängig von anderen Agenten abgearbeitet. Diese Unabhängigkeit führt dazu, dass Agenten wiederverwendbar und leicht in andere Systeme zu integrieren sind (vgl. Eichler u. a. 2017).

2.5.2 Multiagentensystem

Die CSTI Plattform wurde für die Nutzung in Multiagentensystemen (MAS) ausgelegt.
MAS werden durch mehrere eigenständig handelnde Softwarekomponenten (Agenten) gebildet, welche miteinander kommunizieren und entsprechend ihrer Ziele auf Nachrichten reagieren. Durch die Kapselung konkreter Aufgaben in den Agenten, lassen sich diese sehr einfach zu einem verteilten System zusammenfügen (vgl. Wooldridge 2002).
Agenten und Middleware eignen sich hervorragend für das Vorhaben, da die Sensoren in der Regel an mehrere Rechner angeschlossen werden müssen und somit ein verteiltes System unumgänglich ist.

2.5.3 Schnittstellen

Eine gute Plattform zeichnet sich maßgeblich durch die angebotenen Schnittstellen aus. Auch interne Schnittstellen müssen für die Kommunikation der Rechner im MAS entwickelt werden. Für die einfache Erstellung von Schnittstellen bietet das CSTI neben der eigenen Middleware auch einen API-Generator, der Schnittstellen-Bibliotheken für verschiedene Zielsprachen erstellt. Die Bibliotheken definieren mögliche Nachrichten, die über die Middleware verschickt werden können und ermöglichen gleichzeitig automatische Code-Vervollständigung und Type Checking. Die gewünschten Nachrichtenformate werden in einer Domain Specific Language (DSL) angegeben und dann vom Generator in die Sprachen Java, Scala, C++ und Javascript übersetzt. Die Syntax der DSL ist in dem nachfolgenden Beispiel erkennbar.
(vgl. Eichler 2014)

```
name ExampleAPI
version 0.1.0-SNAPSHOT
package de.hawhamburg.csti.example

trait Zahl

//Einzeiler Kommentar
msg Nachricht(attribut : String, liste : Seq[Int], optionalesAttribut : Option[String])

/*
Mehrzeiliger
Kommentar
*/
msg Bruch(zaehler : Int, nenner : Int) extends Zahl
msg Potenz(basis : Int, exponent : Int) extends Zahl
```

```
val msg : Nachricht = Nachricht("ein String", Seq(1,2,3), Some("anderer String"))
val attribut : String = msg.attribut
val liste : Seq[Int] = msg.liste
val optionalesAttribut : Option[String] = msg.optionalesAttribut
```

Listing 2.1: Zugriff auf die definierten Nachrichten in Scala

Die Kombination aus Middleware und API-Generator ermöglicht es ohne großen Aufwand verteilte Systeme zu bauen, da selbst wenn mit verschiedenen Programmiersprachen gearbeitet wird, praktisch keine Arbeit für die Kommunikation der Rechner anfällt.
Die Middleware und der API-Generator bieten somit die in den Voraussetzungen aus Abschnitt 2.4 geforderte Messaging-Struktur.

2.6 Szenarien

Nachfolgend werden Szenarien beschrieben, um die Anwendungsfälle der in Abschnitt 2.3 beschriebenen PTP zu verdeutlichen.

2.6.1 Virtueller Handshake

Das Szenario „Virtueller Handshake" beschreibt eine VR-Erfahrung für zwei Personen, welche jeweils ein Head-Mounted Display tragen. In diesem Szenario sollen die Personen, welche sich im selben Raum befinden, die Hand schütteln. Dabei sollen die Körper der beiden Personen möglichst vollständig und genau in der virtuellen Welt dargestellt werden. Liegen nur Informationen über die Positionen der Hände der beiden Personen vor, so reicht die Darstellung der Hände. Ziel ist es, dass der Handschlag in der virtuellen Welt mit einem Handschlag in der realen Welt einhergeht. Dies ist der Fall, wenn die Körper der Personen, ganz besonders die Hände, möglichst genau verortet und in die virtuelle Welt übersetzt werden.
Dieses Szenario wird von Niklas Gerwens in einer eigenständigen Bachelorarbeit behandelt (vgl. Gerwens 2017).

In diesem Szenario benötigen die beiden Virtual Reality Systeme Informationen über die Positionen der beiden Personen, damit diese in die virtuelle Welt übersetzt werden können. Genau diese Informationen soll die PTP liefern, sodass der Entwickler des Szenarios sich nicht im Detail mit den nötigen Sensoren und Transformationen der Koordinaten auseinandersetzen muss.

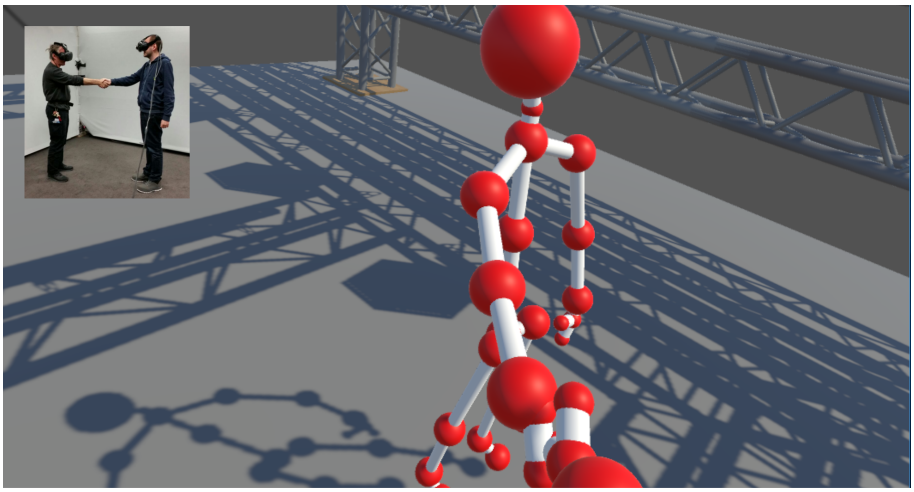

Abbildung 2.7: Virtueller Handshake

2.6.2 Shelldon

„Shelldon" ist der Codename für ein Projekt, welches viele Forschungsbereiche vereint. Hauptsächlich geht es um Smart-Objects und Blended- bzw. Mixed-Reality. Die Smart-Objects sind Schildkröten, welche mit transparentem Filament 3D-gedruckt wurden. Diese wurden dann mit Microcontrollern, Sensoren und LEDs versehen, so dass die Schildkröten Dinge wahrnehmen (sense), daraus Schlussfolgern (reason/plan) und letztendlich Handeln (act) können.

Foto: Jessica Broscheit (CSTI)
3D-Model: pmoews

Abbildung 2.8: Shelldon

Damit ist es den Schildkröten beispielsweise möglich zu erkennen, dass sie auf dem Rücken liegen. Sie sind abhängig vom Wohlbefinden beleuchtet, im Normalzustand sind die Schildkröten grün, dreht man sie auf den Rücken, so leuchten alle rot auf, um eine empathische Reaktion zu demonstrieren.
Die Schildkröten sollen aber ebenfalls die Brücke zwischen realer und virtueller Welt sein. Sie werden in der VR-Welt dargestellt und können dort an einem Strand freigelassen werden. Ein interessanter Effekt entsteht durch die spürbar physische Präsenz der Schildkröte in der Hand, verbunden mit der lebendigen Darstellung in der virtuellen Welt.
Durch die Vernetzung der Schildkröten untereinander bedient sich das Projekt an Aspekten des „Internet of Things"(IoT). Die Schildkröten stellen als Brücke zwischen Realität und Virtueller Welt auch eine Schnittstelle dar, welche dem Bereich der Human-Computer-Interaction (HCI) zuzuordnen ist. Die Behandlung dieser vielen Forschungsbereiche macht Shelldon zu einem interessanten Projekt. Verantwortlich für dieses Projekt ist Gerald Melles, welcher dieses in einem Interview vorgestellt hat (Melles 2017).

Auch in diesem Szenario müssen die Positionen der Schildkröten korrekt in die virtuelle Welt übersetzt werden, so dass der Nutzer die virtuelle Schildkröte auch wirklich vor sich sieht, während er das reale Gegenstück in der Hand hält. Dafür werden die Schildkröten mit AR-Tracking Markern versehen, sodass es möglich wird sie zu tracken. Die Umrechnung der Koordinaten aus dem AR-Tracking System in ein gewünschtes Koordinatensystem könnte wiederum über die PTP erfolgen.

2.6.3 Anwesenheits-Status

Da im CSTI viele Leute ohne feste Zeiten anwesend sind, ist es oft schwierig bestimmte Personen spontan im CSTI zu kontaktieren. Es wäre also praktisch für CSTI-Mitarbeiter eine Online-Übersicht über alle im CSTI anwesenden Mitarbeiter zu haben. Denkbar wäre eine Liste von Mitarbeitern, die in der letzten halben Stunde im CSTI waren.
Mit Hilfe der Erweiterungsschnittstelle für Metadaten wäre es der geplanten PTP möglich dieses Szenario ohne großen Aufwand zu realisieren. Eine Erweiterung wie zum Beispiel eine Gesichtserkennung wäre nötig, damit den getrackten Personen Identitäten zugewiesen werden können. Während das Projekt Shelldon und der virtuelle Handshake Szenarien sind, welche aktuell bearbeitet werden, ist das Szenario des Anwesenheits-Status ein rein theoretisches, welches noch praktisch umgesetzt werden muss.

2.7 Anforderungen

Die PTP hat verschiedenste Anforderungen. Die funktionalen Anforderungen beschreiben dabei das Verhalten und die Funktionalität des Systems, während die nicht-funktionalen Anforderungen bestimmen, unter welchen Voraussetzungen diese Funktionalität gewährleistet werden muss. Im Folgenden werden die an das System gestellten Anforderungen noch einmal aufgezählt und erläutert.

2.7.1 Funktionale Anforderungen

Die in diesem Abschnitt behandelten funktionalen Anforderungen lassen sich aus der in Abschnitt 2.3.2 erläuterten Idee ableiten.

1. *Verarbeitung von Sensordaten*
   Das System muss dazu in der Lage sein die Daten verschiedener Sensoren verarbeiten zu können. Dazu müssen Koordinaten und Richtungsdaten eindeutig bestimmter Körperteile ausgelesen werden.

2. *Bestimmung von Rotation und Translation*
   Die Rotation und die Translation zwischen zwei Koordinatensystemen zweier Sensoren müssen bestimmbar sein. Ob die Bestimmung durch manuelles ausmessen von Winkeln und Distanzen oder durch einen halbautomatischen Prozess der Kalibrierung erfolgt, ist dabei unwichtig.

3. *Anwendung von Rotation und Translation*
   Die berechneten Rotationen und Translationen müssen auf neu eingehende Sensordaten anwendbar sein, damit diese in ein einheitliches Koordinatensystem überführbar sind.

4. *Transformierte Koordinaten bereitstellen*
   Die in das einheitliche Koordinatensystem überführten Koordinaten und Ausrichtungen müssen über eine Schnittstelle verfügbar gemacht werden. Wichtig ist, dass die Koordinaten zusammen mit den Informationen über das Ursprungskoordinatensystem und das Zielkoordinatensystem bereitgestellt werden, damit die Quelle und das Koordinatensystem nach der Transformation noch nachvollziehbar sind.

5. *Anhaltende Identifikation von Personen*
   Aus den bereitgestellten transformierten Koordinaten müssen Personen nachhaltig identifiziert werden. Das heißt, dass eine von den Tracking-Systemen getrackte Person eine

eindeutige ID zugeordnet bekommt, welche sie behält, solange sie den Trackingbereich nicht verlässt.

6. *Editieren von Metadaten von identifizierten Personen*
   Es muss möglich sein einer identifizierten Person beziehungsweise einer vergebenen ID Metadaten hinzufügen zu können. So können zu der ID "5" Informationen über beispielsweise Namen und Alter gespeichert werden.

7. *Bereitstellen von identifizierten Personen*
   Die vom System identifizierten Personen sollen inklusive ihrer Daten über Koordinaten, Ausrichtung und anderer Metainformationen, nach außen bereitgestellt werden.

8. *Speichern und Laden von Konfigurationen*
   Wurden Rotationen und Translationen für einen bestimmten Aufbau von Sensoren in das System integriert, so soll es möglich sein diese Konfiguration zu speichern und bei einem Neustart zu laden.

2.7.2 Nicht-funktionale Anforderungen

Neben den festgestellten funktionalen Anforderungen existieren auch nicht-funktionale Anforderungen, welche in diesem Abschnitt betrachtet werden.

