Albert-Ludwigs-Universität Freiburg
Technische Fakultät
Institut für Informatik
Arbeitsgruppe Autonome intelligente Systeme

Techniken für Virtual Reality Spiele mit Ganzkörperbewegungserfassung

Bachelor-Arbeit

vorgelegt von:	Philipp Ruchti
Matrikelnummer:	2548383
Betreuender Professor:	Prof. Dr. Wolfram Burgard
Zweitgutachter:	Prof. Dr. Matthias Teschner
betreut von:	Jürgen Sturm
Datum:	30. August 2010

Eigenständigkeitserklärung

Hiermit erkläre ich, dass ich diese Bachelor-Arbeit mit dem Titel

Techniken für Virtual Reality Spiele mit Ganzkörperbewegungserfassung

selbständig verfasst habe, keine anderen als die angegebenen Quellen/Hilfsmittel verwendet wurden und alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten Schriften entnommen sind, als solche kenntlich gemacht sind. Darüber hinaus erkläre ich, dass diese Bachelor-Arbeit nicht, auch nicht auszugsweise, bereits anderweitig verwendet wurde.

Philipp Ruchti

Freiburg im Breisgau, den 30. August 2010

Abstract

In der Spielebranche gab es nach dem Erscheinen der Wii einen Aufschwung für Spiele mit Bewegungssteuerung. Eine zentrale Limitierung gegenwärtiger Systeme ist jedoch, dass derzeit keine Anpassung des Spieles an den Spieler vorgenommen wird, das heißt, dass der Spieler vorgegebene Bewegungsabläufe mühsam erlernen muss. Im Zuge dieser Bachelor-Arbeit wurde ein Virtual Reality Spiel mit Ganzkörperbewegungserfassung ausgearbeitet, um Lösungsansätze für dieses Problem zu entwickeln. Hierzu wird ein Ganzkörperbewegungsaufzeichnungsanzug von Xsens sowie eine 3D-Videobrille von Zeiss verwendet. Aus den geglätteten Positions-, Geschwindigkeits- und Beschleunigungsdaten des Anzuges werden Features extrahiert, aus denen ein spielerspezifisches Modell der Aktionen eines Nutzers erlernt werden. Des Weiteren wird eine Methode zur Abschätzung und Vorhersage des eintretenden Klassifikationsfehlers vorgestellt. Mit Hilfe des spielerspezifischen Modells können während des Spiels die Aktionen des Anwenders klassifiziert und zur Spielsteuerung verwendet werden. In der implementierten Spielewelt können Kisten gestapelt und bewegt sowie Münzen eingesammelt werden. In den ausgeführten Experimenten wird die Erkennungsrate der hier verwendeten Technik zum Erlernen solcher Aktionen geprüft und mit der eines konventionellen Schwellwert-Klassifikators verglichen. Auch wird die Übertragbarkeit der Modellparameter von verschiedenen Aktionen zwischen Spielern untersucht. Dabei wurde festgestellt, dass sich Modellparameter nicht zwischen Spielern übertragen lassen. In den Experimenten zeigte sich, dass mit der hier vorgestellten Technik Aktionen mit einer hohen Genauigkeit erkannt werden können und diese Technik einem konventionellen Klassifikator vorzuziehen ist. Daher ist der hier vorgestellte flexible Lernansatz für solche Anwendungen gut geeignet.

Inhaltsverzeichnis

1 Einleitung

Lange Zeit wurden Spiele lediglich mittels Tastatur, Maus oder Joystick gesteuert. In den letzten Jahren widmete sich die Spieleindustrie mehr und mehr der Entwicklung neuer Eingabetechniken. Durch diese wird es dem Spieler ermöglicht, Spiele mit eigenen Körperbewegungen zu steuern. Die erste, mittlerweile weit verbreitete Neuerung auf diesem Gebiet wurde von Nintendo Ende 2006 mit dem System *Wii* vorgestellt (siehe Kapitel 2 sowie Abbildung 1.1). Zur Steuerung der Wii erhält der Spieler ein Eingabegerät, welches Geschwindigkeiten sowie Beschleunigungen messen kann. So ist es dem Spieler möglich, eigene Bewegungen in ein Spiel zu integrieren.

