Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Johann Heitsch

Ein Framework zur Erkennung von dreidimensionalen Gesten

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Johann Heitsch

Ein Framework zur Erkennung von dreidimensionalen Gesten

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Technische Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr. rer. nat. Gunter Klemke

Abgegeben am 13. August 2008

Johann Heitsch

Thema der Bachelorarbeit

Ein Framework zur Erkennung von dreidimensionalen Gesten

Stichworte

Mensch-Maschine-Interaktion, Motion Tracking, Motion Capturing, Gesten, Gestenerkennung, SVM

Kurzzusammenfassung

In dieser Arbeit wird ein Framework erstellt, auf dessen Basis Gestenerkennungsmaschinen realisiert werden können. Es wird ein Überblick über die verfügbaren Motion Tracking Technologien gegeben. Für die Erstellung des Frameworks werden die Anforderungen an dieses analysiert und spezifiziert. Eine Gestenerkennnungsmaschine, die Gesten mit einer Supported Vector Machine klassifiziert, wird erstellt und getestet. Die Ergebnisse der ersten Tests werden beschrieben und bewertet.

Johann Heitsch

Title of the paper

A Framework for the detection of three-dimensional gestures

Keywords

Human-Computer-Interface, Motion Capturing, Motion Tracking, Gestures, Gesture Recognition, SVM

Abstract

This paper deals with the implementation of a Framework for gesture recognition. This framework is the base for the implementation of gesture recognition machines. It provides an overview of the available motion tracking technologies. For the preparation of the framework, the specifications are analyzed and specified. A gesture recognition machine is developed that uses an Supported Vector Machine to classify the gestures. The results of this are described and evaluated.

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einführung

Gesten sind neben der Sprache und der Mimik ein wichtiger Bestandteil der menschlichen Kommunikation. Dabei können Gesten die Sprache unterstützen (z.B. die unterstützende Gestik mit Händen im Dialog) oder auch ersetzen, wie es bei der Gebärdensprache der Fall ist. Emotionen, aber auch komplexe Sachverhalte, können über Gesten einfach und intuitiv vermittelt werden. Zudem ist es auch möglich, Befehle und Signale über Gesten zu geben. Ein Beispiel hierfür sind Schiedsrichter im Sport, die über Gesten Entscheidungen bekannt geben.

Es ist zu beobachten, dass Gesten in der Kommunikation eine wichtige Rolle spielen. Ist die Möglichkeit über Gesten zu kommunizieren nicht gegeben, so können schwierige Sachverhalte nur schwer vermittelt werden und es kann schnell zu Missverständnissen kommen.

In der Mensch-Maschine-Kommunikation (HCI[1]) wird seit einiger Zeit der Nutzen von Gesten für die Interaktion erforscht. Ziel ist es, aus der Mensch-Maschine-Interaktion eine Kooperation zu erschaffen. Die Interaktion mit der Maschine soll dabei ähnliche Strukturen annehmen wie die Kommunikation zwischen Menschen.

Diesen Aspekt fasst das SmartKom Projekt (SmartKom (2007)) auf. In dem Projekt werden neue Formen der Mensch-Computer-Interaktion erforscht mit folgendem Ziel:

> Die bestehenden Hemmschwellen von Computerlaien bei der Nutzung der Informationstechnologie abzubauen und so einen Beitrag zur Nutzerfreundlichkeit und Nutzerzentrierung der Technik in der Wissensgesellschaft zu liefern (aus: SmartKom (2007))

Um das Ziel zu erreichen, werden multimodale Dialoge genutzt, bei denen neben Eingaben über die üblichen Eingabemedien wie z.B. die Tastatur auch die Sprache, Mimik und Gestik genutzt werden. Die Steuerung wird somit unter anderem robuster gegen möglicherweise ungenaue oder falsche Eingaben. Ein weiterer Aspekt des Ziels ist die intuitive und handbuchfreie Bedienbarkeit der Benutzerschnittstellen, welche sich hierdurch nahtlos (seamless) in das tägliche Leben integrieren.

[1] Human Computer Interaction

Die *seamless interaction* ist die Verwirklichung eines mentalen Modells (siehe: Craik (1943)). Die Interaktion findet so statt, wie sich Menschen diese vorstellen. Die Maschinen passen sich den Menschen an und nicht wie bisher die Menschen der Maschine.

Durch die *seamless interaction* (siehe: Ishii u. a. (1994)) verschmelzen die Grenzen zwischen der Informationstechnologie und dem täglichen Leben. Die Forschungen lassen sich somit auch dem Forschungsfeld des *ubiquitous computing*, das von Mark Weiser (Weiser (1991)) geprägt wurde, unterordnen.

1.1 Motivation

Ein Teilgebiet der Mensch-Maschine-Interaktion ist die Interaktion mit großen Displays. Diese Displays weisen Dimensionen auf, welche nicht mehr mit Tastatur und Maus oder Single-/Multitouch Verfahren bedienbar sind. Als Mausersatz kann das Zeigerät gesehen werden, welches in Fischer (2008) entwickelt worden ist. Jedoch verändert dieses nicht die Art der Interaktion, sondern nur das Eingabemedium.

Ein neuer Ansatz, welcher auch in dem Film *Minority Report* von Regisseur Steven Spielberg in einer Zukunftsvision dargestellt worden ist, ist der, die Interaktion mittels Gesten zu realisieren. Die Ideen, für die in dem Film gezeigten Interaktionsmodi, stammen vom *Massachusetts Institute of Technology* (MIT (2008)). Viele Entwickler nutzen die Zukunftsvisionen der Filme als Vorlagen, um diese in der Realität zu verwirklichen (vgl. Clarke (2002)).

Abbildung 1.1: Präsentationsraum bei Airbus Hamburg (Abb. von Christie (2008))

Ausgereifte Konzepte, um Standardsoftware mit Gesten zu steuern, sind derzeit noch nicht zu finden. Es gibt einige Ansätze in speziell auf den Arbeitsbereich zugeschnittene Software,

welche mit gestenbasierter Interaktion gesteuert werden kann. Als Beispiel ist der Präsentationsraum bei Airbus (Abb. 1.1), in dem potentiellen Kunden ein Flugzeug vor Baubeginn vorgeführt werden kann. Die Bewegungen in dem Flugzeug können über Gesten gesteuert werden. Jedoch sind die Gesten sehr einfach und sind nicht mit der Mensch-Mensch-Kommunikation vergleichbar. Ähnliche Szenarien bieten sich in diversen Präsentationsanwendungen. Große Datenmengen sollen interaktiv präsentiert werden.

1.2 Zielsetzung

Im Zusammenhang mit der *seamless interaction* stellt sich die Frage, inwieweit und in welcher Form sich Gesten für die Interaktion mit großen Displays nutzen lassen.

Im theoretischen Teil dieser Arbeit wird analysiert, welche Anforderungen an ein Softwaresystem gestellt werden, das benötigt wird, um Gesten aufzunehmen, zu speichern und wiederzuerkennen. Auch die grundliegenden Fragen, die aufkommen, wenn die genannte Frage beantwortet werden soll, werden dargestellt.

Das praktische Ziel dieser Arbeit ist die Enwicklung eines Softwaresystemes, welches den Anforderungen der Analyse entspricht. Es wird somit ein System für die Erkennung von Gesten erstellt, das die Möglichkeit bietet, die Erkennung mit diversen austauschbaren Algorithmen vorzunehmen.

1.3 Verwandte Arbeiten an der HAW- Hamburg

An der HAW Hamburg beschäftigen sich bereits einige Arbeiten mit der Gestenerkennung.

Lorenzen (2005) beschreibt eine konturenbasierte Gestenerkennung, bei der statische Gesten von einer Kamera aufgenommen und analysiert werden.

Boetzer (2008) beschreibt eine Anwendungssteuerung mit Bewegungen und Gesten. Die Gestenerkennung basiert hierbei auf der Erkennung von Geraden in den Bewegungen.

Rahimi und Vogt (2008) beschreibt, wie auf Basis von Multitouch-Technologie die Interakton mit Computern realisiert werden kann.

J. Boetzer (2008) beschreibt die aktuellen Forschungsergebnisse der Arbeiten an der HAW-Hamburg

1.4 Gliederung der Arbeit

Die Arbeit teilt sich in drei große Abschnitte. In dem Kapitel 2 werden die Grundlagen und die Umgebung beschrieben, die für diese Arbeit benötigt werden. Es werden sowohl grundliegende Begriffe erläutert (2.1), wie aber auch die diversen Technologien, welche auf dem Markt zu finden sind (2.4). Auch die während der Arbeit erstellten Applikationen werden vorgestellt (2.5). Das Kapitel 3 ist die Analyse, in der die Problemstellung identifiziert und erläutert wird. Es werden Anwendungsszenarien genannt (3.1) und es wird auf Gesten im allgemeinen eingegangen (3.3). Für die Interaktion werden die aktuellen Varianten vorgestellt. Zudem werden in der Analyse die Anforderungen an das Framework spezifiziert (3.7). In dem Kapitel 4 werden die Komponenten, aus denen das Framework besteht, beschrieben (4.2) und es wird erläutert, wie das Framework aufgebaut ist. Am Schluss werden die Ergebnisse der Arbeit zusammengefasst und ein Ausblick gegeben (5).

2 Grundlagen

2.1 Definitionen

Motion

Der Begiff “Motion” stammt ursprünglich von dem lateinischen Begriff *movere* (bewegen, erregen, antreiben) ab.
Im Englischen ist Motion als : “The act or process of changing position or place” definiert. Ins Deutsche übersetzt bedeutet es *Bewegung*.

Capture

Der Begriff “Capture” stammt urprünglich von dem lateinischen Begriff *capere* (auffassen, fassen, ergreifen) ab.
Im Englischen ist Capture als: “The act of catching, taking, or winning, as by force or skill” definiert. Ins Deutsche übersetzt bedeutet es *Erfassung*.

Motion Capture

Zusammengesetzt ergeben die beiden Begriffe einen neuen, den des *Motion Capture* . Dabei beschreibt dieser die Methode, Bewegungen von Lebewesen oder Objekten in Echtzeit aufzunehmen und sie auf den Computer zu übertragen. In der Literatur wird Motion Capture häufig mit MoCa oder MoCap abgekürzt.

Im Kontext dieser Arbeit wird der Begriff Motion Capture benutzt, um die Bewegung einer Person und deren Gliedmaßen zu erfassen.

Human Computer Interaction

In dieser Arbeit wird HCI analog zu SIGCHI (1996) definiert:

> Human-computer interaction is a discipline concerned with the design, evaluation and implementation of interactive computing systems for human use and with the study of major phenomena surrounding them (aus SIGCHI (1996)).

Das Forschungsgebiet der HCI befasst sich also mit der Schnittstelle zwischen dem Menschen und den Maschinen.

Seamless Interaktion

Seamless Interaktion bedeutet, wie auch Ubiquitous Computing, dass sich die Interaktion und die Computer nahtlos in das tägliche Leben integrieren. Es steht bei der Verwendung von Computern nicht der Akt der Interaktion im Vordergrund, sondern die Aufgaben, die mit Hilfe der Computer erledigt werden sollen. Die Interaktion ist so natürlich wie nur möglich zu halten, damit ohne aufwändige Einarbeitung mit den Computern gearbeitet werden kann. Das Vorbild für die Interaktion sind natürliche Gesten aus der Mensch zu Mensch Kommunikation.

Multimodale Interaktion

Bei der multimodalen Interaktion werden diverse Interaktionsmodi miteinander kombiniert. Somit kann die Interaktion noch intuitiver gestaltet werden. Eingaben von Texten oder Befehlen über die Sprache sind genauso möglich wie die Manipulation von Objekten mittels Gesten. Eyetracker, die die Blickrichtung von Personen feststellen können, werden genutzt, um Objekte zu selektieren. Mit diesen und weiteren Interaktionsmodi können Interaktionsschnittstellen zwischen Mensch und Computer erstellt werden, die sich ohne große Einarbeitung bedienen lassen.

Computer Supported Collaborative Worksplace

Die Vernetzung von Arbeitsplätzen wird nicht mehr nur für den reinen Datenaustausch genutzt, sondern unterstützt auch das gemeinschaftliche Arbeiten an Problemlösungen. Arbeitsplätze, welche die rechnergestützte gemeinsame Arbeit ermöglichen, werden Computer Supported Collaborative Workspaces (CSCW) genannt. Eine detaillierte Beschreibung über die Forschung und Entwicklungen in diesem Gebiet sind in Napitupulu (2008) zu finden.

Powerwall/CAVE

Diverse Anwendungsbereiche in der Wirtschaft und Industrie benötigen eine Darstellung von Informationen auf sehr großen Flächen. Hierzu werden Powerwalls eingesetzt. Dieses sind große skalierbare Displays mit einem hohen Detaillierungsgrad. Es ist auch möglich, komplette betretbare Räume aus solchen Displays zu erschaffen. Diese nennen sich CAVE[1] (siehe Abb. 2.1). Weitere Einzelheiten zum Thema Powerwall und CAVE sind in Napitupulu (2008) beschrieben.

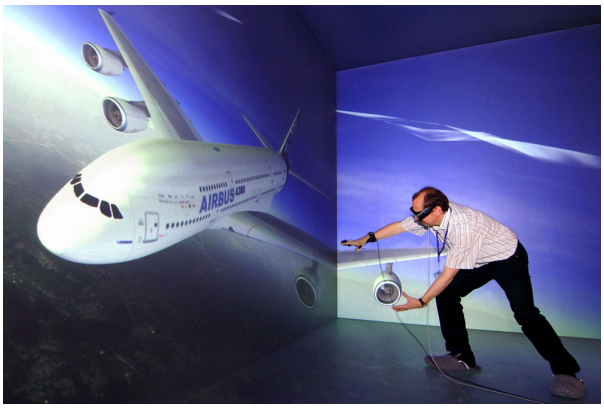

Abbildung 2.1: Cave bei Airbus Hamburg (Abb. von Christie (2008))

[1] Cave Automatic Virtual Environment

2.2 Anwendungen

Motion Tracking/Capturing ist bereits in vielen Bereichen in der Industie zu finden. Im Weiteren werden einige wichtige Bereiche beschrieben.

Animationsproduktion

Realistische Bewegungsabläufe und spektakuläre Stunts werden in Filmen immer beliebter. Die Erstellung der Szenen mittels Keyframing, bei dem der Animator die Position von Körperteilen in Zeitintervallen festlegt und die Bewegungen interpoliert werden, ist zu kostenintensiv. Die Szenen werden deshalb von speziell ausgestatteten Schauspielern nachgespielt und mittels Motion Capture aufgenommen. Das Material kann am Computer nachbearbeitet und in Filme oder Spiele integriert werden.

Abbildung 2.2: Aus der Realität in den Film (Abb. aus Motion Analysis Corporation (2008) und Wikipedia (2008))

Beispiele hierfür sind Filme wie:

- Matrix Reloaded (2003)
- I Robot (2004)
- Sky Captain (2004)
- The Chronicles of Riddick (2004)

Bei den Computerspielen wie z. B. FIFA 08 wurde Motion Capture verwendet, um die Spieler so realistisch wie nur möglich darzustellen.