1. *Latenz*
   Insbesondere Virtual Reality Anwendungen sind sehr latenzkritisch. Bewegungen des Kopfes verlangen eine zeitnahe Anpassung des Blickwinkels in der virtuellen Welt, denn sonst stimmt die visuelle Wahrnehmung nicht mit der vestibulären Wahrnehmung überein und es kann zu sogenannter „Cybersickness" kommen (vgl. Dörner u. a. 2014, S. 56). Da die von der PTP zur Verfügung gestellten Positionsdaten unter anderem zur Visualisierung und Positionierung in der virtuellen Welt genutzt werden, müssen diese ebenfalls mit einer akzeptablen Latenz bereitgestellt werden. Die angestrebte Latenz liegt bei etwa $20\,\mathrm{ms}$, eine Latenz von über $50\,\mathrm{ms}$ sorgt für eine zu Hohe Diskrepanz. Folglich müssen Sensoren die für das Head-Tracking verantwortlich sind eine Frequenz von mindestens $50\,\mathrm{Hz}$ haben (vgl. Fuchs 2017, S. 120).
   Insgesamt sollte auf eine performante Verarbeitung der Sensordaten geachtet werden, um die Latenz und Cybersickness so gering wie möglich zu halten.

2. *Präzision*
   Die Präzision der Positionsdaten spielen eine ebenso wichtige Rolle. Das im Abschnitt

2.6.1 erläuterte Handshake-Szenario ist ein gutes Beispiel. Um dem Nutzer gegenüber erfolgreich die Hand reichen zu können, sollten die transformierten Positionsdaten eine Abweichung von einigen Zentimetern möglichst nicht überschreiten. Bei zu hohen Abweichungen besteht die Möglichkeit, dass die Personen sich beim Versuch die Hände zu geben verfehlen.

3. *Skalierbarkeit*
   Prinzipiell ist die Menge der verwendeten Sensoren für ein VR-/AR-Projekt unbegrenzt, deswegen soll die PTP skalierbar sein. In Zusammenhang mit der Latenz-Anforderung heißt das, dass eine möglichst unbegrenzte Menge an Sensoren verwendbar sein sollen, ohne dass die Latenz darunter leidet.

4. *Erweiterbarkeit*
   Die Erweiterbarkeit des Systems lässt sich in zwei Punkte unterteilen. Erstens soll gewährleistet sein, dass neu entwickelte Sensoren zukünftig integrierbar sind. Diese Integration soll stattfinden können, ohne dass der vorhandene Quelltext dafür geändert werden muss. Zweitens sollen die in 2.3.2 angesprochenen Erweiterungen für das System entwickelbar sein.

2.8 Abgrenzung

Bei dieser Arbeit handelt es sich um ein Forschungsprojekt, bei dem ein theoretisches Konzept als realisierbar bestätigt werden soll. Deswegen findet in diesem Abschnitt eine Abgrenzung von Inhalten statt, die bewusst nicht Bestandteil dieser Arbeit sind.

Sicherheit

Auch wenn Positionsdaten durchaus relevant für eventuelle Angreifer sein könnten, wurde keinerlei Verschlüsselung und Authentifizierung einbezogen, da diese nicht zum Kern des theoretischen Konzeptes beitragen.

Grafische Oberflächen

Grafische Oberflächen (GUIs) dienen der einfacheren Benutzbarkeit des Systems und haben keine Auswirkungen auf die Funktionalität, folglich dienen entwickelte GUIs dem einfacheren Testen, sind sehr minimalistisch gestaltet und werden nicht weiter in dieser Arbeit behandelt.

2.9 Zusammenfassung

Im Analyse Kapitel wurden die Themen Virtual Reality, Augmented Reality und Tracking-Systeme eingeführt und im groben erläutert. Tracking-Systeme sind für VR/AR-Erfahrungen unabdingbar, da mindestens die Positionierung des Kopfes getrackt werden muss. Sollen außerdem Körper realer Personen in die virtuelle Welt übertragen werden, so müssen diese mit Tracking-Systemen erfasst werden und dann in die virtuelle Koordinatenwelt umgerechnet werden. Es wurde das Potential für eine Plattform erkannt, welche die Schnittstellen verschiedener Tracking-Systeme auf eine einheitliche Schnittstelle vereinfacht und den Umrechnungsprozess in die virtuelle Koordinatenwelt übernimmt.
Dabei unterliegt die PTP verschiedenen nicht-funktionalen Anforderungen. Sie muss die ausgehenden Daten mit einer Verzögerung von unter $50\,\mathrm{ms}$ liefern und sollte Abweichungen von wenigen Zentimetern nicht überschreiten. Wünschenswert wären außerdem, dass die PTP mit steigender Anzahl an Tracking-Systemen skalierbar bleibt und dass das integrieren von neuen Tracking-Systemen einfach möglich ist.

3 Design

Aus den im Abschnitt 2.4 identifizierten Voraussetzungen ergab sich, dass eine performante Messaging-Struktur von Nöten ist. Die vom CSTI bereitgestellte Middleware erfüllt genau diese Aufgabe und bildet damit die Basis für das weitere Design der PTP.

3.1 Kalibrierung

Die PTP soll die Koordinaten der verschiedenen Sensoren in ein einheitliches Koordinatensystem überführen. Für die Realisierung einer solchen Transformation müssen die Sensoren mit dem Zielkoordinatensystem in Relation gesetzt werden. Das heißt, dass eine Verschiebung (Translation) und eine Drehung (Rotation) notwendig sind, um die Koordinaten umzurechnen. Für die Ermittlung der Translation und der Rotation existieren mehrere Möglichkeiten. Die simpelste Methode wäre es, die Werte auszumessen. Bei einer hohen Anzahl an Sensoren führt das zu großem Aufwand, da jeder Sensor berücksichtigt werden muss und die Koordinatensysteme der Sensoren nicht immer selbsterklärend liegen. Das Ausmessen jedes einzelnen Sensors relativ zum Zielkoordinatensystem erscheint mühsam, weswegen die Wahl auf einen Kalibrierungsprozess gefallen ist. Dafür werden korrespondierende Koordinatenpunkte der Sensoren gemessen und aus diesen eine Relation hergestellt. Das heißt jedoch gleichzeitig, dass die Sensoren nur in Relation zueinander und nicht zu einem gewünschten Koordinatensystem gesetzt werden. Allerdings können die Koordinaten, sobald sie in ein Koordinatensystem *eines* Sensors umgerechnet wurden, immer noch in das Zielkoordinatensystem überführt werden, indem nur dieser eine Sensor in Bezug zu dem Zielkoordinatensystem gesetzt wird.
Wie die Kalibrierung aus mathematischer Sicht abläuft, wird im Abschnitt 3.6.4 erläutert.

3.2 Komponenten

Das Komponentendiagramm in Abbildung 3.1 liefert einen ersten Überblick über die Agenten und ihre Beziehungen. Zur Übersichtlichkeit wurden einige verwandte Nachrichten und Agenten gruppiert und wenige Nachrichten nicht einbezogen. Ausgehend von einem Akteur

(*Actor*) wird das System gesteuert. Die zentrale Verwaltung wird von dem *Master* übernommen. Das Sammeln der Sensordaten für die Kalibrierung geschieht in den StaticCollector- und DynamicCollector-Agenten. Die Sensordaten kommen von Adaptern für die Agenten der genutzten Tracking-Systeme. Verrechnet werden diese dann im *Calculator* Agenten und die Ergebnisse über den Master an die Agenten geschickt, welche für die Transformation der Koordinaten zuständig sind. Um Kalibrierungen nach einem Neustart nicht zu verlieren, existiert der *Persistance* Agent, welcher Einstellungen im Dateisystem speichern und laden kann. Genauere Beschreibungen über die Abläufe erfolgen in dem Kapitel 3.5, während die einzelnen Agenten im Kapitel 3.6 erläutert werden. Auf den *DynamicCollector*- und *DynamicTransformer*-Agenten wird im Folgenden wenig eingegangen. Diese wurden im Design berücksichtigt, jedoch nicht implementiert, weshalb lediglich grundlegende Designentscheidungen zu ihnen erläutert werden.

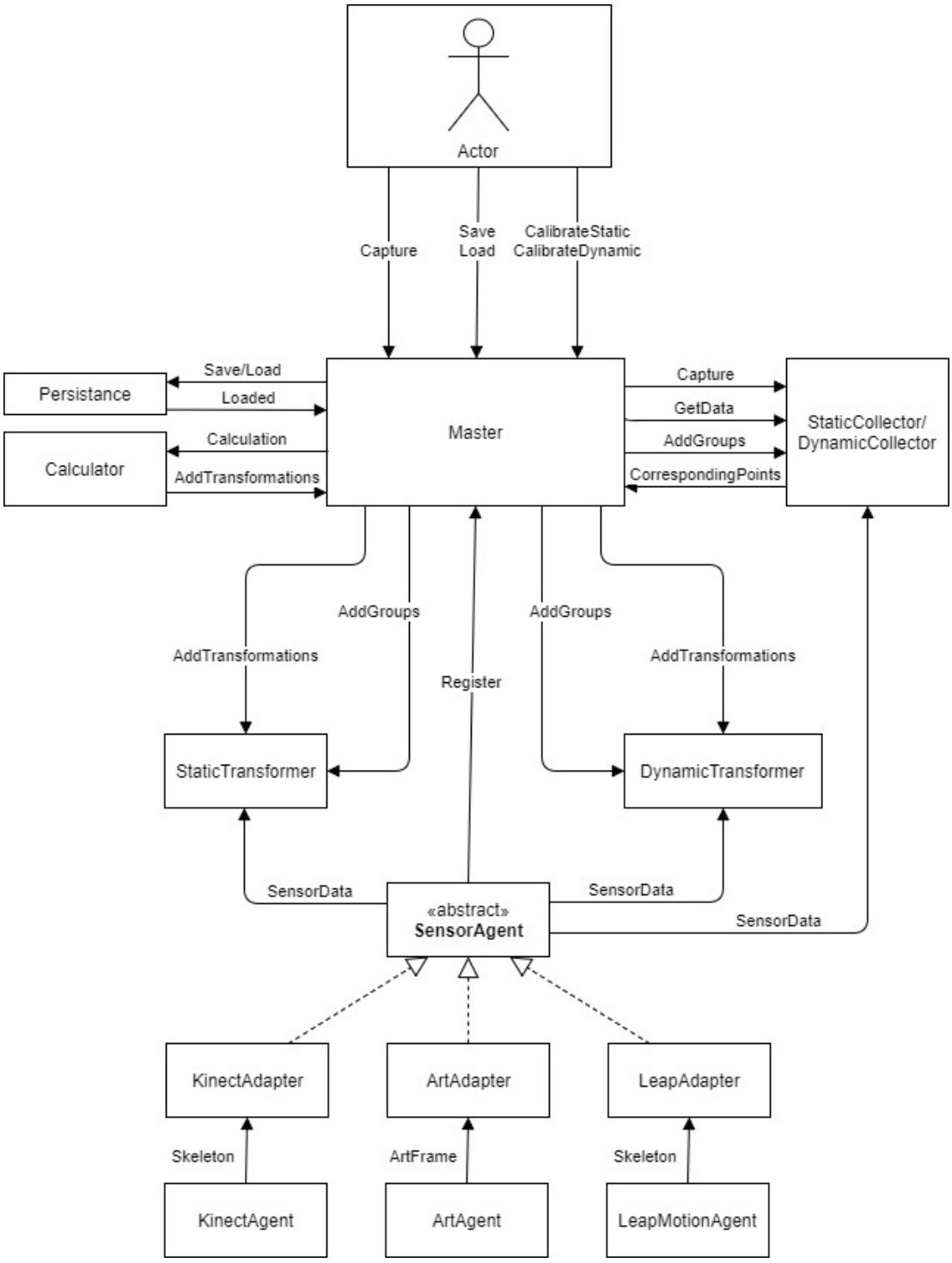

Abbildung 3.1: Agenten-Konstellation der Koordinatentransformation

3.3 Probleme

Während der Entwicklung und Gestaltung der PTP sind eine Vielzahl von Problemen und Schwierigkeiten aufgetreten. Einige dieser Probleme könnten in gleicher Form in ähnlichen Projekten auftreten, weswegen die Probleme in diesem Kapitel aufgezeigt und erläutert werden. Konkrete Lösungsstrategien für die hier benannten Probleme werden in den Abschnitten 3.6 und 3.8 behandelt.

3.3.1 Synchronisation der Sensoren

Die im Abschnitt 3.1 angesprochene Kalibrierung sammelt korrespondierende Punkte verschiedener Sensoren. Korrespondierend heißt, dass die Punkte zum gleichen Zeitpunkt am selben Ort waren, auf Grund der unterschiedlich liegenden Koordinatensysteme der Sensoren, jedoch unterschiedliche Koordinaten haben.
Die Latenzzeiten der Sensoren können variieren und es können zusätzliche Latenzen beim Verarbeiten und Kommunizieren der Daten entstehen. Folglich könnten zwei Punkte vom selben Ort zu verschiedenen Zeiten im System verarbeitet werden. Werden zwei Punkte in Relation gesetzt, welche nicht vom gleichen Zeitpunkt stammen, dann führt dies später zu ungenaueren Ergebnissen, die Genauigkeit der transformierten Koordinaten sinkt also. Dieses Problem der Synchronisation kann auf verschiedene Weisen gelöst werden, welche im Abschnitt 3.4.1 und 3.4.2 behandelt werden.

