Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Christian Blank

Intuitive kontaktfreie Interaktion in Virtual und Mixed Reality

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Christian Blank

Intuitive kontaktfreie Interaktion in Virtual und Mixed Reality

Masterarbeit eingereicht im Rahmen der Masterprüfung

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

Betreuender Prüfer: Prof. Dr. Wendholt
Zweitgutachter: Prof. Dr. habil. Kletschkowski

Eingereicht am: 3. August 2016

Christian Blank

Thema der Arbeit
Intuitive kontaktfreie Interaktion in Virtual und Mixed Reality

Stichworte
Gestenerkennung, Direktmanipulative Interaktion, Interpretation von Gesten, Sensorabstraktion

Kurzzusammenfassung
Die vorliegende Masterarbeit untersucht die Frage, ob eine Kombination von direktmanipulativer Interaktion mit einer Interpretation von Gesten einen Vorteil für den Nutzer darstellt. Es wurde ein Konzept erarbeitet, dass eine Untersuchung dieser Frage ermöglicht. Für eine Evaluierung wurde dieses Konzept prototypisch umgesetzt und mithilfe von Probanden in mehreren Tests und einer Umfrage untersucht. Das Ergebnis zeigt, dass eine Mensch-Computer-Schnittstelle, die auf direktmanipulativer Interaktion basiert, von einer Erweiterung durch eine Gestenerkennung profitieren kann.
Christian Blank

Title of the paper
Intuitive contact free interaction in virtual and mixed reality

Keywords
Gesture Recognition, Direct Manipulation Interaction, Gesture Interpretation, Device Abstraction

Abstract
This master thesis examines the question whether a combination of direct manipulative interaction with an interpretation of gestures is an advantage for the user. We have developed a concept to allow a study of this question. For an evaluation this concept was implemented prototypical. We used volunteers in several tests and a survey to examine the question. The results show that a human-computer interface which is based on direct manipulative interaction, can benefit from an extension by a gesture recognition.

Inhaltsverzeichnis

Danksagung

An dieser Stelle möchte ich die Gelegenheit nutzen, um einigen Personen zu danken, die mich während meines gesamten Studiums und speziell bei dieser Arbeit unterstützt haben.

Mein besonderer Dank gilt Frau Prof. Dr.-Ing. Birgit Wendholt, die mich bei der Anfertigung dieser Masterarbeit betreut hat. Außerdem möchte ich mich bei Herrn Prof. Dr.-Ing. habil. Thomas Kletschkowski für die Bereitschaft zur Erstellung des Zweitgutachtens bedanken.

Ein großes Dankeschön auch an die Projektgruppe I^2E für die langen, konstruktiven Diskussionen und die Unterstützung, während der Anfertigung dieser Arbeit. Nicht zuletzt gebührt meiner Familie und meiner Verlobten Dank. Sie haben mich nicht nur bei der Korrektur unterstützt, sondern standen mir auch persönlich zur Seite, wann immer ich ihre Hilfe benötigt habe.

1 Einleitung

Das Thema kontaktfreie Interaktion ist mit seinen Schwerpunkten in 3D User Interfaces sowie Virtual und Augmented Reality[1] einzuordnen. Für eine genauere Definition sei hier auf Abschnitt 2.1 verwiesen.

1.1 Motivation

VR / AR wurden bereits in den 1960er Jahren erforscht und hatten in den 1990er Jahren einen weiteren Boom [MK94, Rhe91, CNSD93]. 20 Jahre später ist das Thema wieder aktuell. Durch immer bessere Verfahren zur Darstellung und höhere Rechenleistung kann der Nutzer immer weiter in die virtuelle Welt eintauchen.

Mit der zunehmenden Verbreitung von 360°-Kameras wird die Generierung von Content immer einfacher. Mithilfe dieser Kameras kann die Welt so aufgenommen werden, dass der Nutzer selbst entscheiden kann, welchen Teil der Umgebung er sich gerade anschauen möchte. Die Einführung von Google Cardboard und vergleichbaren Produkten anderer Hersteller wirkt als zusätzlicher Katalysator. So gibt es auf Youtube bereits viele Videos, die eine vollständige Umgebung zeigen. Neben dem Abspielen von Videosequenzen, die keine Tiefeninformationen enthalten, wird intensiv an der Digitalisierung von Objekten, Räumen und ganzen Gebieten gearbeitet. Dies geschiet durch Fotogrametrie (Realities.io[2]) und Depth Image Fusion (MobileFusion[OKI15], 3D Modeling on the Go[SSHP15] und DynamicFusion[NFS15]).

Sowohl die Ausgabe über Brillen als auch die Eingabe über Sensoren werden immer mobiler. Im Bereich der Ausgabe sind mit Google Glass[3], Epson Moverio[4], Cardboard-Systemen[5], Meta Glasses[6] usw. viele unterschiedliche Ansätze vertreten. Im Gegensatz zu ihren Vorgängern vor 20 Jahren sind sie nicht mehr so groß wie ein Motoradhelm, sondern passen bequem in die Tasche. Auch die Eingabesensorik wird immer kleiner und mobiler. So hat Google mit ihrem

[1] kurz VR und AR
[2] http://realities.io/
[3] https://developers.google.com/glass/
[4] http://www.epson.com/MoverioBT200
[5] https://vr.google.com/cardboard/
[6] https://www.metavision.com/

Project Soli[7] einen Radarchip in Miniaturausführung entwickelt, der Fingerbewegungen sehr gut erkennen kann. Ebenso gibt es Fortschritte im Bereich der mobilen Tiefenbildkameras. Es ist möglich, eine Leap Motion[8] oder einen Structure Sensor[9] direkt an einem Smartphone zu betreiben[Ebe15]. Ebenso gibt es Ansätze von Ringen[10] und Armbändern, wie etwa das Myo[11] von Thalmic. Diese Technologien ermöglichen einen Fokus auf mobile Konzepte für dreidimensionale Userinterfaces und führen weg von statischen Aufbauten, wie etwa Cave-Systemen oder ähnlichem.

In den vergangenen Jahren wurde ein großes Augenmerk auf die technische Entwicklung einzelner Komponenten und auch ganzer Systeme gerichtet. Jedoch wurde zum Teil vernachlässigt, dass der für den Nutzer wichtigste Part eines Systems die Anwendung ist, die er sieht. Viele Entwicklungen sind nur als Demonstration der Technologie gedacht und nicht auf die eigentlichen Bedürfnisse der Nutzer angepasst. Frühe Adaption durch die Gamingindustrie (Minecraft VR[12]) und verschiedene Künstler und Designer (Tilt Brush[13]) zeigen erste, umfangreichere Anwendungen. Mit der Industrie 4.0 beginnen nun andere Unternehmen aus Wirtschaft und Industrie spezielle Lösungen für AR / VR zu entwickeln. So entwickeln Audi und VW eine Unterstützung bei der Montage und Reparatur von Kraftfahrzeugen. Das Frauenhofer Institut hat mit dem Elbe Dom[14] ein stationäres Großprojektionssystem entwickelt. Zur Interaktion mit den genannten Anwendungen werden Controller benötigt, die der Nutzer in der Hand hält oder am Körper trägt. Eine Weiterentwicklung dieses Interaktionskonzeptes ist die kontaktfreie, dreidimensionale Interaktion.

Eine kontaktfreie, dreidimensionale Interaktion ermöglicht es dem Nutzer ohne Umwege in eine virtuelle Welt einzutauchen und befreit ihn von der Beschränkungen herkömmlicher Interaktionskonzepte, die zumeist statisch sind. Herkömmliche Interaktionselemente sind in aller Regel zweidimensional (Maus, Tastatur, Touchscreen). Die Arbeit mit dreidimensionalen Information über ein zweidimensionales Interaktionselement wirkt für den Nutzer nicht intuitiv und erfordert zusätzliche kognitive Leistung, um die zweidimensionale Aktion in eine dreidimensionale Aktion zu übertragen. Daher haben Forscher Aufbauten vorgeschlagen, in denen dreidimensionale Interaktionen möglich sind. In [HKI+12, WLK+14] werden kontaktfreie Interaktionen genutzt, um dreidimensionale Inhalte (virtuelle Objekte) zu manipulieren.

[7] https://atap.google.com/soli/
[8] https://www.leapmotion.com/
[9] http://structure.io/
[10] http://logbar.jp/ring/en/
[11] https://www.myo.com/
[12] https://www2.oculus.com/experiences/gear-vr/1046887318709554/
[13] http://www.tiltbrush.com/
[14] http://www.iff.fraunhofer.de/de/ueber-fraunhofer-iff/labore/elbe-dom.html

Sowohl HoloDesk als auch MixFab sind dabei Vertreter für feste Installationen. Ein Nutzer muss sich vor einen festen Aufbau stellen und agiert in einem kleinen Bereich hinter einer Sichtfläche, auf der die Szene visualisiert wird. Im Gegensatz dazu finden Entwicklungen statt (RoomAlive[JSM+14]), die eine kontaktfreie Interaktion ermöglichen und den Nutzer trotzdem erlauben, sich möglichst frei im Raum zu bewegen. Der Nutzer ist somit mobil.

Die Forschungsgruppe I^2E arbeitet in diesem schnelllebigen Umfeld und forscht in unterschiedlichen Teilprojekten an möglichen Lösungen für aktuelle Fragestellungen der Forschung und Industrie. Es wird an einer Lösung gearbeitet, in der Nutzer frei und mobil um ein virtuelles Objekt gehen können. Die Arbeitsgruppe hat sich dabei vor allem auf ingenieursnahe Domainen fokussiert und forscht zusammen mit Maschinenbauern, Fahrzeug- und Flugzeugtechnikern und Umwelttechnikern an verschiedenen Lösungen zur Kombination von realen und virtuellen Elementen in einer gemeinsamen Umgebung. Zusätzlich wird daran gearbeitet, eine Möglichkeit zu schaffen, sodass Nutzer über eine Entfernung miteinander an dem gleichen virtuellen Objekt arbeiten können. Das genannte Szenario bedient die Aspekte für eine kontaktfreie Interaktion, in der der Nutzer sich mobil um ein virtuelles Objekt bewegen kann. Aufgrund der Kombination von realen und virtuellen Objekten in einer Mixed Reality (kurz MR) ist die Verwendung eines direktmanipulativen Verfahrens von Vorteil, da ein Nutzer nicht erst ein neues Bedienkonzept erlernen muss, sondern sein bisheriges Wissen nutzen kann.

Bisher bekannte Lösungen ermöglichen es dem Nutzer, virtuelle Objekte zu greifen, zu drehen und zu verschieben. Die Bedienung einer Applikation, die nicht nur als Demo verwendet wird, sondern auch im produktiven Alltag zum Einsatz kommen soll, benötgt mehr als nur diese grundlegenden Interaktionsprimitiven. Neben direktmanipulativer Interaktionen mit virtuellen Objekten werden auch Primitiven für Selektion und Skalierung benötigt. Zusätzlich können unterschiedliche Anwendungen weitere Anforderungen an eine Eingabeinterface stellen.

1.2 Zielsetzung

In diesem Abschnitt wird die Zielsetzung der Arbeit beschrieben. Zunächst wird anhand der These 1 das Hauptziel der vorliegenden Arbeit vorgestellt. Anschließend werden die Teilziele erläutert.

1.2.1 These

These 1 *Durch die Kombination von interpretierten Gesten und direktmanipulativer Interaktion kann ein Benutzer effektiver mit virtuellen, dreidimensionalen Objekten arbeiten als es bei herkömmlichen Lösungen der Fall ist.*

Für diesen Zweck sollen zwei Ansätze der Interaktion miteinander verbunden werden. Interpretierte Gesten werden für erweiterte Interaktionsformen und direktmanipulative Interaktion für die Bewegung von virtuellen Objekten verwendet. These 1 ist die Grundlage der Arbeit und soll im weiteren Verlauf untersucht werden. Dabei wird zunächst eine Lösungsidee konzipiert und anschließend versucht, die These mithilfe der Lösung zu evaluieren.

Eine grundsätzliche Frage muss zusätzlich geklärt werden: Was toleriert der Nutzer noch als rein physikbasierte Interaktion und an welcher Stelle muss eine Interpretation erfolgen oder unterstützen?

Die direktmanipulative Interaktion erfolgt durch die Berechnung der Kollision zwischen den zu manipulierenden virtuellen Objekten und der virtuellen, physikbasierten Repräsentation von Händen und Fingern des Nutzers.

In der Evaluierung werden in einem Konstruktionsszenario die rein auf Physik basierende, direktmanipulative Interaktion mit einer Umsetzung verglichen, die eine zusätzliche Interpretation der Eingaben macht. In dem Szenario kann ein Nutzer Bausteine vor sich aufnehmen und sie zu einer größeren Konstruktion zusammenstecken. Die Konstruktion und die Bausteine könenn beliebig gedreht und verschoben werden. Verbindungen zwischen den Bausteinen können hergestellt und wieder gelöst werden. In der Umsetzung, in der die physikbasierte, direktmanipulative Interaktion um eine Interpretierung erweitert wird, können Bausteine und Konstrukte zudem auch skaliert werden.

1.2.2 Teilziele

Zum Erreichen des Hauptziels, der Bestätigung oder Widerlegung von These 1, müssen einige weitere Aspekte berücksichtigt werden. Diese werden im folgenden Abschnitt als Teilziele der Arbeit vorgestellt.

Alternative Eingabemethode Eine Umsetzung einer alternativen, kontaktfreien Eingabemethode für die Arbeit mit virtuellen Objekten und interaktiven Antwortzeiten wird entwickelt.

Systemaufbau Für die Umsetzung muss zuvor der Aufbau und die Architektur des Systems entworfen werden. Dieses System soll die Eingabe eines Nutzers aufnehmen und so verarbeiten, dass eine Applikation gesteuert werden kann. Ebenfalls soll der Aufbau so gewählt sein, dass der Nutzer möglichst frei in seiner Mobilität ist und keine Sensorik an den Armen und Händen trägt.

Tests und Evaluierung Um eine begründete Aussage zu treffen, wird eine Evaluierung durchgeführt. Zu Sicherstellung der Qualität und Verarbeitungsgeschwindigkeit werden automatisierte und manuelle Tests durchgeführt. Das Vorgehen bei den Tests kann auch auf andere Arbeiten mit ähnlichem Fokus angewendet werden.

Eingliederung in Gesamtprojekt Um nicht nur als Stand-Alone-Lösung zu dienen, soll die Lösung in das Gesamtsystem mit der bestehenden Infrastruktur unter der Nutzung der Middleware eingebunden werden können.

Sensor Fusion Es soll möglich sein, verschiedene Sensoren miteinander zu kombinieren. Somit können bspw. der größere Aufnahmebereich der Microsoft Kinect mit dem detailierten, jedoch kleinen Bereich der Leap Motion kombiniert werden, um dem Nutzer noch mehr Möglichkeiten zu bieten.

Device Abstraktion Durch die Unterstützung von mehreren Sensoren und die Zusammenführung auf einen gemeinsamen Standard kann die Erkennung unabhängig vom Sensor erfolgen. Dadurch ist ein breiteres Einsatzgebiet möglich.

Gestenset Zur Manipulation von virtuellen, dreidimensionalen Objekten muss ein Gestenset definiert werden, welches Primitiven bereitstellt und als Basis für zukünftige Arbeiten dienen kann.

1.3 Aufbau

Die vorliegende Thesis ist in fünf Kapitel unterteilt. In Kapitel 1 wird auf die Motivation zur Entwicklung einer Gestenerkennung und der Hauptthese der Arbeit eingegangen, es wird eine Zielsetzung definiert und es wird die Struktur der Thesis aufgezeigt.

In Kapitel 2 werden vergleichbare Arbeiten vorgestellt. Diese unterteilen sich in Arbeiten zu direktmanipulativer Interaktion und Interpretation von Gesten. Zuvor werden einige Definitionen für den weiteren Verlauf der Arbeit festgelegt. Das Kapitel endet mit einer Bewertung der untersuchten Arbeiten für die Zwecke dieser Thesis.

Kapitel 3 umfasst die Vorstellung eines Konzeptes zur Umsetzung einer Kombination aus direktmanipulativer Interaktion und Interpretierung von Gesten. Dabei wird das Konzept einer Sensorabstraktion genauer erklärt und es wird aufgezeigt, wie die Lösung aufgebaut und umgesetzt ist. Zusätzlich werden Anforderungen an die Lösung gestellt und in funktionale und nicht-funktionale Requirements unterteilt.

Die Evaluierung der Hauptthese wird in Kapitel 4 besprochen. Dabei werden sowohl technische als auch psychologische Aspekte getestet. Es wird zunächst das Testsetting mit den unterschiedlichen Szenarien beschrieben. Anschließend wird auf die Auswahl von Probanden eingegangen und der Fragebogen wird erklärt. Eine Auswertung der Ergebnisse aus findet im Anschluss statt. Neben der Auswertung befindet sich auch eine Interpretation der Ergebnisse und die Beschreibung zusätzlicher Beobachtungen in diesem Kapitel.

Im Kapitel 5 wird ein Fazit zu der Thesis gegeben und die Lessons Learned werden vorgestellt. Schlussendlich erfolgt ein Ausblick auf zukünftige Arbeiten.

2 Vergleichbare Arbeiten

In diesem Kapitel werden vergleichbare Arbeiten untersucht und bewertet. Das Kapitel ist dabei in drei Abschnitte unterteilt. Im ersten Abschnitt werden grundlegende Begriffe definiert (Abschnitt 2.1). In Abschnitt 2.2 werden Arbeiten untersucht, die sich mit direktmanipulative Interaktion auseinander setzen. Anschließend werden Verfahren zur Gestenerkennung untersucht, Abschnitt 2.3. Abschließend werden die Ergebnisse in Abschnitt 2.4 zusammengefasst.

2.1 Definitionen

Um im weiteren Verlauf dieser Arbeit ein einheitliches Verständnis für die einzelnen Begriffe zu haben, werden zunächst einige Definitionen gegeben. Die Begriffe werden in der Literatur teilweise unterschiedlich genutzt und deshalb an dieser Stelle definiert.

2.1.1 Mixed Reality

Mixed Reality (MR) ist ein Teil der virtuellen Realität, in der reale und virtuelle Objekte in einer Szene gemeinsam dargestellt werden. Sie wird zum einen von der realen Umgebung und zum anderen von der virtuellen Umgebung begrenzt, wie in Abbildung 2.1 gezeigt. Augmented Reality (AR) ist die Erweiterung der Realität durch zusätzliche, virtuelle Informationen. Ein Beispiel wäre die Navigation im Straßenverkehr durch ein Head-up-Display, das die korrekte Fahrtrichtung für den Fahrer sichtbar auf die Frontscheibe projeziert und Verkehrsschilder markiert.

Im Gegensatz dazu ist die Augmented Virtuality (AV) eine virtuelle Umgebung, in der reale Daten eingeblendet werden. Diese realen Daten können beispielsweise Webcams sein, deren Videostream in einer virtuellen Welt in einem Fenster gerendert wird.