Die selben Mechanismen werden auch in der Werbeindustrie verwendet.

Bewegungsanalayse

Eine weitere verbreitete Anwendung von Motion Capture ist die Bewegungsanalyse. Diese findet sich in den Bereichen Medizin, Sport (Abb. 2.3)und Forschung wieder.

Abbildung 2.3: Bewegungsanalyse im Sport (Abb. aus Motion Analysis Corporation (2008))

Gangartanalyse: Erkennung von Abnormalitäten und ihren Ursachen beim Gehen.

Rehabilitation: Protokollierung des Heilungsverlaufes nach einer Operation oder Erkennung von Bewegungseinschränkungen am Patienten.

Leistungssport: Bewegungsanalyse für die Optimierung von Bewegungsabläufen und Verminderung des Verletzungsrisikos.

Medizinische Roboter: Robotern können Bewegungen beigebracht werden.

Training: Virtuelles Training von medizinischen Eingriffen.

Biofeedback: Überwachung von Bewegungen, z.B. nach Operationen.

Abbildung 2.4: Virtual Reality: Fallschirmtraining und Virtuelles Design: CATIA V5 (Abb. aus Navy (2002) und Motion Analysis Corporation (2008))

Industrienutzung

Auch in der Industrie wird Motion Capture eingesetzt (Abb. 2.4).

Virtuelles Training (vgl. Napitupulu (2008)): Militäroperationen, Rollenspiele

VR: Virtuelle Realität , virtuelles Prototyping.

Roboter: Roboter können über Motion Capture gesteuert werden.

Ergonomie: Ergonomie und Usablitystudien können durch Motion Capture unterstützt werden.

Virtuelles Design: Interaktive Arbeitsplätze für Entwicklung und Tests.

2.3 Aufnahmetechniken

Auf dem Markt sind diverse Systeme für die Aufnahme von Bewegungen vorhanden. Diese können in folgende drei Typen eingeteilt werden:

Inside-in

Bei diesem System sind Sensoren und die Referenzquellen direkt am Körper angebracht. Es ist somit eine hohe Mobilität vorhanden, eine Positionsbestimmung im Raum ist aber nicht möglich.

Inside-out

Die Sensoren sind bei dieser Technik am Körper angebracht, jedoch gibt es externe Referenzquellen. Die Positionsbestimmung im Raum ist somit möglich.

Outside-in

Im Gegensatz zu den beiden anderen Systemen sind bei dieser Technik die Referenzquellen am Körper angebracht, diese können sowohl aktiv als auch passiv sein. Die Sensoren zur Signalaufnahme befinden sich im Raum verteilt. Es ist sowohl eine Positionsbestimmung möglich als auch das Erkennen von mehreren Objekten.

2.4 Aufnahmesysteme

Elektromechanisch

Bei elektromechanischen Systemen werden die Bewegungen und Positionen der Objekte mittels elektromechanischer Sensoren erfasst. Um dieses möglich zu machen, trägt der Performer eine Art Außenskelett.

Ein Nachteil dieser Systeme liegt in der Kalibrierung. Das Außenskelett muss an jeden Performer individuell angepasst und eingemessen werden. Das Außenskelett beschränkt den Performer in seiner Bewegungsfreiheit, so dass nicht alle Bewegungen möglich sind.

Die Vorteile sind die hohe Portabilität, die keine externen Komponenten benötigt, und hohe Reichweiten, die durch Funkübertragung der Daten erreicht werden. Ein weiterer Vorteil ist

Abbildung 2.5: Mechanisches Motion Capturing von *Meta Motion* (Motion (2004))

die Möglichkeit, mehrere interagierende Personen gleichzeitig aufzunehmen,jedoch mit der Einschränkung, dass absolute Positionsangaben nicht möglich sind.

Elektromagnetisch

Bei elektromagnetischen Systemen werden im Raum Transmitter angeordnet, die ein tieffrequentes magnetisches Feld aussenden. Dieses Feld wird von Sensoren, die am Performer angebracht sind, aufgenommen und die Daten werden an einen zentralen Computer gesendet.

An dieser Stelle verbirgt sich aber auch der Nachteil. Jeder Sensor muss mit einem eigenen abgeschirmten Kabel mit dem Computer verbunden sein. Ist der Computer extern, so wird der Performer durch die Kabel in der Bewegung eingeschränkt, ist jedoch der Computer in z.B. einem Rucksack untergebracht, so wird der Performer durch diesen eingeschränkt. Ein weiterer Nachteil sind Interferenzen und Störungen, die entstehen können, wenn sich zu viele Sensoren auf einem kleinen Raum befinden. Dieses ist der Fall, wenn mehrere Performer dicht zusammenstehen. Aber auch die Umgebung, wie Kabel oder metallische Gegenstände, verändern das Magnetfeld.

Die Vorteile liegen in den geringen Kosten, in der Echtzeit und dem nicht Vorhandensein eines Verdeckungsproblems, welches ensteht, wenn ein Performer durch einen anderen abgeschirmt wird. Diverse Störungen können mit Hilfe von Filtern herausgefiltert werden.

Inertial

Inertiale Systeme arbeiten mit Beschleunigungssensoren und Gyroskopen. Es handelt sich dabei um die Weiterentwicklung der mechanischen Systeme.

Ein Nachteil des Systems ist, dass sich Ungenauigkeiten addieren. Es können keine absoluten Positionsbestimmungen gemacht, sondern nur relative Bewegungen gemessen werden.

Die Vorteile des Systems liegen in der Portabilität und der geringen Einschränkung des Performers.

Akustisch

Bei akustischen Systemen senden Ulltraschallsender, die am Körper des Performers angebracht sind, Impulse aus, welche von Empfängern im Raum empfangen werden.

Dieses System benötigt externe Komponenten und ist somit nicht portabel, was als Nachteil aufzufassen ist. Zudem entstehen Probleme bei der Verdeckung von Sensoren, z.B. durch Reflexionen von Signalen.

Funk und Ultraschall

Mittels des Laufzeitunterschiedes eines Funk- und eines Ultraschallsignales lässt sich die relative Position von Objekten im Raum bestimmen. Auf Basis der Entwicklungen von Gregor (2006) sind an der Hochschule für Angewandte Wissenschaften Hamburg IMAPS[2] Module entworfen worden, welche auch von Hinck (2008) verwendet wurden, um die Position von Händen relativ zum Körper zu bestimmmen. Nach- und Vorteile sind in Hinck (2008) näher beschrieben.

Optisch mit Standardkameras

Bei Systemen mit Standardkameras werden, wie bei den Bilderfassungssystemen, die Bewegungsinformationen aus den Bildern extrahiert. Diese Systeme sind noch sehr selten und unzuverlässig, könnten aber die Motion Capture Systeme revolutionieren.

Es ist ein Nachteil, dass viel Rechenaufwand nötig ist, um Objekte in Bildern und Bildsequenzen zu erkennen und ihre Bewegungen und Positionen zu berechnen.

[2] Indoor Distance Measurement And Positioning System

Die Vorteile liegen auf der Hand. Es sind keine externen Komponenten, ausser den Kameras, nötig und der Performer wird durch keine Sensoren etc. behindert.

Optisch mit Infrarotkameras

Abbildung 2.6: Ein Tänzer, der einen Motion Tracking- Anzug trägt (aus Wikipedia (2008))

Diese Art der Motion Capture Technologie ist die am weitesten verbreiteste. Sie wird sowohl bei der Erstellung von Filmen für Kino und Werbung eingesetzt, wie auch bei der Erstellung von Videospielen. Bei dieser Technologie werden an dem Performer sogenannte Marker angebracht (Abb. 2.6). Hierbei handelt es sich um kleine Reflektoren, die die IR-Strahlen der im Raum angebrachten Kameras reflektieren. Die Kameras sind so im Raum verteilt, dass bei möglichst jeder Bewegung alle Marker sichtbar werden. Eine Software, an der die Kameras angeschlossen sind, berechnet aus den 2D-Positionsdaten der Kameras die Position der Marker im 3D-Raum.

Es kann sowohl die Position von einzelnen Markern bestimmt werden als auch die Position und Ausrichtung von sogenannten *Targets*. Dieses sind feste Anordnungen von Markern, welche aufgrund der Anordnung der Marker eindeutig zu identifizieren sind. Da ein Target immer mindestens vier Marker besitzt, ist es möglich, zu der Position auch seine Lage (Ausrichtung) im Raum zu bestimmen.

Ein Nachteil ist, dass ein Marker immer mindestens von zwei Kameras erkannt werden muss. Daher kann es bei mehreren Performern zu Verdeckungen kommen. Die Fehler, die durch diese Verdeckungen entstehen, können in einer Nachbearbeitung korrigiert werden. Werden die Daten hingegen gleichzeitig verarbeitet, müsssen hier andere Ansätze angewendet werden.

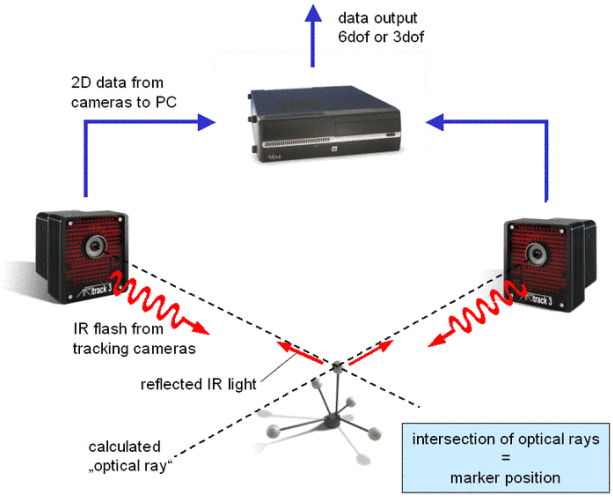

Abbildung 2.7: Systematischer Systemaufbau eines Motion Tracking Systems mit Infrarotkameras (Abb. von A.R.T-GmbH (2008))

Der Vorteil dieser Technologie besteht darin, dass der Performer keinen Einschränkungen unterliegt und somit auch komplexe Bewegungsabläufe erfassbar sind.

Der Aufbau des Systems ist in der Abbildung 2.7 dargestellt.

2.5 Ambient-Labor der HAW Hamburg

Aufbau

Powerwall

Die Powerwall an der Hochschule für Angewandte Wissenschaften Hamburg besteht aus einer Matrix von 3x3 30” Apple Cinema Displays. Der Aufbau hat eine Auflösung von maximal 7500*4800 Pixeln. Das Rendering wird von einem Master- und fünf Renderingrechnern übernommen. Als Software kann sowohl Xinerama als auch Chromium verwendet werden. Der Aufbau ist in Abbildung 2.8 abgebildet.

Motiontracker

Zum Tracken von Targets besteht an der HAW eine Installation von vier *ARTtrack 3* Kameras, die um die Powerwall angeordnet sind. Die Berechnung der Positionen der Targets übernimmt ein PC, auf dem die A.R.T. Software *DTrack* installiert worden ist. In Abbildung 2.8 ist die Installation zu sehen.

Abbildung 2.8: Die Powerwall der HAW Hamburg mit dem Tracking System

Applikationen

Während der Erstellung der Arbeit wurden Anwendungen geschrieben, die dazu dienen, die Möglichkeiten, welche die Powerwall und der Motion Tracker ermöglichen, auszutesten. Bei den Versuchen konnten die Vorteile, die der Aufbau mit sich bringt, genutzt werden.

Headtracking

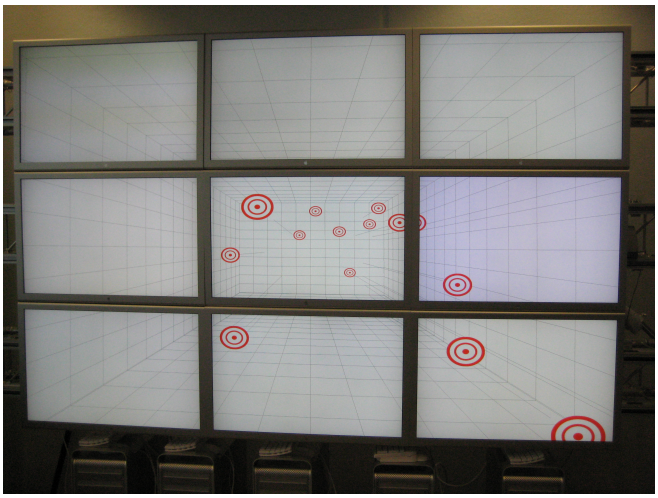

Abbildung 2.9: Headtracking Anwendung. Inspiriert durch Jonny Chung Lee (Lee (2008))

Angeregt durch die Versuche von Johnny Chung Lee (Lee (2008)), der am HCCI[3] an der Carnegie Mellon University tätig ist und dort mit der WiiMote experimentiert, wurde ein Head Tracking implementiert. Bei dieser Anwendung bewegt sich eine Person vor der Powerwall und bekommmt den Eindruck, als würde sie durch ein Fenster in einen Raum schauen.

Um dieses zu realisieren, wird ein Marker auf dem Kopf der Person befestigt, um seine Position im Raum zu bestimmen. Diese recht einfach gehaltene Demo zeigt, dass es mit einfachen Mitteln möglich ist, einen dreidimensionalen Eindruck zu erzeugen. Es ist jedoch noch zu erforschen, in welcher Hinsicht die Perspektive und Position einer Person vor einer Powerwall für die Interaktion genutzt werden können. Die Sicht der Person ist in Abbildung 2.9 abgebildet.

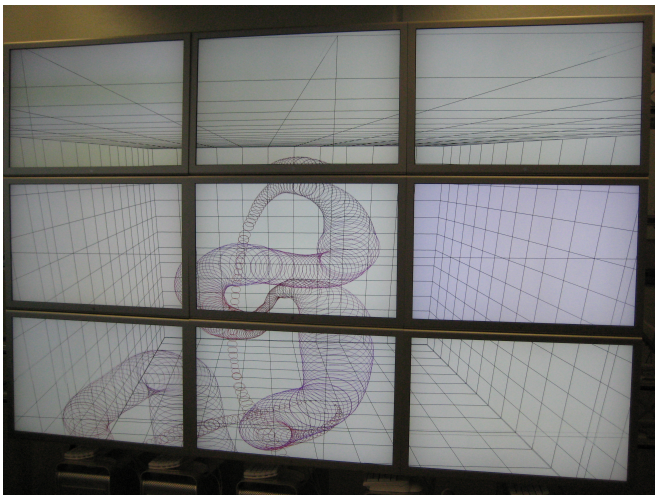

Abbildung 2.10: Virtual Graffity Anwendung. Inspiriert durch das Video “Tagged in Motion” (Jung und Matt/next GmbH (2007))

Virtual Graffity

Angeregt durch das Video “Tagged in Motion", das von der Werbeagentur *Jung von Matt/-next GmbH (Jung und Matt/next GmbH (2007))* erstellt wurde, wurde eine Anwendung (Abb. 2.10) geschrieben, mit der es möglich ist, im Raum virtuelle Graffitis zu zeichnen, die auf der Powerwall dargestellt werden können. Der Unterschied zu dem Original (dem Video) ist die Darstellung. Im Video wird eine Brille mit integrierten Bildschirmen genutzt, wohingegen in der Anwendung die Powerwall zur Anzeige genutzt wird. Beim Testen der Anwendung fiel auf, dass es an Interaktionsmöglichkeiten mit dem Raum fehlt. Derzeit zeichnet der Marker immer eine Linie, eine Unterbrechung der Linie ist nicht möglich. Auch ein Radieren, also ein Löschen von Linien ist nicht möglich, da das Aktivieren und Selektieren bzw. Wechseln von Werkzeugen nicht funktioniert.