3.3.2 Sensor-Anomalien

Anomalien sind Ausbrüche vom erwarteten Verhalten der Sensoren. Sie können beispielsweise auftreten, wenn Objekte den Sichtbereich von Kameras blockieren, oder mehrere Infrarot-Signale miteinander interferieren. Ähnlich wie bei der Latenz der Sensoren, können Anomalien zur Kalibrierungszeit dafür sorgen, dass die berechneten Matrizen für Rotation und Translation größere Abweichungen von den korrekten Matrizen haben. Aber auch nach der Kalibrierung können Anomalien negative Einflüsse haben. So können, auf Grund der fehlerhaften Koordinaten einer Anomalie, Objekte in VR-Anwendungen hin und her springen und die VR-Erfahrung des Nutzers beeinträchtigen.

3.3.3 Debugging

Das Debugging stellte ein großes Problem dar. Das prüfen der Korrektheit von Matrizen und dreidimensionalen Koordinaten erwies sich als schwierig. Die Menge an Zahlen ist unüber-

sichtlich und schwierig in Relation zu setzen, weswegen eine Lösung gesucht wurde, welche das Debugging erleichtern sollte. Gelöst wurde dies durch die in Abschnitt 3.4.3 erläuterte Visualisierung.

3.3.4 Verschiedene Sensoren

Die verschiedenen Sensoren tracken verschiedene Körperteile und stellen ihre Daten auf unterschiedliche Art und Weise zur Verfügung. Dies macht eine einheitliche Verarbeitung schwierig und den dazugehörigen Code unübersichtlich. Um die Sensordaten einfacher verarbeiten zu können, wurde eine einheitliche Schnittstelle für alle Sensoren entwickelt, welche in Abschnitt 3.6.2 genauer beschrieben wird.

3.4 Lösungen

3.4.1 Zeitstempel und Zuordnen

Die Problematik der zeitversetzten Verarbeitung von korrespondierenden Sensordaten aus Abschnitt 3.3.1 kann gelöst werden, indem erfasste Sensordaten direkt nach der Erfassung mit Zeitstempeln versehen werden. Dann ist es später möglich die einzelnen Datenpakete der verschiedenen Sensoren über die Zeitstempel miteinander in Verbindung zu setzen. Wird mit mehreren Rechnern gearbeitet, so muss darauf geachtet werden, dass diese miteinander synchron sind.

3.4.2 Mittelwertbildung über ein Zeitfenster

Da die in Abschnitt 3.3.2 angesprochen Sensor-Anomalien jederzeit auftreten können sollte man sich nicht auf einzelne Sensorwerte verlassen. Deswegen ist eine Mittelwertbildung über die Sensorwerte, welche über ein größeres Zeitfenster gebildet werden verlässlicher als Einzelwertbetrachtungen. Dabei muss darauf geachtet werden, dass die getrackten Punkte sich während des aktiven Zeitfensters möglichst nicht bewegen. Die einzelnen Werte, also auch die Anomalien, verlieren an Gewicht und somit an negativen Einfluss auf das Ergebnis. Auch das Problem der Synchronisation aus Abschnitt 3.3.1 wird mit der Mittelwertbildung gelöst.

3.4.3 Visualisierung

Anhand von Matrizen und Vektoren zu erkennen, ob die Berechnungen korrekt sind erwies sich als schwierig. Visualisierungen lassen leichter erkennen, ob Achsen vertauscht oder gespiegelt wurden oder ob ähnliche Fehler vorliegen. Deswegen wurden die transformierten Koordinaten

mit Hilfe von Unity visualisiert. Eine bereits existierende Szene mit dem Traversensystem des CSTI wurde von Niklas Gerwens um eine Anbindung zur Middleware des CSTI erweitert. Somit war es möglich die transformierten Koordinaten über die Middleware an Unity zu schicken und dort anzeigen zu lassen. Die genaue Verarbeitung der ankommenden Daten in Unity wird in der Bachelorarbeit von Niklas Gerwens thematisiert (vgl. Gerwens 2017).

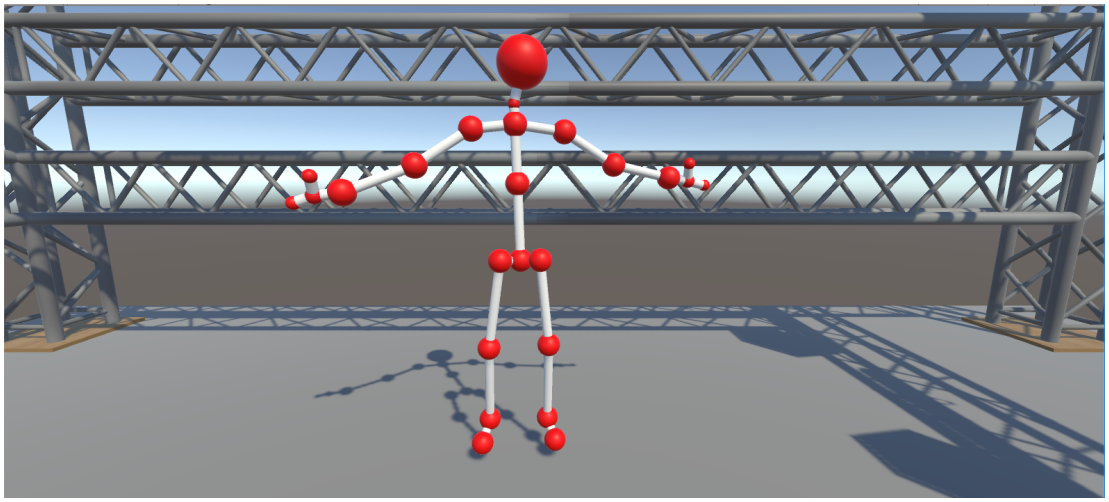

Abbildung 3.2: Visualisierung von Koordinaten in Unity

3.5 Die Abläufe

Der Kern des Systems wird durch drei verschiedene Prozesse gebildet. Das Ergebnis des Kalibrierungsprozesses bildet die Grundlage für den Transformationsprozess, welcher wiederum die Basis für den Identifikationsprozess ist. Die genauen Abläufe werden in diesem Abschnitt in Form von Sequenzdiagrammen veranschaulicht und ergänzend erklärt.

3.5.1 Kalibrierung

Der Kalibrierungsprozess wird durch einen Nutzer manuell ausgelöst, indem eine *CalibrateStatic*-Message an den *Master*-Agenten geschickt wird. Diese enthält Informationen über die Sensoren, welche berücksichtigt werden müssen und gibt an, welche Körperteile für die Kalibrierung herangezogen werden können. Der *Master* informiert dann einen verfügbaren *StaticCollector*-Agenten mit einer *AddGroups*-Message über die Middleware-Gruppen, welche abonniert werden müssen, damit die Sensordaten empfangen werden.

Der *StaticCollector* ist nun für den praktischen Prozess der Kalibrierung vorbereitet. Der Nutzer schickt eine *Capture*-Message an den *Master*, welcher diese an den *StaticCollector* weiterleitet. Bei Ankunft wird ein Intervall gestartet, in dem korrespondierende Koordinatenpunkte der Sensoren gesammelt werden.
Beendet wird der Sammel-Prozess durch eine *GetData*-Message vom Nutzer an den *Master*. Dieser leitet die Nachricht erneut weiter und erhält als Antwort die gesammelten korrespondierenden Koordinatenpunkte. Diese werden mit einer *Calculation*-Message an einen *Calculator*-Agenten geschickt und dort in die nötigen Translationen und Rotationen umgerechnet. Die Translations- und Rotationspaare werden mit Informationen über Quell- und Zielkoordinatensystem in Form einer *AddTransformations*-Message an den Master übermittelt. Dieser speichert diese Informationen ab und leitet den Transformationsprozess ein, welcher nachfolgend erläutert wird.

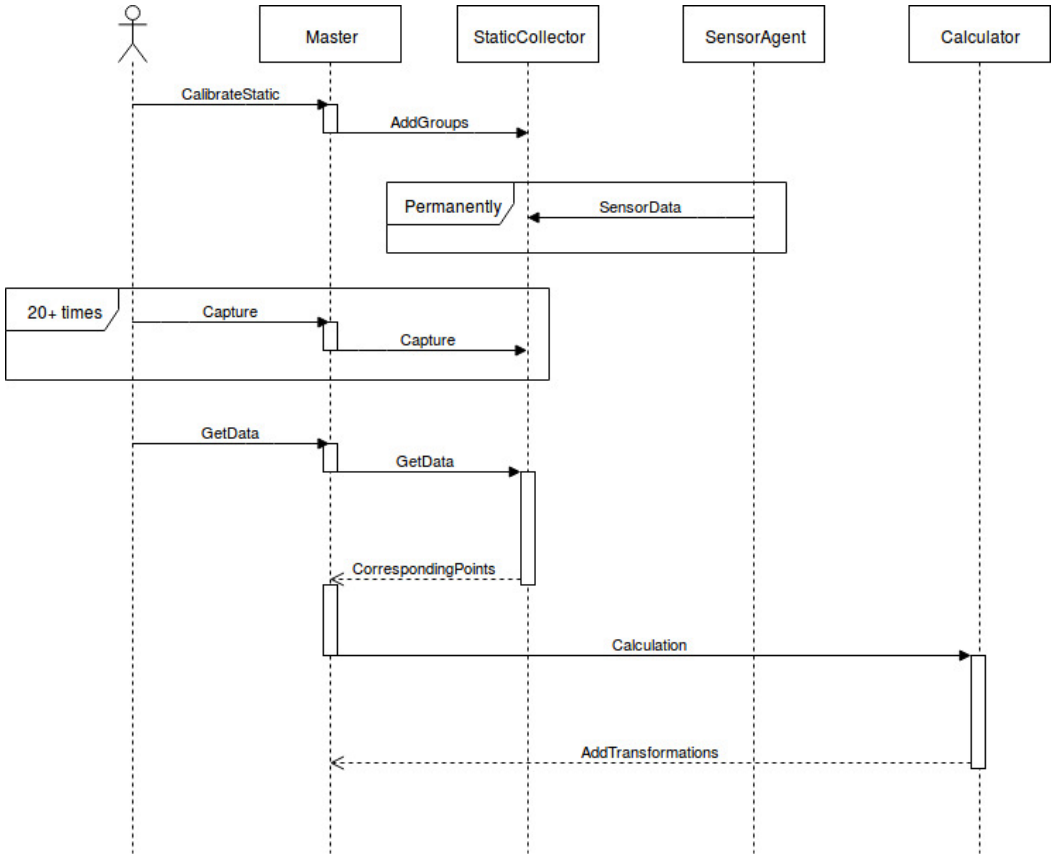

Abbildung 3.3: Ablauf der Kalibrierung

3.5.2 Transformation

Die im Kalibrierungsprozess berechneten Translationen und Rotationen werden vom *Master* an einen *StaticTransformer*-Agent gesendet. Außerdem werden dem *StaticTransformer* die notwendigen Middleware-Gruppen der Sensoren vom *Master* mit einer *AddGroups*-Message mitgeteilt. Danach ist der *StaticTransformer* bereit neu eingehende Sensordaten zu transformieren. Bei Eingang von *SensorData*-Messages werden die Koordinaten transformiert und dann in Form der *LocationPackage*-Message in einer Middleware-Gruppe veröffentlicht. Diese Nachricht wird im Identifikationsprozess genutzt und weiter verarbeitet, könnte aber jedoch auch von anderen Systemen als Tracking-Information genutzt werden.

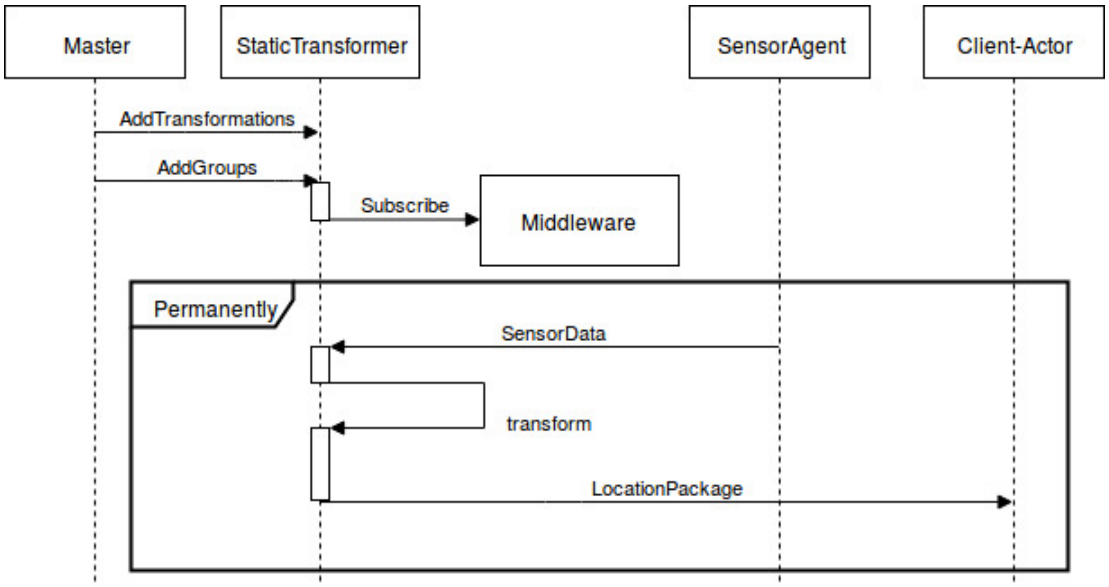

Abbildung 3.4: Ablauf der Transformation

3.5.3 Identifikation

Der Identifikationsprozess dient der langfristigen Identifizierung von Personen. Der *MetadataCore*-Agent ist die zentrale Verwaltungseinheit der Metadaten über identifizierte Personen. Empfängt er eine *LocationPackage*-Message aus dem Transformationsprozess, so stößt er eine Identifizierung an, indem er eine *IdentificationJob*-Message an einen *Identifier*-Agenten schickt. Dieser analysiert daraufhin um welche Person es sich handelt oder ob es gegebenenfalls eine neue Person ist. Das Ergebnis der Analyse schickt er in Form einer *Identification*-Message wieder zurück an den *MetadataCore*. Dieser veröffentlicht die ihm verfügbaren Informationen über identifizierte Personen und dazugehörige Metadaten in einem festgelegten Intervall in einer Middleware-Gruppe.