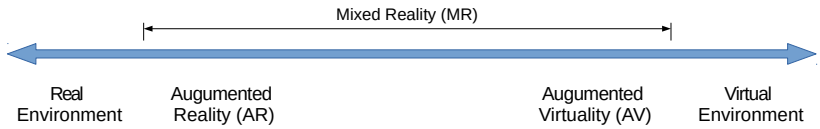


Abbildung 2.1: Der Bereich zwischen vollständig realer und vollständig virtueller Welt wird als Mixed Reality bezeichnet (Quelle: [MK94])

2.1.2 Virtual Reality

Eine virtuelle Welt oder virtuelle Realität ist eine rein virtuelle Umgebung, in der keinerlei realer Input zu finden ist. In der Abbildung 2.1 ist sie der aüßerste, rechte Teil des Kontinuums. Bekannte Vertreter aud der virtuellen Realität sind Computerspiele und Animationsfilme.

2.1.3 Kontaktfreie Interaktion

Die Interaktion im freien Raum ohne eine direkte Verbindung zu Sensoren oder Controllern wird als kontaktfreie Interaktion bezeichnet. Diese Art der Interaktion wird durch optische Messverfahren unterstützt, wie etwa die Leap Motion, Microsoft Kinect oder Lichtfeldkameras. Neben elektromagnetischen Wellen können auch Schallwellen als Medium verwendet werden. Beispiele für nicht kontaktfreie Interaktionen sind die Nutzung von Maus und Tastatur oder die Verwendung eines Gamepads oder Joysticks.

2.1.4 Interpretierte Geste

Eine Geste ist eine Abstraktion von Bewegungsmustern, denen eine Bedeutung zugewiesen wird (vgl. [McN92, LM90]). In dieser Arbeit werden interpretierte Gesten für virtuelle, dreidimensionale Objekte untersucht. Beispiele für Gesten sind die Zeigegeste, eine Geste zum Vergrößern oder Rotieren eines Objektes. In Abgrenzung zu Gesten stehen Posen. Posen sind eine statische Haltung und ändern sich nicht über die Zeit.

2.1.5 Direktmanipulative Interaktion

In der Literatur wird unter direktmanipulativer Interaktion ein System verstanden, dass drei Techniken beinhaltet. “(1) Bereistellung eines physikalisch direkten Weges, um einen Zeiger zu bewegen oder ein Objekt zu manipulieren. (2) Präsentation einer konkreten grafischen Repräsentation des Objektes und sofortige Anpassung der Sicht, um Operationen widerzuspiegeln. (3) Nutzung von Kommandosprache und Abhängigkeit von Operationen meiden, die

auf das kognitive Modell angewendet werden können, welches dargestellt wird." ([Shn82]) Als direktmanipulative Interaktion wird in dieser Arbeit der Umgang mit virtuellen, dreidimensionalen Objekten bezeichnet. Dabei haben die Bewegungen der Hände des Nutzers einen *direkten* Einfluss auf ein oder mehrere virtuelle Objekte.

2.2 Direktmanipulative Interaktion

Die Unterstützung von direktmanipulativer Interaktion, wie sie in Abschnitt 2.1 beschrieben ist, kann durch verschiedene Ansätze erreicht werden. Grob sind diese in Partikelsysteme (Abschnitt 2.2.1), aktorbasierte (Abschnitt 2.2.2) und modellbasierte Ansätze (Abschnitt 2.2.3) zu unterscheiden.

Die klare Herausforderung von direktmanipulativer Interaktion ist die Unterstützung von komplexen, feingranularen Interaktionen, unter der vollständige Utilisierung der Hand. Darunter fallen Kombinationen aus Greifen, Rotieren, Verschieben und Loslassen von virtuellen Objekten durch die Hand eines Nutzers. Die Abweichungen vom Sollwert bei Positionierungen sollte dabei möglichst gering sein.

Die untersuchten Verfahren nutzen zur Simulation Physikengines. Die Umsetzungen sind in der Regel performant, stoßen aber bei der Nachbildung realer Phänomene an ihre Grenzen. Zur besseren Unterstützung der Physikengines nutzen einige Verfahren deshalb Masse-Feder-Systeme (Abschnitt 2.2.4). Ein weiterer Punkt ist die Abwägung zwischen Realismus und Latenz. Wird ein sehr detailiertes Modell der Hand genutzt und werden alle Kräfte vollständig berechnet, stoßen aktuelle Lösungen auch mit moderner Hardware an ihre Leistungsgrenzen. Im Gegensatz dazu stehen Approximationen, die eine deutlich geringere Latenz besitzen, da sie die Form der Hand vereinfachen. Je nach Grad der Vereinfachung kann es für den Nutzer dabei zu ungewohntem Verhalten kommen.

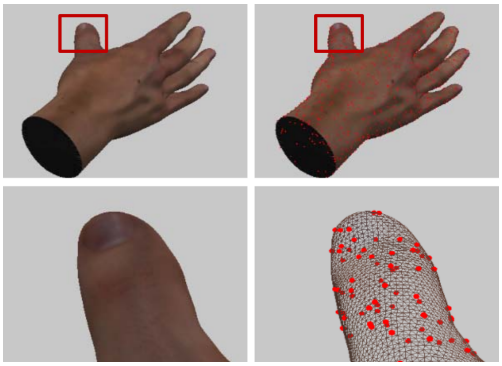
2.2.1 Partikelsysteme

Partikelsysteme vereinfachen die Berechnung komplexer, kontinuierlicher Modelle, indem sie ein kontinuierliches Modell mithilfe einer Vielzahl an kleinen, regulären Objekten nachbilden und somit ein diskretes Modell erzeugen. Ursprünglich wurden Partikelsysteme zur detailgetreuen Simulation von weichen Materialien, Gasen und Flüssigkeiten verwendet (vgl. [Ree83, WH94]). "Partikelsystem können als Finite-Elemente-Methode höherer Ordnung angesehen werden." ([MNKW07]) Der Lebenszyklus eines Partikels besteht aus der Registrierung, mehreren Aktualisierungen, nach denen jeweils ein Simulationsschritt folgen kann und einer möglichen Löschung. In der Registrierung wird eine Position für jedes Partikel festgelegt,

sodass die Gesamtheit aller Partikel ein dreidimensionales Objekt möglichst gut beschreibt. Die Aktualisierung bestimmt die neue Position jedes Partikels in einem Frame.

Eine Adaption zur Verwendung von Partikelsystemen zur Simulation von direktmanipulativer Interaktion wird in [HKI+12, WIH+08] behandelt. In [IKH+11] wurde ebenfalls ein Partikelsystem eingesetzt, dass jedoch nicht dynamisch verändert wird, sondern statisch nach dem Scannen eines starren Objektes berechnet wird. Die Interaktion in [WIH+08] wird auf einem Touchscreen mit einem zweidimensionalen Display durchgeführt. Erst in [HKI+12] findet eine echte Interaktion im dreidimensionalen Raum statt. Hilliges et al. nutzt ein Partikelsystem, um physikalische Objekte, etwa die Hände des Nutzers, im Interaktionsraum nachzustellen, zu sehen in Abbildung 2.2a.

(a) Die Partikel modellieren die Hand und alle anderen physikalischen Objekte vollständig. Positionsupdates werden über umfangreiche Berechnungen ermittelt (Quelle: [HKI+12])



(b) Partikel befindet sich auf dem Oberflächen-Mesh der Hand und werden durch Manipulation des Gesamtmeshes in ihrer Position verändert (Quelle: [KP15])

Abbildung 2.2: Zwei Beispiele für Partikelsysteme mit unterschiedlichem Ansatz zum Update der Partikel

Ein großer Vorteil von Partikelsystemen zeigt sich in der Vielfalt an realistischen Interaktionsmöglichkeiten zwischen verschiedenen realen und virtuellen Objekten. So ist es beispielsweise möglich, einen virtuellen Ball auf einem realen Papier herunterrollen zu lassen, um ihn anschließend in der Hand aufzufangen oder ihn in die Luft zu werfen.

Ein Nachteil von Partikelsystemen ist die rechenintensive Vorverarbeitung und das Update der Partikel aufgrund der großen Anzahl an zu berechnenden Partikeln. [HKI+12] versucht

dieses Problem zu lösen, indem die meisten Berechnungen auf die Grafikkarte ausgelagert werden, die für diese Art von Berechnungen gut geeignet ist.

Die Funktionsweise zur Aktualisierung von Partikelsystemen wird anhand das Verfahrens **Depth-Aware Optical Flow** erläutert. Dabei wird der Versatz von Pixeln nicht direkt auf einem Tiefenbilder berechnet, sondern es wird das RGB-Bild verwendet, da es besser texturiert ist und somit robustere Ergebnisse liefert. Die nachfolgende Auflistung gibt die einzelnen Schritte des Algorithmus wieder.

1. Rektifizierung von RGB- und Tiefenbild in Frame i und $i + 1$
2. Vordergrundsegmentierung der RGB-Bilder
3. Berechnung des optischen Flusses unter Verwendung der einer Energiefunktion zur Minimierung des Intensitätsunterschiedes in einem Pixel
4. Berechnung des Versatzes im Tiefenbild
5. Aktualisierung der Positionen der Partikel

Die Partikel als Gesamtheit ergeben eine gute Approximation der realen Objekte und können sowohl Festkörper als auch verformbare Objekte widerspiegeln. Durch sie können Kräfte in der Physiksimulation modelliert werden, wie etwa seitlich wirkende Kräfte oder Reibungskräfte.

Auf die zuvor stattfindende Registrierung wurde an dieser Stelle nicht eingegangen. Hilliges et al. greifen auf eine vereinfachte Form mittels Rasterisierung zurück. Da sie nur eine Halbschale des dreidimensionalen Objektes, nämlich die der RGBD-Kamera zugewandten Seite modellieren müssen, ist dieser Ansatz zielführend. Andere Ansätze müssen wie Eingangs erwähnt ein Problem höherer Ordnung (nicht-lineare Systeme) lösen. Nach der Registrierung und der Aktuallisierung der Partikel nutzen alle vorliegenden Partikelsysteme die Funktionalität von vorhandenen Physikengines.

Im Gegensatz zu reinen Partikelsystemen verfolgt [KP15] mit der Kombination aus modellbasiertem Ansatz und Partikelsystem einen anderen Weg. Ihr Partikelsystem ist fest in das detailierte Oberflächen-Mesh der Hand integriert (siehe Abbildung 2.2b). Somit entfällt die Registrierung der Partikel und ein späteres Löschen komplett. Eine Aktualisierung der Position erfolgt in drei Schritten.

1. Aktualisierung des Skelettes der Hand auf Basis von Skelettdaten über einen Sensor
2. Repositionierung des Meshes auf Basis des Skelettes

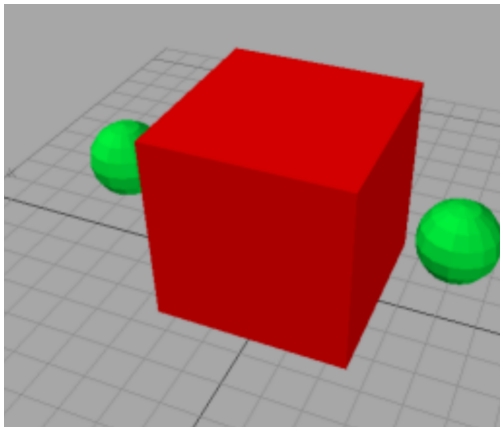
3. Aktualisierung der Positionsdaten der Partikel auf Basis der Meshdaten

Durch diesen Ansatz können mehrere rechenintensive Schritte übersprungen werden. Es ist möglich, die Berechnungen auf der CPU durchzuführen. Die von den Autoren durchgeführten Tests haben gezeigt, dass mit diesem Ansatz die Arbeit mit dreidimensionalen, virtuellen Objekten mit geringer Fehlertoleranz möglich ist.

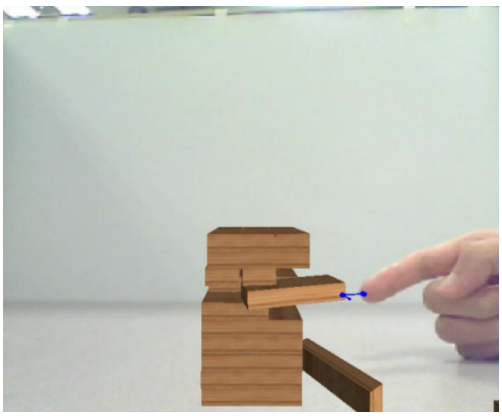
2.2.2 Aktorenbasierter Ansatz

Eine Interaktion auf Basis von Aktoren ist eine sehr einfache Form und vereinfacht das Konzept der Partikelsysteme durch die Anzahl von $n = 1$ Partikeln. Der einzelne Partikel wird dann Aktor genannt und ist zumeist größer als ein Partikel in einem Partikelsystem. In [Pot11] wird ein Aktor pro Hand verwendet, um mit virtuellen Objekten zu interagieren, siehe Abbildung 2.3a. In [SYW08] besitzt die Spitze des Zeigefingers der rechten Hand einen Aktor, mit dem der Nutzer mit der virtuellen Welt interagieren kann (siehe Abbildung 2.3b).

Aktorenbasierte Ansätze sind sehr leicht zu erstellen und zu berechnen. Ihr Nachteil liegt in der starken Vereinfachung der Hand als Werkzeug und dem damit einhergehenden Verlust der Funktionalität. Einfache Interaktionsformen wie Verschieben oder Drehen sind damit bis zu einer gewissen Genauigkeit durchführbar. Feingranulare Selektion, einhändiges oder zweihändiges Greifen lassen sich mit diesem Ansatz jedoch nicht erreichen.



(a) Die Aktoren repräsentieren die Hände des Nutzers. Das virtuelle Objekt kann damit transliert und rotiert werden (Quelle: [Pot11])



(b) Ein Aktor wird in der Fingerspize registriert und kann mit virtuellen Objekten interagieren (Quelle: [SYW08])

Abbildung 2.3: Zwei Beispiele für aktorbasierte Ansätze für direktmanipulative Interaktion

Registrierung und Update sind bei einem aktorbasierten Ansatz zusammengefasst. So wird in [SYW08] in jedem Kameraframe zunächst die Hand und anschließend die Fingerspitze gesucht. Der Aktor wird an die Position der Fingerspitze gesetzt. In [Pot11] wird der Mittelpunkt der Fläche gesucht, die sich zwischen den Fingern einer Hand aufspannt. An dieser Stelle wird ein Aktor positioniert. In jedem Frame wird der Mittelpunkt neu gesucht und mit den vorherigen Werten verglichen. Aus der Differenz kann die Geschwindigkeit berechnet werden. Der Ansatz über Aktoren wird in der Grafikbearbeitung, genauer dem Sculpting verwendet [SBS06], um durch Druck und Zug ein Objekt zu verformen (ähnlich wie Knete oder Ton).

2.2.3 Modellbasierter Ansatz

Unter modellbasierten Ansätze werden verschiedene Lösungen verstanden, die die Hand als ein Modell nachbilden. Dabei sind zwei Strömungen zu erkennen. Auf der einen Seite wird versucht, einen möglichst hohen Detailgrad der Hand zu verwenden. Auf der anderen Seite wird das Handmodell sehr stark durch Basiskörper approximiert.

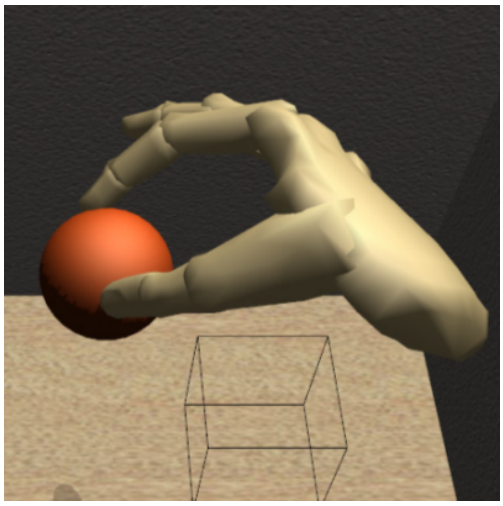
Ein modellbasierter Ansatz bildet die Eigenschaften der Hand am genauesten nach und erlaubt so ein gewohntes Arbeiten. Das Problem liegt in der komplizierten Berechnung der Zustände in der Physikengine und die Möglichkeit, in nicht auflösbare Zusände zu gelangen, in denen das Verhalten nicht spezifiziert ist.

Ein komplett gemeshtes Handmodell beschreibt die Hand eines Nutzers vollständig und repräsentiert die Hand in der virtuellen Welt somit am Besten. Die Berechnungen bei einer Kollision sind jedoch extrem umfangreich und werden von vielen Physikengines nicht unterstützt, da zu viele Vertizen in dem Kollider enthalten sind.

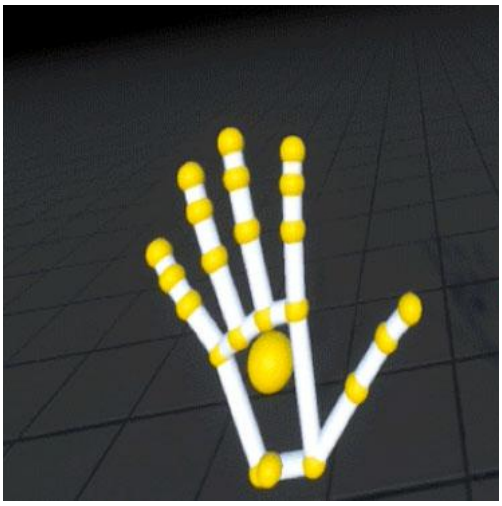
Approximationen schaffen an dieser Stelle Abhilfe und erlauben eine vereinfachte Berechnung. Durch die Vereinfachung kann es für den Nutzer jedoch zu unverständlichem Verhalten kommen, obwohl die Approximation der Hand durch Primitiven eine deutliche Verbesserung gegenüber einfachen Aktoren darstellt.

[PB11] zeigt ein Modell bestehend aus unterschiedlichen Primitiven, die zusammen eine sehr genaue Approximation der Hand ergeben, siehe Abbildung 2.4a. Diese Primitiven dienen als Kollider in einer Physiksimulation und ermöglichen die Interaktion mit virtuellen Objekten.

Eine weitaus generischeres Modell bietet die Leap Motion. Sie modelliert die Hand aus Säulen und Kugeln, wobei die Approximation relativ ungenau ist (Abbildung 2.4b). Dadurch kann es zu ungewohntem Verhalten kommen.



(a) Unterschiedliche Primitve bilden zusammen ein geschlossenes Modell (Quelle: [PB11])



(b) Eine Hand modelliert durch Säulen und Kugeln. Modell ist nicht geschlossen (Quelle: Leap Motion Inc.)

Abbildung 2.4: Zwei Beispiele für modellbasierte Ansätze zur direktmanipulative Interaktion

2.2.4 Masse-Feder-System

Ein großes Problem bei der Arbeit mit Physikengines ist das Clapping von Kollidern in andere virtuelle Objekte aufgrund des direkten Setzens von Positionen durch den Nutzer. Es kommt zu einem invaliden Zustand, den die Engine nicht sauber auflösen kann. Eine Lösung sind Masse-Feder-Systeme (vgl. [KP15, HKI+12, PB12a, PB11, PB12b]).