[3] Human-Computer Interaction Institute

3 Analyse

3.1 Anwendungsszenarien

Um in den Kontext dieser Arbeit einzuführen und zu zeigen, wo die Probleme sind, die mit dieser Arbeit gelöst werden sollen, folgt nachstehende Beschreibung: Am Anfang wird mit den Szenarien begonnen, welche den Einsatz der Motion Capturing und Gestenerkennung erfordern.

3.1.1 Interaktion mit CAD Zeichnungen

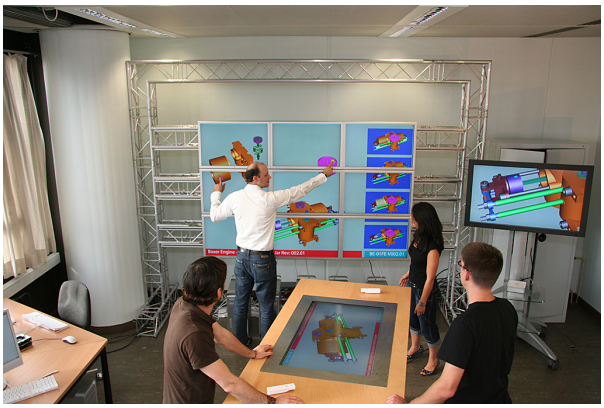

Abbildung 3.1: Interaktion mit einer Powerwall

Ein Ingenieur, der komplexe 3D CAD Zeichnungen erstellt, möchte diese auf einer Powerwall bearbeiten und präsentieren (Abb. 3.1). Ähnlich sind die Anforderungen eines Architekten oder eines Designers, der z.B. den Innenraum eines Flugzeuges gestaltet, nur dass hier noch eine Navigation im Raum hinzukommt. Hierzu ist es notwendig, die Zeichnungen/Entwürfe im Raum zu bewegen, rotieren zu lassen und zu skalieren. Zudem sollen Manipulationen an den dargestellten Objekten möglich sein.

3.1.2 Interaktion in Grafical Information Systems (GIFs)

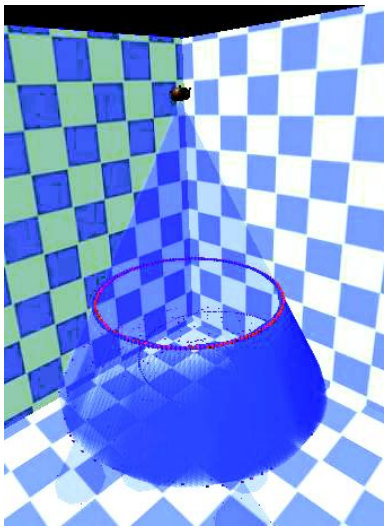

Abbildung 3.2: Conetree mit über 700 Einträgen (aus Norman I. Badler (1998))

Ein Stadtplaner möchte die Möglichkeit haben, in virtuellen Städten zu navigieren. Auch das Rescue Szenario aus Piening (2007a) und Piening (2007b) gehört mit in dieses Szenario. Hier ist eine Interaktion mit z.B. Google Maps (Google (2008)) nötig. Abbildung 3.1 zeigt eine Powerwall mit einem GIF.

3.1.3 Interaktion mit Desktops

Um sehr große hierarchisch geordnete Datenmengen zu visualisieren, wurden sogenannte Conetrees (Norman I. Badler (1998), George G. Robertson (1991)) entwickelt (siehe Robertson (1991)). Bei dieser Art der Visiualisierung werden die Daten in dreidimensionalen Darstellungen präsentiert, wodurch diese einfacher handelbar werden und es somit möglich ist, zu navigieren. Über die Interaktion und die Möglichkeiten, die sich durch eine solche 3D Darstellung ergeben, wird in Patterson (2007) berichtet. Abbildung 3.2 zeigt einen reinen Conetree. In Abbildung 3.3 ist eine Art Desktop abgebildet, der die Dateistruktur als Conetree zeigt.

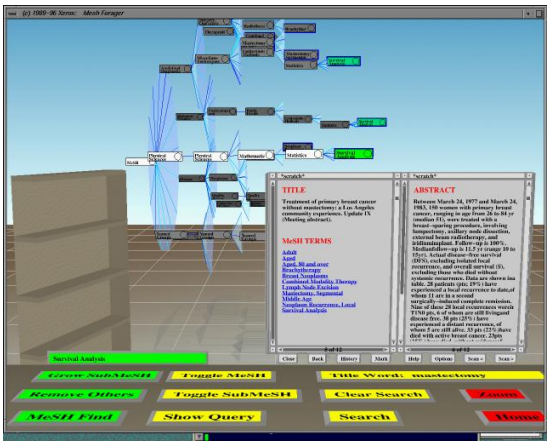

Abbildung 3.3: Desktop mit Conetree (aus Marti A. Hearst (1997))

3.2 Auswertung der Szenarien

In allen drei Szenarien besteht das Problem, dass mit den dargestellten Objekten interagiert werden soll. Sowohl eine Selektierung als auch eine Manipulation sollen möglich sein. Eine derzeitige Variante der Interaktion wird in 3.4.1 beschrieben. Jedoch ist diese Art der Interaktion nicht auf die reale Mensch-Mensch-Interaktion zurückzuführen. Die Interaktion soll daher mit Gesten ermöglicht werden.

Nicht nur Gesten zur Selektierung und Manipulation von Objekten sind notwendig, sondern auch, wie das Szenario des Architekten und Stadtplaners zeigt, eine Bewegung durch virtuelle Räume. Abbildung 3.4 zeigt weitere Einsatzszenarien von Gesten.

Abbildung 3.4: Einsatzszenarien von Gesten (aus: Rizzo (2007))

Es besteht noch kein fundiertes Wissen über das Aussehen der Gesten, die in solchen Szenarien benötigt werden, sondern nur über die Aktionen, die durch die Gesten ausgelöst werden sollen. Daher sind einige grundlegende Fragen zu beantworten:

Wie interagieren Personen mit Anwendungen / in virtuellen Räumen? Es ist notwendig herauszufinden, wie Personen die Manipulationen und Bewegungen vornehmen. Mit anderen Worten, wie sehen die Gesten aus?

Haben die Gesten verschiedener Personen Gemeinsamkeiten? Gibt es Gemeinsamkeiten zwischen den von den Personen gemachten Gesten oder sind die Gesten verschiedener Personen grundlegend verschieden?

Was sind die Gemeinsamkeiten? Sind Gemeinsamkeiten in den Gesten vorhanden, so ist zu analysieren, um welche es sich handelt und ob diese verallgemeinert werden können.

Wie können Gesten beschrieben werden? Sind Gemeinsamkeiten vorhanden, so ist eine Beschreibungsprache zu finden, welche die Gesten beschreibt.

3.3 Gesten

3.3.1 Taxonomie

Gesten sind Bewegungen von Körperteilen, denen eine Bedeutung zugeteilt worden ist. Gesten können in zwei Klassen eingeteilt werden. In der Literatur werden diverse Klassifizierungen vorgeschlagen, jedoch ist die zweckmäßigste, die auf der Taxonomie von Queck basierende Einteilung von Vladimir Pavlovic, Rajeev Sharma und Thomas Huang (Pavlovic u. a. (Jul 1997)).

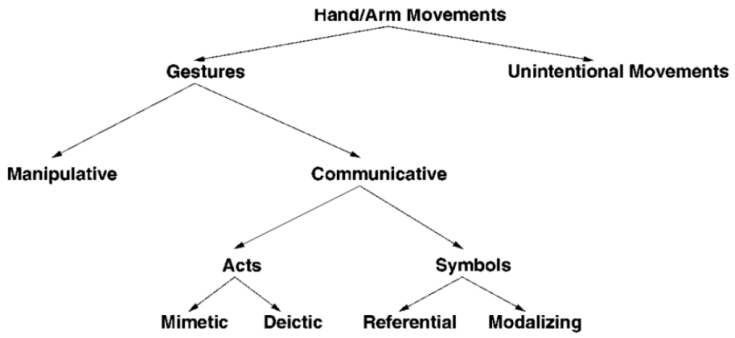

Abbildung 3.5: Taxonomie of Handgestures for HCI (aus Pavlovic u. a. (Jul 1997))

In Abbildung 3.5 ist die Taxonomie dargestellt. Die *unintentional Movements* sind nicht unterteilt und werden in dieser Arbeit nicht weiter untersucht, jedoch spielen sie eine große Rolle bei der Start-End-Erkennung, da die *unintentional Movements* genau die Bewegungen sind, welche aus einem Bewegungsfluss ausgefiltert werden sollen. Die Gesten werden in manipulative und kommunikative Gesten eingeteilt.

Der manipulative Zweig wird in der Abbildung 3.5 nicht weiter unterteilt, jedoch in der Ausarbeitung über die gestenbasierte Interaktion (J. Boetzer (2008)) . Die Unterteilung in der Ausarbeitung über die gestenbasierte Interaktion dient für diese Arbeit als Grundlage. Deshalb wird diese im Folgenden noch einmal erläutert und um einige Aspekte, die in der dreidimensionalen Interaktion hinzukommen, ergänzt.

Bewegungsverfolgung: Bei dieser Kategorie von Gesten handelt es sich um manipulative Gesten, bei welchen die Bewegung einen direkten Einfluss auf die Position und/oder Orientierung eines Objektes hat. Es können bei der Bewegungsverfolgung nur festgelegte Eigenschaften eines Objektes modifiziert werden (hier: Position, Orientierung). Sie gleichen Gesten aus der Realität, jedoch sind diese im dreidimensionalen Raum nicht ohne Einschränkungen verwirklichbar. Die Gründe hierfür werden in 3.3.2 weiter erläutert.

Kontinuierliche Gesten: Dieses sind Gesten, welche in der realen Welt nicht vorhanden sind und die vom Benutzer gelernt werden müssen. Es handelt sich bei kontinuierlichen Gesten um eine Mischform aus der Bewegungsverfolgung und den symbolisch-manipulativen Gesten.

Die Objekte werden kontinuierlich modifiziert, jedoch, im Unterschied zu der Bewegungsverfolgung, wird die Art der Modifikaton (die zu modifizierenden Eigenschaften) durch die Bewegung bestimmt.

Symbolisch-manipulative Gesten: Bei diesen Gesten handelt es sich um Gesten, welche vom Benutzer erlernt werden müssen. Es sind Gesten, die Objekte hinsichtlich diverser Eigenschaften verändern können. Für alle Eigenschaften sind eigene Gesten notwendig. Diese Gesten werden benötigt, um Steuerbefehle zu realisieren.

3.3.2 Start-End-Erkennung

Bei allen Gesten ist ein fest definierter Start- und Endpunkt notwendig. Die Bewegung zwischen diesen beschreibt hierbei die Geste.

Im Gegensatz zu zweidimensionalen Gesten z.B. auf Touchscreens, wo der Startpunkt durch das Aufsetzen und der Endpunkt durch das Heben des Fingers von dem Bildschirm erkannt werden können, gibt es das Problem im dreidimensionalen Raum, dass der Start- und Endpunkt einer Geste nicht einfach definiert werden kann.

Bei den symbolisch-manipulativen Gesten ist das Ergebnis der Erkennung erst nach dem Beenden der Gesten berechenbar. Dieses ist vergleichbar mit einem Mausklick auf eine Schaltfläche, bei der die Aktion auch erst nach dem Klick ausgeführt wird. Ein weiteres Beispiel für symbolisch-manipulative Gesten sind die Mausgesten in dem Browser Opera (Opera (2008)).

Bei der Bewegungsverfolgung und den kontinuierlichen Gesten muss die Art der Geste zu Beginn bekannt sein, da sich bei diesen Gesten zu modifizierende Objekte synchron zur Bewegung verändern sollen. Ein Beispiel hierfür ist das Verschieben eines Icons auf dem Desktop. Hierbei folgt das Icon der Maus synchron der Bewegung.

Ein Beispiel für die Start- und Endpunkterkennung bei der Bewegungsverfolgung und den kontinuierlichen Gesten ist in Boetzer (2008) beschrieben. Hier werden der Start- und Endpunkt über die Ausrichtung der Hände im Raum definiert.

Diese Arbeit hat den Fokus auf die Erkennung von symbolisch-manipulativen Gesten.

Die Bewegungsverfolgung, die auf den symbolisch-manipulativen Gesten aufbaut, wird nicht weiter bearbeitet. Für die Bewegungsverfolgung ist je eine Geste für das Selektieren (Mausklick im Beispiel) und das Verschieben (Mausbewegung im Beispiel) notwendig.

Die kontinuierlichen Gesten können mit der Einschränkung erkannt werden, dass das Ergebniss der Manipulation erst nach der Beendigung der Geste vorgenommen werden kann.

3.4 Derzeitige Methoden

In diesem Abschnitt werden zwei Techniken vorgestellt, welche derzeit im produktiven Einsatz verwendet werden.

3.4.1 Flystick

Abbildung 3.6: Der Flystick 1 von A.R.T-GmbH (2008)

Ein Flystick (Abb. 3.6) ist eine Art kabelloser Joystick, welcher vom Motiontracking System erkannt wird. Damit der Benutzer die Möglichkeit hat Eingaben im System zu machen, besitzt der Flystick diverse Knöpfe und z.T. zusätzlich auch noch einen kleinen zusätzlichen Joystick.

Bei Airbus wird der Flystick genutzt, um mit den vorhandenen Caves und Powerwalls zu interagieren. Dabei wird der Flystick durch ein Objekt in der virtuellen Welt repräsentiert. Es ist möglich, durch das Drücken einer Taste das Objekt in dem virtuellen Raum zu fixieren. Alle mit gedrückter Taste durchgeführten Bewegungen werden dann auf die Welt angewandt. Auch die Navigation durch Menues ist möglich.

Es handelt sich hierbei um eine einfache und schnell erlernbare Form der Interaktion.

Die Start- und End-Erkennung werden über den Knopf realisiert und die Bewegung wird direkt auf das Objekt übertragen. Es handelt sich hierbei also um eine reine Bewegungsverfolgung.