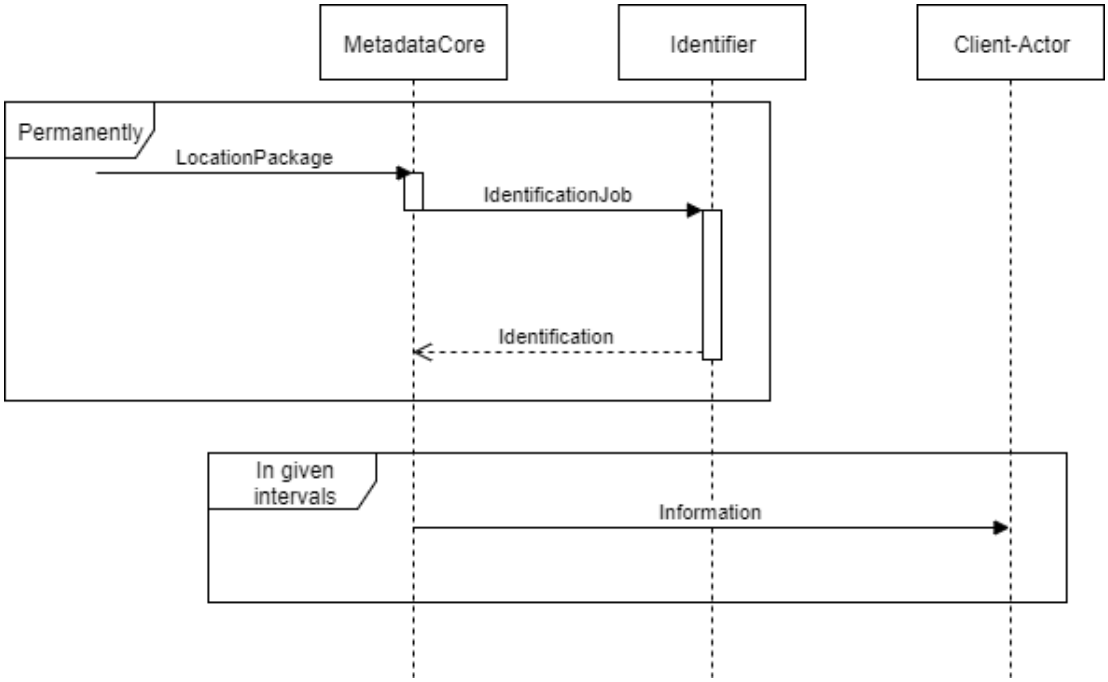

Abbildung 3.5: Ablauf der Identifikation

3.6 Agenten der Koordinatentransformation

In diesem Abschnitt werden die einzelnen Agenten, die an der Koordinatentransformation beteiligt sind genauer durchleuchtet. Die zwischen den Agenten ablaufenden Prozesse werden erklärt und Entscheidungen für bestimmte Verfahren begründet.

3.6.1 Master

Der *Master*-Agent bildet den zentralen Kern für die Kalibrierung und die Koordinatentransformation. Er nimmt Benutzereingaben entgegen, setzt Prozesse in Gang und verwaltet registrierte Sensoren (vgl. 3.6.2) und Transformationen (vgl. 3.6.5). Der Master ist in allen Prozessen beteiligt und wird im Kontext der anderen Agenten erklärt.

3.6.2 SensorAgent

Das in Abschnitt 3.3.4 angesprochene Problem der verschiedenen Sensoren, welche ihre Daten in unterschiedlichen Formaten liefern, wurde durch eine einheitliche Schnittstelle für alle Sensoren gelöst. Diese erfordert das veröffentlichen der Sensordaten in Form eines *SensorData*-Objektes, welches folgende Attribute hat:

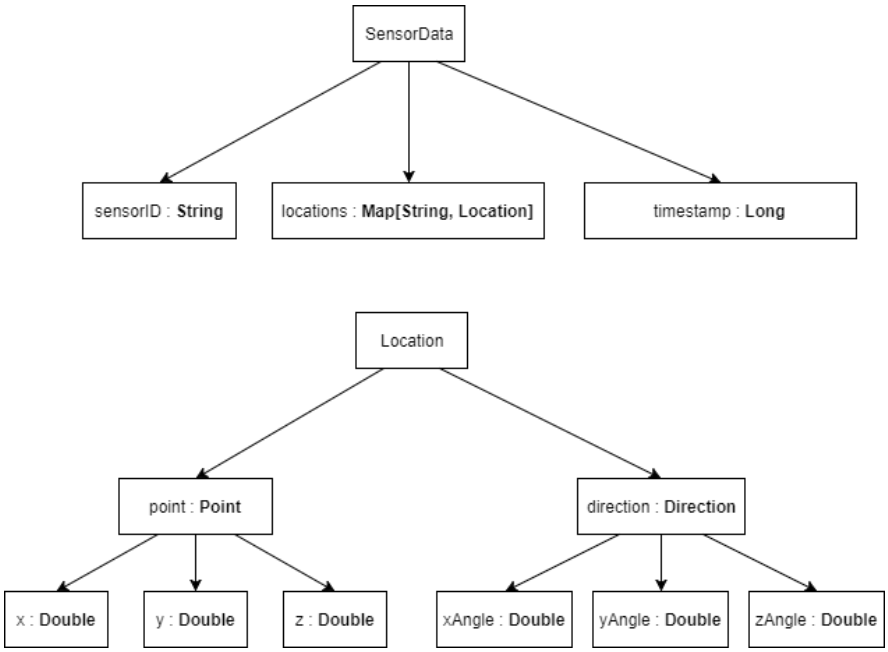

Abbildung 3.6: Aufbau der SensorData-Objekte

Die *sensorID* ist ein Attribut, welches notwendig ist, damit die einzelnen Sensoren eindeutig identifiziert werden können. Ohne dieses Attribut wäre es für die Nutzer der PTP nicht möglich den Ursprung der transformierten Koordinaten nachzuvollziehen. Angenommen es befinden sich zwei Nutzer mit Headtracking im Trackingbereich, dann werden zwar zwei verschiedene Kopf-Koordinaten bereitgestellt, aber der Nutzer wüsste nicht, wie diese zuzuordnen sind. Deswegen muss sich jeder Sensor zu Beginn beim System registrieren.
Die vom Sensor erfassten Daten sind in dem Attribut *locations* enthalten. Diese Map bildet eine ID auf eine *Location* ab. Die ID repräsentiert einen Körperteil, eine Übersicht über alle Körperteile und ihre zugehörigen IDs befindet sich im Anhang (vgl. 6.1). Die *Location* beinhaltet die Koordinaten und die Ausrichtung dieses Körperteils. Diese Schnittstelle muss von allen Sensoren implementiert werden.
Die Sensoren lassen sich in statische und dynamische bzw. mobile Sensoren unterscheiden. Statische Sensoren verändern ihre Position nach der Kalibrierung nicht, während dynamische Sensoren, wie beispielsweise die Leap Motion, nach der Kalibrierung in Bewegung sind. Da für die Koordinatentransformationen der mobilen Sensoren die Veränderung der Position berücksichtigt werden muss, unterscheiden sich die Registrierungen der dynamischen Sensoren von denen der statischen. Die Positionsveränderung muss von einem anderen Sensor erfolgen, deswegen müssen zusätzlich zwei IDs angegeben werden. Eine dieser IDs identifiziert den Sensor und die andere den Körperteil an dem sich der mobile Sensor befindet.

```
1 msg StaticSensor(sensorID : String, bodyPartIDs : Seq[String], isTarget : Boolean) extends Sensor
2 msg DynamicSensor(sensorID : String, bodyPartIDs : Seq[String], attachedToID : String, attachedToPart :
      String) extends Sensor
```

Die Sensor-Objekte mit denen sich beim *Master* registriert wird

3.6.3 StaticCollector

Der *StaticCollector*-Agent ist dafür zuständig, die Sensordaten für die Berechnung der Rotationsmatrizen und der Translationsvektoren für Transformationen statischer Sensoren zu sammeln. Dem *StaticCollector* werden zu Beginn eine Menge von *sensorIDs* und *bodypartIDs* mitgeteilt, nur diese Sensoren und Körperteile werden berücksichtigt.
Das Sammeln erfolgt in folgendem Schema:

1. Der Agent empfängt ein Signal zum Sammeln von Daten. Es öffnet sich ein Zeitfenster, indem alle eingehenden Sensordaten gespeichert werden.

2. Das Zeitfenster schließt sich. Es werden vorerst keine weiteren Daten gespeichert.

3. Die Daten des letzten Zeitfensters werden verarbeitet. Es werden Mittelwerte über die Koordinaten jeder Kombination aus Sensor und Körperteil gebildet. Die Mittelwerte werden als korrespondierende Punkte gespeichert.
4. Der Agent empfängt das nächste Signal zum Sammeln und der Prozess wiederholt sich.
5. Dem Agent wird signalisiert den Prozess zu beenden und die gesammelten korrespondierenden Punkte mitzuteilen.

Damit dieser Prozess funktioniert muss die kalibrierende Person nach Start des Zeitfensters möglichst still halten. Das Berechnen des Mittelwertes über die empfangen Daten löst die Probleme aus Abschnitt 3.3.1 und 3.3.2, da eventuelle Ausbrüche, durch Latenzen und Anomalien, mit der Mittelwertbildung an Relevanz verlieren.

3.6.4 Calculator

Die Berechnungen der Rotationsmatrizen und der Translationsvektoren geschehen im *Calculator*-Agenten. Der *Calculator* bekommt vom *Master* die vom *StaticCollector* oder *DynamicCollector* gesammelten korrespondierenden, Punkte in Form einer Map, welche die *sensorIDs* auf Listen mit den Punkten abbildet.

Die Berechnung der Rotation und der Translation zwischen zwei Datensets geschieht wie folgt:

1. Mittelpunkt aller gesammelten Koordinatenpunkte beider Sensoren finden
2. Beide Koordinatensets mit Hilfe des Mittelpunktes in Richtung Ursprung bringen
3. Die optimale Rotationsmatrix berechnen
4. Die Translation mit Hilfe der Rotationsmatrix und der Mittelpunkte berechnen

(vgl. Ho)

Nach den Berechnungen werden die Rotationen und Translationen, zusammen mit den jeweiligen *sensorIDs* für Quelle und Ziel, in einer Liste an den *Master* gegeben.

3.6.5 StaticTransformer

Empfängt der *Master* die Transformationsvorschiften von dem *Calculator*, dann werden diese einem *StaticTransformer* mitgeteilt. Zu jeder *sensorID* werden die dazugehörigen *sensorIDs* der Zielkoordinatensysteme und die jeweilige Rotation und Translation gespeichert. Die Transformer sind dann dafür zuständig diese Transformationen auf die eingehenden Sensordaten anzuwenden. Dazu wird die *sensorID* der *SensorData*-Nachricht genutzt, um in den gespeicherten Transformationen die Zielkoordinatensysteme zu finden. Für jedes gefundene Zielkoordinatensystem wird dann die Rotation auf die eingegangenen Koordinaten angewandt. Anschließend wird der Translationsvektor addiert.

Letztendlich werden die transformierten Koordinaten über zwei verschiedene Schnittstellen wieder zur Verfügung gestellt. Eine dieser Schnittstellen ist die „Skeleton-API", welche bereits viele Anwendungsfälle im CSTI hat.

```
1 msg Vector3D(x: Double, y: Double, z: Double)
2
3 msg Joint(position: Vector3D, direction: Vector3D, confidence: Double)
4 msg Skeleton(id: Int, joints: Map[String, Joint])
```

Die Skeleton-API

Die Skeleton-API ist eine sinnvolle ausgehende Schnittstelle, da sie bereits von einigen anderen Projekten unterstützt wird. Die Information von welchem Sensor die Daten erfasst wurden und zu welchem Koordinatensystem die Koordinaten gehören, gehen jedoch verloren. Deshalb ist eine weitere Schnittstelle, welche diese Informationen berücksichtigt sinnvoll. Diese erweiterte Schnittstelle wird im folgenden Kapitel beschrieben.