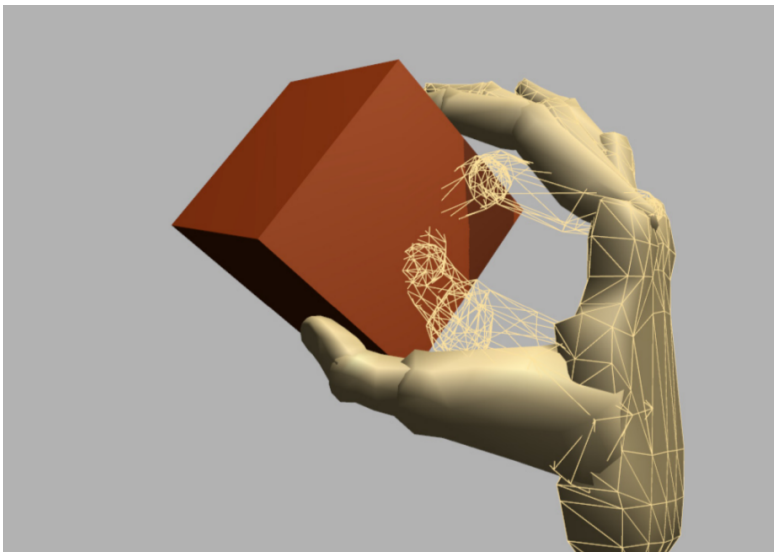


Abbildung 2.5: Die Kollider drücken auf die Fläche des virtuellen Objektes ohne das Objekt zu penetrieren (Quelle: [PB11])

Dabei werden die Kollider nicht direkt bewegt, sondern sie werden durch Federn in der virtuellen Welt gezogen. Treffen sie auf ein virtuelles Objekt, dann dringen sie nicht in dieses ein, sondern drücken mit einer zunehmenden Kraft auf deren Oberfläche in Richtung der Normale. Zusätzlich erhöht sich die tagential wirkende Reibungskraft, beispielhaft zu sehen in Abbildung 2.5.

2.3 Interpretierte Gesten

Die Interpretation von Gesten, wie sie in Abschnitt 2.1 definiert sind, ist notwendig, um weiterführende Interaktionskonzepte zu ermöglichen. Eine Beschränkung allein auf direktmanipulative Interaktion zwingt dem Nutzer ein Bedienkonzept auf, das künstlich eingeschränkt wird. Umfangreiche Anwendungen können mit einem solchen Konzept nicht umgesetzt werden. So ist es nur durch eine Gesteninterpretation möglich ein Menüsystem zu entwerfen, das mit den Händen gesteuert werden kann. Rotation und Translation von dreidimensionalen, virtuellen Objekten sind Transformantionen, die mithilfe von direktmanipulativer Interaktion umgesetzt werden können. Es ist jedoch nicht möglich, allein durch diese Form der Interaktion Skalierung und Scherung zu modellieren.

Neben der Nutzung von Menüs und der Transformation von Objekten können interpretierte Gesten auch zur Unterstützung von schwierigeren direktmanipulativen Interaktionen, wie dem Greifen und Loslassen von Objekten verwendet werden.

Interpretierte Gesten besitzen einen dynamischen Anteil und können keine, eine oder mehrere Posen enthalten. Die Bewegung ist dabei Teil der Information und hat sonst keinen weiteren Zweck. Der Unterschied zwischen Geste und Pose wurde in Abschnitt 2.1.4 geklärt.

In der Literatur werden häufig Systeme zur Gestenerkennung vorgestellt, die per Definition keine Gesten sondern Posen erkennen [RKSI+14, SKR+15, OWL15, CRDR13]. Die der vorliegenden Arbeit zugrunde liegenden Definitionen werden genutzt, um eine klare Unterscheidung zu erhalten. Arbeiten zur Erkennung von Posen, also eine über die Zeit unveränderte Haltung, können als Basis für Gestenerkennungen dienen, sind aber nicht mit ihnen zu vergleichen.

Diese Arbeit untersucht Gesten, die im freien Raum ausgeführt werden können. Einer der wichtigsten Unterschiede zwischen zweidimensionalen Gesten, die mit dem Stift, dem Finger oder der Maus durchgeführt werden, und räumlichen Gesten ist die Erkennung des Anfangs- und Endpunktes. Im Fall von zweidimensionalen Gesten ist dieses Problem nicht vorhanden. Erst durch den Kontakt mit der sensitiven Oberfläche bzw. die Betätigung der Maustaste beginnt eine Geste. Das Ende ist durch das Abheben des Stifts oder Fingers bzw. das Loslassen der Maustaste gekennzeichnet. Im Falle von dreidimensionalen Gesten ist die Erkennung deutlich schwieriger. Das Start-Ende-Problem (auch König-Midras-Problem) ist nicht einfach zu lösen. Der Suchraum wächst aber nicht nur durch die fehlende temporale Begrenzung, sondern auch durch die zusätzliche Dimension. Ein weiteres Problem ist die stark schwankende Größe der Gesten, die durch unterschiedliche Nutzer ausgeführt wird. Medien zur Eingabe von zweidimensionalen Gesten haben eine natürliche Einschränkung, etwa das Display eines Smartphones, das Touchpad eines Notebooks oder das Fenster einer Applikation. Räumliche Gesten besitzen diese Einschränkungen in der Regel nicht.

In den nachfolgenden Abschnitten werden verschiedene Ansätze zur Interpretation von Gesten untersucht. Beginnend mit Templatematching in Abschnitt 2.3.1, Trajektorie in Abschnitt 2.3.2 und Convolutional Neural Network in Abschnitt 2.3.3 wird die Funktionsweise beschrieben und es werden unterschiedliche Beispiele für die jeweiligen Ansätze mit ihren Besonderheiten vorgestellt.

2.3.1 Templatematching

Ein verbreiteter Ansatz zur Erkennung von zweidimensionalen Gesten ist die Verwendung von Templatematching [AB10, FC15, KD11, NY95]. Dabei wird eine Menge von Gestentemplates, dem Gestenset, mit der aktuellen Eingabe auf Basis einer Distanzfunktion verglichen. Anhand der Distanz wird entschieden, welches der vorhanden Templates am Besten zu der Eingabe passt. Den Ansatz des Templatematchings kann man ebenfalls für räumliche Gesten verwenden [KNQ12].

Templatematching kann mithilfe einfacher mathematischer Konzepte implementiert werden. Es ist leicht erweiterbar, da in den meisten Fällen nur ein weiteres Template in einem vorgegebenen Format[1] definiert werden muss. Templatematching nutzt keine Lernphase und benötigt deshalb keine zusätzlichen Lerndaten.

Gesten, die durch Templates definiert werden, sind in der Regel sehr abstrakt und ähneln Schriftzeichen. Das Erlernen einer solchen Geste ist daher für Expertensysteme geeignet und nicht für Nutzer, die nur kurz mit einem System interagieren (vgl. [AB10]).

Kristensson et al. stellen in ihrer Arbeit eine Methode vor, mit der Gesten, die mit den Händen und Armen ausgeführt werden, erkannt werden können. Dabei verwenden sie einen auf Wahrscheinlichkeitsverteilung basierenden Algorithmus, der in jedem Schritt eine Prognose erstellt. Der Algorithmus vergleicht dabei die Bewegung des Nutzers mit einer Sammlung von Gestentemplates und ermittelt die Wahrscheinlichkeit für jede Geste ([KNQ12]) im Gestenset. Als Eingabe wird das komplette Skelett der Kinect-Kamera verwendet, aber für die Erkennung der Geste werden nur die Bewegungen der Handflächen analysiert. Eine Ausrichtung des Nutzers oder die Körperhaltung wird ebenso nicht berücksichtigt wie die Bewegung der Finger. Ein weiterer Punkt, der beachtet werden muss, ist die einfache Projektion $R^4 \rightarrow R^2$ durch $(x, y, z, t) \rightarrow (x, y)$. Nutzereingaben in der Tiefe werden somit komplett verworfen.

[AB10] zeigen einen Algorithmus zur Ermittlung der Skalierung einer Geste, bevor sie beendet wurde. Da die Ermittlung a-priori arbeitet, kann sie zur Unterstützung von Nutzern verwendet werden. Als Distanzfunktion wird ein der die Turning-Angle-Distanz verwendet.

[FC15] erstellen aus den Templates zunächst *Polylines*. Die Eingabe wird ebenfalls so behandelt. Bei einem Vergleich werden zwei Polylines abgeglichen. Bevor die Distanz berechnet wird, wird die Polyline der Eingabe rotiert, sodass sie mit dem ersten Segment der Polyline des Templates übereinstimmt. Die Distanz wird über die Multiplikation eines Strafwertes mit der aufsummierten Längen der Fehlervektoren gebildet, siehe 2.1. Der Strafwert entsteht bei dem Abgleich der Polylines. Der Fehlervektor ist die Differenz der einzelnen Abschnitte auf den Polylines, wobei die Länge gleich der relativen Länge eines Abschnittes ist und die Steigung durch die Orientierung gegeben ist.

$$D(P, Q) = penalty * \sum_{i=1}^{n} |\overrightarrow{P_i} - \overrightarrow{Q_i}| \tag{2.1}$$

[1] Oftmals ist das verwendete Format SVG.

2.3.2 Trajektorie

Die Trajektorie ist die Beschreibung der Bahn eines Objektes. In der Gestenerkennung kann sie genutzt werden, um den dynamischen Anteil einer Geste mathematisch zu beschreiben. Da die Trajektorie, ähnlich wie das Template, nur den Verlauf beschreibt, wird sie mit unterschiedlichen Verfahren zur Auswertung kombiniert. Die Arbeiten von [WXB+14, EAHK+07, EAHAM08] nutzen *Hidden-Markov-Modelle*[2] zur Bestimmung der passenden Geste. Ein weiterer, sehr verbreiteter Ansatz ist die Kombination von Trajektorie zur Beschreibung von dreidimensionalen Gesten und die Nutzung einer *Support Vector Machine*[3] zur Klassifizierung (vgl. [LD13, BPS+14, CXBT12]).

Im Gegensatz zu Templates können Gestenbeschreibungen über Trajektorie eine zeitliche Komponente enthalten. Die Beschreibung von dreidimensionalen Bewegungen als Bahnkurve fällt einem Computer sehr leicht. Die Umrechnung von einer grafischen Darstellung in eine besser verarbeitbare Form muss nicht erfolgen. Die Trajektorie zur Beschreibung einer Geste ist sehr gut zur Kombination mit statistischen Methoden, wie den HMM, SVM und auch *Recurrent Neural Networks*[4] (vgl. [XC16]) geeignet.

Trajektorie wird in der Regel nicht direkt zur Erkennung genutzt, sondern als Repräsentation für eine Geste und zur Beschreibung des Nutzerinputs. Durch die Nutzung von HMM, SVM oder RNN entsteht eine zusätzliche Trainingsphase. Eine Erweiterung durch neue Gesten ist somit in der Regel nicht möglich.

2.3.3 Convolutional Neural Network

Faltungsnetzwerke[5] sind eine spezielle Form der neuronalen Netzwerke, genauer der vorwärtsgekoppelten Netzwerke. Sie werden in der Literatur für viele Bereiche der Bilderkennung und Spracherkennung genutzt. Allgemein eignen sie sich für die Erkennung von Mustern aller Art (siehe [Nie15]). So können auch allgemeine Bewegungen und Handgesten Handgesten durch die Verwendung eines CNN erkannt werden (vgl. [JXYY13, HCLC16, WLC+12, NDC+11, MGKK15]).

Ihr großer Vorteil liegt darin, dass sie selbstständig ihre Hidden-Layer formen und Features aus den Input-Daten ermitteln. Somit ist es nicht mehr nötig, manuell Features für die Trainingsdaten und die späteren Eingabedaten zu definieren. CNN eignen sich zur Verarbeitung von großen Stichprobenräumen, den Sampling Spaces. Da sie im Gegensatz zu anderen neuro-

[2] kurz HMM
[3] kurz SVM
[4] kurz RNN
[5] kurz CNN

nalen Netzen, werden in einem CNN nicht alle Neuronen eines Layers mit allen Ausgängen des vorherigen Layers verbunden. Dadurch nimmt die interne Größe stark ab.

Um ein CNN verwenden zu können, muss es zuvor mit passenden Daten trainiert werden. Wie auch schon in Abschnitt 2.3.2 erwähnt, bedeutet eine Trainingsphase auch eine schwierige Erweiterbarkeit. Ebenso ist die Berechnung eines CNN aufwändig und erfordert die Ausnutzung moderner Hardware zur Parallelisierung von Berechnungen auf der GPU.

Molchanov et al. kombiniert ein hochaufgelöstes und ein niedrigaufgelöstes Netzwerk, um eine robuste Klassifizierung von Handgesten unter veränderlichen Lichtbedingungen und von unterschiedlichen Positionen zu erkennen (siehe [MGKK15]). Zu diesem Zweck werden die Gradienten verwendet. Die einzelnen Kanäle wurden normalisiert um die Berechnung zu beschleunigen. Um die Trainingsdaten zu erweitern, damit keine Überanpassung stattfindet, wurden die vorhanden Daten durch Augmentation erreicht. Dabei wurden die vorhanden Daten zum einen gespiegelt und in unterschiedlicher Reihenfolge abgespielt und zum anderen durch Transformation, Deformation und Entfernen einiger Bildteile. Die Menge der Trainingsdaten wurde somit erheblich gesteigert.

Die direkte Verwendung von Tiefenbildern in einem CNN ist ebenso möglich, wie Liu et al. zeigt. In [LZT16] werden sowohl Tiefenbilder als auch Skelettdaten verwendet, um Körperbewegungen zu erkennen. Die Sequenz von Tiefenbilder werden in einem CNN zu einem High-Level-Feature transformiert. Die Skelettdatensquenz wird getrennt verarbeitet und es wird ein Gelenksvektor-Feature berechnet. Die beiden Features werden in separaten SVM klassifiziert und in einem anschließenden Fusion zusammengefasst. Die Fusion bildet dabei gewichtete Summen der Ergebnisse der SVM und gibt das Ergebnis mit der höchsten Wahrscheinlichkeit weiter. Parallel dazu nutzen Wu et al. Skelettdaten, Tiefen- und Farbbilder zur Segmentierung und Erkennung von Gesten. In ihren Ergebnissen stellen sie fest, dass eine Kombination von Modalitäten zur Erkennung von Gesten bessere Ergebnisse erzielt, als die Modalitäten für sich allein gestellt. Die Late Fusion erzielte dabei leicht bessere Ergebnisse, als eine Fusion in der Mitte der Verarbeitung (siehe [WPK+16]).

2.4 Zusammenfassung

In den vorangegangenen Abschnitten wurden verschiedene Systeme zur Interpretation von Gesten und zur Verarbeitung von direktmanipulativer Interaktionen untersucht. Keines der untersuchten Arbeiten bietet eine Kombination aus direktmanipulativer Interaktion und interpretierten Gesten. Daher werden diese Verfahren separat ausgewertet und es wird ein Ausblick auf die kombinierte Verwendung gegeben.

Direktmanipulative Interaktion kann durch verschiedene Ansätze verarbeitet werden. Partikelsysteme bieten eine realistische Interaktion und können mit Physikengines kombiniert werden. Die benötigten Berechnungen zur Initialisierung und Aktualisierung sind jedoch aufwendig und können nur in einfachen Setups (vgl. [HKI+12, WLK+14]) effektiv berechnet werden. Eine Lösung über Aktoren ist leicht zu implementieren, bietet aber keine umfangreichen Interaktionsmöglichkeiten. Zusätlich ist die Nutzung eines einzelnen Aktors für die gesamte Hand eines Nutzers eine zu grobe Auflösung. Ein modellbasierter Ansatz bietet, wie auch Partikelsysteme, eine realistische Interaktion und ist zumindest für die Initialisierung und die Aktualisierung einfach zu berechnen. Ohne eine geeignete Approximation, ist die Verwendung in einer Physikengine jedoch schwierig. Die Kombination von detailiertem Modell zur Visualisierung und Nutzung von Partikeln zur Interaktion mit dem virtuellen, dreidimensionalen Objekt, wie in [KP15] vorgestellt, verspricht einen guten Kompromiss aus realistischer Simulation und einfacher Berechenbarkeit. Durch Zuhilfenahme eines Masse-Feder-Systems (siehe Abschnitt 2.2.4) kann die Simulation zusätzlich unterstützt werden.

Die Erkennung von Gesten und deren Interpretation kann bspw. durch Templatematching, Trajektore oder CNN erfolgen. Templatematching in Kombination mit Distanzfunktionen wird häufig zur Erkennung zweidimensionaler Gesten verwendet. Die Erkennung von dreidimensionalen Gesten gestaltet sich als schwierig. Kristensson et al. schlägt ein Konzept vor, dass zwar dreidimensionale Eingaben im freien Raum zulässt, jedoch nur zwei Dimensionen auswertet (vgl. [KNQ12]). Durch eine zusätzliche Dimension müssen deutlich mehr Templates für das gleiche Kommando verwendet werden, da beispielsweise eine Skalierung nun nicht mehr in der Ebene, sondern im Raum stattfindet. Die Verwendung einer Distanzfunktion erlaubt jedoch die Erstellung einer Gestenerkennung ohne zusätzliche Trainingsphase, was zu einer besseren Erweiterbarkeit führt. Durch Trajektorie können auch dreidimensionale Gesten beschrieben und zusätzliche temporale Abhängigkeiten definiert werden. Die Erkennung arbeiten dabei in der Regel auf Basis eines statistischen Verfahrens, wie HMM, SVM oder RNN, sodass eine vorherige Lernphase notwendig ist. Bei der Verwendung von CNN entfällt die Bestimmung von Features vollständig. Die Berechnung der CNN gestaltet sich jedoch bei größerem Umfang der Trainingsdaten als sehr umfangreich. Somit ist auch die Erweiterbarkeit eingeschränkt. Die besten Ergebnisse bei Berücksichtigung der Anforderungen in Abschnitt 3.1 werden bei einer Kombination aus Trajektorie zur Beschreibung der Geste in Verbindung mit Distanzfunktionen zur Gestenerkennung erwartet. Die Kombination unterstützt dreidimensionale Gesten vollständig und ist zudem leicht erweiterbar.

In Kapitel 3 wird ein Konzept zur Umsetzung eines Systems erarbeitet, dass eine Interpretation von dreidimensionalen Gesten und eine Verarbeitung von direktmanipulativer Interaktion erlaubt.

3 Konzept

In den vorangegangenen Kapiteln wurde die These aufgestellt, dass eine Kombination aus direktmanipulativer Interaktion und interpretierten Gesten für den Umgang mit dreidimensionalen, virtuellen Objekten in Mixed-Reality-Anwendungen Vorteile gegenüber herkömmlichen Ansätzen hat. Dazu wurde zunächst der Anwendungsfall näher umrissen und anschließend wurden vergleichbare Arbeiten untersucht und bewertet. Dabei hat sich gezeigt, dass keine der bisherigen Verfahren eine echte Kombination nutzt.

Somit muss zunächst eine geeignete Lösung erarbeitet werden. In diesem Kapitel wird das Konzept der Umsetzung vorgestellt. Zunächst werden in Abschnitt 3.1 die Anforderungen näher erläutert. Anschließend wird in Abschnitt 3.2 ein Überblick über die konzipierte Lösung gegeben. Wie in Kapitel 2.4 bereits vorweggenommen, wird die Lösung einen modellbasierten Ansatz zur Umsetzung von direktmanipulativer Interaktion nutzen. In den nachfolgenden Abschnitten 3.3 und 3.4 wird die Sensorabstraktion respektive die Komponente zur Gestenerkennung vorgestellt. Ein detailiertes Konzept zur Unterstützung von direktmanipulativer Interaktion ist in Abschnitt 3.5 zu finden. Mehr Informationen zur Interpretation von Gesten befindet sich in Abschnitt 3.6.