3.4.2 Laserpointer / WiiMote

Die Interaktion mit einem Laserpointer oder der WiiMote (Nintendo (2008)) wird in den Arbeiten König u. a. (2007) und Fischer (2008) beschrieben. An dieser Stelle soll nicht weiter auf die Technik eingegangen werden. Jedoch kann auch mit einem Laserpointer ein Gesteninterface realisiert werden, welches den Mausgesten aus den Browser Opera ähnelt.

Abbildung 3.7: Newsmap gesteuert durch Laserpointer (aus König u. a. (2007))

Die Start-End-Erkennung geschieht hier entweder durch zusätzliche Bedienelemente, wie Knöpfen oder einfachen Gesten, wie dem Verharren auf einem Objekt, um es zu selektieren (siehe: Boetzer (2008)).

3.5 Auswertung

Die Versuche und die Recherche haben gezeigt, dass es ein großer Aufwand ist, die Interaktion mittels Gesten zu ermöglichen. Es kristallisieren sich vier Hauptprobleme heraus:

1. Es sind nur wenige Gesten bekannt.
2. Die Gesten liegen nicht in einer maschinenlesbaren Form vor und können somit nicht verarbeitet werden.
3. Der Start- und Endpunkt einer Geste sind nicht bekannt.
4. Eine Beschreibungssprache für Gesten ist nicht bekannt.

Um die ersten zwei Probleme lösen zu können, ist es nötig, Gesten aufzunehmen und in einem geeignetem Format zu speichern. Somit kann eine Gestendatenbank aufgebaut werden. Hierzu können Probanden Aufgaben gestellt werden, welche diese mit Hilfe von Gesten lösen sollen. Beispiele für solche Aufgaben zeigt die Abbildung 3.4.

Der Start- und Endpunkt einer Geste wird manuell vorgegeben.

Um Gesten wiederzuerkennen, müssen diese beschrieben werden können. Aus der Bildverarbeitung sind diverse Verfahren bekannt, die es ermöglichen, komplexe Muster einfach zu beschreiben. Nachdem eine Musterbeschreibung vorhanden ist, muss für die Wiedererkennung eine *Distanz* zwischen den Mustern bestimmt werden. Anhand dieser Distanz ist dann eine Geste wiederzuerkennen.

3.6 Ziel dieser Arbeit

Ziel dieser Arbeit ist es, Anwendungsentwicklern eine Gestenerkennungsmaschine zur Verfügung zu stellen. Mit deren Hilfe ist es möglich, eine Gestendatenbank zu erstellen und Gesten wiederzuerkennen.

Um die Gestendatenbank zu erstellen, enthält die Gestenerkennungsmaschine einen Modus, in dem Gesten aufgenommen und gespeichert werden können. In diesem Schritt wird den Gesten ein symbolischer Name zugeordnet.

Für die Wiedererkennung der Gesten gibt es einen weiteren Modus. In diesem Modus können, anhand der Gesten in der Datenbank, Gesten einer Klasse von Gesten zugeordnet werden. Das Ergebnis der Klassifizierung kann in anderen Applikationen weiterverwendet werden.

Diese Gestenerkennungsmaschine soll konfigurierbar sein. Dieses bezieht sich auf die Algorithmen, die verwendet werden für die Vorverarbeitung, Feature Exraktion und Klassifizierung der Gestendaten. Die mitgelieferten Algorithmen sollen auf eine einfache Weise gegen neue Implementationen ausgetauscht werden können.

Damit die Gestenerkennungsmaschine in beliebige Applikationen integriert werden und auch die Austauschbarkeit der Komponenten realisiert werden kann, wird diese als ein Framework implementiert.

3.7 Anforderungen

3.7.1 Anforderungen an die Gestenerkennungsmaschine

Die Anforderungen an die Gestenerkennungsmaschine sind aus zwei Perspektiven zu betrachten. Es ist sowohl nötig, die Sicht eines Anwendungsentwicklers zu betrachten, der eine Anwendung entwickelt, die über Gesten gesteuert werden soll und für die Erkennung die Gestenerkennungsmaschine nutzt, wie auch die Sicht des Erstellers (Konfigurators) der Gestenerkennungsmaschine. Der Ersteller hat die Aufgabe, die richtigen Algorithmen für die Erkennung der Gesten zu wählen und ggf. zu implementieren.

Sicht des Anwendungsentwickler: Es wird eine Schnittstelle benötigt für den Infomationsaustausch über die erkannten Gesten. Eine Applikation soll benachrichtigt werden über Gesten, die erkannt wurden.

Sicht des Konfigurators: Der Konfigurator des Systems soll das System nach seinen Wünschen gestalten können. Ein wichtiger Aspekt hierbei ist die Austauschbarkeit der Komponenten. Zudem ist es notwendig, ein Mittel zur Verfügung zu stellen, mit denen eine Gestendatenbank erzeugt werden kann.

Funktionale Anforderungen

Nachdem im vorherigen Teil der Arbeit die groben Anforderungen an eine Gestenerkennungsmaschine festgelegt wurden, wird in diesem Teil spezifiziert, was das System für Möglichkeiten bietet. Diese werden im Folgenden mit Hilfe der Anwendungsfälle (Use-Cases) näher betrachtet.

Anwendungsfälle

Aufnahme: Die Rohdaten, die Gesten repräsentieren, sollen aufgenommen werden.

Wiedererkennung: Aufgenommene Gesten werden mithilfe eines trainierten Systems identifiziert.

3.7.2 Anforderungen an das Framework

Das Framework ist die Basis für die Gestenerkennungsmaschinen und muss somit den Anforderungen an eine Gestenerkennungsmaschine gerecht werden.

Das Framework soll einem Konfigurator die Möglichkeit geben, über eine Konfigurationsdatei das Framework und somit die Gestenerkennungsmaschine zu erstellen. Dabei sollen die Algorithmen für die Lösung der im Folgenden beschriebenden Problemstellungen, durch Algorithmen, welche im Framework vorhanden sind, wie auch durch eigene Implementationen realisiert werden könnnen.

Datenanbindung: Die Gestendaten können sowohl von dem Motiontracker kommen als auch aus Dateien gelesen werden. Diese beiden Anbindungen soll das Framework mindestens realisieren.

Targetauswahl: Es ist notwendig, eine Auswahl treffen zu können, welche Targets die Gesten darstellen und somit analysiert werden sollen. Auch die Berechnung von virtuellen Targets aus realen Targets soll möglich sein.

Gestensteuerung: Um die Zeitpunkte, welche den Start und das Ende einer Geste beschreiben, dem Framework bekannt zu machen, soll das Framework eine Möglichkeit bieten, diverse Eingabegeräte zu unterstützen.

Beschreibung von Gesten: Um eine Geste beschreiben zu können, ist es notwendig, aus den Gestendaten mit einem Algorithmus, Featuresets zu extrahieren. Das Featureset beschreibt eine Geste eindeutig. Jedes einzelne Feature eines Featuresets ist aussagekräftiger und besser für die Weiterverarbeitung geeignet als die Menge einzelner Gestendaten. Um diverse *Features Extraktoren* testen zu können, soll das Framework vom Benutzer implementierte Extraktoren aufnehmen können.

Unterscheidung von Gesten: Um zwischen den Featuresets, die aus den Gesten extrahiert wurden, eine Distanz zu bestimmen, werden, wie auch bei der Extraktion, Algorithmen benötigt. Die Distanz segmentiert die Gesten in Klassen, daher werden die Algorithmen zur Berechnung einer Distanz im weiteren als Klassifikatoren bezeichnet. Um diverse Klassifikatoren testen zu können, soll das Framework eine Schnittstelle zur Verfügung stellen.

Nicht funktionale Anforderungen

An das Framework direkt werden keine funktionalen Anforderungen beschrieben, da dieses die Basis darstellt, funktionale Anforderungen zu realsieren.

Austauschbarkeit und Konfigurierbarkeit

Die Komponenten für die Erkennung der Gesten sollen Austauschbar sein. Hierzu soll das Framework über eine Konfigurationsdatei konfigurierbar sein. In der Konfiguration sollen die realisierten Komponenten angegeben werden, welche für die Anwendung genutzt werden sollen.

Über die Konfigurierbarkeit und Austauschbarkeit sollen auch neue funktionale Anforderungen und Anwendungsfälle integriert werden können.

Portierbarkeit

Die Software wird in Kontext des UbiComp- und Ambiend Living Labores eingesetzt werden. Da in diesen Laboren diverse Betriebssysteme verwendet werden, soll die Software in einer plattformunabhängigen Sprache geschrieben sein. Als Betriebssysteme werden sowohl Windows und Linux als auch Mac OS X verwendet.

Umgebung

Derzeit befindet sich an der Hochschule für Angewandte Wissenschaften Hamburg die in 2.5 beschriebene Installation. Das Framework soll hauptsächlich für eine solche Installation verwendet werden. Aber auch abweichende Installationen, wie z.B. ein anderes Trackingsystem sollen unterstützt werden können. Es soll auch die Möglichkeit vorhanden sein, andere Systeme, die durch Gesten gesteuert werden, wie z.B. Touchscreens, an das System anzubinden.

4 Design und Realisierung

In diesem Kapitel wird das Design und die Realisierung des Frameworks beschrieben. Zu Beginn werden die Komponenten identifiziert und näher beschrieben. Bei der Beschreibung wird auf die Aufgaben der einzelnen Komponenten eingegangen. Zusätzlich wird jeweils beschrieben, wie die Komponenten implementiert wurden, bzw. was nötig ist, um eigene Komponenten zu entwickeln.

4.1 Allgemein

Das Framework ist in der Programmiersprache Java geschrieben. Durch Verwendung des JNI[1] ist es möglich, Algorithmen etc., welche in C/C++ geschrieben sind, in das Framework zu integrieren.

4.2 Komponenten

Aus den in der Analyse identifizierten Anforderungen lassen sich folgende Kernkomponenten ableiten:

Datenquellen sind Komponenten, die Gestendaten liefern.

Vorverarbeitung sind Komponenten, die die Gestendaten vorverarbeiten können.

Feature Extraktoren sind Komponenten, die benötigt werden, um aus Gestendaten Features zu extrahieren.

Klassifikatoren sind Komponenten, die Distanzen zwischen Featurevektoren ermitteln und diese somit in Klassen einteilen können.

Start-End-Erkennung sind Komponenten, in welchen die Start- End- Erkennung von Gesten stattfindet.

Kernel ist die Hauptkomponente, in der die Arbeitsabläufe gesteuert werden.

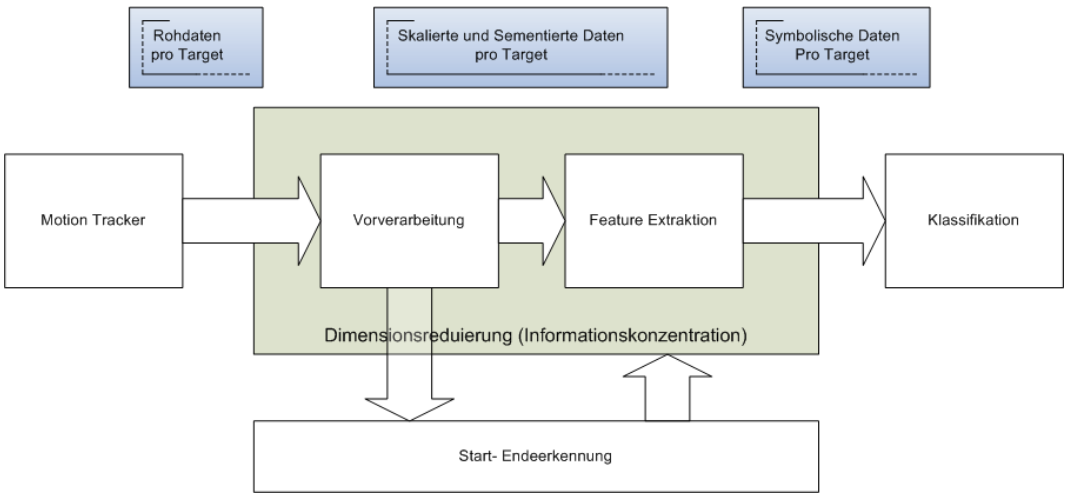

Abbildung 4.1: Übersicht über den Ablauf der Klassifizierung

Aus der Bildverarbeitung ist ein genereller Ablauf bekannt, der in der Abbildung 4.1, an die Anforderungen des Frameworks angepasst, dargestellt ist. Aus diesem Ablauf kann entnommmen werden, welche Komponenten miteinander in Verbindung stehen. Eine grobe Übersicht zeigt die Abbildung 4.2.

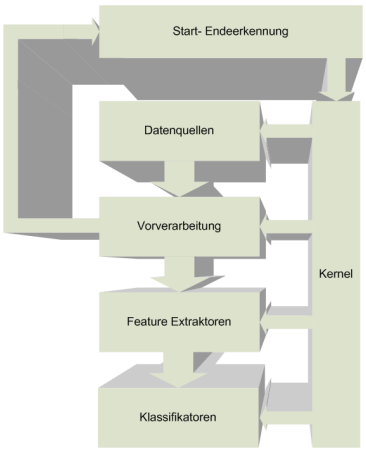

Abbildung 4.2: Übersicht über die Komponenten

Im Folgenden werden die Komponenten beschrieben. Weitere, als die hier aufgeführten, Informationen über die Klassen sind der jeweiligen JavaDoc zu entnehmen.

[1] Java Native Interface

4.2.1 Features und Featurevektoren

Features in der Mustererkennung sind markante Merkmale. Mit deren Hilfe ist es möglich, Muster, in dem Kontext dieser Arbeit Gesten, zu beschreiben.

Formale Beschreibung

Ein Muster kann als Vektor von Messgrößen aufgefasst werden.

$$\vec{f_i} = (f_{i1}, f_{i2}, f_{i3}, \ldots, f_{in}) \tag{4.1}$$

Dabei ist

$\vec{f_i}$ der Merkmalsvektor (Featurevektor) und
$f_{i1}, f_{i2}, f_{i3}, \ldots, f_{in}$ sind die n Merkmale (Features) der Geste.

Realisierung

Für die Abbildung von Featurevektoren in der Realisierung wurde ein Array von *double* Werten verwendet. Jeder Eintrag in dem Array entspricht dabei einem Feature. In der Implementierung steht für einen Featurevektor die Klasse *Featurevektor* zur Verfügung.

4.2.2 Datenquellen

Aus der Analye ist bekannt, dass Gestendaten sowohl direkt vom Tracker als auch aus im System gepeicherten Daten verarbeitet werden sollen.

Für die persistenten Daten ist in der Architektur eine Persistensschicht vorgesehen, die das Speichern, Laden und Verwalten der Gestendaten übernimmt.

Die Daten, welche vorab aufgenommen wurden, werden mit Metadaten versehen, um eine Sortierung und Gruppierung vornehmen zu können. In den Metadaten muss erkennbar sein, aus welcher Klasse die Geste ist. Damit ein Klassifikator diese über die Metadaten einordnen kann, werden die Metadaten für die Klassifizierung benötigt.