3.7 Schnittstelle der Koordinatentransformation

```
1  //Input
2  msg CalibrateStatic(sensors : Seq[String], bodyparts : Seq[String])
3  msg CalibrateDynamic(sensorID : String, targetID : String, bodyparts : Seq[String])
4  msg Capture()
5  msg GetData()
6  msg Undo()
7  msg Save(filename : Option[String])
8  msg Load(filename : Option[String])
9  msg Register(sensor : Sensor)
10 msg Unregister(sensor : Sensor)
11
12 //Datentypen
13 msg StaticSensor(sensorID : String, bodyPartIDs : Seq[String], isTarget : Boolean) extends Sensor
14 msg DynamicSensor(sensorID : String, bodyPartIDs : Seq[String], attachedToID : String, attachedToPart :
       String) extends Sensor
15
16 msg Location(point : Point, direction : Direction)
17 msg Point(x : Double, y : Double, z : Double)
18 msg Direction(xAngle : Double, yAngle : Double, zAngle : Double)
19
20 //Position eines bestimmten Körperteils
21 msg SingleLocation(source : String, target : String, bodyPartID : String, location : Location)
22
23 //Positionen mehrerer Körperteile einer Person
24 msg LocationPackage(source : String, target : String, locations : Map[String, Location])
```

Die API der Koordinatentransformation

Datentypen

- **StaticSensor**
  StaticSensor ist ein Datentyp, welcher die statischen Sensoren beschreibt. Die *sensorID* identifiziert den Sensor eindeutig im System, während in *bodyPartIDs* IDs zu allen Körperteilen enthalten sind, die getrackt werden können. Über *isTarget* kann angegeben werden, ob dieser Sensor das Zielkoordinatensystem für zukünftige Transformationen ist.

- **DynamicSensor**
  DynamicSensor beschreibt die dynamischen Sensoren. Die *sensorID* und *bodyPartIDs* dienen der gleichen Funktion wie beim *StaticSensor*. Die *attachedToID* gibt an, welcher Sensor die Bewegung des dynamischen Sensors trackt, während *attachedToPart* angibt, an welchem Körperteil dieser Sensor befestigt ist.

- **Location, Point und Direction**
  Location beschreibt die Position eines Körperteils. *Point* gibt mit seinen Koordinaten die Position in Millimetern an, während *Direction* die Ausrichtung anhand von Winkeln mit Werten zwischen null und eins angibt.

Input

- **CalibrateStatic**
  Mit *CalibrateStatic* wird die statische Kalibrierung für die Sensoren mit den IDs aus *sensors* eingeleitet. Berücksichtigt werden die Körperteile, mit den IDs aus *bodyPartID*.

- **CalibrateDynamic**
  Mit *CalibrateDynamic* wird der Sensor mit *sensorID* in das Koordinatensystem des Sensors mit der *targetID* kalibriert. Über *bodyparts* werden die Körperteile angegeben, welche zur Kalibrierung genutzt werden.

- **Capture**
  Capture startet ein Zeitfenster in dem Koordinaten gesammelt werden. Der Mittelwert dieser Koordinaten wird für jeden beteiligten Sensor gespeichert und später für die Berechnung der Rotation und Translation benutzt.

- **GetData**
  GetData beendet das Sammeln von korrespondierenden Koordinatenpunkten. Die gesammelten Punkte werden anschließend verrechnet und die Rotationsmatrizen und Translationsvektoren an die Agenten zur Transformation der Koordinaten gegeben.

- **Undo**
  Undo löscht die zuletzt gespeicherten Koordinaten einer *Capture*-Nachricht. Das kann notwendig sein, sofern sich während eines Zeitfensters bewegt wurde, eine weitere Person in den Trackingbereich gelaufen ist oder ähnliche Probleme aufgetreten sind.

- **Save**
  Save speichert alle aktuellen Transformationsregeln in einer Datei im JSON Format. Der Dateiname kann optional im Feld *filename* angegeben werden.

- **Load**
  Load lädt eine zuvor gespeicherte Datei mit Konfigurationsregeln und stellt diesen Zustand wieder her.

- **Register**
  Register registriert einen Sensor bei der PTP. Die übergebene *sensorID* muss einzigartig sein, sodass der Sensor im System eindeutig identifiziert werden kann.

- **Unregister**
  Unregister hebt die Registrierung eines Sensors auf, sodass dieser im System nicht mehr genutzt wird.

Output

- **SingleLocation**
  SingleLocation gibt mit *location* die Position des Körperteils mit der *bodyPartID* an. Außerdem werden mit *source* und *target* Informationen über das ursprüngliche und das letztendliche Koordinatensystem gegeben.

- **LocationPackage**
  LocationPackage gibt mit *locations* im Vergleich zum *SingleLocation*-Objekt die Positionen mehrerer Körperteile an. Dabei werden in der Map die IDs der Körperteile auf die Positionen abgebildet.

3.8 Agenten der erweiterten Identifizierung

Die Agenten der erweiterten Identifizierung bauen auf den transformierten Koordinaten auf. Durch das einheitliche Koordinatensystem ist es nun möglich weitere Aussagen über die gesammelten Koordinaten zu treffen. Ziel ist es, die Koordinateninformationen so zu erweitern, das Personen langfristig getrackt werden. Die transformierten Koordinaten geben Auskunft darüber, dass zu einem bestimmten Zeitpunkt eine beliebige Person an der Stelle dieser Koordinaten war. Es fehlt aber eine Identifizierung dieser Person über einen Zeitraum. Genau diese Identifizierung soll von den nachfolgenden Agenten zur Verfügung gestellt werden.

3.8.1 Core

Der *Core* ist der zentrale Agent zum Speichern der Metadaten. Er veröffentlicht die Informationen über die identifizierten Personen, ihre Koordinaten und ihre dazugehörigen Metadaten. Dafür kann ein Intervall in der Konfiguration angegeben werden, welche die Publikationsrate bestimmt. Da VR/AR-Anwendungen sehr zeitkritisch sind, empfiehlt es sich einen Intervallwert von unter 33 Millisekunden zu benutzen. Dies würde in etwa 30 Ausgaben pro Sekunde entsprechen. Ein Wert von mindestens 20 Millisekunden, also 50 Ausgaben pro Sekunde, ist demnach anzustreben. Sinnvoll ist dies nur, solange genügend Sensoren zum Tracking aktiv sind. Ist lediglich ein Sensor mit 30 Hz aktiv, so werden die 50 Hz des *Cores* nicht ausgenutzt.

Hinzugefügt werden Metainformationen über die vom *Core* bereitgestellte Schnittstelle. Metainformationen können entweder über die Angabe einer konkreten *ID* oder über die Angabe einer konkreten Position hinzugefügt werden. Wird eine Position angegeben, so werden die gewünschten Metadaten der *ID* hinzugefügt, welche sich am dichtesten an dieser Position befindet.

Die Zuweisung von *IDs* zu konkreten Koordinaten erfolgt über sogenannte *Identifier*-Agenten. Diese werden bei eingehenden Koordinaten gestartet und bekommen Informationen über vorhandene *IDs* und deren Positionen.

3.8.2 Identifier

Die *Identifier* müssen den eingegangen Koordinaten die richtige *ID* zuweisen. Da die *Identifier* eigenständige Agenten sind, können sie mit verschiedenen Algorithmen implementiert werden. Eine einfache Implementierung könnte die neuen Koordinaten derjenigen *ID* zuordnen, deren letzte Koordinaten am dichtesten waren. Bei Überschreitung einer bestimmten Distanz könnte eine neue *ID* zugewiesen werden.
Es wäre aber auch eine Implementierung durch sogenannte Constraints denkbar. Diese würden beispielsweise die maximale Bewegungs- und Drehgeschwindigkeit von Menschen berücksichtigen.

3.9 Schnittstelle der erweiterten Identifizierung

```
1 //Input
2 msg IdEdit(id : String, metadata : Map[String,String])
3 msg LocationEdit(location : LocationPackage, metadata : Map[String,String])
4
5 //Datentyp
6 msg Data(locations : LocationPackage, metadata : Map[String, String])
7
8 //Output
9 msg Information(map : Map[String, Data])
```

Die API der erweiterten Identifizierung

- **IdEdit**
  Mit *IdEdit* werden der Datenstruktur, welche die übergebene *id* hat, die Metadaten hinzugefügt, welche in der *metadata* Map gespeichert sind. Ist ein Schlüssel schon mit einem Wert belegt, so wird dieser mit dem neuen Wert überschrieben.

- **LocationEdit**
  Mit *LocationEdit* wird nicht wie beim *IdEdit* über die *id* ermittelt welche Datenstruktur zu ändern ist, sondern darüber welche der übergebenen *location* am dichtesten ist. Es wird demnach erst geprüft, welche *id* der *location* am dichtesten ist und anhand dieser *id* dann ein *IdEdit* durchgeführt.

- **Data**
  Data umfasst die zu jeder *id* gespeicherten Informationen. Dazu gehört die letzte ermittelte Positon (*location*) und die dazugehörigen Metadaten (*metadata*).

- **Information**
  Information stellt die ausgehende Schnittstelle dar. Die Map speichert zu jeder *id* das dazugehörige *Data*-Objekt. Diese Information wird periodisch veröffentlicht, das Intervall kann in der Konfigurationsdatei eingestellt werden.

3.10 Programmiersprache

Zur Zeit unterstützt die Middleware die Sprachen Java, Scala, C++ und Javascript (vgl. Eichler u. a. 2017). Das ausgearbeitete Design lässt sich in allen genannten Sprachen implementieren, jedoch hat Scala gegenüber den anderen einen großen Vorteil. Scala bietet das Konzept des Pattern Matchings, welches sich wie folgt beschreiben lässt:

> Pattern matching is a mechanism for checking a value against a pattern. A successful match can also deconstruct a value into its constituent parts. It is a more powerful version of the switch statement in Java and it can likewise be used in place of a series of if/else statements.
> Tour of Scala - Pattern Matching

Das Pattern Matching von Scala passt sehr gut zum Konzept von Multiagentensystemen, da sich die kommunizierten Nachrichten der Agenten mit Hilfe des Pattern Matchings übersichtlich verarbeiten lassen. Zu erkennen ist das am besten an einem Beispiel.

3.10.1 Nachrichtenverarbeitung

Anhand der Code-Ausschnitte lässt sich erkennen, dass die Nachrichtenverarbeitung in Scala wesentlich übersichtlicher ist als in Java. In Java muss die eingehende Nachricht erst auf ihren Typ geprüft (instanceof) und anschließend noch zugesichert werden. Das Pattern Matching von Scala kombiniert beides und sorgt somit für eine kompaktere Schreibweise. Die Case Class „MeineCaseClass" soll verdeutlichen, dass dies nicht nur für primitive Datentypen funktioniert. Die Deklaration der Case Class bzw. der Klasse wurde im Java Beispiel weggelassen, da diese deutlich umfangreicher ist als in Scala. Das ist jedoch kein ausschlaggebender Grund für die Nutzung von Scala, da die Klassen von dem im Abschnitt 2.5.3 erwähnten API-Generator erzeugt werden.

```
1 case class MeineCaseClass(feld : String)
2
3 override def receive: Receive = {
4 case msg : Int => println("Die Nachricht ist vom Typ Integer! " + msg)
5 case msg : Boolean => println("Die Nachricht ist vom Typ Boolean! " + msg)
6 case msg : Double => println("Die Nachricht ist vom Typ Double! " + msg)
7 case msg : MeineCaseClass => println("Die Nachricht ist vom Typ MeineCaseClass! " + msg)
8 case msg => println("Die Nachricht entspricht keinem der berücksichtigten Typen! " + msg)
9 }
```

Scala Beispiel

```java
public void onReceive(Object msg) throws Exception {
if(msg instanceof Integer) {
int i = (int) msg;
System.out.println("Die Nachricht ist vom Typ Integer! " + i);
} else if(msg instanceof Boolean) {
boolean b = (boolean) msg;
System.out.println("Die Nachricht ist vom Typ Boolean! " + b);
} else if (msg instanceof Double) {
Double d = (Double) msg;
System.out.println("Die Nachricht ist vom Typ Double! " + d);
} else if (msg instanceof MeineCaseClass) {
MeineCaseClass mcc = (MeineCaseClass) msg;
System.out.println("Die Nachricht ist vom Typ MeineCaseClass");
} else { System.out.println("Die Nachricht entspricht keinem der berücksichtigten Typen! " + msg); }
}
```

Java Beispiel

4 Evaluation

Dieses Kapitel evaluiert die PTP anhand der im Kapitel 2 analysierten Anforderungen. Dazu werden erhobene Messwerte dargestellt und bewertet. Außerdem erfolgt eine Einschätzung darüber, wie die PTP in Zukunft erweitert werden kann.