Das System zur Kombination von direktmanipulative Interaktion und interpretierten Gesten wird im Folgenden mit **Gesture Recognition** abgekürzt.

3.1 Anforderungen

In diesem Abschnitt werden die Anforderung für ein System zur Gestenerkennung beschrieben. Neben den rein funktionalen Anforderungen in Abschnitt 3.1.1 werden auch die nichtfunktionalen Anforderungen in Abschnitt 3.1.2 dargestellt.

3.1.1 Funktionale Anforderungen

Funktionale Anforderungen beschreiben allein die Funktionsweise eines Systems. Qualitative Aussagen werden dabei nicht getroffen.

Verarbeitung von Handbewegungen

Das zu entwickelnde System soll die Handbewegung eines Nutzers verarbeiten und ein Handmodell erzeugen, das in einer Physikengine genutzt werden kann, um virtuelle, dreidimensionale Objekte zu manipulieren.

Interpretation von Handbewegungen

Die Bewegungen der Hand eines Nutzers im dreidminesionalen Raum sollen durch das System interpretiert bzw. als Geste erkannt werden. Der Nutzer muss keine Aktivierungspose oder Geste durchführen. Es sollen grundsätzlich beliebige Gesten und Posen erkannt werden können. Das System benötigt keine Trainingsphase und erkennt Gesten und Posen robust auf Basis einer analytischen Beschreibung.

Mobile Nutzung

Der Nutzer soll sich während der Bedienung bewegen können und trotzdem keine Einschränkung bei der Nutzung erleben. Es wird eine nicht-stationäre Lösung erwartet (vgl. [HKI+12, WLK+14]).

Come as you are

Ein Nutzer benötigt keine zusätzlichen Marker oder Geräte an seinem Körper, um mit dem System zu interagieren. Es ist eine kontaktfreie Interaktion (siehe Abschnitt 2.1) mit dem System möglich.

Device Independence

Die Lösung soll unabhängig von einem bestimmten Sensor arbeiten.

Bereitstellung der Ergebnisse an Schnittstelle

Das Ergebnis der Analyse bestehend aus Kollidern und Ergebnissen der Interpretation soll zur Nutzung durch überliegende Applikation an einer Schnittstelle zur Verfügung gestellt werden.

3.1.2 Nicht-funktionale Anforderungen

Zu den nicht-funktionalen Anforderungen gehören Anforderungen, die qualitative Aussagen über ein System treffen können und nicht den funktionalen Anforderungen zuzuordnen sind.

Erlernbarkeit

Das System soll für den Nutzer einfach zu erlernen sein. Es soll für Einsteiger und Experten gleichermaßen geeignet sein und kein reines Expertensystem bieten.

Antwortzeitverhalten

Das System soll interaktive Antwortzeiten besitzen. Somit ist eine Verzögerung von mehr als 100 ms nicht mehr akzeptabel.

3.2 Übersicht

Die hier präsentierte Lösung vereinigt zwei Ansätze der räumlichen Interaktion: direktmanipulative, dreidimensionale Interaktion und Interpretation von räumlichen Gesten. Die Begriffe wurden bereits in Kapitel 2.1 erläutert.

Bewegungen des Nutzers werden durch einen oder mehrere Sensoren aufgezeichnet und durch die Geräteabstraktion Trame weiterverabeitet. Trame stellt ein uniformes Skelettmodell bereit, dass zur Weiterverarbeitung genutzt werden kann. An dieser Stelle setzt die Gesture Recognition an und greift die bereitgestellten Skelettdaten ab (zu sehen in Abbildung 3.1).

Die Gesture Recognition ist in zwei Pipelines unterteilt, die als hellblaue Linien in der Abbildung 3.1 dargestellt sind. Die Verarbeitung von direktmanipulativen Eingaben arbeitet parallel zu der Interpretation von Gesten (rote Kasten). Als Eingabe dient bei beiden Pipelines das Skelettmodell (Vergrößerter Bereich), das von Trame (orangener Kasten) geliefert wird. Die Verarbeitung direktmanipulativer Eingaben ist in Gleichung 3.1 definiert. Als Ergebnis gibt sie ein kinetisches Hand-Arm-Modell zurück, welches zum Beispiel durch eine Physikengine verarbeitet werden kann. Das Ergebnis der Interpretation ist ein Gestenobjekt, woraus sich die Funktionsdefiniton 3.2 ergibt.

$$DirectManipulative := Skeleton[] \rightarrow KineticHandArmModel \tag{3.1}$$

$$Interpretation := Skeleton[] \rightarrow GestureObject \tag{3.2}$$

Ein Service stellt die Ergebnisse der beiden Pipelines anschließend für eine Weiterverarbeitung zur Verfügung, dargestellt als blauer, abgerundeter Kasten. Die gesamte Gesture Recognition, sowie auch Trame, können als Bibliothek in ein Projekt eingebunden werden oder selbstständig als dedizierter Service über eine Netzwerkschnittstelle kommunizieren.

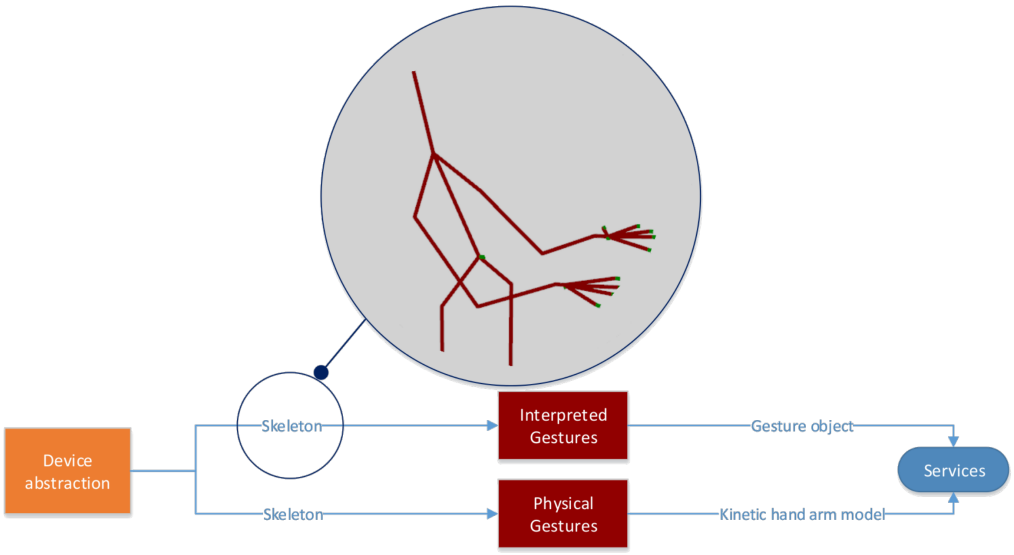


Abbildung 3.1: Übersicht der Verarbeitungspipeline mit Geräteabstraktion Trame (blau) und Gesture Recognition. Zur Gesture Recognition gehören die Pipelines (hellblau), Modellerstellung (rot), Interpretierung von Gesten (rot) und die Ausgabe (blau) (Quelle: http://i2e.informatik.haw-hamburg.de)

3.3 Geräteabstraktionsschicht

Ein wichtiger Bestandteil der Gesture Recognition sind verlässliche Sensordaten. Die Vergangenheit hat gezeigt, dass es innerhalb von kurzer Zeit zu großen Änderungen in diesem Bereich kommen kann. So ist beispielsweise das Opensource-Projekt OpenNI von PrimeSense nicht mehr direkt verfügbar und auch die Erweiterung Nite zur Erkenung von Skeletten kann nicht mehr verwendet werden. Bei der LEAP Motion wurde die API mit einer neuen Version des SDKs verändert. Diese Entwicklung macht deutlich, dass eine Gestenerkennung von den Sensordaten abstrahiert werden sollte. Eine Gestenerkennung ist allgemeiner als ein Sensordevice und sollte somit nicht direkt mit Daten der Sensoren arbeiten. An diesem Punkt setzt *Trame* an. Es ist eine Abstraktionsschicht für Sensordaten von verschiedenen Sensoren, die die räumliche Position von Skelettdaten ermitteln. Dabei ist es egal, ob die Sensoren am Körper getragen oder die Daten durch ein Kamerasystem ermittelt werden.

Bereits in [EGG+03] wurde auf die Verwendung einer Abstraktionsschicht eingegangen, um bei der Erkennung von Gesten unabhängig von den eigentlichen Sensoren zu sein. Diese Idee

soll auch in dieser Arbeit aufgegriffen werden. Als Abstraktionsschicht kommt Trame[1] zum Einsatz. Durch Trame kann die Gesture Recognition immer auf dem gleichen Skelettmodell arbeiten und ist unabhängig von Änderungen der Sensoren oder der Schnittstellen.

Dieser Abschnitt behandelt die Abstraktionsschicht für Eingabegeräte. Zunächst wird das Skelettmodell in Abschnitt 3.3.1 beschrieben. In Abschnitt 3.3.2 wird der Aufbau und die Funktionsweise erklärt und zuletzt werden die Vorteile bei der Verwendung der Abstraktionsschicht aufgezeigt, siehe Abschnitt 3.3.3.

3.3.1 Skelettmodell

Trame stellt ein Skelett zur Verfügung, dass neben ID und Zeitstempel auch Informationen zur Validität eines Skelettes bietet. Ein Skelett, wie in Abbildung 3.2 dargestellt, besteht aus mindestens 18 Gelenken die in einer Map-Struktur im Speicher gehalten werden. Jedes Gelenk besitzt eine eindeutige ID und kann anhand dieser ID mit einer Komplexität von $O(1)$ gelesen werden.

Jede ID besitzt in ihr kodierte Informationen, sodass auf Basis der ID Gruppen gebildet werden können. So kann bspw. zwischen linker und rechter Seite, Oberkörper und Beinen oder den Händen unterschieden werden.

Der Körperschwerpunkt wird als Center bezeichnet und trägt die ID 10000. Jedes Gelenk besitzt neben der ID eine Position im dreidimensionalen Raum und eine Orientierung in Form einer Quaternion. Position und Orientierung sind absolut. Ein Gelenk kann ein oder kein Elterngelenk und kein oder mehrere Kindgelenke besitzen. Viele der Gelenke besitzen genau ein Kindgelenk, wie etwa die Knie, die Ellenbogen oder die Schultern.

In Listing 3.1 wird ein kleiner Ausschnitt (nur drei Gelenke) aus einem vollständigen Skelett gezeigt. Ein wichtiger Wert ist neben der Orientierung und der Position die ID eines Gelenks.

3.3.2 Aufbau und Funktionsweise von Trame

Trame besteht aus den drei Komponenten trame, trame.serialization und trame.skeleton, die im Folgenden näher beschrieben werden.

trame

Die Hauptkomponente *trame* gibt nach außen das Interface ICameraAbstraction, auf das sich Nutzer über ein Eventsystem mit einem Callback einschreiben können. Der Callback wird anschließend immer dann aufgerufen, wenn neue Daten vorhanden sind. Diese Daten können

[1] Git-Repository: https://github.com/i2e-haw-hamburg/trame

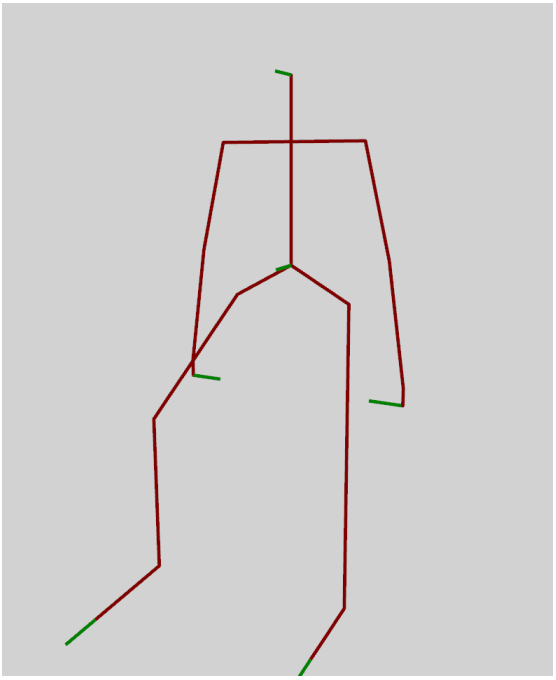


Abbildung 3.2: Eine Visualisierung der Skelettdaten durch den in Trame enthaltenen skeleton-viewer (Quelle: eigene Arbeit).

dann zur Weiterverarbeitung verwendet werden. Zusätzlich besitzt das Interface Methoden zum expliziten Starten und Stoppen der unterliegenden Devices. Damit kann sichergestellt werden, dass alle Resourcen vor Beendigung freigegeben werden, zu sehen in Abbildung 3.3.

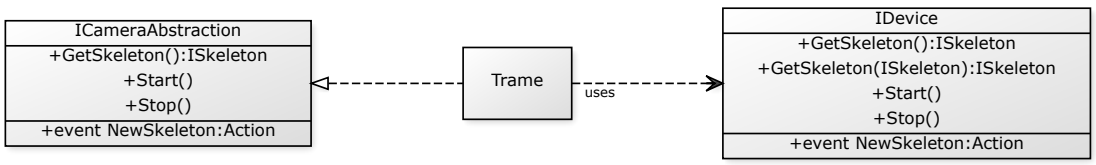


Abbildung 3.3: Klassendiagram von trame (Quelle: eigene Arbeit).

Die Hauptkomponente besitzt die Implementierung für verschiedene Sensoren, die über das Interface IDevice gekapselt sind. Devices können unterschiedliche Adapter verwenden, um kompatibel zu verschiedenen SDK-Versionen zu sein. Adapter kapseln die Logik zur Ansteuerung der Devices und unterscheiden sich von Device zu Device in ihrem Umfang.

Trame nutzt Dependency Injection und Inversion-of-Control, um konfigurierbar zu sein. Dieses Konzept erlaubt auch eine nachträgliche Erweiterung um neue Devices durch Dritte.

trame.serialization

Trame ist als Bibliothek konzipiert und gut für die Verwendung als eigenständigen Service geeignet. Zur einfachen Kommunikation über das Netzwerk steht eine Serialisierungskomponente bereit. Mit ihr ist es möglich, Skelettdaten in verschiedenen Formaten zu serialisieren und zu deserialisieren. Es werden verschiedene Formate unterstützt. Ein einfaches Format ist die sprachabhängige Serialisierung des .NET-Frameworks. Zusätzlich werden sprachunabhängie Formate, wie JSON[2] und Protobuf[3] unterstützt. Weitere Formate können durch Dritte jederzeit registriert werden. Als sinnvolle Erweiterungen wären BSON[4] und YAML[5] zu nennen.

trame.skeleton

Die Skeleton-Komponente umfasst Funktionalitäten zum Erstellen, Bearbeiten und Vergleichen von Skeletten und das Skelettmodell selbst. Zusätzlich ist eine Mathematik-Bibliothek mit erweiterten Konzepten für Vektoren, Matrizen und Quaternions integriert, sodass trame.skeleton[6] unabhängig von anderen Bibliotheken verwendet werden kann.

[2] http://www.json.org/
[3] https://developers.google.com/protocol-buffers/
[4] http://bsonspec.org/
[5] http://yaml.org/
[6] Git-Repositiory: https://github.com/i2e-haw-hamburg/trame.skeleton

```
{
  "id": 1,
  "root": {
    "children": [
      {
        "children": [{
          "children": [...],
          "orientation": [1, 0, 0, 0],
          "position": [0, 1280, 0],
          "type": 1
        }],
        "orientation": [1, 0, 0, 0],
        "position": [0, 1550, 0],
        "type": 5 // type of joint
      },...
    ],
    "orientation": [1, 0, 0, 0],
    "position": [0, 1100, 0], // absolute coordinates
    "type": 10000
  },
  "timestamp": 4215765565 // unix timestamp of creation
}
```

Listing 3.1: Ausschnitt aus serialisiertem Skelett in JSON-Format

Das Skelettmodell wurde bereits im vorherigen Abschnitt behandelt. Eine formale Beschreibung des Modells mithilfe von UML ist der Abbildung 3.4 zu entnehmen. Aus Modellsicht entsteht aus den gegebenen Interfaces eine Baumstruktur. Dieses Verhalten ist bewusst gewählt und erleichtert die Verwendung von Teilabschnitten des Skelettes. So kann sehr einfach der komplette Arm referenziert werden.

Performance-Messungen haben ergeben, dass Zugriffe auf eine Baumstruktur nicht so performant umgesetzt werden können, wie Zugriffe in einer Map ($O(\log n)$ respektive $O(1)$). Der steigende Verwaltungsaufwand für eine Map ist bei einer kleinen Anzahl von Elementen ($n < 100$) gering und kann deshalb vernachlässigt werden.

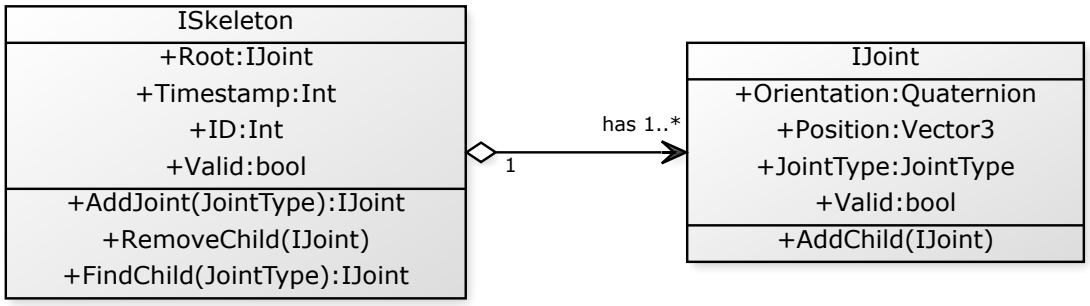


Abbildung 3.4: Klassendiagram von trame.skeleton (Quelle: eigene Arbeit).

3.3.3 Vorteile

Trame besitzt eine Reihe von Vorteilen, die die Arbeit mit Skeletten im Kontext von Gestenerkennung vereinfachen. Die wichtigsten werden nachfolgend aufgelistet und kurz erläutert.

Unterstützung verschiedener Sensoren

Trame erlaubt die Verwendung verschiedener Sensoren[7], SDK-Versionen und Treibern und erstellt eine uniformes Skelettmodell aus den Daten. Diese Daten können von überliegenden Applikationen verwertet werden, ohne eine ständige Anpassung vornehmen zu müssen.

Einfache Erweiterbarkeit

Das Konzept von Trame zielt auf eine einfache Erweiterbarkeit und bietet viele Schnittstellen, über die neue Sensoren und Treiber integriert werden können. Je nach Anwendungsfall müssen dafür neue Device-Klassen oder Adapter-Klassen implementiert werden. Diese können der Abstraktionsschicht anschließend durch Dependency Injection übergeben werden. Es ist ebenso möglich, neue Serialisierungsformate zu definieren und zu implementieren.

Nutzung mehrerer Sensoren zur Auswertung

Trame unterstützt die Fusion von Sensordaten auf Skelettebene. Somit ist es möglich, verschiedene Sensoren zur gleichen Zeit zu verwenden und die Daten als ein gemeinsames Skelett zurückzugeben. Ein Anwendungsfall wäre die Kombination von Kinect-Daten mit Daten der LEAP Motion. Somit kann ein komplettes Skelett des Nutzers einschließlich eines feingranularen Handskelettes genutzt werden.