Formale Beschreibung

Die Datenquellen liefern Matrizen M_i. Der Index i gibt die Geste an. Die Position des Targest s zum Zeitpunkt t der Geste g_i wird durch das Tupel

$$v_{ist} = (x, y, z) | x, y, z \in [-1, 1] \tag{4.2}$$

beschrieben. Durch die Start-End-Erkennung werden der Startzeitpunkt $t = 1$ und der Endzeitpunkt d der Geste festgelegt. Die Anzahl der Targest s ist m. Eine Geste $g_i \in G$ ist durch die Matrix

$$M_i = \begin{pmatrix} v_{i11} & \dots & v_{im1} \\ \vdots & \dots & \vdots \\ v_{i1d} & \dots & v_{imd} \end{pmatrix} \tag{4.3}$$

komplett beschrieben.

$$g_i = (M_i, \lambda) \tag{4.4}$$

wobei das λ die Art (Klasse) der Geste beschreibt.

Realisierung

Für die Realisierung einer Datenquelle steht das Interface *IDataSource* zur Verfügung.

Es gibt einen Unterschied zwischen dem Laden der Daten aus einer Datei und dem Laden der Daten von einem Tracker. Die Gestendaten aus einer Datei können Metainformationen enthalten, die zusätzlich ausgelesen werden müssen. Zudem kann eine Datei mehere Gesten enthalten. Daher werden zwei weitere Interfaces zur Verfügung gestellt, die beide das *IDataSource* Interface erweitern und im Folgenden beschrieben werden.

IFileDataSource Das Interface ist für die Anbindung an Datenquellen gedacht, die auf persistente Daten zurückgreifen und somit mehrere Gestendatensätze auf einmal händeln können. Dazu sind zusätzliche Methoden vorhanden, um die Gestenmetadaten auszulesen und bestimmte Gesten zu selektieren. Es werden alle Gestenrohdaten auf einmal und nicht sequentiell geladen.

ITrackerDataSource Das Interface ist für die Anbindung an *live*-Quellen gedacht. Dieses sind z.B. Trackingsysteme. Damit das verbundene System gestartet und gestoppt werden kann, stellt das Interface entsprechende Methoden zur Verfügung. Da Klassen, die dieses Interface implementieren, aktive Klassen sind, die Events verschicken, werden Methoden für die Verwaltung von *IRawDataListener* bereitgestellt.

In dem Framework sind zwei Implementationen zu finden.

DTrackDataSource implementiert das *ITrackerDataSource* Interface und stellt die Verbindung zu dem Motion Tracking System her.

SerializeDataSource implementiert das *IFileDataSource* Interface und kann die im Filesystem abgelegten Gestendaten lesen.

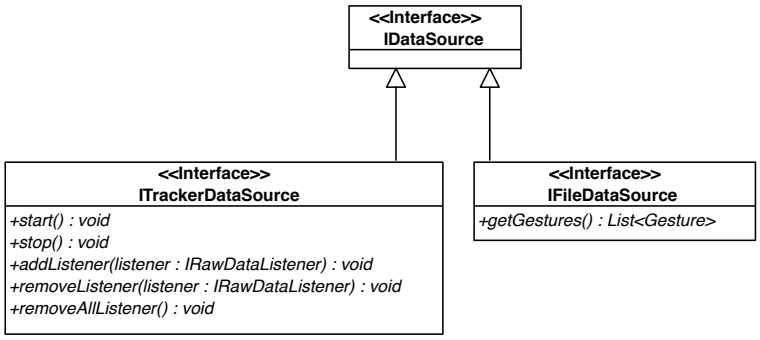

Abbildung 4.3: Übersicht über die Interfaces für die Datenquellen

4.2.3 Vorverarbeitung

Die Vorverarbeitungskomponente wird benötigt, um virtuelle Targets zu erstellen. Dieses sind Targets, die nicht real vorhanden sind, sondern aus den Positionen diverser anderer Targets berechnet werden. Sie werden benötigt, um Positionen der Targets relativ zu z.B. dem Performer zu generieren. Dieses ist notwendig, um von der absoluten Position im Raum zu abstrahieren. Weierhin können hier die Rohdaten skaliert und transformiert werden, wenn dieses erforderlich ist.

Realisierung

Für die Realisierung einer Vorverarbeitungskomponente steht das Interface *IPreProcessor* zur Verfügung.

Es wird zwischen zwei Arten von Vorverarbeitungskomponenten unterschieden. Der Grund dafür ist, dass zum einen die einzelnen Gestenrohdaten von einer *ITrackingDataSource* verarbeitet werden müssen, wie auch die gesammten Rohdaten einer kompletten Geste, welche von einer *IFileDataSource* stammen. Die beiden hierfür zur Verfügung stehenden Interfaces werden im Folgenden beschrieben.

ISequentialPreProcessor Dieses Interface erweitert das *IRawDataListener* Interface. Es wird als Listener bei der Datenquelle angemeldet und erhält somit die Gestenrohdaten. Die Gestenrohdaten sind einzele Targetpositionen ohne Bezug zur kompletten Geste. Nach der Verarbeitung benachrichtigt die Vorverarbeitungskomponente seine Listener über neue vorzuverarbeitene Rohdaten. Für die Verwaltung der Listener stellt das Interface passende Methoden zur Verfügung.

IParallelPreProcessor Dieses Interface ist für die Vorverarbeitung von kompletten Gestenrohdatensätzen zuständig. Es enthält eine Methode, in welcher aus Gestenrohdaten neue veränderte Gestenrohdaten erstellt werden können.

Im Framework sind drei Vorverarbeitungskomponenten vorhanden:

CalculateNewTargetPreProcessor berechnet aus der Position von zwei Targets die Position eines neuen virtuellen Targets.

ScalePreProcessor skaliert die Targetpositionen auf ein gegebenes Intervall. Wird die Klasse als *ISequentialPreProcessor* implementiert, ist dieses jedoch nicht immer möglich, da von einer Targetposition aus keine Aussage über die Gesamtgröße einer Geste gemacht werden kann.

ReduceDimensionPreProcessor reduziert die Anzahl der Dimensionen des Positionsvektors der einzelnen Targets.

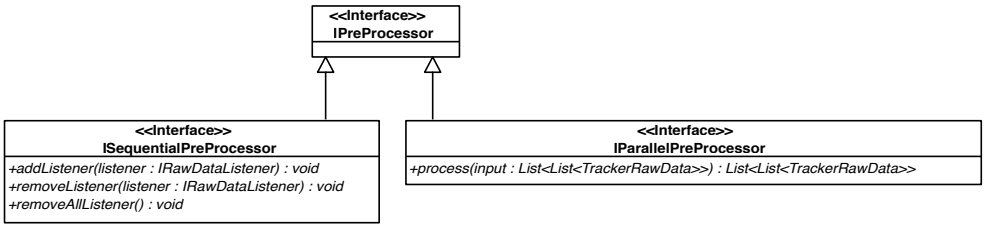

Abbildung 4.4: Übersicht über die Interfaces für die Vorverarbeitung

4.2.4 Feature Extraktoren

Feature Extraktoren sind Komponenten, die es ermöglichen, aus der Menge von Daten, die eine Geste darstellen, relevante Features zu extrahieren.

Durch die Extraktion von Features soll die Anzahl der Daten maximal reduziert werden. Wenige sehr markante Features sagen mehr über eine Geste aus als die einzelnen Target Positionen, die vom Tracker geliefert werden.

Formale Beschreibung

Der Feature Extraktor f bekommt als Eingangsvariable die Matrix M_i einer Geste g_i. Der Extraktor extrahiert die Features $f_i s$ der Geste g_i. Diese werden in dem Featurevektor $\vec{f_i} \in F$ zusammengefasst. Der Index s ist der Index des Features im Featurevektor und n die Anzahl der Features ($dim(\vec{f_i}) = n$).

$$\vec{f_i} = \left(f_{i1}, \ldots, f_{in}\right) = f(M_i) \tag{4.5}$$

Realisierung

Für die Realisierung einer Feature Extraktor- Komponente steht das Interface *IFeatureExtractor* zur Verfügung. Es definiert eine Methode, in welcher die Extraktion der Features vorgenommen wird.

Im Framework ist der Feature Extraktor, welcher unter 4.4.2 beschrieben wurde, in die Klasse *ApproximationFeatureExtractor* implementiert.

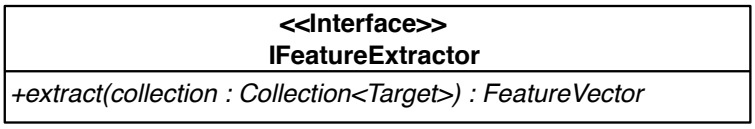

Abbildung 4.5: Übersicht über die Interfaces für die Feature Extraktoren

4.2.5 Klassifikatoren

Klassifikatoren sind Komponenten, die es ermöglichen, die Distanz zwischen den Featurevektoren zu berechnen. Über diese Distanzbeziehung können die Featurevektoren in Klassen eingeteilt werden.

Durch diese Klassifizierung ist es nicht nur möglich, unbekannte Gesten einer Klasse zuzuordnen, sondern es können auch Gemeinsamkeiten von unterschiedlichen Gesten analysiert werden.

Der Klassifikator kann eine Einordnung der Gesten anhand der Featurevektoren vornehmen als auch anhand der Gestenmetadaten.

Formale Beschreibung

Ein Klassifizierer definiert aus der Menge der Gestenvektoren F eine Relation

$$R : R \subseteq FxF$$

Diese Relation R ist eine Äquivalenzrelation und damit transitiv, reflexiv und symmetrisch. Die Äquivalenzrelation teilt die Menge der Featurevektoren F in Äquivalenzklassen. Zwei Äquivalenzklassen der Relation sind entweder identisch oder disjunkt. Dieses führt dazu, dass jeder Geste eine Klasse eindeutig zuzuordnen ist und somit jede Geste wiedererkannt werden kann.

Die Berechnung der Relation R geschieht anhand der Repräsentanten in der Gestendatenbank.

Realisierung

Für die Realisierung von Klassifizierern gibt es in dem Framework das *IClassifier* Interface. Das Interface enthält eine Methode, in der die Klassifizierung durchgeführt wird.

Im Framework ist der in 4.4.3 beschriebene Klassifikator implementiert. Zu finden ist dieser in der Klasse *SVMClassifier*

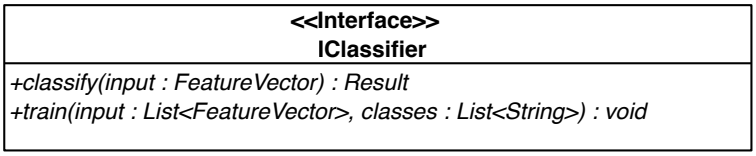

Abbildung 4.6: Übersicht über die Interfaces für Klassifikatoren

4.2.6 Start-End-Erkennung

Die Komponenten der Start-End-Erkennung werden benötigt, um aus dem Gestendatenstrom, der direkt vom Tracker oder aus der Vorverarbeitung kommt, den Start und das Ende einer Geste herauszufiltern.

In der Analyse wird eine Fernbedienung gefordert, um den Start und das Ende einer Geste manuell dem Framework bekannt zu machen. Dieses kann mit der Start-End- Komponente realisiert werden.

Realisierung

Für die Komponente, welche den Start und das Ende einer Geste bestimmt, steht das Interface *IStartEndDetection* zur Verfügung. Das Interface stellt Methoden für die Verwaltung von *IStartEndListener* Listenern bereit. Es steht dem Entwickler frei, ob eine Start-End-Erkennungskomponente zusätzlich das *IRawDataListener* implementiert. Wenn das Interface erweitert wird, so bekommt die Komponente die Rohdaten der Datenquelle oder Vorverarbeitung mitgeteilt. Wird das Interface nicht implementiert, so hat die Komponente keinen Zugriff auf die Gestendaten.

Im Framework ist in der Klasse *WiiMoteStartEndDetection* eine Start- End-Erkennungskomponente implementiert. Mit dieser ist es möglich, über eine WiiMote den Start und den Endpunkt einer Geste zu steuern.

4.2.7 Kernel

Der Kernel ist ein fester Bestandteil des Frameworks und ist nicht austauschbar. Im Kernel werden die Arbeitsabläufe realisiert.

Alle genannten Komponenten werden im Kernel vereinigt. Auf welche Art die Komponenten miteinander verknüpft und welche Komponenten von den Gestendaten durchlaufen werden, wird in einer Konfigurationsdatei festgelegt.

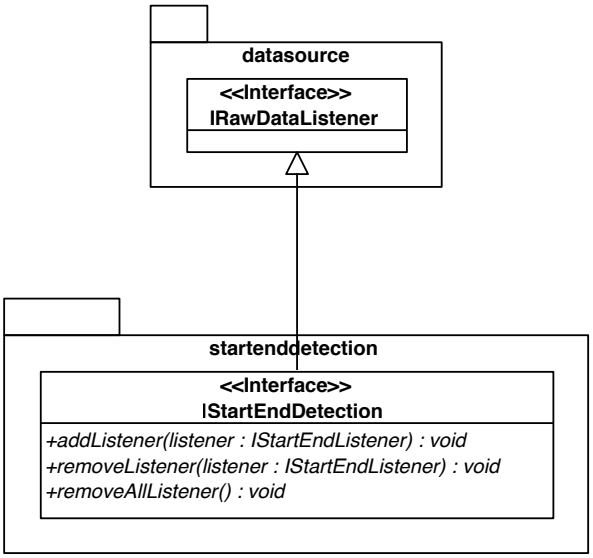

Abbildung 4.7: Übersicht über die Interfaces für die Start-End-Erkennung

Abläufe

Für die Abläufe der einzelnen Anwendungsfälle sind jeweile einzelne Prozessklassen zuständig. Diese müssen das Interface *IProcess* implementieren und haben Zugriff auf alle Komponenten. Die Verwaltung der Listener und das Weiterleiten der Daten wird in den den jeweiligen Prozessklassen übernommen. Im Folgenden werden die derzeit vorhandenen Prozessklassen kurz vorgestellt.

RecordProcess Diese Klasse implementiert die Abläufe, die benötigt werden, um Gestenrohdaten von einem Tracker zu speichern. Hierzu meldet sich die Komponente bei der Datenquelle als Listener an. Auch die Start-End-Erkennung wird gestartet. Wird der Prozess beendet, so werden die Gestendaten festgeschrieben. Derzeit wurde kein Format, wie das BioVision Hierachie (BVH) (bvhformat) verwendet, sondern der Serialisierungsmechanismus von Java. Die Steuerung kann über die, mit dem Prozess verknüpften, GUI vorgenommen werden.

RecognitionProzess Diese Klasse implementiert alle Abläufe, die nötig sind, um Gesten wiederzuerkennen bzw. zu klassifizieren. Als Trainingsdaten können diverse Gestenrohdaten geladen werden. Die Komponenten werden anhand einer gewählten Konfiguration initialisiert. Es ist möglich, alle Trainingsdaten auf einmal zu verarbeiten oder jede Geste einzeln zu trainieren. Über die mit dem Prozess verknüpften GUI lassen sich dann die Ausgaben der jeweiligen Komponenten betrachten.