Abbildung 1.1: Rechts: Nintendo Wii. Links: Microsofts Kinect für die XBox360.

Ende dieses Jahres wird von Microsoft das System *Kinect* (siehe Kapitel 2 sowie Abbildung 1.1) eingeführt, welches bei der Steuerung von Spielen einen neuen Ansatz wählt. Bei Kinect wird der Spieler mittels Daten einer 3D-Kamera erfasst und kann dadurch ohne ein Eingabegerät zu berühren, mit vollem Körpereinsatz mit einem Spiel interagieren. Ein System zur Steuerung eines Spieles mit Ganzköperbewegungserfassung wird auch in dieser Arbeit vorgestellt.
Bei diesen Systemen handelt es sich um *Virtual Reality* Anwendungen. Mit dem Begriff Virtual Reality bezeichnet man eine computersimulierte Umgebung, welche entweder die reale Welt nachbildet oder aber völlig fiktiv ist. Der Benutzer nimmt die Virtual Reality über ein Display oder ähnliche Vorrichtungen visuell wahr. Zusätzlich ist es auch möglich, dass er die Welt hört oder mittels haptischer Aktoren sogar spürt. Mit der virtuellen Welt kann der Benutzer interagieren. Hierzu sind

verschiedenste Möglichkeiten gegeben (siehe Kapitel 2). Die Möglichkeiten reichen von Eingabe per üblicher Mittel, wie Tastatur und Maus, bis hin zu Ganzkörperbewegungserkennungssystemen, die den Spieler mit seinem ganzen Körper ins Spiel integrieren. Ein kommerzielles Beispiel für letztere ist das oben bereits erwähnte System Kinect von Microsoft.

Virtual Reality ist aber nicht nur in der Spieleindustrie von Interesse. Ein weiterer Anwendungszweig der Virtual Reality sind Simulatoren. Hier findet sie professionelle Anwendung zu Ausbildungszwecken. Beispielsweise werden Piloten oder Kapitäne an Simulatoren ausgebildet (siehe Abbildung 1.2). Ein Vorteil hierbei ist, dass teure Ausbildungsflüge oder -fahren eingespart werden können. Außerdem kann der Ausbilder seinen Schüler gezielt in bestimmte Szenarien versetzen. Auch in der Medizin, bei der Armee oder bei der Katastrophenbekämpfung werden Simulatoren vielseitig eingesetzt.

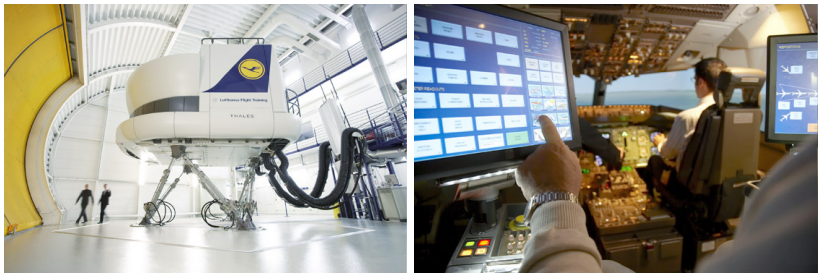

Abbildung 1.2: Ein Flugsimulator der Lufthansa Flight Training GmbH.