[7] PrimeSense Carmine, LEAP Motion v1, v2 und Orion, Microsoft Kinect v1 und v2 (Stand 21.07.2016)

Die Fusion geschieht in zwei Schritten. Zunächst werden die Skelettmodelle von Kinect und LEAP Motion ermittelt und anschließend werden sie kombiniert. Die Kombination erfolgt an der Basis der Hände. Die Orientierung wird dabei von den jeweiligen Sensoren übernommen. Eine vorherige Kalibrierung könnte noch genauere Ergebnisse erzielen.

Schnelle Verarbeitung

Eine Abstraktion mit seiner Indirektion führt zu einem langsameren Laufzeitverhalten als eine direkte Nutzung der Schnittstellen. Bei der Entwicklung wurde darauf geachtet, so wenig Zeit wie möglich für die Verarbeitung zu verwenden. In dem zur Verfügung gestellten GitHub-Repository von Trame können verschiedene Messungen zum Laufzeitverhalten betrachtet werden.

Generierung von Standardskeletten

Sensoren liefern nicht immer ein korrektes Ergebnis, sondern zum Teil invalide Daten. Trame erkennt solche Fehler und führt das Ergebnis in einen Initialwert zurück, sodass ein Programm, dass Trame verwendet, sich nicht zwangsläufig um die Validierung kümmern muss.

Das Standardskelett basiert dabei auf der Veröffentlichung zur Untersuchung von menschlichen Proportionen und Maßen in der heutigen Gesellschaft (siehe [Jür04]). Kann ein Sensor, wie die LEAP Motion, nur Teile eines Skelettes wiedergeben, dann wird das Standardskelett verwendet, um die fehlenden Daten bereitzustellen.

3.4 Gesture Recognition

Dieser Abschnitt behandelt die Verarbeitung der durch die Geräteabstraktion Trame bereitgestellten Skelettdaten (siehe Abschnitt 3.3). Es wird auf die Architektur der Gesture Recognition (Abschnitt 3.4.1), auf die technische Umsetzung (Abschnitt 3.4.2) und auf die Bereitstellung der Daten eingegangen, siehe Abschnitt 3.4.3.

3.4.1 Architektur

Die Gesture Recognition abstrahiert von der internen Verarbeitung und den Pipelines mit einem Event- oder auch Callbacksystem, siehe Abbildung 3.5. Es stellt einen Callback zur Verfügung, um neue Skelettdaten in die Gesture Recognition zu laden und erlaubt die Registrierung von Callbacks für durchgeführte Gesture Objects. Sobald ein Gesture Object oder auch Command erstellt wurde, werden alle registrierten Callbacks aufgerufen. Skelettdaten werden in der Regel

von Trame geliefert, könnten aber auch aus einer Datenbank oder einer anderen Datenquelle geladen werden.

Ein Controller in der Gesture Recognition sortiert die Skelettdaten in eine freie Verarbeitungspipeline. Der Controller ist dafür verantwortlich, dass eintreffende Skelettdaten nur mit Skelettdaten des gleichen Nutzers verarbeitet werden. Jeder Nutzer besitzt in der Gesture Recognition eine eigene Pipeline.

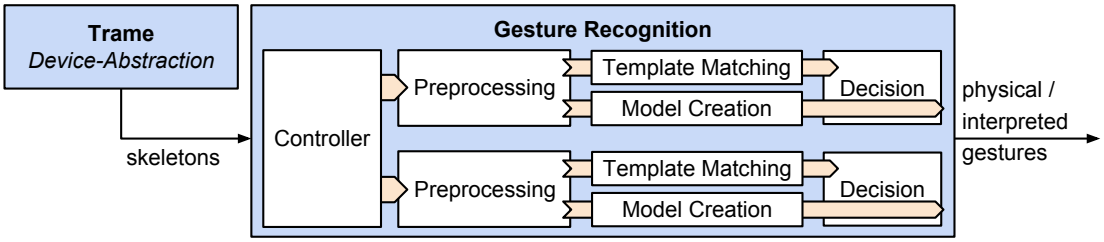


Abbildung 3.5: Übersicht über Architektur der Gesture Recognition und Zusammenspiel mit Nachbarsystemen (Quelle: eigene Arbeit).

Der erste Schritt innerhalb einer Pipeline dient der Vorverarbeitung der Skelettdaten (siehe Abbildung 3.6). In diesem Schritt der Vorverarbeitung werden Messfehler und Zittern in den Daten reduziert, indem der Mittelwert der Skelettbewegung innerhalb eines Fensters von Frames berechnet wird. Um die Anzahl der Frames nicht zu reduzieren, wird ein *Sliding Window* verwendet. Das Ergebnis wird für alle weiteren Berechnungen genutzt. Durch diese Maßnahmen soll die Robustheit der Lösung gesteigert werden.

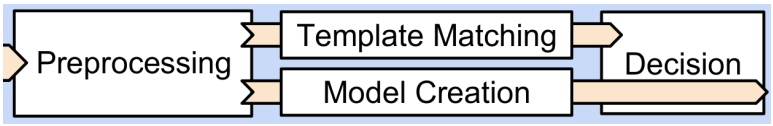


Abbildung 3.6: Aufbau einer Pipeline der Gesture Recognition (Quelle: eigene Arbeit).

Anschließend werden die Skelettmodelle sowohl in die Pipeline zur Interpretierung mithilfe von Templatematching als auch in die Pipeline zur Erstellung eines kinetischen Hand-Arm-Modells, welches zur Durchführung direktmanipulativer Interaktion benötigt wird, geladen. Das Hand-Arm-Modell wird direkt von der Gesture Recognition an entsprechende Callbacks propagiert. Das Ergebnis des Templatematchings wird in einem weiteren Schritt, der Decision, verarbeitet und gibt nur unter der Voraussetzung, dass ein neues Command erkannt wurde, seine Daten weiter an die Gesture Recognition.

3.4.2 Umsetzung

Die gesamte Gesture Recognition ist als Pipeline aufgebaut, wie in den Abbildungen 3.5 und 3.6 zu sehen ist. Die *Gesture Recognition* implementiert das Interface *IGestureRecognition*. Als Input kann der Handle OnNewSkeleton verwendet werden. Um sich auf eine Geste oder ein PhysicCommand einzuschreiben, kann man sich mit einem Callback auf das entsprechende Command registrieren.

Intern nutzt GestureRecognition eine Implementation von *IController*, um die Skelettdaten auf die zuvor initialisierten Pipelines zu verteilen (siehe Abbildung 3.7). IController kapselt dabei die Verarbeitung innerhalb der Pipeline. Die Berechnung der direktmanipulativen Interaktion (Abschnitt 3.5) und die Gesteninterpretation werdem parallel ausgeführt. Somit kann die Zeit, die für die Verarbeitung einer Eingabe benötigt wird, reduziert werden. Die Implementierung greift dabei auf das Concurrent-Producer-Consumer-Pattern zurück.

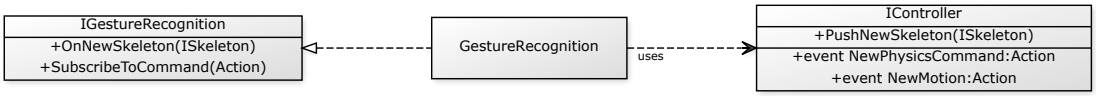


Abbildung 3.7: Klassendiagramm der Gesture Recognition (Quelle: eigene Arbeit).

3.4.3 Bereitstellung der Daten

Die erzeugten Commands aus den beiden Pipelines werden an alle Subscriber, die sich auf die entsprechenden Commands eingeschrieben haben, verteilt. Dabei wird das PhysicsCommand mit dem kinetischen Hand-Arm-Modell direkt weitergegeben. Gefunde Gesten werden zunächst in einer weiteren Stufe, dem Decider, ausgewertet und nur weitergegeben, wenn alle Prädikate, etwa die Überschreitung von Schwellwerten, erfüllt sind.

3.5 Direktmanipulative Interaktion

In diesem Abschnitt wird auf die Verarbeitung der direktmanipulativen Interaktion eingegangen. Wie bereits in Abschnitt 2.4 erläutert, wird ein modellbasierter Ansatz verfolgt. Zu Beginn wird das verwendete Hand-Arm-Modell beschrieben (Abschnitt 3.5.1). In Abschnitt 3.5.2 werden die zur Berechnung der Daten für die Kollider aus den Rohdaten des Skelettmodells. Anschließend wird Konzept vorgestellt, mit dem die Toleranz in den Latenzen nicht überschritten wird kann (Abschnitt 3.5.3).

3.5.1 Kinetisches Hand-Arm-Modell

Das kinetische Hand-Arm-Modell beschreibt eine Anzahl vorgegebener regelmaßiger Elemente der Hand und des Unterarms. Jedes Element des Modells kann als *BodyPart* beschrieben werden und repräsentiert einen Knochen im Skelettmodell. Neben Position, Rotation und Länge besitzen die BodyParts eine Geschwindigkeit, Beschleunigung, Winkelgeschwindigkeit und Winkelbeschleunigung, zu sehen in Abbildung 3.8. Zusätzlich haben sie eine innerhalb eines Modells eindeutige ID. Es stehen ebenfalls die Identifier für den vorherigen und den nachfolgenden BodyPart zur Verfügung.

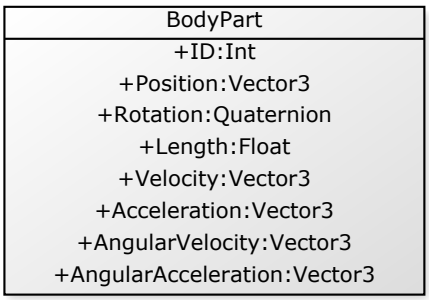


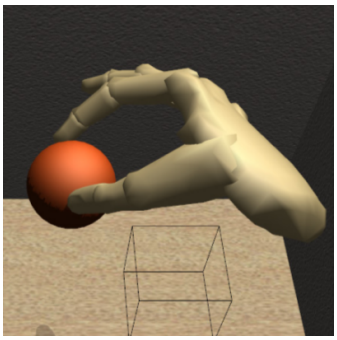
Abbildung 3.8: Klassendiagramm eines BodyParts (Quelle: eigene Arbeit)

Dieses Modell können Physikengines nutzen, um ein Handmodell abzuleiten und dadurch eine physikalisch korrekte Interaktion zwischen den Händen eines Nutzers und einem virtuellen, dreidimensionalen Objekt zu erzeugen. Abhängig von der Umsetzung in der entsprechenden Applikation kann das entstehende Modell unterschiedliche Charakteristiken besitzen. Die Gesture Recognition liefert dabei das kinetische Hand-Arm-Modell, aus dem die Applikation Kollider erstellt und diese an eine Physikengine übergibt. Ein Beispiel für unterschiedliche Umsetzungen des gleichen Hand-Arm-Modells sind die Ansätze von [OKA11, PB11] sowie das Handmodell der Leap Motion. Die Ergebnisse ihrer Arbeiten können in Abbildung 3.9 verglichen werden.

3.5.2 Berechnung

Dieser Abschnitt erläutert die Berechnung der Daten, die von der Physikengine benötigt werden, um eine realistische Simulation zu erzeugen. Dabei werden Geschwindigkeit $\overrightarrow{v}$, Beschleunigung $\overrightarrow{a}$, Winkelgeschwindigkeit $\overrightarrow{\omega}$ und Winkelbeschleunigung $\overrightarrow{\alpha}$ jeweils vektoriell betrachtet. Diese Werte sind für jeden BodyPart separat zu berechnen. Da es keine direkten Abhängigkeiten gibt, können alle BodyParts vollständig parallel berechnet werden.

(a) Darstellung eines approximierten Handmodells durch Ellipsoide (Quelle: [OKA11])



(b) Darstellung eines approximierten Handmodells durch Ellipsoide und unregelmäßige Körper (Quelle: [PB11])

(c) Darstellung eines approximierten Handmodells durch Kugeln und Säulen (Quelle: Leap Motion Inc)

Abbildung 3.9: Vergleich zweier Ansätze Approximation eines Handmodells

Ein BodyPart besitzt eine Position $\overrightarrow{P}$, repräsentiert durch einen dreidimensionalen Vektor und eine Rotation q, repräsentiert durch eine Quaternion, zu sehen in Abbildung 3.8.

Die Geschwindigkeit $\overrightarrow{v}$ kann über die Position ($\overrightarrow{P_0}$ und $\overrightarrow{P_1}$) zweier BodyParts und die zeitliche Differenz Δt der beiden zugrunde liegenden Skelettmodelle berechnet werden (siehe Gleichung 3.3).

$$\overrightarrow{v}(\overrightarrow{P_0}, \overrightarrow{P_1}, \Delta t) = \frac{\overrightarrow{P_1} - \overrightarrow{P_0}}{\Delta t} \tag{3.3}$$

Die Beschleunigung $\overrightarrow{a}$ eines BodyParts ist die Änderung der Geschwindigkeit über die Zeit. Gleichung 3.4 verdeutlicht diesen Zusammenhang, wobei $\overrightarrow{P_0}$, $\overrightarrow{P_1}$ und $\overrightarrow{P_2}$ die Position eines BodyParts zum Zeitpunkt t_i ist. $\Delta t_i = t_{i+1} - t_i$ ist die zeitliche Veränderung.

$$\overrightarrow{a}(\overrightarrow{P_0}, \overrightarrow{P_1}, \overrightarrow{P_2}, \Delta t_0, \Delta t_1) = \frac{\overrightarrow{v}(\overrightarrow{P_1}, \overrightarrow{P_2}, \Delta t_1) - \overrightarrow{v}(\overrightarrow{P_0}, \overrightarrow{P_1}, \Delta t_0)}{\Delta t_1} \tag{3.4}$$

Zur Beschreibung von Rotationen werden Quaternionen verwendet. Eine Quaternion besteht aus vier Elementen (Gleichung 3.5). Die Differenz zwischen zwei Quaternionen kann durch eine Multiplikation des einen Quaternions mit dem Inversen des zweiten Quaternions berechnet werden.

$$\begin{aligned} q &:= [q_0, q_1, q_2, q_3] \\ \Delta q(q_a, q_b) &= q_a * q_b^{-1} \end{aligned} \tag{3.5}$$

Um die Winkelgeschwindigkeiten in allen drei Dimensionen zu berechnen, müssen zunächst ϕ, θ und ψ definiert werden. Sie werden als Eulerwinkel bezeichnet und geben die Rotation um x-, y- und z-Achse an (Gleichung 3.6).

$$\begin{aligned} \phi(q) &= atan2(2(q_0 q_1 + q_2 q_3), 1 - 2(q_1^2 + q_2^2)) \\ \theta(q) &= \arcsin(2(q_0 q_2 - q_3 q_1)) \\ \psi(q) &= atan2(2(q_0 q_3 + q_1 q_2), 1 - 2(q_2^2 + q_3^2)) \end{aligned} \tag{3.6}$$

Mithilfe der Eulerwinkel kann die Winkelgeschwindigkeit $\overrightarrow{\omega}$ berechnet werden. Sie ist als Eulerwinkel ϕ, θ und ψ der Differenz zweier Quaternionen Δq über die Zeit Δt, wie in Gleichung 3.7.

$$\overrightarrow{\omega}(\Delta q, \Delta t) = \begin{pmatrix} \phi(\Delta q) \\ \theta(\Delta q) \\ \psi(\Delta q) \end{pmatrix} / \Delta t \tag{3.7}$$

Analog zur $\overrightarrow{a}$ eines BodyParts wird die $\overrightarrow{\alpha}$ als Änderung der Winkelgeschwindigkeit $\overrightarrow{\omega}$ über die Zeit definiert. Gleichung 3.8 macht diesen Zusammenhang deutlich.

$$\overrightarrow{\alpha}(\Delta q_a, \Delta q_b, \Delta t_0, \Delta t_1) = \frac{\overrightarrow{\omega}(\Delta q_b, \Delta t_1) - \overrightarrow{\omega}(\Delta q_a, \Delta t_0)}{\Delta t_1} \tag{3.8}$$

3.5.3 Vermeidung von Versatz

Für eine direktmanipulative Interaktion muss eine Lösung genau und schnell arbeiten. Eine interaktive Antwortzeit bedeutet, dass ein neues PhysicsCommand nicht später als 100 ms abgesendet werden darf. Da zusätzlich noch eine Verarbeitung durch die Applikation und die Physikengine erfolgt, sollte ein Wert von etwa 50 ms als Grenze gewählt werden. Die Berechnung der einzelnen BodyParts kann performant implementiert werden. Ein Versatz entsteht durch die Nutzung eines Mittelwertes zur Glättung der Bewegungen. Geht man von einer Framerate von 50 Hz und einer Fenstergröße von drei Bildern aus, beträgt der Versatz in etwa 48 ms. Ein zusätzlicher Versatz von einem Frame entsteht, wenn die Beschleunigung eines Körpers berechnet wird, da dafür die Geschwindigkeit des vorherigen Elementes notwendig ist. Somit entsteht ein Versatz, der Größer als 50 ms ist.

Um dieses Problem zu lösen, wird nicht gewartet, bis alle Skelette zur Verfügung stehen, um die Glättung innerhalb eines Fensters durchzuführen, sondern es werden nicht vorhandene Skelette mit dem Standardskelett ersetzt. Somit müssen in der Programmierung keine Sonderfälle

betrachtet werden. Das gleiche Konzept kann ebenfalls zur Berechnung der Beschleunigung genutzt werden.

Durch dieses Konzept kann ein zu hoher Versatz zwischen Eingabe und Ausgabe vermieden werden.

3.5.4 Abschluss

In diesem Abschnitt wurde das Konzept zur Verarbeitung von Eingaben zur direktmanipulativen Interaktion von dreidimensionalen Objekten erläutert. Es wurde auf die Aufbereitung der Daten zur Nutzung in einer Physikengine eingegangen. Zusätzlich wurde eine Strategie zur Vermeidung eines zu großen Versatzes vorgestellt.

Die Gliedmaßen zwischen den Gelenken können durch Ellipsoide, Säulen oder Quader approximiert werden. Eine Approximation durch Quader hat den Vorteil, dass eine größere Kontaktfläche zwischen virtuellem Objekt und BodyPart entsteht, als es bei abgerundeten Flächen der Fall ist (siehe Abbildung 3.10. Durch die Verwendung dieser einfachen Approximation kann ein zu großer Rechenaufwand vermieden werden, wie es etwa bei der Partikelverfolgung der Fall gewesen wäre (vgl [HKI+12]). Im Gegensatz zu [SYW08], bei dem nur eine Fingerspitze verfolgt wurde, erhält dieses Modell einen größeren Detailgrad, der ein natürlicheres Verhalten bei der Interaktion mit virtuellen Objekten bewirken soll. Auf Basis der bereitgestellten Daten kann ebenfalls eine kombinierte Lösung, auf Basis von modellbasiertem Ansatz mit Partikelsystem, genutzt werden (vgl. [KP15]). Die Umsetzung eines Masse-Feder-Systems ist ebenso möglich (siehe Abschnitt 2.2.4), liegt aber außerhalb des Scopes dieser Arbeit.