4.3 Konfiguration

Da das Framework zum Testen von Komponenten gedacht ist, ist es notwendig, dass diese miteinander in beliebiger Kombination verknüpft werden können. Um dieses zu ermöglichen, werden die Komponenten in einer Konfigurationsdatei miteinander verbunden. Somit ist es möglich, ohne ein erneutes Übersetzen eine neue Komponentenkonfiguration zu erstellen.

Um die Erstellung der Konfiguration einfach zu halten, wurde das XML[2]-Format genutzt. Ein Beispiel ist im Folgenden dargestellt:

```
<?xml version="1.0" encoding="utf-8"?>
<gesture_framework_config>
    <start_end_detection
        class="org.heitsch.ba.startenddetection.WiiMoteStartEndDetection"/>
    <tracker_data_source
        class="org.heitsch.ba.datasource.DTrackDataSource">
        <property name="usemulticast" value="true"/>
        <property name="multicastaddress" value="230.230.230.230"/>
        <property name="port" value="5230"/>
    </tracker_data_source>

    <process class="org.heitsch.ba.kernel.RecordProcess">
        <property name="filenameextension" value="_gesture.ges"/>
        <property name="gesturetype" value="arrow"/>
    </process>
</gesture_framework_config>
```

Im Folgenden wird erläutert, was die Tags beschreiben.

start_end_detection definiert die Klasse für die Start-End-Erkennungskomponente

file_data_source definiert die Klasse für die Datenquellenkomponente, die gepeicherte Rohdaten liefert

tracker_data_source definiert die Klasse für die Datenquellenkomponente, die die Anbindung an einen Tracker herstellt.

pre_processor definiert die Klasse für die Vorverarbeitungskomponente

feature_extractor definiert die Klasse für die Feature Extraktorkomponente

classifier definiert die Klasse für die Klassifizierungskomponente

[2] eXtensible Markup Language

process definiert die Klassen, die die Abläufe im Framework implementiert. Hiermit wird ausgewählt, welcher Anwendungsfall angewandt werden soll.

Bei der Initialisierung des Kernels wird die Konfiguration eingelesen und das System entsprechend zusammengestellt. Damit es möglich ist, Komponenten zu parametrisieren, können Properties für die Komponenten angegeben werden. Im Beispiel ist dieses bei der Klassifizierungskomponente geschehen.

4.4 Beispielalgorithmen für die Komponenten

Im Design wurden die beiden Komponenten Feature Extraktoren und Klassifikatoren beschrieben. Um die Aufgaben der beiden Komponenten besser zu verstehen und Anregungen für die Implementierung zu geben, werden hier je zwei Beispiele für die Realisierung der beiden Komponenten gezeigt. Je ein Beispiel wird nur kurz beschrieben, um eine Grundlage für einen kurzen Vergleich zu geben. Das andere Beispiel wird genau erläutert und in der Realisierung implementiert. Für die Vorverarbeitung werden mögliche Funktionen beschrieben, mit denen einige Probleme der Vorverarbeitung verarbeitet werden können.

4.4.1 Vorverarbeitung

Nicht nur das Berechnen von neuen virtuellen Targets kann die Aufgabe der Vorverarbeitung sein, sondern auch das Skalieren der Trackerdaten. Aus dem Design ist bekannt, dass die Trackerdaten aus dem Intervall $[-1, 1]$ stammen. (siehe Gleichung (4.2)). Jedoch werden die Gesten nicht von allen Probanden mit den gleichen Dimensionen durchgeführt. Verschiedene Personen werden z.B. eine Kreisgeste mit verschiedenen Radien machen.

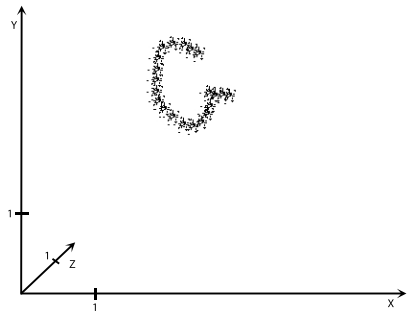

Abbildung 4.8: Gestenrohdaten

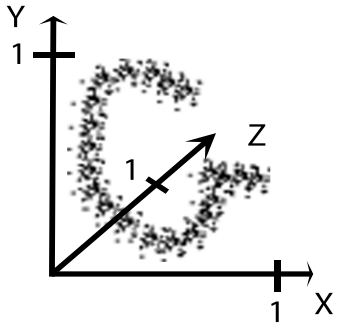

Abbildung 4.9: Skalierte Gestendaten

Um eine einheitliche Beschreibung der Geste zu ermöglichen, ist eine Skalierung notwendig, bei der die Min- und Maximalwerte der Positionen ermittelt werden und die komplette Geste auf eine Einheitsgröße transformiert wird. Abbildung 4.8 zeigt die Geste im nicht normierten Raum. Abbildung 4.9 zeigt die Geste nach der Vorverarbeitung, welche in das benötigte Intervall eingepasst ist.

Die Skalierung kann mit der *affinen Transformation* gelöst werden.

Die Basis der affinen Transformation sind lineare Polynome:

$$x_z = a_0 + a_1 * x_q + a_2 * y_q + a_3 * z_q \tag{4.6}$$
$$y_z = b_0 + b_1 * x_q + b_2 * y_q + b_3 * z_q \tag{4.7}$$
$$z_z = c_0 + c_1 * x_q + c_2 * y_q + c_3 * z_q \tag{4.8}$$

oder in der Matrixschreibweise:

$$\begin{bmatrix} x_z \\ y_z \\ z_z \end{bmatrix} = \begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x_q \\ y_q \\ z_q \end{bmatrix} + \begin{bmatrix} a_0 \\ b_0 \\ c_0 \end{bmatrix} \tag{4.9}$$

Es bleibt noch die Berechnung der optimalen Zielkoordinaten. Eine mögliche Variante ist:

$$\begin{aligned}
(x_{q1}, y_{q1}, z_{q1}) &= (min(x_q), min(y_q), min(z_q)) \Rightarrow (x_{z1}, y_{z1}, z_{z1}) = (0, 0, 0) \\
(x_{q1}, y_{q1}, z_{q1}) &= (max(x_q), min(y_q), min(z_q)) \Rightarrow (x_{z1}, y_{z1}, z_{z1}) = (1, 0, 0) \\
(x_{q1}, y_{q1}, z_{q1}) &= (min(x_q), max(y_q), min(z_q)) \Rightarrow (x_{z1}, y_{z1}, z_{z1}) = (0, 1, 0) \\
(x_{q1}, y_{q1}, z_{q1}) &= (min(x_q), min(y_q), max(z_q)) \Rightarrow (x_{z1}, y_{z1}, z_{z1}) = (0, 0, 1)
\end{aligned}$$

4.4.2 Feature Extraktoren

Um die Anzahl der Features einer Geste möglichst gering zu halten, werden Algorithmen benötigt, die eine Geste mit wenigen aussagekräftigen Werten beschreiben. Eine Methode, die auch in der Bildmesstechnik verwendet wird, ist die Beschreibung der Geste als geometrische Figur, die sich aus n Geraden zusammensetzt. Bei dieser Methode wird die Geste stückweise linear approximiert. Dieses wird im Folgenden im zwei-dimensionalen Raum gezeigt, da hier dieselben Probleme wie im drei-dimensionalen Raum auftreten. Eine Darstellung im zwei-dimensionalen Raum ist sehr viel einfacher möglich. Um die dreidimensionalen Positionsdaten in das zwei-dimensionale System zu überführen, kann folgende Abbildung ($\mathbb{R}^3 \rightarrow \mathbb{R}^2$) genutzt werden:

$$f \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x \\ y \\ 0 \end{pmatrix} \tag{4.10}$$

Somit ist die dritte Dimension nicht mehr vorhanden und die Geste kann wie eine zweidimensionale Geste betrachtet werden.

Es besteht bei dieser Art der Beschreibung das Problem der Findung von den Geraden bzw. der Start-und Endpunkte der Geraden.

In der Bildverarbeitung wird häufig festgelegt, dass die Geraden so gewählt werden müssen, dass ein Maximalfehler nicht überschritten wird. Dieses ist der Fall, wenn jeder Punkt nicht weiter als der Maximalfehler von einer Geraden entfernt ist.

In diesem Fall ist diese Art der Beschreibung jedoch nicht brauchbar, da eine Geste durch eine feste Anzahl von Geraden beschrieben werden soll, was bei der eben vorgestellten Variante nicht sichergestellt ist.

Da bei den Gesten in dieser Arbeit die Start-und Endpunkte bekannt sind, kann eine neue Bedingung aufgestellt werden. Die Geste wird über die Zeit approximiert. Dieses bedeutet, dass die Start-und Endpunkte der einzelnen Geraden über die Dauer der Geste berechnet werden. Aus 4.2.2 ist bekannt, dass der Startpunkt t und der Endpunkt d einer Geste bekannt sind. Die gesuchten n Geraden haben ihren Startpunkt (x_j, y_j) und Endpunkt (x_k, y_k) somit in dem Intervall $[v_{ist}, v_{isd}]$.

$$(x_j, y_j), (x_k, y_k) \in [v_{ist}, v_{isd}], \text{mit } j < k \tag{4.11}$$

Eine Variante ist es, die Geste in n zeitlich gleichgroße Intervalle einzuteilen. Auf diese Art

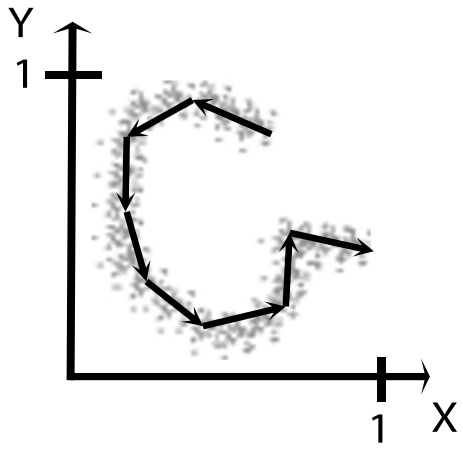

Abbildung 4.10: Zerlegung einer Geste in Geraden

wird gleichzeitig von der Zeit abstrahiert, die somit keine Rolle mehr spielt, unter der Vorraussetzung, dass die Geste in einer gleichmäßigen Geschwindigkeit durchgeführt wird. Abbildung 4.10 zeigt eine Geste, welche in Geraden zerlegt wurde.

Allgemein gilt für eine Gerade durch die zwei Punkte (x_j, y_j) und (x_k, y_k) die Gleichung:

$$\mathbf{x}(y_i - y_k) + \mathbf{y}(x_k - x_i) + y_k * x_i - y_i * x_k = 0 \tag{4.12}$$

Als Features können z.B. die Winkel der Geraden zueinander oder die Winkel der Geraden zur X-Achse genutzt werden.

Eine andere Variante ist es, die Polygonbeschreibung der Geste, also die Form, mit Fourierdeskriptoren zu beschreiben. Diese Variante ist invariant gegen Translation, Rotation und Skalierung. Ob alle diese Eigenschaften von Vorteil sind, ist noch zu untersuchen.

4.4.3 Klassifikatoren

Um die Gesten bzw. die Featurevektoren der Gesten in Klassen einteilen zu können, sind Klassifikatoren notwenig, die die in dem Design *Distanz* genannnte Eigenschaft zwischen den Featurevektoren berechnet. Eine direkte Berechnung einer solchen Distanz ist nicht möglich, daher spielen andere Strategien eine große Rolle. Die SVM werden ausfühltich beschrieben. Um einen Vergleich zu ermöglichen werden im Anschluss die SOM beschrieben.

Supported Vector Machines (SVM)

Im Folgenden werden die Grundlagen von SVM beschrieben. Eine detaillierte Beschreibung ist in Bernhard Schölkopf (2002) zu finden. Auch in der Arbeit Dosenbach (2007) werden die SVM beschrieben und genutzt.

Supported Vector Machines sind keine Maschinen im eigentlichen Sinne, sondern sind ein mathematisches Modell. Für eine SVM ist es notwendig, dass bei jedem Trainingsobjekt bekannt ist, welcher Klasse dieses zugehört. Die Trainingsobjekte sind in dieser Arbeit die Gesten, die durch ihre Featurevektoren repräsentiert werden. Aufgabe der SVM ist es, eine mehrdimensionale Hyperebene zwischen den Featurevektoren zu erzeugen, welche diese in ihren Klassen zusammenfasst.

Die Hyperebene beschreibt eine lineare Trennline zwischen den Klassen. Da zwei Klassen nicht immer linear voneinander trennbar sind, wird auf die Daten der sogenannte *Kernel*-Trick angewandt. Weiteres hierzu wird nach der Beschreibung der *linearen SVM* erläutert.

SVM können in der Grundversion nur zwei Klassen voneinander trennen. Das Multiklassenproblem wird im späteren Verlauf erklärt.

Lineare SVMs Es ist ein Trainingsdatensatz gegeben:

$$\mathcal{D} = \{(x_i, y_i)|x_i \in \mathbb{R}^p, y_i \in \{-1, 1\}\}_{i=1}^n \tag{4.13}$$

Die Variable y_i ist -1 oder 1 und beschreibt die Klasse, x_i ist der p-dimensionale Vektor. Gesucht ist die Hyperebene, die die Klassen bestmöglich voneinander trennt. Es sind jedoch sehr viele Hyperebenen möglich (Abb. 4.11). Um einen Maßstab für die Güte zu definieren, wird der Begriff des Margin eingeführt. Dieser beschreibt einen Rand um die Hyperebene, der die Klassen berührt (Abb. 4.12). Die Featurevektoren, die am Rand einer Klasse liegen und somit von dem Margin geschnitten werden, werden Supportvektoren genannt. Die beste Hyperebene ist die mit dem größten Margin (Abb. 4.12).