4.1 Bewertung der Anforderungen

4.1.1 Latenz

Die Tracking-Daten der PTP müssen mit geringer Latenz bereitgestellt werden. Aus der Analyse ergab sich, dass die Tracking-Daten eine Latenz von $50\,\mathrm{ms}$ möglichst nicht überschreiten sollten. Zu beachten ist hierbei, dass neben den Latenzen in der PTP auch noch Latenzen in den Tracking-Systemen und Client-Anwendungen entstehen, auf die kein Einfluss genommen werden kann. Zur Bestimmung der Gesamtlatenz wurde diese aufgeteilt um eventuelle Problembereiche zu identifizieren. Diese setzt sich aus den folgenden Aufgaben zusammen:

1. Transformation der Sensordaten
2. Starten der Identifizierung
3. Identifizierung einer ID anhand der transformierten Koordinaten
4. Aktualisieren des Standortes der entsprechenden ID
5. Summe der Latenzen für Kommunikation (6 Nachrichten über die Middleware)

Die Latenzen der Punkte 1 bis 4 wurden auf verschiedenen Computern gemessen und in Diagrammen veranschaulicht (s. Abbildung 4.1 bis 4.4). Die für das Testszenario relevanten Hardware-Spezifikationen sind in der Tabelle 4.1.1 dargestellt. Die Messungen erfolgten durch den Vergleich der Systemzeit vor und nach Ausführung der relevanten Code-Abschnitte. Das Kreuz markiert den Mittelwert, während die Trennlinie innerhalb des Rechtecks den Median darstellt.

Im Folgenden wird sich auf die Messwerte des Labor-Rechners des CSTI bezogen (s. Abbildung 4.1 bis 4.4, roter Graph). Die Summe der Mittelwerte der gemessenen Latenzen beträgt etwa $1.73\,\mathrm{ms}$. Davon entstanden $1.45\,\mathrm{ms}$ Latenz durch die Transformation der Koordinaten. Vernachlässigt man vereinzelte hohe Ausreißer und betrachtet den Median statt des Mittelwertes, so erhält man sogar einen Gesamtwert von nur $1.43\,\mathrm{ms}$.

Die Kommunikation der Nachrichten verursacht ebenfalls Latenzen. Die Middleware des CSTI ist für Smart Homes und VR-Projekte entwickelt worden und hat selbst bei der gleichzeitigen Benutzung von 10.000 Agenten Latenzzeiten von etwa $5.5\,\mathrm{ms}$ (vgl. Eichler u. a. 2017). Unter der Annahme einer stark ausgelasteten Middleware fallen für die Kommunikation somit maximal $33\,\mathrm{ms}$ ($6 \times 5.5\,\mathrm{ms}$) Latenz an. Insgesamt wird deutlich, dass der entscheidende Faktor durch die Kommunikation gebildet wird. Ein schnelles Netzwerk ist somit auf jeden Fall notwendig. Dennoch liegt die Gesamtlatenz unter der $50\,\mathrm{ms}$ Grenze und bietet noch Spielraum für Verarbeitungszeit auf Seiten der Sensoren und Client-Systeme.

Prozess	**Median (in ms)**	**Mittelwert(in ms)**
Transformation	1,1756425	1,44857294
Starten der Identifizierung	0,1820975	0,18804451
Identifizierung	0,0638045	0,08108838
Aktualisierung	0,010557	0,0114176

Tabelle 4.1: Latenzwerte des Labor-Rechners

Bezeichnung	**Prozessor**	**Arbeitsspeicher**
Gaming PC	Intel Core i7-4790K (4 x 4,0-4,4GHz)	16GB
Labor-Rechner	Intel Core i7-5820K (6 x 3,3-3,6GHz	16GB
Laptop	Intel Core i7-2760QM (4 x 2,4-3,5GHz	8GB

Tabelle 4.2: Hardware-Spezifikationen

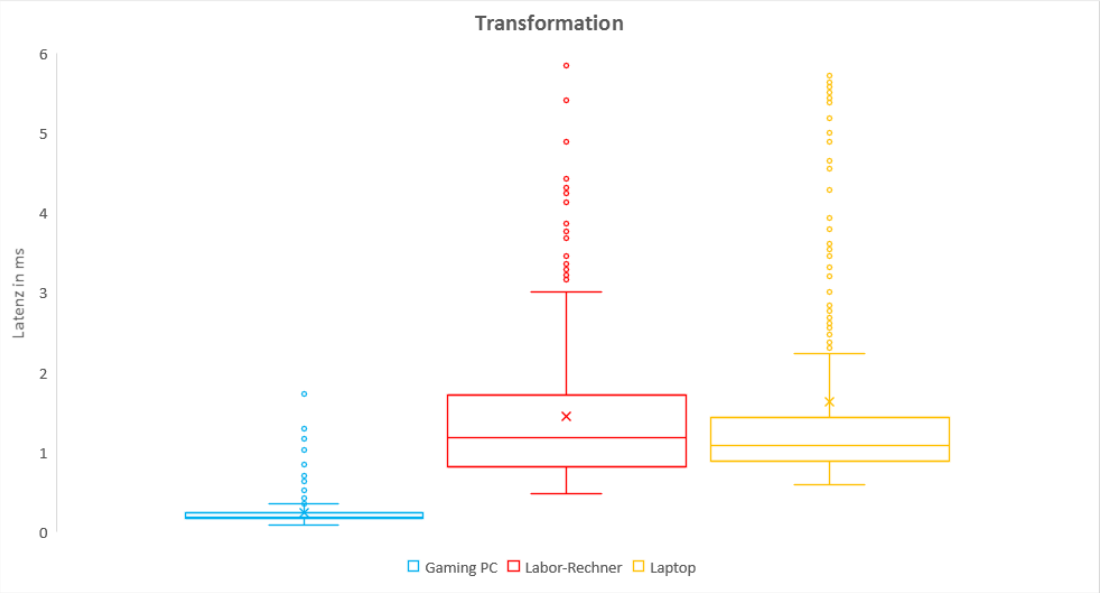

Abbildung 4.1: Gemessene Latenzen während der Transformation

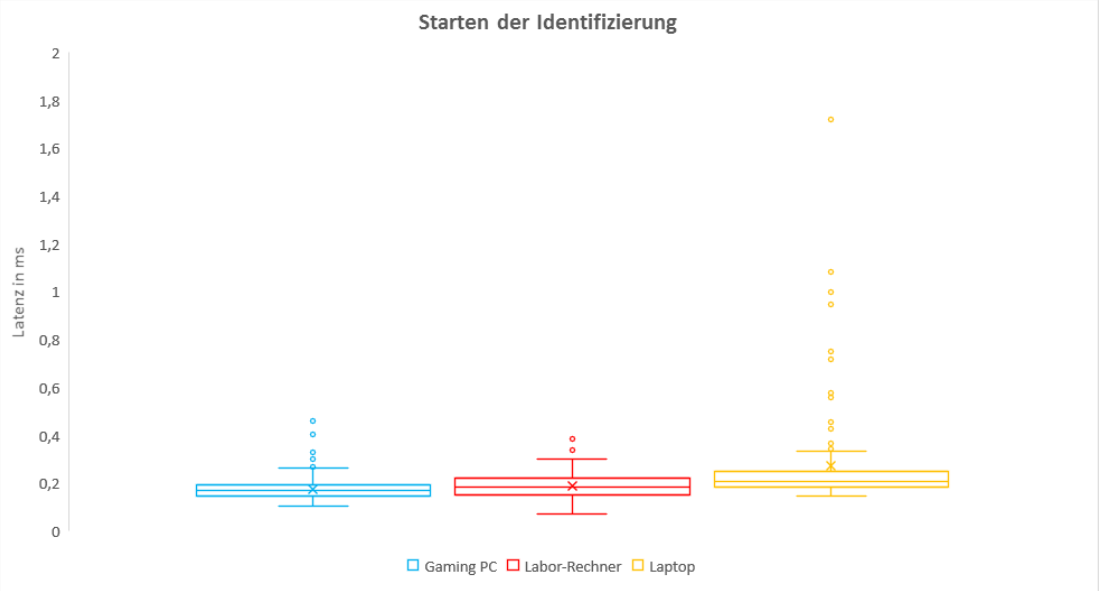

Abbildung 4.2: Gemessene Latenzen während des Starts der Identifizierung

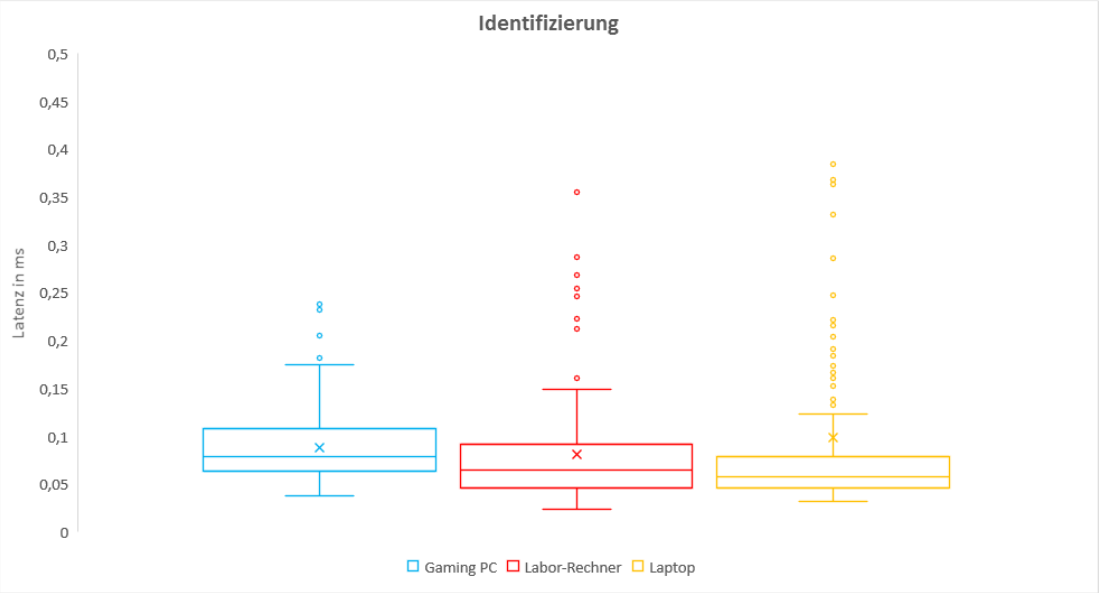

Abbildung 4.3: Gemessene Latenzen während der Identifizierung

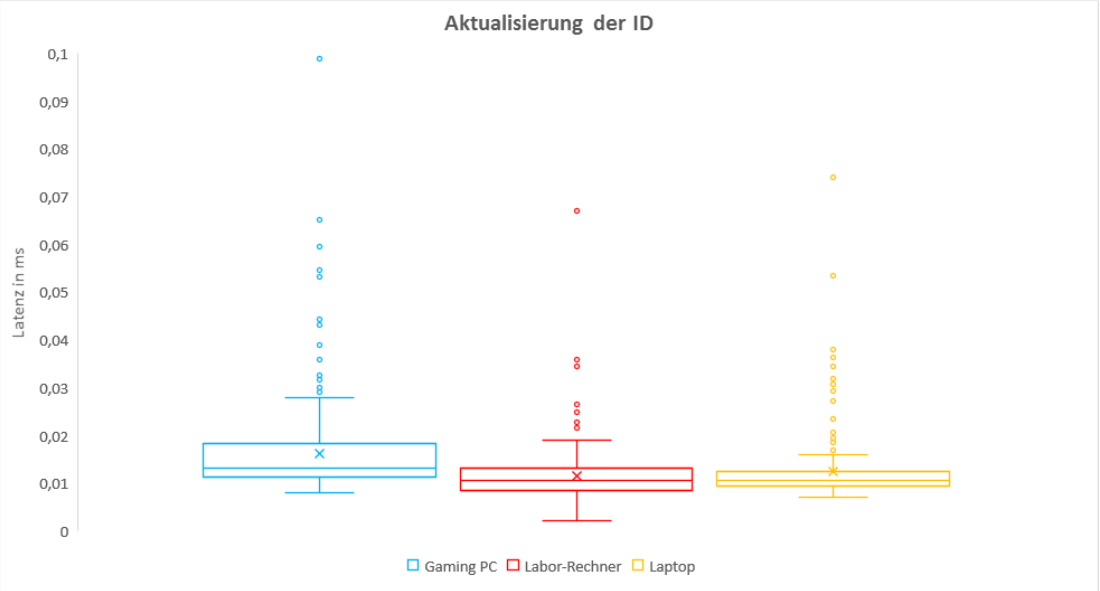

Abbildung 4.4: Gemessene Latenzen während der Aktualisierung

4.1.2 Präzision

Die Präzision der PTP ist ein maßgeblicher Faktor für die Brauchbarkeit der Plattform. Weichen die transformierten Koordinaten zu stark von den tatsächlichen ab, so wird die Interaktion in VR/AR schwierig. Um die Präzision der PTP zu bestimmen wurden zwei Testszenarien aufgebaut. Diese werden im Folgenden beschrieben und ausgewertet.