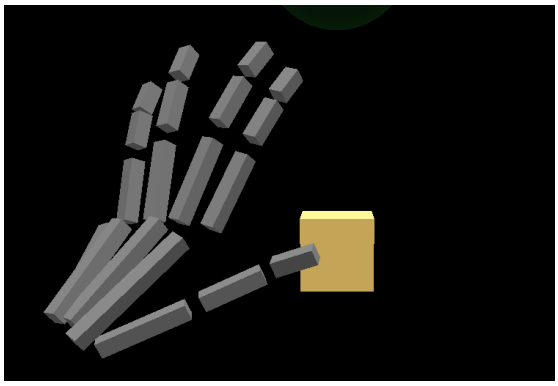


Abbildung 3.10: Darstellung eines approximierten Handmodells durch Quader (Quelle: eigene Arbeit)

3.6 Interpretierte Gesten

Die Interpretation von Gesten ist Gegenstand dieses Abschnittes. Es wird gezeigt, wie Gesten auf Basis von Trajektorie beschrieben werden (Abschnitt 3.6.1) und wie die Erkennung mittels Distanzfunktionen durchgeführt wird (Abschnitt 3.6.2). In Abschnitt 3.6.3 wird auf den Entscheidungsprozess eingegangen. Eine abschließende Bewertung der Lösung befindet sich in Abschnitt 3.6.4.

3.6.1 Beschreibung einer Geste

Die Beschreibung einer Geste besteht im Wesentlichen aus drei Elementen. Es gibt eine Anfangs- und Endbedingung sowie einen dynamischen Anteil, der die Bewegung des Skelettmodells beschreibt. Das Konzept unterstützt sowohl Hand- als auch Armgesten. Aus diesem Grund beinhaltet die Beschreibung einer Geste auch die Information über die zugehörigen Körperteile. Abbildung 3.11 zeigt die Beschreibung für eine Geste formal. Zusätzlich zu der eigentlichen Information über die Bewegung beinhaltet die Geste auch das zugehörige Command, also der Bedeutung der Geste.

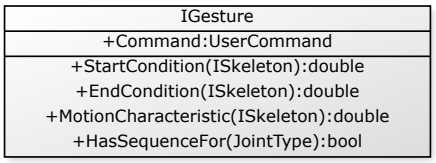


Abbildung 3.11: Klassendiagramm eines BodyParts (Quelle: eigene Arbeit)

Die Start- und Endbedingungen beschreiben eine Pose, die über das System erkannt werden kann. Die unterstützten Posen sind abhängig vom genutzten Sensor. Es werden *geschlossene Hand*, *Zangengriff*, *zueinander zugewandte Handflächen*, *linke Hand in Eingabebereich*, *rechte Hand in Eingabebereich* und *Handfläche dem Gesicht zugewandt* als Posen erkannt. Die Sicherheit, mit der eine Pose erkannt wurde, ist der Wahrscheinlichkeitswert für das Zutreffen der Start- bzw Endbedingung.

Als Beispiel für die Bereitstellung einer Geste wird die Geste zur Skalierung verwendet. Eine Skalierung kann nur stattfinden, wenn ein Objekt zwischen den beiden zueinander zugewandten Händen exisitert. Sie endet, wenn sich das Objekt nicht mehr zwischen den Händen befindet oder die Hände nicht mehr zugewandt sind. Eine Bewegung kann nur linear auf der von den beiden Händen definierten Geraden verlaufen. Auf Basis dieser Beschreibung kann eine Geste erstellt werden. Die Startbedingung ist die Pose *zueinander zugewandte Handflächen*. Da die Gesture Recognition nicht über das Wissen der Applikation verfügt, ist der zweite Teil

der Bedingung (Objekt zwischen den Händen) nicht relevant. Die Endbedingung kann parallel dazu definiert werden und ist die negierte Form der Startbedingung. Die Bewegung kann in zwei Segmenten pro Hand definiert werden und startet jeweils im Koordinatenursprung.

Die Gesture Recognition besitzt keinen Kontext der Applikation und kann deshalb nicht entscheiden, ob sich ein Objekt zwischen den Händen befindet. Die Applikation ist für diese Prüfung verantwortlich. In dem bereitgestellten ScaleCommand werden alle nötigen Informationen weitergegeben. Es werden die aktuellen Positionen der Hände und des Zentrums der Geste sowie der Skalierungsfaktor übergeben. Der Skalierungsfaktor ist das Verhältnis zwischen der Entfernung der beiden Hände zu Beginn der Geste und der Entfernung zum Ende der Geste, wie in Gleichung 3.9 gezeigt. P ist die Position als Vektor. Die Indizes von P werden genutzt, um die Seite der Hand und den Zeitpunkt, an dem die Hand an entsprechnder Position gewesen ist, zu definieren,

$$Scale = \frac{|\overrightarrow{P}_{right,t_n} - \overrightarrow{P}_{left,t_n}|}{|\overrightarrow{P}_{right,t_0} - \overrightarrow{P}_{left,t_0}|} \tag{3.9}$$

3.6.2 Gesten-Matching

Das Matching ist der Vergleich zwischen einem Strom von Eingabedaten und jedem Element in einem vorhandenen Gestenset. Jede mögliche Geste wird dabei mit der Eingabe verglichen. Als Ergebnis entstehen drei Wahrscheinlichkeitswerte pro Geste. Jeweils ein Wert für Startbedingung, Bewegung und Endbedingung. Die Ergebnisse werden gesammelt und an den Entscheidungsalgorithmus weitergegeben.

Um unnötige Berechnungen zu vermeiden, werden Gesten, die eine Wahrscheinlichkeit der Startbedingung $P(start) < 0,5$ besitzen ebenso aussortiert, wie Gesten, die für die vorhandenen Gelenkpunkte keine Sequenz besitzen.

Der Vergleich zwischen Geste ω_j und Eingabe I erfolgt auf Basis der Arbeit von Kristensson et al., wurde jedoch angepasst, um auch dreidimensionale Bewegungen als Eingabe zu erlauben. Dazu werden Eingabe und Gesten in einer vorgesetzen Schritt ausgerichtet (vgl. [AB10]) und normalisiert. Die Ausrichtung erfolgt dabei durch die Rotation der Eingabe um die y-Achse, sodass das erste Segment der Eingabe mit dem ersten Segment der Geste deckungsgleich ist. Das Verhältnis aus Länge eines Segmentes zur Gesamtlänge ist die normalisierte Länge des Segmentes. Durch diese beiden Maßnahmen sind die Gesten skalierungsinvariant. Eine Rotationsinvarianz gilt jeweils nur für die obere und untere Halbschale.

Im ersten Schritt wird die Wahrscheinlichkeit $P(I_i|\omega_j)$ für jede Geste ω_j im Gestenset ω anhand der Nutzer-Eingabe I berechnet. Die Eingabe I besteht dabei aus n Punkten und I_i ist

ein Teilinput mit $0 < i \leq n$. Gleichung 3.10 beschreibt die Berechnung der Wahrscheinlichkeit. S_j ist dabei die Menge aller Teilsequenzen und $D()$ ist eine Distanzfunktion.

$$P(I_i|\omega_j) = arg \max_{S_k \in S_j \in \omega_j} D(I, S) \tag{3.10}$$

Im zweiten Schritt wird der Satz von Bayes für jede Geste ω_j angewendet, wie Gleichung 3.11 zeigt. Die a-priori-Wahrscheinlichkeit für eine Geste ist $P(\omega_j)$. k ist der Index für die Iteration über das Gestenalphabet. Der Algorithmus wird für jeden Punkt der Eingabe wiederholt und berechnet die Wahrscheinlichkeit für jede Geste.

$$P(\omega_j|I_i) = \frac{P(\omega_j)P(I_i|\omega_j)}{\sum_k P(\omega_k)P(I_i|\omega_k)} \tag{3.11}$$

Als Distanzmaß (Gleichung 3.12) wird der Drehwinkel verwendet. Die Parameter I und S sind zwei Vektoren, die verglichen werden sollen. I repräsentiert die Eingabe des Nutzers und S ist eine Geste. Sie bestehen aus den Punkten $a_1, a_2, ..., a_n$ respektive $b_1, b_2, ..., b_n$. d_t ist der Winkel zwischen zwei Linien in Radiant (vgl. [KD11]).

$$x_t = \frac{1}{n-1} \sum_{i=2}^{n} d_t(a_i, a_{i-1}, b_i, b_{i-1}) \tag{3.12}$$

Das Ergebnis des zweiten Schrittes ist die Wahrscheinlichkeit für die Bewegung einer Geste. Diese wird zusammen mit den Wahrscheinlichkeitswerten für Start- und Endbedingung and die Entscheidungsfunktion weitergereicht.

3.6.3 Entscheidungsalgorithmus

Der Entscheidungsalgorithmus wertet die Ergebnisse der Matchingphase aus und leitet ein UserCommand an die Gesture Recognition weiter, falls eine Geste gefunden wurde. Die Entscheidung basiert dabei auf der größten Gesamtwahrscheinlichkeit für eine Geste. Die Gesamtwahrscheinlichkeit $P(total_j)$ für eine Geste ω_j ist das Produkt der Einzelwahrscheinlichkeiten, wie in Gleichung 3.13 zu sehen.

$$P(total_j) = P(start_j) * P(\omega_j) * P(end_j) \tag{3.13}$$

Die Geste mit der höchsten Wahrscheinlichkeit wird mit einem Schwellwert verglichen. Sollte der Schwellwert überschritten worden sein, wird das zur Geste gehörende UserCommand erstellt und an die Gesture Recognition weitergeleitet. Diese informiert anschließend alle Subscriber über die registrierten Callbacks von dem neuen Command. Zusätzlich werden alle

bisher in der Pipeline des Nutzers enthaltenen Daten entfernt, sodass auf Basis der gleichen Bewegung nicht mehrere Gesten erkannt werden können.

3.6.4 Abschluss

In der vorgestellten Lösung werden Bewegungen im dreidimensionalen Raum ausgewertet und mit vorhandenen Gesten verglichen. Eine Geste besteht dabei aus statischen und dynamischen Elementen. Der dynamische Abschnitt ist als Trajektorie modelliert. Für einen skalierungsunabhängigen Vergleich werden Eingabe und Geste zunächst normalisiert. Eine Ausrichtung erfolgt, um eine hohe Fehlertoleranz gegen Rotation zu erreichen. Der Vergleich erfolgt anschließend auf Basis des Matchingverfahrens aus Kristensson und Denby. Durch die Vorverarbeitung werden nicht nur zweidimensionale Gesten, sondern auch Gesten in allen drei Dimensionen erkannt. Bei der Lösung von Kristensson et al. wurden nur die Handflächen verfolgt. In der vorgestellten Lösung können alle enthaltenen Gelenkpunkte verfolgt und in einer Geste einbezogen werden.

Auf die Schwierigkeiten bei der Erkennung von dreidimensionalen Gesten durch das Start-Ende-Problem, wurde bereits in Abschnitt 2.3 eingegangen. Die Verwendung von Start- und Endbedingungen, der Verwurf von Daten nach einer erfolgreichen Erkennung und der zeitlichen Begrenzung der Validität von Eingabedaten vereinfacht das Problem jedoch, da es den Suchraum deutlich einschränkt.

4 Evaluierung

Das zuvor in Kapitel 3 entwickelte Konzept zur Evaluierung der These 1 wurde prototypisch umgesetzt. Die Ergebnisse wurden als Library auf GitHub[1] frei zur Verfügung gestellt und sind als Nuget-Package[2] nutzbar.

Mithilfe der Library wurde eine Testsetup entwickelt, das verschiedene Szenarien bereitstellt. Die unterschiedlichen Szenarien und das gesamte Testsetting wird in Abschnitt 4.1 vorgestellt. An der Durchführung der Untersuchung haben 20 Probanden teilgenommen. Sie wurden dabei in zwei Gruppen für einen A/B-Test unterteilt. Gruppe A führte alle Aufgaben mit direktmanipulativen Gesten aus. Gruppe B wurde durch interpretierte Gesten unterstützt. Mehr dazu kann in Abschnitt 4.1 nachgelesen werden.

Die abschließende Auswertung der Tests erfolgt in drei Stufen, zu finden in Abschnitt 4.2. Zunächst wurden die Zeitmessungen analysiert, um ein objektives Ergebnis als Grundlage für die endgütlige Bewertung zu erhalten. In einem weiteren Schritt werden alle Fragebögen ausgewertet. Anschließend werden eigene Beobachtungen und zusätzliches Feedback hinzugezogen. Das Ziel ist die Bestätigung oder Widerlegung der These.

4.1 Testsetting

Dieser Abschnitt beschreibt das Testsetting zur Evaluierung der Haupthese dieser Arbeit. Wie bereits eingangs erwähnt, wurde ein A/B-Test entworfen. In dem Test müssen Probanden eine Reihe von Aufgaben in unterschiedlichen Szenarien lösen. In diesem Abschnitt wird erläutert, wie das Feld der Probanden zusammengesetzt ist (Abschnitt 4.1.1), welche Aufgaben sie in den Szenarien lösen müssen (Abschnitt 4.1.2) und wie die Leistungen der Probanden gemessen wird (Abschnitt 4.1.3). Anschließend wird auf den Fragebogen eingegangen, den die Probanden nach Beendigung der Testreihen ausfüllen konnten (Abschnitt 4.1.4).

[1] https://github.com/i2e-haw-hamburg/gesture-recognition
[2] https://www.nuget.org/packages/i2e.gesture-recognition/

4.1.1 Auswahl und Zusammensetzung der Testpersonen

Bei der Auswahl der Probanden wurde darauf geachtet, eine möglichst gleichmäßige Verteilung im Bezug auf die Erfahrung der Probanden mit Mixed Reality zu erreichen. Das Ergebnis sollte auch für Nutzer gelten, die zuvor noch nicht mit einem vergleichbaren System gearbeitet haben.

In einer informellen Befragung gaben 85% der 20 Probanden an, selbst technisch affin oder sehr affin zu sein. 65% besitzen einen akademischen Abschluss. 32% der Probanden waren feminin und 63% maskulin. Das Durschnittsalter lag bei 31 Jahren mit einer Standardabweichung von $\sigma_{Alter} = 9,36$.

4.1.2 Testszenarien

Die Testszenarien sollen Aktionen nachstellen, die ein Nutzer eines Systems zur Manipulation von dreidimensionalen, virtuellen Objekten in einer Mixed-Reality-Umgebung ausführen könnte. Diese Aktionen beinhalten Rotation und Translation von einzelnen Objekten, Komposition von mehreren Objekten und Entfernung von Objekten aus einer Komposition, sind aber nicht auf diese beschränkt. Der Umfang von Aktionen bei der Nutzung von Gesten ist größer als der Umfang von Aktionen von direktmanipulativer Interaktion. So ist es etwa nicht möglich, allein durch direktmanipulative Interaktion ein Objekt zu skalieren oder zu selektieren. Um trotzdem vergleichbar zu sein, wurden die Szenarien so konzipiert, dass sie durch beide Varianten gelöst werden können.

Nachfolgend werden die einzelnen Szenarien vorgestellt und ihre Umsetzung innerhalb einer, zur Evaluierung der These entwickelten, Applikaton gezeigt. Abbildung 4.1 zeigt den Aufbau der Applikation mit dem Menü zur Auswahl der Szenarien auf der linken Seite. Die Applikation erlaubt das Aktivieren und Deaktivieren von zusätzlicher Gesteninterpretation.

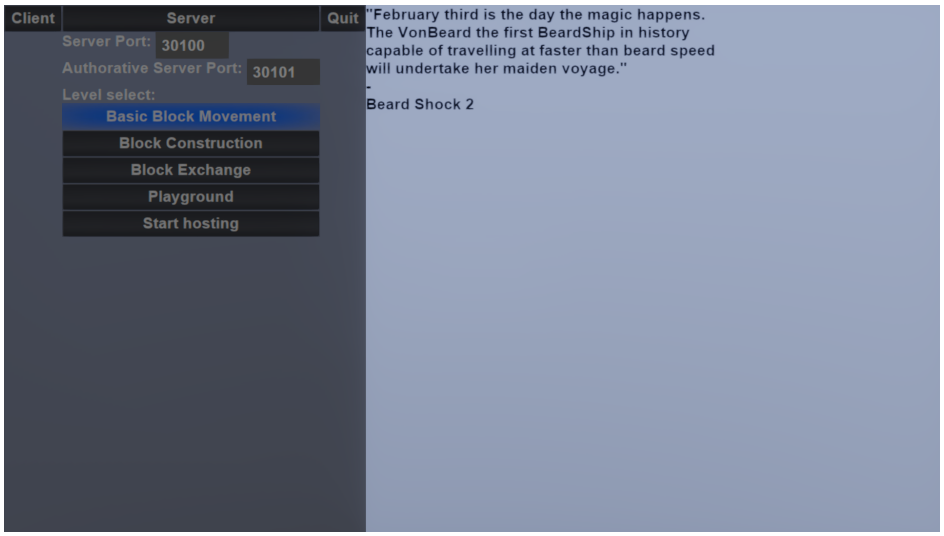


Abbildung 4.1: Hauptmenü der Applikation zur Evaluierung der Arbeit (Quelle: eigene Arbeit).

Szenario 1

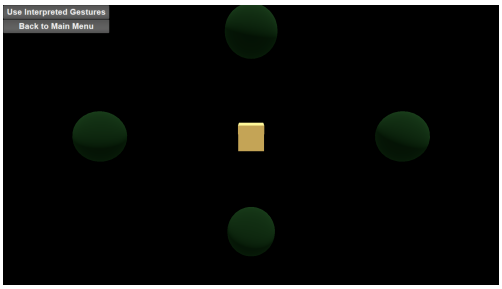
Das erste Szenario verlangt von dem Nutzer die Translation eines Objektes an unterschiedliche Positionen. Die Translationen von Objekten ist ein oft durchgeführter Task in einer Applikation zur Arbeit mit dreidimensionalen Objekten.

Die Umsetzung als Aufgabe für die Probanden wird in Abbildung 4.2 gezeigt. Ein einzelner Baustein muss von der Testperson nacheinander zu vier Markern transportiert werden, die quer über den Arbeitsbereich verteilt sind (Abbildung 4.2a). Sobald die Person den Baustein berührt hat, wird die Zeit gestartet. Der Durchgang endet, wenn der letzte Marker mit dem Baustein berührt wurde (Abbildung 4.2c).

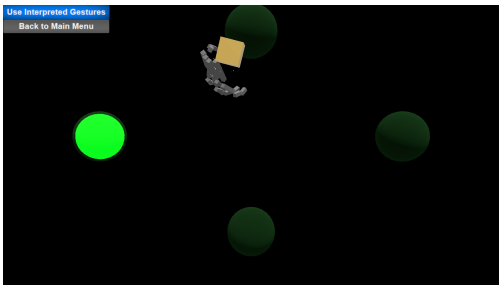
Szenario 2

Der Proband muss im zweiten Szenario Translation und Rotation nutzen, um eine Komposition von virtuellen Objekten zu erhalten. Die Komposition von virtuellen zu einem Konstrukt wird in vielen Bereichen der 3D-Modellierung angewendet.