Diese kann wie folgt dargestellt werden:

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0, \mathbf{x}, \mathbf{w} \in \mathbb{R}^p, b \in \mathbb{R} \tag{4.14}$$

Die zugehörige Entscheidungsfunktion lautet:

$$f(\mathbf{x}) = sgn(\langle \mathbf{w}, \mathbf{x} \rangle + b) \tag{4.15}$$

Um die optimal trennende Hyperebene zu finden, können **w** und b so skaliert werden, so dass für die Supportvektoren **x** gilt:

$$|\langle \mathbf{w}, \mathbf{x} \rangle + b| = 1 \tag{4.16}$$

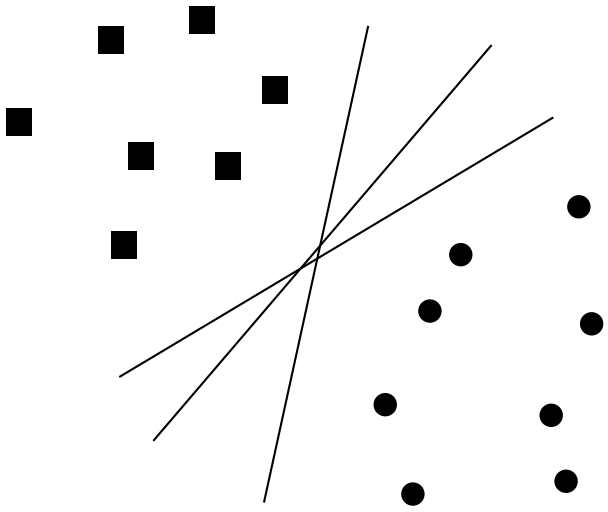

Abbildung 4.11: Viele Hyperebenen sind möglich

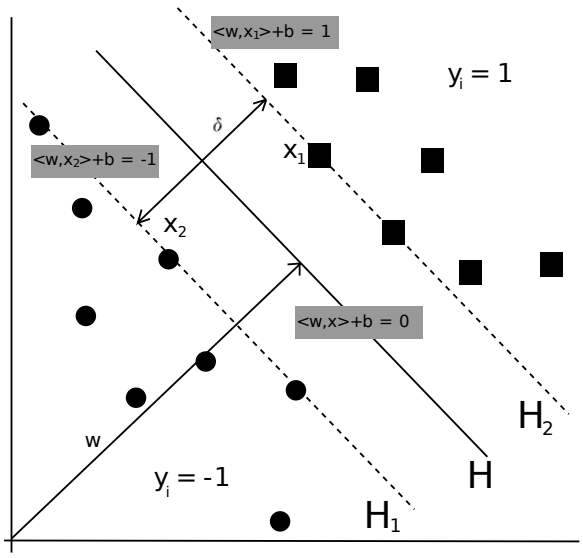

Abbildung 4.12: Bestmögliche Hyperebene

Für die Distanz δ zwischen den Ebenen H_1 und H_2 gilt:

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_1 \rangle + b &= +1 \\ \langle \mathbf{w}, \mathbf{x}_2 \rangle + b &= -1 \\ \Rightarrow \langle \mathbf{w}, (x_1 - x_2) \rangle &= 2 \\ \Rightarrow \langle \frac{\mathbf{w}}{\|\mathbf{w}\|}, (x_1 - x_2) \rangle &= \frac{2}{\|\mathbf{w}\|} \end{aligned} \tag{4.17}$$

Für die Bestimmung der Hyperebene müssen somit die folgenden Bedingungen erfüllt sein:

- Die Distanz δ soll maximiert werden. Aus der Gleichung (4.17) ist zu erkennen, dass das Problem gleichbedeutend mit der Minimierung von $\|\mathbf{w}\|$ ist.
- Zwischen den Ebenen H_1 und H_2 sind keine Featurevektoren vorhanden.

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i \rangle + b &\geq 1 && \text{für alle } y_i = 1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b &\leq -1 && \text{für alle } y_i = -1 \end{aligned}$$

daraus folgt:

$$\Rightarrow y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1, i = 1, \ldots, n \tag{4.18}$$

Zusammengefasst lässt sich das Problem wie folgt beschreiben

$$\min \frac{1}{2}\|\mathbf{w}\|^2 \text{ unter der Bedingung, dass Gleichung 4.18 erfüllt ist} \tag{4.19}$$

Das Optimierungsproblem kann mit Hilfe des Lagrange-Ansatzes gelöst werden:

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{m} \alpha_i (y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1) \tag{4.20}$$

unter der Bedingung, dass die *Lagrange Multiplikatoren* α folgende Bedingung erfüllen:

$$\alpha_i \geq 0$$

Unter Anwendung der *Karush-Kuhn-Tucker (KKT)* Bedingungen lässt sich die Entscheidungsfunktion (4.15) wie folgt darstellen:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} y_i \alpha_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b \right) \tag{4.21}$$

mit

n :	Anzahl der Supportvektoren
x_i :	Supportvektor
α_i :	Lagrange-Multiplikator
y_i :	Klassenlabel des Supportvektors $\mathbf{x}_i$

Aus der KKT-Bedingung

$$\alpha_i [y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) - 1] = 0, \text{ für alle } i = 1, \ldots, m \tag{4.22}$$

ist zu erkennen, dass die Hyperebene nur von den Supportvektoren abhängig ist, die auf dem Margin liegen und somit ein $\alpha > 0$ haben. Dieses resultiert daraus, dass Featurevektoren, die nicht auf dem Margin liegen, weiter von der Hyperebene entfernt sind. Für sie gilt: $y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1]$

So kann, anhand von wenigen Featurevektoren, die Lage der Hyperebene berechnet werden.

Nichtlineare SVMs Es ist nun möglich, ein Klassifizierungsproblem, wie es in der Abbildung 4.11 zu sehen ist, zu lösen. Jedoch sind die wenigsten Klassifizierungsprobleme nicht verschränkt. Abbildung 4.13 zeigt ein Beispiel für Featurevektoren, welche ineinander verschränkt

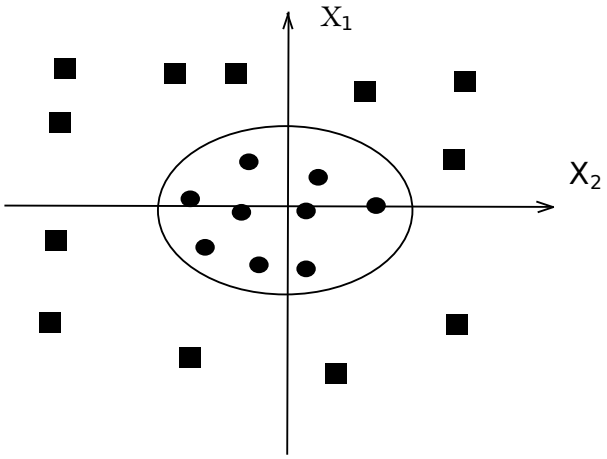

Abbildung 4.13: Originaldaten

sind. Um auch diese klassifizieren zu können, wird der Featurevektorraum in einen Vektorraum höherer Dimension transformiert. Die transformierten Daten sind in Abbildung 4.14 zu sehen.

Formal bedeutet dieses, dass die Featurevektoren $\mathbf{x} \in \mathcal{X}$ mit einer nichtlinearen Abbildung in den sogenannten Featurespace $\mathcal{F}$ projiziert werden:

$$\begin{aligned} \Phi : \mathcal{X} &\to \mathcal{F} \\ x &\mapsto \mathbf{x} := \Phi(x) \end{aligned}$$

Es ist sehr viel Rechenaufwand notwendig, um die Projektion in den höherdimensionalen Raum durchzuführen. Umgehen lässt sich dieses damit, dass nicht das Skalarprodukt von $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle$ berechnet wird, sondern stattdessen eine *Kernel-Funktion* k genutzt wird, so dass gilt:

$$k(\mathbf{x}_i, \mathbf{x}) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle \tag{4.23}$$

Die bekannte Entscheidungsfunktion (4.21) lässt sich nun wie folgt definieren:

$$f(\mathbf{x}) = sgn\left(\sum_{i=1}^{m} y_i \alpha_i k\langle \mathbf{x}_i, \mathbf{x} \rangle + b \right) \tag{4.24}$$

Es gibt viele verschiedene Kernel-Funktionen. Die wichtigsten werden im Folgenden benannt:

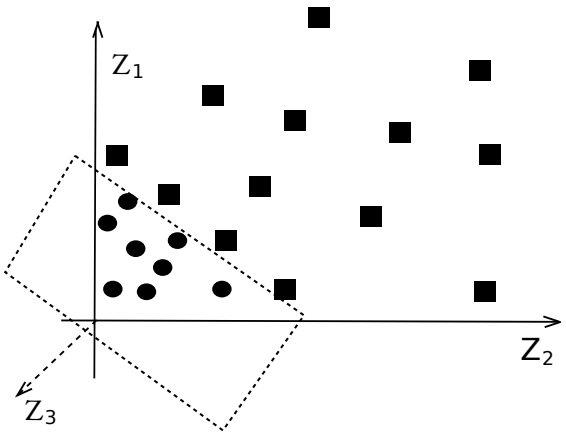

Abbildung 4.14: In einen höherdimensionierten Raum transformierte Featurevektoren

Linear-Kernel

$$k(\mathbf{x}_i, \mathbf{x}) = \langle \mathbf{x}_i, \mathbf{x} \rangle \tag{4.25}$$

Polynomal-Kernel

$$k(\mathbf{x}_i, \mathbf{x}) = \langle \mathbf{x}_i, \mathbf{x}^d \rangle, d \in \mathbb{N} \tag{4.26}$$

Imhomogen-Polynomal-Kernel

$$k(\mathbf{x}_i, \mathbf{x}) = (\langle \mathbf{x}_i, \mathbf{x} + c)^d \rangle, d \in \mathbb{N}, c \geq 0 \tag{4.27}$$

RBF-Kernel

$$k(\mathbf{x}_i, \mathbf{x}) = \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{x}\|^2}{2\sigma^2} \right), \sigma > 0 \tag{4.28}$$

Sigmoid-Kernel

$$k(\mathbf{x}_i, \mathbf{x}) = \tanh(\kappa \langle \mathbf{x}_i, \mathbf{x} \rangle + \vartheta), \kappa > 0, \vartheta < 0 \tag{4.29}$$

Multiklassenproblem Bislang ist es nur möglich, zwei Klassen voneinander zu trennen. In der Praxis hingegen ist es meistens notwendig, mehrere Klassen voneinander zu trennen. Im Folgenden werden zwei Methoden vorgestellt, mit deren Hilfe es möglich ist, mehrere Klassen voneinander zu trennen.

One Versus the Rest Um einen M-Klassen Klassifizierer zu realisieren, werden M Klassifikatoren benötigt. Für jede Klasse einen, der die Klasse gegen alle anderen Klassen

abgrenzt. Um zu entscheiden, zu welcher ein unbekannter Featurevektor gehört, werden alle M Klassifikatoren getestet. Der Klassifizierer mit dem höchsten Wert bestimmt die Klassenzugehörigkeit.

Pairwise Classifikation Um einen M-Klassen Klassifizierer zu realisieren, werden $\frac{(M-1)M}{2}$ Klassifikatoren benötigt. Für jede mögliche Klassenpaarung wird ein Klassifizierer erstellt. Um einen unbekannten Featurevektor zu klassifizieren, ist es nicht nötig, alle Klassifizierer zu testen. Es kann ein azyklisch gerichteter Graph (DAG) aufgebaut werden. Weitere Informationen hierzu sind Bernhard Schölkopf (2002) zu entnehmen. Der Vorteil dieser Methode ist, dass weniger Trainingsdaten und weniger Supportvektoren für die Abgrenzung nötig sind, da zwischen Klassen eine einfachere Hyperebene findbar ist.

Clusteranalyse mit Kohonennetzen (Self-Organizing-Maps (SOM))

Die Kohonennetze werden hier nicht im Detail erläutert, sondern nur kurz beschrieben, so dass die grundlegenden Unterschiede zu den SVM sichtbar werden.

Die Kohonennetze sind eine Art von künstlichen neuronalen Netzen. Sie basieren darauf, dass Strukturen, die zusammengehören, ähnliche Strukturen aufweisen. Das Training der Kohonennetze ist unüberwacht. Es ist somit nicht zuvor bekannt, zu welcher Klasse ein Datum gehört.

Ein Kohonennetz bildet Cluster (Klassen) um die ihn bekannten Objekte. Nach dem Training kann die Klasse zu einem unbekannten Datum ermittelt werden. Weitere Informationen zu Kohonennetzen können Meisel (2008) entnommen werden.

Vergleich

Beide Verfahren ermöglichen es, eine Menge von Daten zu klassifizieren. Beim Training unterscheiden sich die Verfahren durch die Arten. Die SVM besitzen ein überwachtes Lernen, bei dem für jedes Datum die Klasse bekannt ist. Die SOM haben einen unüberwachten Trainingsalgorithmus, bei dem einem Datum eine Klasse zugeordnet wird.

Dieser Unterschied ist der Entscheidene bei der Anwendung in der Gestenerkennung. In der Analyse wird eine Untersuchung der Gesten hinsichtlich der Gemeinsamkeiten gefordert. Dieses ist mit den SOM möglich, da bei diesem Verfahren Gesten mit ähnlichen Eigenschaften derselben Klasse zugeordnet werden können. Die Merkmale, nach denen die Gesten gruppiert werden sollen, werden durch Feature Extraktoren festgelegt.

4.5 Evaluation

Um das Framework zu testen, ist eine Gestenerkennungsmaschine auf Basis des Frameworks erstellt worden. Im Weiteren wird deren Aufbau beschrieben.

4.5.1 Aufbau des Prototypen

Für den Prototyp werden die in 4.4 beschriebenen Algorithmen verwendet. Als Datenquelle dient die Klasse *DTrackDataSource*, welche die Trackerdaten via UDP von dem *dtrack* Server empfängt und an die angemeldeten Listener weiter gibt.

Für die Vorverarbeitung wird die Klasse *ReduceDimensionPreProcessor* genutzt, die, wie in 4.4.2 beschrieben, den 3D-Raum auf einen 2D-Raum reduziert. Eine weitere Vorverarbeitung ist nicht nötig, da der eingesetze Feature Extraktor keine weitere Vorverarbeitung benötigt.

Als Feature Extraktor wird ein einfacher Algorithmus genutzt, der die Ideen aus 4.4.2 implementiert. Die Gestenrohdaten werden in eine einstellbare Anzahl von Gruppen gruppiert. Das erste Gestendatum einer Gruppe ist dabei der Startpunkt und das letzte Datum der jeweiligen Gruppe der Endpunkt einer Geraden. Somit wird die gesammte Geste durch wenige Geraden beschrieben (vgl. Abb. 4.10). Als Feature wurde der Winkel der Geraden und der X-Achse genutzt, woduch die Extraktion nicht rotationsinvariant ist, die Lage der Gesten im Raum somit erkannt wird. Die Implementierung ist in der Klasse *ApproximationFeatureExtractor* zu finden

Als Klassifikator wurde eine SVM genutzt. Die Library LibSVM Chang und Lin (2001) wurde in das Framework eingebunden. Die Klasse *LibSVMClassifier* implementiert die Klasse *AbstractClassifier* und nutzt die LibSVM für die Klassifikation. Für die Paramter der SVM wurden Standardwerte verwendet. Als Kernel kommt der RBF-Kernel zum Einsatz.

Für die Visualisierung der Ergebnisse der Klassifikation ist ein kleines Tool entwickelt worden, welches den Namen der erkannten Geste auf einem Display anzeigt.

4.5.2 Resultate

Mit der erstellten Gestenerkennungsmaschine ist es möglich, die Gesten, die vorher aufgenommen worden sind, wiederzuerkennen. Für die Evaluation wurden einfache einhändige Gesten genutzt. Die Gesten wurden bei den Versuchen in zehn Geraden zerlegt. Es zeigte sich, dass die Gesten trotz dieser sehr geringen Anzahl von Geraden gut zu erkennen sind.