Im ersten Testszenario wurden die Koordinaten der Kinect V2 in das Koordinatensystem des ART überführt. Dazu wurde eine Kalibrierung mit 20 korrespondierenden Koordinatenpunkten durchgeführt. Die daraus berechnete Rotation und Translation wurden für die Transformation der Kinect Koordinaten genutzt. Die Präzision wurde bestimmt, indem die Kinect Koordinaten nach der Transformation mit den zuletzt eingetroffenen ART-Koordinaten verglichen wurden. Die Distanz zwischen den Punkten wurde dann als Messwert gespeichert und im folgenden Diagramm aufbereitet.

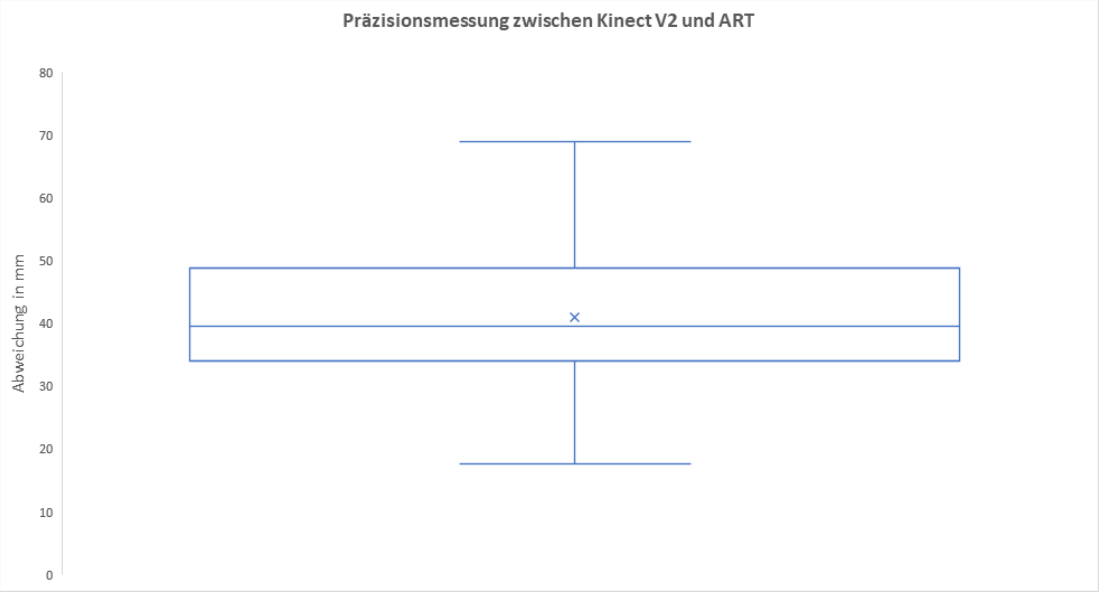

Abbildung 4.5: Gemessene Abweichung zwischen Kinect V2 und ART

Aus der Abbildung 4.5 geht hervor, dass die mittlere Abweichung bei etwa $4\,\mathrm{cm}$ liegt. Dieser Wert erfüllt die Anforderungen, dabei ist jedoch zu beachten, dass die Messung der Präzision und die Kalibrierung zwischen den beiden Systemen schwierig ist. Das ART trackt lediglich die in Abschnitt 2.2.2 vorgestellten Targets, während die Kinect V2 den Körper selbst trackt. Misst

man also die Koordinaten der rechten Hand und hält das Target anders als in der Kalibrierung, so entstehen bereits Abweichungen von wenigen Zentimetern.
Im zweiten Szenario wurde das ART also durch eine zweite Kinect V2 ausgetauscht. An dem Vorgehen hat sich nichts geändert. Mit einer Abweichung von etwa 3 cm fiel diese wie erwartet etwas geringer aus.

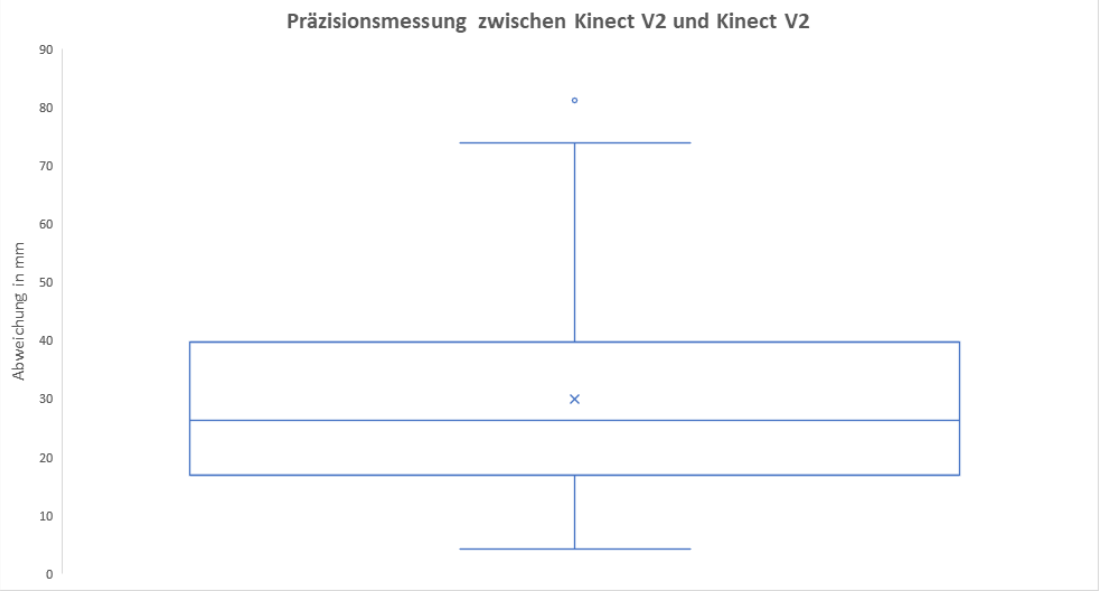

Abbildung 4.6: Gemessene Abweichung zwischen Kinect V2 und Kinect V2

4.1.3 Skalierbarkeit

Die Forderung an die PTP skalierbar zu sein bedeutet, dass das hinzufügen weiterer Sensoren die Latenz des Systems nicht maßgeblich erhöht. Das Hinzufügen weiterer Sensoren bedeutet, dass mehr Koordinatentransformationen durchgeführt werden müssen. Da diese Aufgabe in eigenständigen Agenten gekapselt ist, können bei Bedarf einfach weitere Instanzen der Agenten auf weiteren Rechnern ausgeführt werden. Als Folge steigt auch die Auslastung bei der Identifikation. Der Identifizierungs-Algorithmus ist in dem *Identifier*-Agenten gekapselt, so dass auch hier weitere Instanzen gestartet werden könnten. Da die Ergebnisse der *Identifier*-Agenten jedoch zentral in einer einzelnen Instanz eines *Core*-Agenten zusammengeführt werden entsteht hier ein Flaschenhals. Der hier anfallende Rechenaufwand ist jedoch eher gering. Dennoch sollte hierfür eine Lösung gefunden werden, welche die Auslastung weiter aufteilt.

4.1.4 Erweiterbarkeit

Die PTP sollte um neue Sensoren erweiterbar sein, damit sie nicht veraltet ist, sobald die verwendeten Tracking-Systeme von neuen Technologien überholt sind. Diese Erweiterbarkeit ist durch die *SensorAgent*-Schnittstelle gewährleistet. Für neue Sensoren müssten Adapter geschrieben werden, welche die Sensordaten auf die *SensorData*-Nachricht abbilden (vgl. Abbildung 3.1).
Außerdem sollte es möglich sein die PTP um Erweiterungen wie beispielsweise eine Gesichtserkennung zu erweitern. Diese Option wird durch die Schnittstelle aus Abschnitt 3.9 gewährleistet. Die Realisierung könnte wie folgt aussehen:

1. Starten der Gesichtserkennung
2. Lokalisierung der Kamera (manuell oder automatisch)
3. Erkennen eines Gesichts
4. Mitteilen der zu dem Gesicht gespeicherten Daten an die PTP (per *LocationEdit*-Nachricht)
5. PTP: identifiziert die dichteste ID zu der übergebenen Position und passt die Metadaten an

4.2 Zukunft

Diese Arbeit behandelt das Design der PTP auf einem Basisniveau. Im folgenden werden weitere Anknüpfungspunkte für zukünftige Projekte thematisiert.

4.2.1 Wartefenster für transformierte Koordinaten

Im aktuellen Design werden die einzelnen transformierten Koordinaten einer Person zugeordnet und der Standort für die Person dann aktualisiert. Liefern mehrere Sensoren Daten zu einer gleichen Person, so könnte man diese nach der Transformation miteinander verrechnen, so dass die Genauigkeit der Koordinaten steigt.

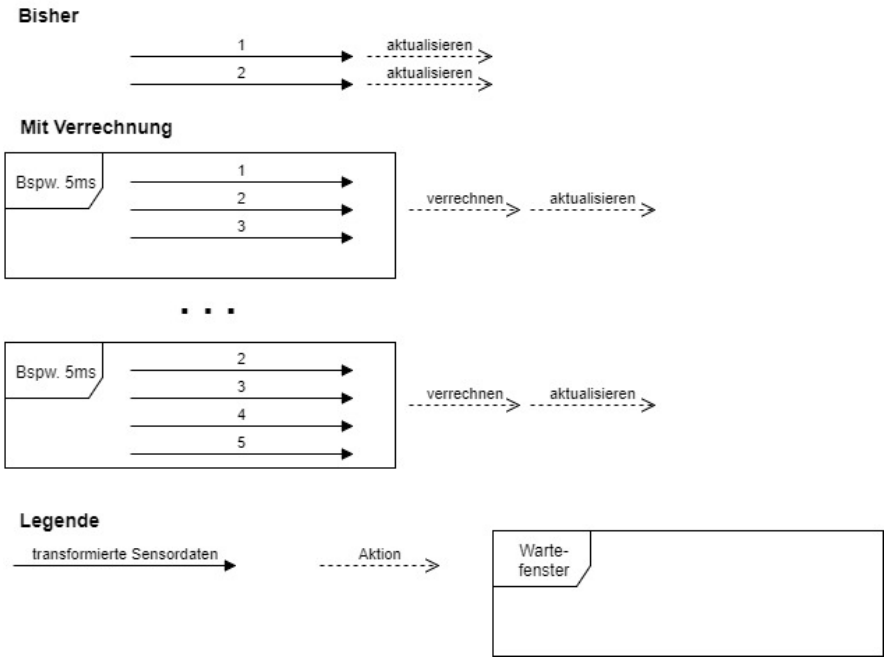

Abbildung 4.7: Darstellung des Wartefensters

Das würde bedeuten, dass die Koordinaten nicht sofort berücksichtigt werden, sondern auf weitere potenzielle Koordinaten zur Verrechnung gewartet wird. Demnach entsteht eine weitere Latenz die der Wartedauer zuzüglich der Verrechnungsdauer entspricht. Bei der Verrechnung müssten die einzelnen Koordinaten nicht in gleichem Maße gewichtet werden. Ob sich das Wartefenster lohnt, welche Gewichtungen sinnvoll sind und wie es umzusetzen ist, könnte in einem weiteren Projekt erforscht werden.

4.2.2 Identifizierungsalgorithmen

Die Identifizierung der Personen anhand der transformierten Sensordaten kann auf verschiedene Weisen erfolgen. Dabei müssen unterschiedliche Aspekte beachtet werden. Beispielsweise sollen Personen die dicht beieinander stehen nicht vertauscht werden, Wie wird damit umgegangen wenn ein Sensor kurzzeitig ausfällt und die Person nicht getrackt wird, wird sie dann wiedererkannt oder bekommt sie eine neue ID? Um möglichst genaue Identifizierungen zu machen könnten verschiedene Algorithmen implementiert und getestet werden. Eine Kombination verschiedener Algorithmen wäre ebenfalls denkbar. Aufgrund des Designs lässt sich der *Identifier*-Agent hierfür einfach austauschen.

4.2.3 Transitivität von Transformationen

Eine weitere nützliche Funktionalität wäre eine Unterstützung von Transitivität. Das bedeutet, dass wenn die Sensordaten eines Sensors A in das Koordinatensystem eines Sensors B überführbar sind und das gleiche für die Daten von B nach C gilt, automatisch auch eine Überführung von A nach C existiert.