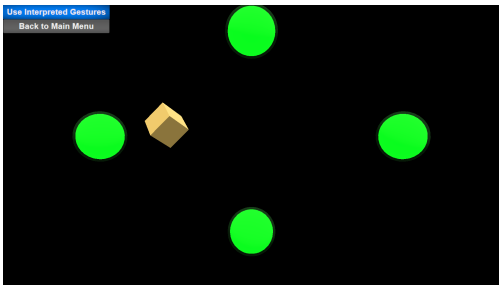
Der Proband muss nacheinander vier Bausteine aus einer Spawn-Area nehmen (Abbildung 4.3a). Sobald ein Baustein entnommen wurde, wird ein neuer erstellt, bis insgesamt vier Bausteine in der Szene vorhanden sind (Abbildung 4.3b). Der Proband hat nun die Aufgabe, die vier Steine zu einer 2x2-Mauer zusammenzufügen (Abbildung 4.3c). Die Zeit beginnt, wenn der erste Baustein berührt wurde und endet mit der Fertigstellung des Konstruktes.



(a) Beginn von Szenario 1: Ein Baustein ist in der Mitte des Arbeitsbereiches, vier Marker sind über den Arbeitsbereich verteilt (Quelle: eigene Arbeit)



(b) Ein Marker wurde aktiviert und der Proband verschiebt den Baustein zum nächsten Marker (Quelle: eigene Arbeit)



(c) Ende von Szenario 1: Alle Marker sind aktiviert worden (Quelle: eigene Arbeit)

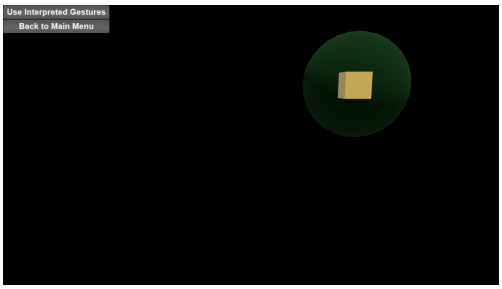
Abbildung 4.2: Szenario 1 in unterschiedlichen Stadien

Szenario 3

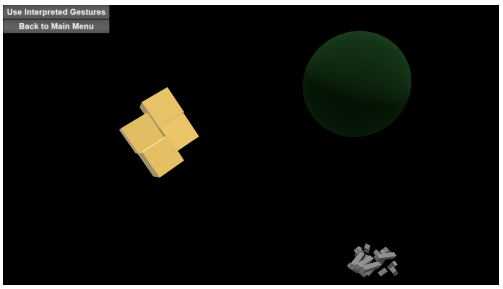
Die Auflösung von Kompositionen und das anschließende Entfernen von virtuellen Objekten ist ein Punkt, der von einem System unterstützt werden muss, wenn es eine Möglichkeit zur Komposition geben soll. In Szenario 3 müssen zwei rot markierte Steine aus einem Konstrukt entfernt und gelöscht werden (Abbildung 4.4a). Zum Löschen eines Bausteins muss dieser in eine markierte Zone geschoben werden. Ziel ist es, die beiden Bausteine so schnell wie möglich zu entfernen, ohne dabei das restliche Konstrukt zu beschädigen (Abbildung 4.4b). Die Zeit wird gestartet, wenn der erste Baustein berührt wurde und endet, wenn der zwei markierte Baustein gelöscht ist (Abbildung 4.4c).

4.1.3 Messungen der Aufgabe

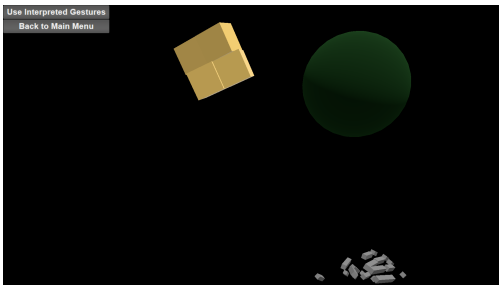
Die Zeiten, die die Probanden für die Lösung einer Aufgabe benötigten wurde erfasst und zur späteren, Auswertung notiert. Es wurde darauf geachtet, dass die Probanden zuvor keine Gewöhnung an das System erhalten hatten. Jeder Proband hatte einen möglichen Versuch pro



(a) Zu Beginn ist in der Spawn-Area (grün) ein Baustein (Quelle: eigene Arbeit)



(b) Der Proband nimmt nacheinander vier Bausteine aus der Spawnarea (Quelle: eigene Arbeit)



(c) Als Ergebnis soll der Proband eine 2x2-Mauer aus den Bausteinen erstellen (Quelle: eigene Arbeit)

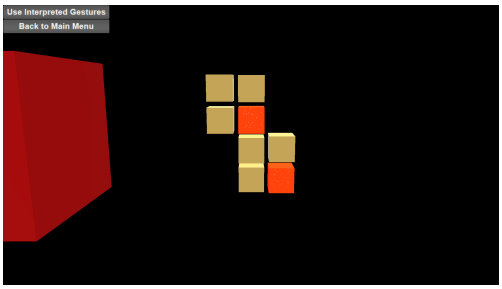
Abbildung 4.3: Szenario 2 in unterschiedlichen Stadien

Szenario. Die Möglichkeit einer automatischen Messung der Ergebnisse wurde verworfen, da eine Auswertung des gewünschten Zielzustandes zu umfangreich gewesen wäre. Vor allem in Szenario 2 würde der Nutzen einer automatischen Zeitmessung nicht mit dem Aufwand der Umsetzung im Verhältnis stehen.

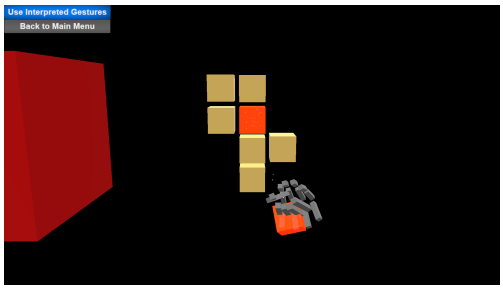
4.1.4 Fragebogen

Zusätzlich zur objektiven, zeitlichen Messung der Aufgaben wurde ein Fragebogen erarbeitet, den die Probanden im Anschluss an die Tests ausfüllen konnten. Der Fragebogen wurde ausgefüllt, bevor ein informelles Interview stattfand.

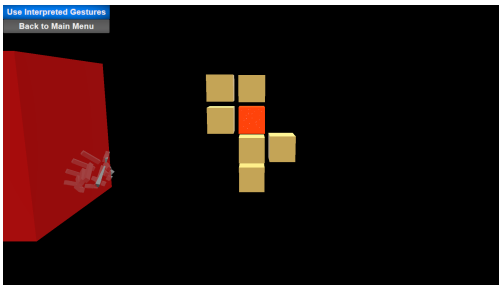
Der Aufbau des Fragebogens bestand aus einer kleinen Anzahl persönlicher Fragen und den zehn Fragen für die *System Usability Scale*, kurz SUS, einer Skala, die 1986 von Digital Equipment Corp. entwickelt wurde. SUS eignet sich sehr gut für die Auswertung von Umfragen mit einer kleinen Anzahl an Probanden (vgl. [TS04]). Die Fragen werden dabei mit Punkten auf einer Skala von eins (Starker Widerspruch) bis fünf (Starke Zustimmung) beantwortet. Die



(a) Aufbau von Szenario 3 mit dem Konstrukt, aus dem die beiden rot markierten Objekte entfernt werden sollen (Quelle: eigene Arbeit)



(b) Eines der markierten Elemente wird durch Greifen des Objektes aus dem Konstrukt gezogen (Quelle: eigene Arbeit)



(c) Das zu löschende Objekt wird in den roten Bereich gezogen, um es zu löschen (Quelle: eigene Arbeit)

Abbildung 4.4: Szenario 3 in unterschiedlichen Stadien

Punkte aus den Fragen eins, drei, fünf, sieben und neun werden um eins dekrementiert, sodass Werte zwischen null und vier entstehen. Die Punkte für die Fragen zwei, vier, sechs, acht und zehn werden von fünf abgezogen. Somit liegt auch hier das Ergebnis zwischen null und vier. Die Ergebnisse werden aufsummiert und mit 2,5 multipliziert, sodass ein Endergebnis entsteht, dass zwischen 0 und 100 liegt (vgl. [B+96]).

Nachfolgend sind die einzelnen Fragen aufgelistet, wobei der Begriff System nicht auf die benutzte Applikation bezogen ist, sondern das Konzept der Steuerung einer Applikation zur Betrachtung und Manipulation von dreidimensionalen, virtuellen Objekten mithilfe kontaktfreier Interaktion meint.

1. Eine regelmäßige Nutzung des Systems ist gewünscht.
2. Das System ist unnötig komplex.
3. Das System ist leicht zu nutzen.

4. Zur Nutzung des Systems wird die Unterstützung einer technisch versierten Person benötigt.
5. Die unterschiedlichen Funktionen sind gut in das System integriert.
6. Das System ist zu inkonsistent.
7. Die meisten Nutzer werden die Bedienung des Systems sehr schnell lernen.
8. Das System ist sehr mühsam zu nutzen.
9. Die Nutzung des Systems fühlt sich vertraut an.
10. Um mit dem System zu Arbeiten müssen zuvor viele Dinge gelernt werden.

Der Fragebogen wurden den Probanden als Onlineumfrage über Typeform[3] zur Verfügung gestellt. Die Ergebnisse von Testgruppe A und B wurden gesondert ausgewertet und der Durchschnittswert sowie die Standardabweichung wurden bestimmt.

4.2 Ergebnisse

Nachdem in dem vorherigen Abschnitt auf das gesamte Testsetting eingegangen wurde, werden die dadurch gesammtelten Daten in diesem Abschnitt ausgewertet und es wird abgeleitet, zu welcher Aussage sie in Bezug auf die eingangs gestellte These führen.

Dieser Abschnitt ist in drei Teile untergliedert. Zunächst werden die erhobenen Daten vorgestellt und grob ausgewertet (Abschnitt 4.2.1). Im Anschluss werden die, aus der Auswertung gewonnenen Informationen interpretiert (Abschnitt 4.2.2). Abschließend werden zusätzliche Beobachtungen und Feedback der Probanden aufgelistet (Abschnitt 4.2.3).

4.2.1 Auswertung

Die Ergebnisse aus den Messungen und den Befragungen werden an dieser Stelle kurz zusammengefasst. Tabelle 4.1 zeigt die Ergebnisse für die direktmanipulative Interaktion (Gruppe B). Die Angaben sind in Sekunden.

Tabelle 4.2 zeigt die Ergebnisse für die Probanden, die sowohl eine Kombination aus direktmanipulative Interaktion und interpretierten Gesten (Gruppe B) nutzen konnten. Die Angaben sind ebenfalls in Sekunden.

[3] https://www.typeform.com/

Tabelle 4.1: Ergebnisse der Messung für die Szenarien 1-3. Die Probanden konnten nur direktmanipulative Interaktion verwenden. Die Zeiten sind in Sekunden.

Szenario 1	**Szenario 2**	**Szenario 3**
25,7	58,9	81,0
27,7	26,6	12,9
70,0	141,4	160,3
40,7	168,6	102,4
25,8	98,3	51,0
31,3	174,9	51,5
61,0	127,5	36,3
48,7	34,4	21,6
13,9	81,3	23,8
10,3	75,1	36,1

Aus den vorhandenen Daten können Durchschnitt und Abweichung für Gruppe A und Gruppe B in den drei Szenarien berechnet werden, zu sehen in Tabelle 4.3. Die Ergebnisse der SUS betragen für Gruppe A $70,56$ mit einer Standardabweichung $\sigma_{SUS_A} = 12,86$ und für Gruppe B $61,5$ mit einer Standardabweichung von $\sigma_{SUS_B} = 16,04$.

4.2.2 Interpretation

Anhand der im vorherigen Abschnitt gezeigten Daten können einige Aussagen zu der aufgestellen These 1 getroffen werden. Die Auswertung der Messdaten aus Tabelle 4.3 zeigt, dass Gruppe B deutlich schneller Szenario 1 und Szenario 3 lösen konnten. In Szenario 2 brauchte Gruppe B im Vergleich zu Gruppe A im Schnitt länger. Es zeigt sich, dass vor allem die Translation und die Rotation durch eine zusätzliche Interpretation deutlich verbessert werden können. Gruppe B konnte Szenario 3 beispielsweise in $54,5\%$ der Zeit lösen, die Gruppe A für dieselbe Aufgabe benötigt hat.

Im Gegensatz zu den objektiven, zeitlichen Messungen stehen die Ergebnisse der Umfragen. Die Auswertung der SUS zeigt deutlich, dass Gruppe A $(70,56)$ mit dem System besser zurechtgekommen ist als Gruppe B $(61,5)$. Beide Werte sind gut und zeigen, dass die Bedienung einer Applikation zur Betrachtung und Manipulation von dreidimensionalen, virtuellen Objekten mit den Händen grundsätzlich ein gutes Konzept ist. Eine Differenz um 9 Skalenpunkte zugunsten von Gruppe A macht aber auch deutlich, dass die Probanden, die eine Kombination aus direktmanipulativer Interaktion und interpretierten Gesten genutzt haben, weniger von der Nutzung überzeugt waren. Der Grund dafür hängt höchstwahrscheinlich mit den schlechten Ergebnissen in der Gruppe B in Szenario 2 zusammen. Einige der Probanden aus dieser Gruppe

Tabelle 4.2: Ergebnisse der Messung für die Szenarien 1-3. Die Probanden konnten direktmanipulative Interaktion und interpretierte Gesten verwenden. Die Zeiten sind in Sekunden.

Szenario 1	**Szenario 2**	**Szenario 3**
18,6	21,0	13,2
26,3	70,4	56,3
11,7	241,0	41
31,1	205,4	21,5
26,2	253,2	77,4
18,5	105,7	12,7
16,9	37,7	9,1
9,4	65,7	43,6
60,7	100,9	17,9
5,3	159,2	21,7

Tabelle 4.3: Vergleich zwischen den Durchschnittswerten und Abweichungen für Gruppe A und B in den einzelnen Szenarien (Angaben in Sekunden).

	Szenario 1	**Szenario 2**	**Szenario 3**
Durchschnitt Gruppe A	35,51	98,70	57,69
Abweichung Gruppe A	19,47	52,83	45,38
Durchschnitt Gruppe B	22,47	126,02	31,44
Abweichung Gruppe B	15,66	83,93	22,43

hatten sehr große Schwierigkeiten damit die gestellte Aufgabe zu lösen, sodass zwei der zehn Probanden mehr als 240 Sekunden zur Lösung der Aufgabe benötigt haben.

Die Ergebnisse, sowohl die Fragebögen als auch die Zeitmessung, zeigen jedoch, dass die Kombination aus direktmanipulativer Interaktion und interpretierten Gesten grundsätzlich eine Verbesserung gegenüber einer rein direktmanipulativen Interaktion ist. Sowohl die Arbeitsgeschwindigkeit als auch der Umfang der möglichen Aktionen wird gesteigert. Daraus folgt, dass die Hauptthese (These 1) dieser Arbeit bestätigt werden konnte. Die Ergebnisse zeigen aber auch, dass der gewählte Ansatz möglicherweise nicht vollständig den gestellten Anforderungen und den Zielen genügt.

4.2.3 Spontanes Feedback und Beobachtungen

Zusätzlich zu den Messungen und Fragebögen wurden Gespräche geführt und die Probanden wurden während der Tests beobachtet. Ein wichtiger Punkt, der auch die Ergebnisse in Szenario 2 miterklärt, war die Problematik mehrdeutiger Aktionen, die das System falsch erkannt hat.

So wollten die Probanden der Gruppe B Objekte mit zwei Händen verschieben. Da sie aber eine zusätzliche Unterstützung durch die Interpretation von Gesten hatten, wurde in vielen Fällen die Aktion der Probanden mit zwei Händen als Skalierung oder Rotation identifiziert. Dadurch wurde es für die Probanden schwieriger, die Aufgabe in einer möglichst kurzen Zeit zu lösen.

Ein weiteres Problem bestand in fehlerhaften Daten der Sensoren. Es wurde eine Leap Motion zur Handerkennung genutzt. Als Treiber kam der Orion-Treiber[4] zum Einsatz, der aktuell in der Beta-Version vorliegt (Stand: 3. August 2016). Dieser Sensor identifiziert die linke Hand in einigen Fällen als die rechte Hand und umgekehrt. Daraus folgen inkonsistente Zustände in der Simulation, die dazu führten, dass die Nutzer teilweise keine gute Kontrolle über die Applikation hatten.

Das gewählte Handmodell besaß keinerlei Beschränkungen und konnte somit zerspringen, falls fehlerhafte Daten von dem Sensor geliefert wurden. Dieser Umstand führte zu einer Verunsicherung der Probanden und manipulierte in einigen Fällen zusätzlich den aktuell erreichten Stand in einer Aufgabe. Bei der Translation eines virtuellen Objektes ist es vorgekommen, dass die Hand teilweise in dem virtuellen Objekt gesteckt hatte. Ein komplexeres Handmodell könnte diese Probleme lösen.

Das Greifen eines Objektes funktionierte sowohl mit als auch ohne Unterstützung durch Gesteninterpretation. Bei einer zusätzlichen Unterstützung konnte das virtuelle Objekt deutlich besser transliert werden, jedoch hat sich das Freigeben eines Objektes als schwierig erwiesen. Die Probanden haben deshalb in Szenario 2, obwohl sie die Unterstützung durch Gesten hatten, lieber auf direktmanipulative Interaktion zurückgegriffen.

Das Löschen von Objekten mithilfe eines markierten Bereiches, in den man ein Objekt ziehen muss, wurde oft als umständlich bezeichnet. Bei einer kombinierten Verwendung von direktmanipulativer Interakion und interpretierten Gesten kann eine Geste oder ein Kontextmenü definiert werden, mit dem ein Objekt sofort gelöscht werden kann.

[4] https://developer.leapmotion.com/orion

5 Zusammenfassung

In den vorangegangenen Kapiteln wurde sich mit der Frage beschäftigt, ob die Unterstützung eines Nutzers durch die zusätzliche Interpretation als Erweiterung von direktmanipulativer Interaktion zu einem besseren Ergebnis bei der Bedienung in Augumented und Virtual Reality führt. Dabei wurde der Fokus auf Applikationen zur Betrachtung und Veränderung von virtuellen, dreidimensionalen Objekten gelegt.

In Kapitel 2 wurden vergleichbare Arbeiten auf ihre Eignung für die Verwendung in der vorliegenden Arbeit untersucht. Als Ergebnis konnte festgestellt werden, dass es sowohl im Bereich der direktmanipulativen Interaktion als auch im Bereich der Interpretation von Gesten mögliche Ansätze gibt, die aber jeweils für sich betrachtet, keine Kombination bieten. Aus diesem Grund wurde in Kapitel 3 ein Konzept erarbeitet, das die direktmanipulative Interaktion mithilfe eines modellbasierten Ansatzes umsetzt und mit einer Gesteninterpretation kombiniert. Die Gesteninterpretation nutzt Trajektorie zur Beschreibung von dreidimensionalen Gesten und vergleicht Gesten über eine Distanzfunktion. Eine Evaluierung dieses Verfahrens wurde in Kapitel 4 beschrieben. Dabei wurde auf das Testsetting mit den einzelnen Szenarien, den Probanden und dem Fragebogen eingegangen. Zusätzlich wurde in diesem Kapitel die Auswertung und Interpretation der, durch die Evaluierung gewonnen, Daten vorgestellt. Es wurde gezeigt, das die Hauptthese 1 dieser Arbeit korrekt ist.