Bei den Tests konnte eine Problematik festgestellt werden. Durch die Verwendung der SVM wird jede unbekannte Geste einer Klasse zugeordnet. Jedoch auch dann, wenn die Geste zu

keiner der bekannten Gesten passt. Dieses resultiert daraus, dass die SVM den Featureraum linear auftrennt. Es gibt aber keine Bereiche im Featureraum keinen Klassen zugeordnet. Auch eine Güte der Klassifizierung ist derzeit nicht vorhanden. Es ist somit notwenig, auch Gesten, die nicht verwendet werden sollen bzw. die im Kapitel 3.3 genannten *unintentional Movements*, zu trainieren. Was in der Praxis jedoch nicht möglich ist.

5 Schluss

5.1 Zusammenfassung

Im Rahmen dieser Arbeit wird ein Framework zur Erkennung von dreidimensionalen Gesten erstellt. Zusätzlich wird auf Basis des Frameworks eine Gestenerkennungsmaschine erstellt, die prototypisch die Funktion des Frameworks zeigt. Mit Hilfe der Gestenerkennungsmaschine ist es möglich, Gesten zu erkennen, die zuvor in einer Gestendatenbank abgelegt worden sind. Auch für die Erstellung einer solchen Gestendatenbank sind Funktionen im Framework vorhanden.

Im Kapitel 2 sind die Grundlagen für diese Arbeit beschrieben. Nach einer Erläuterung, welchem Forschungsgebiet die Arbeit zugeordnet ist, wird auf die derzeit vorhandenen Technologien eingegangen. Auch der Aufbau der Installation an der HAW Hamburg wird beschrieben, der in dieser Arbeit genutzt wird.

In Kapitel 3 werden drei Szenarien vorgestellt, die unter anderem als Motivation dienen. Mit den Szenarien wird analysiert, welche Anforderungen an die Gestenerkennung gestellt werden. Nach der Analyse der Anforderungen wird das Ziel der Arbeit herausgearbeitet und beschrieben. Aus dem Ziel und den globalen Anforderungen werden die Anforderungen an Gestenerkennungsmaschinen spezifiziert. Da es noch an Wissen fehlt, wie die ideale Gestenerkennungsmaschine aussieht, sind die Anforderungen so spezifiziert, dass es möglich ist, die Algorithmen für die Erkennung auszutauschen. Um dieses zu realisieren, basiert die Gestenerkennungsmaschine auf einem Framework, das wesentlicher Bestandteil dieser Arbeit ist. Die Anforderungen an das Framework werden im Abschnitt 3.7 beschrieben.

Im Kapitel 4 werden die Komponenten, aus denen sich die Gestenerkennung zusammensetzt, identifiziert und beschrieben. Auch werden die Komponenten mathematisch definiert. Auf Basis der Komponentenbeschreibung wurde ein Grobdesign für Framework entwickelt. In den jeweiligen Realisierungsabschnitten zu den Komponenten wird beschrieben, wie diese implementiert wurden bzw. zu implementieren sind.

5.2 Ausblick

Das Framework wird in dieser Arbeit in dem Labor der HAW- Hamburg genutzt. Dabei werden die Gestenrohdaten vom *dTrack*- Server geliefert, der vier Infrarotkameras für die Positionbestimmung von Targets nutzt (vgl. Kapitel 2.5). Im Folgenden werden Anwendungsmöglichkeiten genannt, bei denen das Framework unterstützend eingesetzt werden kann.

Anbindung an andere Eingabemedien

Derzeit ist die Anbindung an einen *dTrack*- Server realisiert. Durch die offene Schnittstelle ist es aber auch möglich, diverse andere Eingabegeräte einzubinden. Beispiele wären:

Nindento WiiMote: Erkennung von Handgesten, die mit der Nintendo WiiMote gemacht wurden.

Kamerabasiertes Handtracking: Erkennung von Gesten mit Standardkameras, wobei die Position der Hände von einer Person aus Bildern extrahiert wird.

Single-/Multitouchscreen: Erkennung von Gesten auf Touchscreens.

Anwendung in anderen Kontexten

In dieser Arbeit wurden Gesten genutzt, um mit der Powerwall zu interagieren. Es sind jedoch noch viele weitere Anwendungen vorstellbar. Eine davon ist die Steuerung des Umfeldes im Haushalt mittels Gesten. So könnten z.B. Gardinen oder das Fernsehprogramm mittels Gesten gesteuert werden.

Literaturverzeichnis

bvhformat : *BHVformat*. Webseite. – URL http://www.cs.wisc.edu/graphics/Courses/cs-838-1999/Jeff/BVH.html. – Zugriffsdatum: 12.08.2008

A.R.T-GmbH 2008 A.R.T-GMBH: *Advanced Realtime Tracking GmbH*. Webseite. 2008. – URL http://www.ar-tracking.de/. – Zugriffsdatum: 12.08.2008

Bernhard Schölkopf 2002 BERNHARD SCHÖLKOPF, Alexander J. S.: *Learning with Kernels. Supported Vector Machines, Regularization, Optimization, and eyond*. MIT Press, 2002. – ISBN 0-262-19475-9

Boetzer 2008 BOETZER, Joachim: *Bewegungs ung gestenbasierte Applikationssteuerung auf Basis eines Motion Trackers*, HAW Hamburg, Bachelorarbeit, 2008. – URL http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/boetzer.pdf. – Zugriffsdatum: 12.08.2008

Chang und Lin 2001 CHANG, Chih-Chung ; LIN, Chih-Jen: *LIBSVM: a library for support vector machines*, 2001. – Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Christie 2008 CHRISTIE: *Providing Digital and Film Projectors for Any Application*. Webseite. 2008. – URL http://www.christiedigital.co.uk/EMEAEN/. – Zugriffsdatum: 12.08.2008

Clarke 2002 CLARKE, Darren J.: *MIT grad directs Spielberg in the science of moviemaking*. Webseite. July 2002. – URL http://web.mit.edu/newsoffice/2002/underkoffler-0717.html. – Zugriffsdatum: 12.08.2008

Craik 1943 CRAIK, K.: *The Nature of Explanation*. Cambridge University Press, 1943

Dosenbach 2007 DOSENBACH, Kerstin: *Klassifikation von Audiosignalen mit Support Vector Machines*, HAW Hamburg, Diplomarbeit, 2007. – URL http://www.informatik.haw-hamburg.de/fileadmin/Homepages/ProfFohl/Abschlussarbeiten/DA_Dosenbach.pdf. – Zugriffsdatum: 12.08.2008

Fischer 2008 FISCHER, Christian: *Entwicklung eines multimodalen Interaktionssystems für computergestützte Umgebungen*, HAW Hamburg, Masterarbeit, 2008. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/fischer.pdf`. – Zugriffsdatum: 12.08.2008

George G. Robertson 1991 GEORGE G. ROBERTSON, Stuart K. C.: Cone Trees: animated 3D visualizations of hierarchical information. In: *CHI '91: Proceedings of the SIGCHI conference on Human factors in computing systems*. New York, NY, USA : ACM, 1991, S. 189–194. – ISBN 0-89791-383-3

Google 2008 GOOGLE: *Google Maps*. Webseite. 2008. – URL `http://maps.google.com`. – Zugriffsdatum: 12.08.2008

Gregor 2006 GREGOR, Sebastian: *Entwicklung einer Hardwareplattform für die Ermittlung von Positionsdaten innerhalb von Gebäuden*, HAW Hamburg, Bachelorarbeit, 2006. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/bachelor/gregor.pdf`. – Zugriffsdatum: 12.08.2008

Hinck 2008 HINCK, Steffen: *Prototypische Entwicklung eines 3D-Eingabegeräts für stationäre und mobile Anwendungen*, HAW Hamburg, Masterarbeit, 2008. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/hinck.pdf`. – Zugriffsdatum: 12.08.2008

Ishii u. a. 1994 ISHII, Hiroshi ; KOBAYASHI, Minoru ; ARITA, Kazuho: Iterative design of seamless collaboration media. In: *Commun. ACM* 37 (1994), Nr. 8, S. 83–97. – ISSN 0001-0782

J. Boetzer 2008 J. BOETZER, M. Vogt P. Wendt K.: Gestenbasierte Interaktion mit Hilfe von Multitouch und Motiontracking. In: CLEVE, Jürgen (Hrsg.): *WIWITA 2008* Hochschule Wismar (Veranst.), Hochschule Wismar, 2008, S. 38–49

König u. a. 2007 KÖNIG, Werner A. ; BIEG, Hans-Joachim ; REITERER, Harald: Laserpointer-Interaktion für große, hochauflösende Displays. In: *Mensch & Computer 2007: Interaktion im Plural, 7. Konferenz für interaktive und kooperative Medien*, Oldenbourg Verlag, Sep 2007, S. 69 – 78. – URL `http://hci.uni-konstanz.de/intehrdis/koenig_LaserInteraktion_MC07.pdf`. – Zugriffsdatum: 12.08.2008. – ISBN 978-3-486-58496-7

Lee 2008 LEE, Jonny C.: *Jonny Chung Lee*. Webseite. 2008. – URL `http://www.cs.cmu.edu/~johnny/`. – Zugriffsdatum: 12.08.2008

Lorenzen 2005 LORENZEN, Jonas: *Konturbasier te Gestenerkennung mit Hilfe der Dynamischen Programmierung*, HAW Hamburg, Diplomarbeit, 2005

Marti A. Hearst 1997 MARTI A. HEARST, Chandu K.: Searching and browsing text collections with large category hierarchies. In: *CHI '97: CHI '97 extended abstracts on Human factors in computing systems*. New York, NY, USA : ACM, 1997, S. 301–302. – ISBN 0-89791-926-2

Jung und Matt/next GmbH 2007 JUNG UND MATT/NEXT GMBH: *Nextwall*. Webseite. 2007. – URL `http://nextwall.de`. – Zugriffsdatum: 12.08.2008

Motion Analysis Corporation 2008 MOTION ANALYSIS CORPORATION: *Motion Analysis Corporation, the Motion Capture Leader*. Webseite. 2008. – URL `http://www.motionanalysis.com/`. – Zugriffsdatum: 12.08.2008

Meisel 2008 MEISEL, Prof. Dr.-Ing. A.: *Musterklassifikation mit Neuronalen Netzen*, HAW Hamburg, Vorlesungsskript, 2008. – URL `http://www.informatik.haw-hamburg.de/fileadmin/Homepages/ProfMeisel/Vorlesungen/WP_RobotVision/V/RV07.pdf`. – Zugriffsdatum: 12.08.2008

MIT 2008 MIT: *Massachusetts Institute of Technology*. Webseite. 2008. – URL `http://web.mit.edu/`. – Zugriffsdatum: 12.08.2008

Motion 2004 MOTION, Meta: *Gypsy Motion capture system*. Webseite. 2004. – URL `http://www.metamotion.com/`. – Zugriffsdatum: 12.08.2008

Napitupulu 2008 NAPITUPULU, Jan: *Ein System mit skalierbarer Visualisierung zur Entwicklung kollaborativer Serious Games*, HAW Hamburg, Masterarbeit, 2008. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/arbeiten/master/napitupulu.pdf`. – Zugriffsdatum: 12.08.2008

Navy 2002 NAVY, United S.: *Navy NewsStand- Eye on the Fleet*. Webseite. 2002. – URL `http://www.news.navy.mil/view_single.asp?id=3523`. – Zugriffsdatum: 12.08.2008

Nintendo 2008 NINTENDO: *Wii*. Webseite. 2008. – URL `http://wii.com/`. – Zugriffsdatum: 12.08.2008

Norman I. Badler 1998 NORMAN I. BADLER, Roger W. W.: SIGGRAPH VRML 3D Ph.D Conetree. In: *SIGGRAPH Comput. Graph.* 32 (1998), Nr. 3, S. 43–44. – ISSN 0097-8930

Opera 2008 OPERA: *Opera*. Webseite. 2008. – URL `http://www.opera.com`. – Zugriffsdatum: 12.08.2008

Patterson 2007 PATTERSON, Dale: 3D SPACE: using depth and movement for selection tasks. In: *Web3D '07: Proceedings of the twelfth international conference on 3D web technology*. New York, NY, USA : ACM, 2007, S. 147–155. – ISBN 978-1-59593-652-3

Pavlovic u. a. Jul 1997 PAVLOVIC, V.I. ; SHARMA, R. ; HUANG, T.S.: Visual interpretation of hand gestures for human-computer interaction: a review. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 19 (Jul 1997), Nr. 7, S. 677–695. – ISSN 0162-8828

Piening 2007a PIENING, Andreas: *RESCUE Leitstand für Desaster-Szenarien*, HAW Hamburg, Ausarbeitung Projekt, 2007. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-proj/piening/report.pdf`. – Zugriffsdatum: 12.08.2008

Piening 2007b PIENING, Andreas: *RESCUE Leitstand für Desaster-Szenarien*, HAW Hamburg, Ausarbeitung Anwendungen 2, 2007. – URL `http://users.informatik.haw-hamburg.de/~ubicomp/projekte/master06-07-aw/piening/report.pdf`. – Zugriffsdatum: 12.08.2008

Rahimi und Vogt 2008 RAHIMI, Mohammadali ; VOGT, Matthias: *Gestenbasierte Computerinteraktion auf Basis von Multitouch-Technologie*, HAW Hamburg, Bachelorarbeit, 2008

Rizzo 2007 RIZZO, Prof. A.: *Body and Gesture Tracking, Representation and Analysis: Implications for HCI, Human Motor Function & Virtual Human Action*. Webseite. 2007. – URL `http://imsc.usc.edu/research/project/gesttrack/gesttrack_nsf7.pdf`. – Zugriffsdatum: 12.08.2008

Robertson 1991 ROBERTSON, Jock Mackinlay Stuart C.: Animated 3D Visualizations of Hierarchical Information. In: *SIGCHI 91 Proceedings*, 1991, S. 189–194

SIGCHI 1996 SIGCHI, ACM: *Definition and Overview of Human-Computer Interaction*. Webseite. 1996. – URL `http://sigchi.org/cdg/cdg2.html`. – Zugriffsdatum: 12.08.2008

SmartKom 2007 SMARTKOM: *SmartKom*. Webseite. 2007. – URL `http://www.smartkom.org`. – Zugriffsdatum: 12.08.2008

Weiser 1991 WEISER, Mark: The Computer for the 21st Century. In: *Scientific American* (1991), Februar. – URL `http://www.ubiq.com/hypertext/weiser/SciAmDraft3.html`. – Zugriffsdatum: 12.08.2008

Wikipedia 2008 WIKIPEDIA: *Motion capture*. Webseite. 2008. – URL `http://en.wikipedia.org/wiki/Motion_capture`. – Zugriffsdatum: 12.08.2008

Versicherung über Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 13. August 2008	
Ort, Datum	Unterschrift