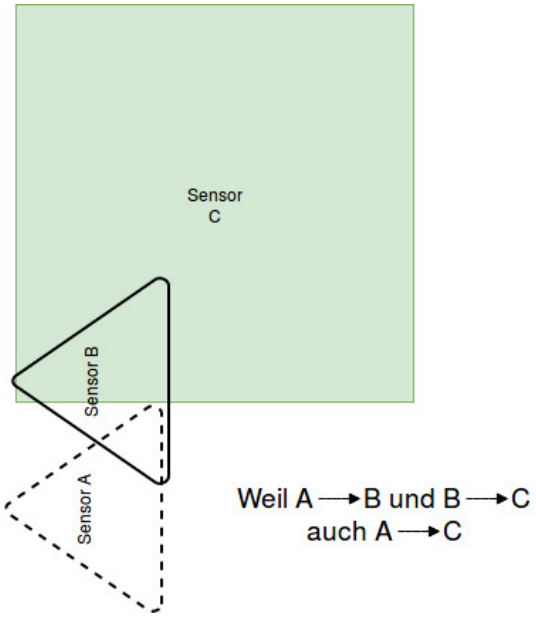

Abbildung 4.8: Darstellung der Transitivität

Damit wäre es eine Transformation von Koordinaten möglich, obwohl kein überlappender Tracking-Bereich vorliegt und somit keine Möglichkeit besteht diese Systeme miteinander zu kalibrieren.

5 Schluss

5.1 Zusammenfassung

In dieser Arbeit wurde eine Plattform für Personentracking entworfen, welche Entwickler von VR- und AR-Anwendungen unterstützen soll. Die Plattform transformiert die Sensordaten verschiedener Sensoren mit unterschiedlichen Koordinatensystemen in ein einzelnes Koordinatensystem und stellt diese über eine einheitliche Schnittstelle zur Verfügung.

In der Analyse (s. Kapitel 2) wurden Anforderungen identifiziert, die von der Plattform erfüllt werden sollten. VR- und AR-Anwendungen sind sehr zeitkritisch, weshalb eine Latenz von $50\,\mathrm{ms}$ nicht überschritten werden sollte. Damit die Interaktion in der virtuellen Welt reibungslos abläuft müssen auch die Koordinaten möglichst genau transformiert werden. Deswegen wurde eine Präzision von wenigen Zentimetern gefordert. Skalierbarkeit soll dafür sorgen, dass die Performance des Systems nicht sinkt, wenn weitere Sensoren hinzugefügt werden. Damit die Plattform auch mit neuen Tracking-Systemen genutzt werden kann, soll sie um diese erweiterbar sein.

Die Plattform und dazugehörige Schnittstellen wurden anhand der ermittelten Anforderungen entworfen und in Kapitel 3 vorgestellt. Es wurde sich dafür entschieden die Relation der Sensoren über einen halbautomatischen Kalibrierungsprozess herzustellen. Dafür werden auf Signal eines Anwenders korrespondierende Koordinatenpunkte der Sensoren gesammelt und genutzt um daraus Rotationsmatrizen und Translationsvektoren zur berechnen, die zur Transformation zukünftiger Sensordaten genutzt werden können. Um die Koordinaten für die Darstellung von Avataren oder die Interaktion zu nutzen, müssen diese erst Personen zugeordnet werden. Damit diese Aufgabe nicht in jeder Client-Anwendung geschehen muss, ermittelt das System aus den transformierten Koordinaten die getrackten Personen und stellt die Koordinaten zusammen mit Identifikationsnummern für die einzelnen Personen bereit.

Die Evaluation (s. Kapitel 4) erfolgte auf Basis der Analyse identifizierten Anforderungen. Für die Latenz und Präzision wurden Messwerte in Testszenarien aufgenommen und anschließend aufbereitet und ausgewertet. Mit einer Latenz von maximal 33 Millisekunden und einer Präzision von etwa drei bis vier Zentimetern erfüllt die Plattform diese Anforderungen. Die Erweiterbarkeit wird sichergestellt, indem neue Sensoren über Adapter in das bestehende System eingegliedert werden können. Skalierbar ist die Plattform, weil sie als Multiagentensystem entworfen wurde und es somit möglich ist, die Auslastung auf mehrere Instanzen der Agenten zu verteilen. Zuletzt wurden Anknüpfungspunkte für zukünftige Projekte vorgestellt, die Potenzial für weitere Forschung bieten.

5.2 Ausblick

Im Jahr 2016 brachten HTC, Oculus und Sony ihre Head-Mounted Displays auf den Verbrauchermarkt. Während diese bislang zum Großteil im Entertainmentbereich genutzt werden, finden sich immer mehr Anwendungsgebiete für VR und AR in anderen Bereichen.
In der Medizin können beispielsweise Angstzustände von Patienten behandelt werden, während neue Fachkräfte an virtuellen Patienten ausgebildet werden. Aber auch in anderen Bereichen können VR und AR zu Bildungszwecken genutzt werden. Die Möglichkeit Objekte ohne Gefahr oder Konsequenz dreidimensional zu erkunden bietet viele Vorteile. Reparaturen können geübt werden ohne, dass- dabei etwas kaputt geht und die Effekte eines Herzdefektes können anschaulich visualisiert werden. Die Stärke von VR und AR liegt allgemein darin Dinge zu visualisieren und erkundbar zu machen. In der Industrie können diese Aspekte genutzt werden um Produkte dreidimensional zu modellieren.

Insgesamt lässt sich feststellen, dass VR und AR ein großes Spektrum von Anwendungsgebieten findet. Noch haben sie sich aber nicht vollständig etabliert. Zu erwarten sind Technologien, welche haptisches Feedback bei Interaktionen mit der virtuellen Welt geben. Die Interaktion im Allgemeinen bietet noch Luft nach oben. Entwickelte Controller bieten nicht die gleiche Immersion wie die Interaktion mit den eigenen Händen, während letztere noch nicht optimiert und oft sehr grob umgesetzt ist. Auch das Übertragen von Personen in virtuelle Welten ist noch nicht vollständig ausgereift. Hier könnte eine Plattform wie die in dieser Arbeit vorgestellten PTP helfen. Erkenntnisse wie die des Redirected Walking sorgen jedoch für stetige Besserung.

Betrachtet man die Entwicklung von VR und AR seit 2016, so machen die Fortschritte Hoffnung auf eine erfolgreiche Durchsetzung. Mit der Entwicklung von neuen Displays, Interaktions-

geräten, Tracking-Systemen und neuer Software lassen sich vermutlich schon bald weitere Projekte in den Alltag eingliedern. Ein neues Projekt im CSTI arbeitet daran Möbel in Form einer Augmented Reality Anwendung im Raum anzuzeigen. Damit könnten IKEA-Möbel in Zukunft direkt in den eigenen vier Wänden mit der dazugehörigen Einrichtung betrachtet und ausgesucht werden. Ob VR und AR sich allerdings so stark im Alltag verankern wie der Computer oder das Smartphone bleibt abzuwarten. Auszuschließen ist es aber in keinem Fall.

6 Anhang

6.1 Körperteile und ihre Identifikationsnummern

ID	Körperteil
0	Spine Base
1	Spine Mid
2	Neck
3	Head
4	Shoulder Left
5	Elbow Left
6	Wrist Left
7	Hand Left
8	Shoulder Right
9	Elbow Right
10	Wrist Right
11	Hand Right
12	Hip Left
13	Knee Left
14	Ankle Left
15	Foot Left
16	Hip Right
17	Knee Right
18	Ankle Right
19	Foot Right
20	Spine Shoulder
21	Hand Tip Left
22	Thumb Left
23	Hand Tip Right
24	Thumb Right

Tabelle 6.1: Körperteile und ihre Identifikationsnummern

Literaturverzeichnis

[Advanced Realtime Tracking GmbH a] Advanced Realtime Tracking GmbH: *AR-Tracking*. – URL http://www.ar-tracking.com/products/markers-targets/. – Zugriffsdatum: 19.07.2017

[Advanced Realtime Tracking GmbH b] Advanced Realtime Tracking GmbH: *Hand & Tree Targets*. – URL http://www.ar-tracking.com/products/markers-targets/targets/passive/. – Zugriffsdatum: 19.07.2017

[CSTI a] CSTI: *About CSTI*. – URL https://csti.haw-hamburg.de/about/. – Zugriffsdatum: 25.07.2017

[CSTI b] CSTI: *Das //CSTI*. – URL https://csti.haw-hamburg.de/. – Zugriffsdatum: 04.11.2017

[Dörner u. a. 2014] Dörner, Ralf ; Broll, Wolfgang ; Grimm, Paul ; Jung, Bernhard (. ; Dörner, Ralf ; Broll, Wolfgang ; Grimm, Paul ; Jung, Bernhard: *Virtual und Augmented Reality (VR / AR) - Grundlagen und Methoden der Virtuellen und Augmentierten Realität*. 1. Aufl. Berlin Heidelberg New York : Springer-Verlag, 2014. – ISBN 978-3-642-28903-3

[Eichler 2014] Eichler, Tobias: *Agentenbasierte Middleware zur Entwicklerunterstützung in einem Smart-Home-Labor*, Hochschule für Angewandte Wissenschaften Hamburg, Masterarbeit, Oktober 2014

[Eichler u. a. 2017] Eichler, Tobias ; Draheim, Susanne ; Grecos, Christos ; Wang, Qi ; von Luck, Kai: Scalable Context-Aware Development Infrastructure for Interactive Systems in Smart Environments. In: *Fifth International Workshop on Pervasive and Context-Aware Middleware 2017 (PerCAM'17)*. Rome, Italy, Oktober 2017

[Fuchs 2017] Fuchs, Philippe: *Virtual Reality Headsets - A Theoretical and Pragmatic Approach*. 1. Aufl. Boca Raton, Fla : CRC Press, 2017. – ISBN 978-1-351-80307-6

[Gerwens 2017] Gerwens, Niklas: *Virtual Handshake - Interaktion in Multi-User VR-Systemen*, Hochschule für Angewandte Wissenschaften Hamburg, Bachelorarbeit, 2017

[Google Inc.] Google Inc.: *„virtual reality" Suchtrend.* – URL https://trends.google.de/trends/explore?date=today%205-y&q=virtual%20reality. – Zugriffsdatum: 20.07.2017

[Ho] Ho, Nghia: *FINDING OPTIMAL ROTATION AND TRANSLATION BETWEEN CORRESPONDING 3D POINTS.* – URL http://nghiaho.com/?page_id=671. – Zugriffsdatum: 16.08.2017

[Jeworutzki] Jeworutzki, André: *Raumplan des CSTI.* – URL https://csti.haw-hamburg.de/projekt/. – Zugriffsdatum: 25.07.2017

[Leap Motion, Inc a] Leap Motion, Inc: *Leap Motion.* – URL https://www.leapmotion.com/. – Zugriffsdatum: 18.07.2017

[Leap Motion, Inc b] Leap Motion, Inc: *Leap Motion und Halterung.* – URL https://www.leapmotion.com/#112. – Zugriffsdatum: 18.07.2017

[Melles 2017] Melles, Gerald: privates Interview. 2017. – geführt am 18.10 im CSTI

[Microsoft] Microsoft: *Joints.* – URL https://msdn.microsoft.com/de-de/library/windowspreview.kinect.jointtype.aspx. – Zugriffsdatum: 19.07.2017

[Microsoft Corporation] Microsoft Corporation: *Kinect.* – URL https://developer.microsoft.com/de-de/windows/kinect/hardware. – Zugriffsdatum: 19.07.2017

[Moore 2006a] Moore, G. E.: Cramming more components onto integrated circuits, Reprinted from Electronics, volume 38, number 8, April 19, 1965, pp.114 ff. In: *IEEE Solid-State Circuits Society Newsletter* 11 (2006), Sept, Nr. 5, S. 33–35. – ISSN 1098-4232

[Moore 2006b] Moore, G. E.: Progress in digital integrated electronics [Technical literaiture, Copyright 1975 IEEE. Reprinted, with permission. Technical Digest. International Electron Devices Meeting, IEEE, 1975, pp. 11-13.]. In: *IEEE Solid-State Circuits Society Newsletter* 20 (2006), Sept, Nr. 3, S. 36–37. – ISSN 1098-4232

[pmoews] pmoews: *Garden Turtle.* – URL https://www.thingiverse.com/thing:421809/#files. – Zugriffsdatum: 02.12.2017

[Steinicke u. a. 2010] Steinicke, F. ; Bruder, G. ; Jerald, J. ; Frenz, H. ; Lappe, M.: Estimation of Detection Thresholds for Redirected Walking Techniques. In: *IEEE Transactions on Visualization and Computer Graphics* 16 (2010), Jan, Nr. 1, S. 17–27. – ISSN 1077-2626

[Sutherland 1968] Sutherland, Ivan E.: A Head-mounted Three Dimensional Display. In: *Proceedings of the December 9-11, 1968, Fall Joint Computer Conference, Part I.* New York, NY, USA : ACM, 1968 (AFIPS '68 (Fall, part I)), S. 757–764. – URL http://doi.acm.org/10.1145/1476589.1476686

[Tour of Scala - Pattern Matching] : *TOUR OF SCALA - PATTERN MATCHING*. – URL https://docs.scala-lang.org/tour/pattern-matching.html. – Zugriffsdatum: 14.09.2017

[Wooldridge 2002] Wooldridge, Michael: *An Introduction to MultiAgent Systems*. 1. Aufl. John Wiley & Sons, 2002. – ISBN 047149691X

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

Hamburg, 04.12.2017 Marc Kirchner