Dieses Kapitel fasst die Ergebnisse und das Erlernte aus der Analyse, dem Konzeptentwurf und der Evaluierung in Abschnitt 5.1 zusammen. Es wird ein Ausblick auf mögliche Weiterentwicklungen in Abschnitt 5.2 gegeben.

5.1 Fazit

Die vorliegende Arbeit hatte verschiedene Ziele und Anforderungen definiert, auf die in diesem Abschnitt eingegangen werden. Zusätzlich wird gezeigt, welche neuen Erkenntnisse und Erfahrungen bei der Bearbeitung des Themas zustande gekommen sind.

Das wichtigste Ziel dieser Arbeit war die Bestätigung oder Widerlegung der These 1. In der Evaluierung der Arbeit (Kapitel 4) konnte die These mit einigen Einschränkungen bestätigt werden. Vor allem die subjektive Einschätzung der Probanden zeigt, dass die erarbeitete

Lösung Möglichkeiten zur Verbesserung hat. Die Anzahl der Probanden ist aber statistisch nicht repräsentativ und es muss ihn nachfolgenden Untersuchungen geprüft werde, ob die hier vorgestellten Ergebnisse und Aussagen auch bei einer größeren Sample Size haltbar sind. Durch die Kombination von direktmanipulativer Interaktion und interpretierten Gesten konnte eine alternative Eingabemethode entwickelt werden, die von den Probanden akzeptiert wird, wie die Auswertung des Fragebogens mithilfe von SUS zeigen. Es wurde ein verteilter und flexibler Systemaufbau angestrebt und umgesetzt. Ein Großteil der Nutzer aus Testgruppe A hat versucht, die virtuellen Objekte mit der flachen Hand zu schieben. Nur wenige haben versucht, das Objekt zu greifen. Testgruppe B hat durch die Unterstützung eher eine Form des Greifens genutzt. Daraus kann man Ableiten, dass Schieben, Heben und Rotieren von Objekten Aktionen sind, die Nutzer mit der direktmanipulativer Interaktion durchführen wollen. Komplexere Aktionen, wie etwa das Greifen von Objekten, benötigen jedoch eine Unterstützung durch eine Gesteninterpretation. Die zuvor erwähnten Einschränkungen bezüglich der Aussagekraft dieser Studie sind auch hier gültig.

Gesture Recognition kann sowohl als eigenständige Service oder als Teillösung genutzt werden und ist darauf ausgelegt, die berechneten Daten an einer definierten Schnittstelle auszugeben. Um die Funktionsweise zu sichern, wurden verschiedene, automatische Tests entwickelt, die auf realen Daten arbeiten. Die Lösung ist in die bestehende Infrastruktur der Projektgruppe I^2E integrierbar und kann dort in der Zunkunft für weitere Arbeiten eingesetzt werden. Mit der Entwicklung von Trame konnte eine Lösung zur Geräteabstraktion entwickelt werden, die zusätzlich die Möglichkeit bietet, Ergebnisse von Sensoren zu kombinieren. Durch die Unterstützung von bildbasierten Tiefensensoren, wie etwa der Microsoft Kinect oder der Leap Motion, ist eine mobile Nutzung möglich. Zusätzlich muss der Nutzer keine Sensoren am Körper tragen und kann kontaktfrei mit dem System interagieren. Die Gesture Recognition verarbeitet Handbewegungen direkt und ermöglicht somit eine direktmanipulative Interaktion. Zusätzlich werden Handbewegungen interpretiert und können genutzt werden, um die möglichen Aktionen eines Nutzers zu erweitern. Es wurde ein Gestenset mit verschiedenen Gesten für die Arbeit mit virtuellen, dreidimensionalen Objekten entwickelt. Die Evaluierung hat gezeigt, dass das Bedienkonzept schnell zu erlernen ist und nicht nur als Expertensystem eingesetzt werden kann. Eine Analyse der Verarbeitungszeit hat zudem gezeigt, dass die Antwortzeit unterhalb der kritischen Grenze von 100 ms liegt.

Lessons Learned

Durch die Arbeit an der Thesis wurden neue Konzepte erlernt. Ein Großteil der Programmierung ist in C# erfolgt. Aus diesem Grund wurde sich intensiv mit der Sprache und den verschiede-

nen Konzepten auseinandergesetzt. Dazu gehören LINQ, asychrone Eventprogrammierung, Multithreading, Taskqueues und Streamprocessing. Durch die Erstellung einer Abstraktionsschicht wurde sich mit verschiedenen Sensoren und Treibern auseinander gesetzt. Es wurden Verteilungsstrategien genutzt, um einen hohen Durchsatz zu erzielen. Es wurden mehrere Konzepte zur Gestenerkennung analysiert und zum Teil auch umgesetzt. Zur Vorbereitung auf die Evaluierung wurden verschiedene Quellen studiert, die sich mit der Erstellung von Fragebögen und der Befragung von Probanden befassen. Zusätzlich wurden vergleichbare Arbeiten herangezogen, um das Vorgehen in diesen Arbeiten zu analysieren. Ebenfalls wurde sich mit kontinuierlicher Integration im Zusammenhang mit Opensource-Software beschäftigt.

5.2 Ausblick

Die erfolgreichen Ergebnisse der Arbeit zeigen, dass bei der Entwicklung des Konzeptes bereits einige richtige Entscheidungen getroffen wurden. In diesem Abschnitt wird ein Ausblick auf mögliche Verbesserungen und Erweiterungen gegeben.

In weiteren Untersuchungen sollten sowohl mehr Probanden als auch mehrere Durchläufe genutzt werden, um ein besser verwertbares und statistisch relevantes Ergebnis zu erhalten. Die Testaufgaben sollten so designt sein, dass der Proband sie in kurzer Zeit durchlaufen kann. Alle Messdaten sollten automatisiert erhoben werden. Zusätzliche Beobachtungswerte, wie etwa die Genauigkeit und die Fehlertoleranz könnten ebenfalls ausgewertet werden (vgl. [KP15]).

Die Abstraktionsschicht Trame besitzt an mehreren Stellen ein Potential zur Weiterentwicklung. Das derzeitige Konzept zur gleichzeitigen Unterstützung von unterschiedlichen Sensorversionen erfordert die Nutzung von *extern-Alias*[1]. Durch eine Auskopplung der einzelnen Devices in eigene Packages könnte dieses Konzept vereinfacht werden. Wie bereits in dem Konzept zu Trame beschrieben, können weitere Devices und Serialisierungsformate entwickelt werden. Ebenso ist ein neues Konzept für die Fusion von Skelettdaten geplant. Dabei können Skelette nicht nur verschnitten werden, sondern sie können auch Messfehler, die durch andere Sensoren entstanden sind, ausgleichen.

Während der Evaluierung wurden alle Roheingaben der Probanden vor der Verarbeitung gesichert. Die gespeicherten Messdaten können für nachfolgende Untersuchungen genutzt werden. Sie müssen zunächst vorverarbeitet und sortiert werden. Anschließend sind sie ideal für das Training und die Testphase von Erkennungen auf Basis von Maschinenlernen, wie etwa SVM, RNN, HMM oder CNN, geeignet. Das Matching und die Beschreibung der Gesten,

[1] https://msdn.microsoft.com/de-de/library/ms173212.aspx

wie in dieser Arbeit verwendet, haben sich als gut geeignet für die gestellten Anforderungen gezeigt. Nicht in jedem Fall muss auf eine Lernphase verzichtet werden. Es ist zu testen, ob die Verwendung eines CNN bessere Ergebnisse erzielt.

In unterschiedlichen Konzepten zur Erkennung multimodalen Nutzereingaben werden mehrstufige Systeme zur Erkennung verwendet. In vereinfachter Form geschieht dieses in Gesture Recognition bereits mit der Entscheidungsphase. Zukünftige Arbeiten könnten diesen Ansatz weiterverfolgen und eine zusätzliche Erkennung innerhalb der Applikation bereitstellen, die den Kontext der Applikation besitzt. Mehrdeutige Eingaben können so besser aufgelöst werden. Zusätzlich besteht die Möglichkeit, ein Eventsystem zu entwickeln, in dem sich die Applikation auf ein Event für ein bestimmtes virtuelles Objekt einschreibt und informiert wird, sobald der Nutzer in der gefragten Weise mit dem Objekt interagiert. Ähnliche Konzepte sind bereits aus der Entwicklung für Applikationen auf mobilen Endgeräten bekannt und können zu einer deutlichen Vereinfachung bei der Entwicklung von neuen Mixed-Reality-Applikationen führen.

Literaturverzeichnis

[AB10] Caroline Appert and Olivier Bau. Scale detection for a priori gesture recognition. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 879–882. ACM, 2010.

[B+96] John Brooke et al. Sus-a quick and dirty usability scale. *Usability evaluation in industry*, 189(194):4–7, 1996.

[BPS+14] Lorenzo Baraldi, Francesco Paci, Giuseppe Serra, Luca Benini, and Rita Cucchiara. Gesture recognition in ego-centric videos using dense trajectories and hand segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 688–693, 2014.

[CNSD93] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the cave. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 135–142, New York, NY, USA, 1993. ACM.

[CRDR13] Ankit Chaudhary, Jagdish Lal Raheja, Karen Das, and Sonia Raheja. Intelligent approaches to interact with machines using hand gesture recognition in natural way: A survey. *CoRR*, abs/1303.2292, 2013.

[CXBT12] Jun Cheng, Can Xie, Wei Bian, and Dacheng Tao. Feature fusion for 3d hand gesture recognition by learning a shared hidden space. *Pattern Recognition Letters*, 33(4):476 – 484, 2012. Intelligent Multimedia Interactivity.

[EAHAM08] M. Elmezain, A. Al-Hamadi, J. Appenrodt, and B. Michaelis. A hidden markov model-based continuous gesture recognition system for hand motion trajectory. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, pages 1–4, Dec 2008.

[EAHK+07] M. Elmezain, A. Al-Hamadi, G. Krell, S. El-Etriby, and B. Michaelis. Gesture recognition for alphabets from hand motion trajectory using hidden markov mo-

dels. In *2007 IEEE International Symposium on Signal Processing and Information Technology*, pages 1192–1197, Dec 2007.

[Ebe15] Björn Eberhardt. Distributed streaming and compression architecture for point clouds from mobile devices. Master's thesis, Hochschule für Angewandte Wissenschaften Hamburg, 2015.

[EGG+03] Jacob Eisenstein, Shahram Ghandeharizadeh, Leana Golubchik, Cyrus Shahabi, Donghui Yan, and Roger Zimmermann. Device independence and extensibility in gesture recognition. In *Virtual Reality, 2003. Proceedings. IEEE*, pages 207–214. IEEE, 2003.

[FC15] Vittorio Fuccella and Gennaro Costagliola. Unistroke gesture recognition through polyline approximation and alignment. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 3351–3354, New York, NY, USA, 2015. ACM.

[HCLC16] M. Han, J. Chen, L. Li, and Y. Chang. Visual hand gesture recognition with convolution neural network. In *2016 17th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD)*, pages 287–291, May 2016.

[HKI+12] Otmar Hilliges, David Kim, Shahram Izadi, Malte Weiss, and Andrew Wilson. Holodesk: direct 3d interactions with a situated see-through display. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, pages 2421–2430. ACM, 2012.

[IKH+11] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 559–568, New York, NY, USA, 2011. ACM.

[JSM+14] Brett Jones, Rajinder Sodhi, Michael Murdock, Ravish Mehra, Hrvoje Benko, Andrew Wilson, Eyal Ofek, Blair MacIntyre, Nikunj Raghuvanshi, and Lior Shapira. Roomalive: Magical experiences enabled by scalable, adaptive projector-camera units. In *Proceedings of the 27th annual ACM symposium on User interface software and technology*, pages 637–644. ACM, 2014.

[Jür04] Hans W Jürgens. *Erhebung anthropometrischer Maße zur Aktualisierung der DIN 33 402-Teil 2*. Wirtschaftsverl. NW, Verlag für Neue Wiss., 2004.

[JXYY13] Shuiwang Ji, Wei Xu, Ming Yang, and Kai Yu. 3d convolutional neural networks for human action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 35(1):221–231, 2013.

[KD11] Per Ola Kristensson and Leif C Denby. Continuous recognition and visualization of pen strokes and touch-screen gestures. In *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, pages 95–102. ACM, 2011.

[KNQ12] Per Ola Kristensson, Thomas Nicholson, and Aaron Quigley. Continuous recognition of one-handed and two-handed gestures using 3d full-body motion tracking sensors. In *Proceedings of the 2012 ACM International Conference on Intelligent User Interfaces*, IUI '12, pages 89–92, New York, NY, USA, 2012. ACM.

[KP15] Jun-Sik Kim and Jung-Min Park. Physics-based hand interaction with virtual objects. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 3814–3819. IEEE, 2015.

[LD13] Jing Lin and Yingchun Ding. A temporal hand gesture recognition system based on hog and motion trajectory. *Optik-International Journal for Light and Electron Optics*, 124(24):6795–6798, 2013.

[LM90] Brenda Laurel and S. Joy Mountford. *The Art of Human-Computer Interface Design*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1990.

[LZT16] Zhi Liu, Chenyang Zhang, and Yingli Tian. 3d-based deep convolutional neural network for action recognition with depth sequences. *Image and Vision Computing*, 2016.

[McN92] David McNeill. *Hand and mind: What gestures reveal about thought*. University of Chicago press, 1992.

[MGKK15] Pavlo Molchanov, Shalini Gupta, Kihwan Kim, and Jan Kautz. Hand gesture recognition with 3d convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2015.

[MK94] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321–1329, 1994.

[MNKW07] M. Meyer, B. Nelson, R. Kirby, and R. Whitaker. Particle systems for efficient and accurate high-order finite element visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(5):1015–1026, Sept 2007.

[NDC+11] J. Nagi, F. Ducatelle, G. A. Di Caro, D. Cireşan, U. Meier, A. Giusti, F. Nagi, J. Schmidhuber, and L. M. Gambardella. Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347, Nov 2011.

[NFS15] Richard A Newcombe, Dieter Fox, and Steven M Seitz. Dynamicfusion: Reconstruction and tracking of non-rigid scenes in real-time. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 343–352, 2015.

[Nie15] Michael A Nielsen. Neural networks and deep learning. *URL: http://neuralnetworksanddeeplearning. com/.(visited: 01.08. 2016)*, 2015.

[NY95] Wayne Niblack and John Yin. A pseudo-distance measure for 2d shapes based on turning angle. In *Image Processing, 1995. Proceedings., International Conference on*, volume 3, pages 352–355. IEEE, 1995.

[OKA11] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Markerless and efficient 26-dof hand pose recovery. In *Computer Vision–ACCV 2010*, pages 744–757. Springer, 2011.

[OKI15] Peter Ondruska, Pushmeet Kohli, and Shahram Izadi. Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones. *Visualization and Computer Graphics, IEEE Transactions on*, 21(11):1251–1258, 2015.

[OWL15] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. *CoRR*, abs/1502.06807, 2015.

[PB11] Mores Prachyabrued and Christoph W Borst. Dropping the ball: Releasing a virtual grasp. In *3D User Interfaces (3DUI), 2011 IEEE Symposium on*, pages 59–66. IEEE, 2011.

[PB12a] Mores Prachyabrued and Christoph W Borst. Virtual grasp release method and evaluation. *International Journal of Human-Computer Studies*, 70(11):828–848, 2012.

[PB12b] Mores Prachyabrued and Christoph W Borst. Visual interpenetration tradeoffs in whole-hand virtual grasping. In *3D User Interfaces (3DUI), 2012 IEEE Symposium on*, pages 39–42. IEEE, 2012.

[Pot11] Olaf Potratz. Ein system zur physikbasierten interpretation von gesten im 3d-raum, 2011.

[Ree83] W. T. Reeves. Particle systems—a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, April 1983.

[Rhe91] Howard Rheingold. *Virtual Reality: Exploring the Brave New Technologies*. Simon & Schuster Adult Publishing Group, 1991.

[RKSI+14] Grégory Rogez, Maryam Khademi, JS Supančič III, Jose Maria Martinez Montiel, and Deva Ramanan. 3d hand pose detection in egocentric rgb-d images. In *Computer Vision-ECCV 2014 Workshops*, pages 356–371. Springer, 2014.

[SBS06] Jia Sheng, Ravin Balakrishnan, and Karan Singh. An interface for virtual 3d sculpting via physical proxy. In *GRAPHITE*, volume 6, pages 213–220, 2006.

[Shn82] Ben Shneiderman. Direct manipulation: A step beyond programming languages. *ACM SIGSOC Bulletin*, 13(2-3):143, 1982.

[SKR+15] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim Christoph Rhemann Ido Leichter, Alon Vinnikov Yichen Wei, Daniel Freedman Pushmeet Kohli Eyal Krupka, Andrew Fitzgibbon, and Shahram Izadi. Accurate, robust, and flexible real-time hand tracking. In *Proc. CHI*, volume 8, 2015.

[SSHP15] Thomas Schops, Torsten Sattler, Christian Hane, and Marc Pollefeys. 3d modeling on the go: Interactive 3d reconstruction of large-scale scenes on mobile devices. In *3D Vision (3DV), 2015 International Conference on*, pages 291–299. IEEE, 2015.

[SYW08] Peng Song, Hang Yu, and Stefan Winkler. Vision-based 3d finger interactions for mixed reality games with physics simulation. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '08, pages 7:1–7:6, New York, NY, USA, 2008. ACM.

[TS04] Thomas S Tullis and Jacqueline N Stetson. A comparison of questionnaires for assessing website usability. In *Usability Professional Association Conference*, pages 1–12. Citeseer, 2004.

[WH94] Andrew P Witkin and Paul S Heckbert. Using particles to sample and control implicit surfaces. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 269–277. ACM, 1994.

[WIH+08] Andrew D Wilson, Shahram Izadi, Otmar Hilliges, Armando Garcia-Mendoza, and David Kirk. Bringing physics to the surface. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 67–76. ACM, 2008.

[WLC+12] Jiang Wang, Zicheng Liu, Jan Chorowski, Zhuoyuan Chen, and Ying Wu. Robust 3d action recognition with random occupancy patterns. In *Computer vision–ECCV 2012*, pages 872–885. Springer, 2012.

[WLK+14] Christian Weichel, Manfred Lau, David Kim, Nicolas Villar, and Hans W. Gellersen. Mixfab: A mixed-reality environment for personal fabrication. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 3855–3864, New York, NY, USA, 2014. ACM.

[WPK+16] D. Wu, L. Pigou, P. J. Kindermans, N. D. H. Le, L. Shao, J. Dambre, and J. M. Odobez. Deep dynamic neural networks for multimodal gesture segmentation and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(8):1583–1597, Aug 2016.

[WXB+14] Qianqian Wang, Yuan-Rong Xu, Xiao Bai, Dan Xu, Yen-Lun Chen, and Xinyu Wu. Dynamic gesture recognition using 3d trajectory. In *2014 4th IEEE International Conference on Information Science and Technology*, pages 598–601. IEEE, 2014.

[XC16] R. Xie and J. Cao. Accelerometer-based hand gesture recognition by neural network and similarity matching. *IEEE Sensors Journal*, 16(11):4537–4545, June 2016.

Tabellenverzeichnis

Abbildungsverzeichnis

Listings

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

Hamburg, 3. August 2016 Christian Blank