Zu einem Virtual Reality System gehören nach Brooks [10] verschiedenste Komponenten. Er unterteilt die Bestandteile in zwei Gruppen: Essentielle und optionale Komponenten. Eine wichtige Komponente eines solchen Systems ist ein Display oder eine andere Visualisierungstechnik, die den Nutzer visuell in die Virtual Reality versetzt und möglicherweise von dieser abschirmt. Dazu gibt es verschiedenste Systeme. Neben Videobrillen mit oder ohne Durchsicht gibt es Beamerwände und virtuelle Tische. Des Weiteren bedarf es eines Bewegungserkennungssystems, welches die Bewegungen des Nutzers an den Computer überträgt. Auch hier gibt es viele verschiedene Lösungen, von Maus und Tastatur, einfachen Joysticks über optische Marker bis hin zu Bewegungsaufzeichnungsanzügen. Der Computer muss aus den Daten des Bewegungserkennungssystems die entsprechende Sicht des Nutzers und die Änderungen in der virtuellen Welt berechnen und danach die Sicht der Welt auf dem Display anzeigen. Darüber hinaus können Elemente wie haptisches Feed-

back, die Bewegung einer ganzen Simulatorkabine oder räumlicher Sound hinzu kommen, um noch mehr Realitätsgefühl zu schaffen.

Das Aufkommen von Ganzkörperspielen führt auch im wissenschaftlichen Bereich zu neuen Fragestellungen und Problemen. Durch die individuelle Bewegungsart eines jeden Spielers ist die Eingabe nicht mehr eindeutig, wie es etwa beim Drücken einer Taste der Fall ist. Es stellt sich die Frage wie man mit der Menge der Bewegungserfassungsdaten umgeht. Wie passt man das Spiel an den Spieler an? Welche Parameter beschreiben die Anpassung des Spieles an den Spieler, welche die Aktionen des Spieles und wie können diese erlernt werden?
In herkömmlichen Spielen beschränkt sich die Anpassung des Spieles an den Spieler auf eine mehr oder minder frei wählbare Belegung von Tasten, zu verschiedenen vorgegebenen Aktionen. Der Nachteil dieses Verfahrens ist, dass es abgesehen von diesen Einstellungen lediglich zu einer Anpassung des Spielers an das Spiel kommt. Mit dem Aufkommen von Bewegungserfassung in der konventionelle Spielewelt durch Systeme wie die Wii oder Kinect ist auch eine Anpassung des Spieles an den Spieler erforderlich. In Spielen mit Bewegungserfassung ist es notwendig, dass auch das Spiel lernt, um sich den Eigenarten des Spielers anzupassen. Nur so ist es möglich, Spiele auf verschieden Nutzer einzustellen. Dadurch wird mit kommenden Spielen die Frage aufkommen, wie solche Anpassungen an den Spieler vorgenommen werden und wie sich diese am besten parametrisieren lassen.

In dieser Arbeit wird einigen der oben genannten Fragen nachgegangen. Es werden ein selbst entwickeltes Virtual Reality Framework sowie einige Techniken vorgestellt, die in Ganzkörperspielen verwendet werden können. Dabei wird der Frage nachgegangen, welche Techniken nötig und hilfreich sind, um ein Spiel mit Ganzkörperbewegungserfassung umzusetzen und an den Spieler anzupassen. In diesem Kontext wird unter anderem das an den Spieler angepasste Erlernen von Aktionen gezeigt und mit der naiven Technik des festen Programmierens von Aktionen anhand von Schwellwerten verglichen. Des Weiteren werden zwei Techniken vorgestellt, um anhand von verrauschten Positionsmessungen geglättete Positionsdaten sowie Werte für Geschwindigkeit und Beschleunigung abzuleiten. Auch wird eine Vorhersage der Erkennungsrate eines erlernten Parametersatzes vorgestellt. Anschließend werden diese Techniken sowie die in dieser Arbeit vorgestellte Lernmethode an mehreren Aktionen getestet. Zum Testen des Lernprozesses werden für mehrere Testpersonen Aktionen erlernt und die Güte des Ergebnisses evaluiert.

In das in dieser Arbeit implementierte Virtual Reality Spiel wird der Spieler mittels Daten eines Bewegungsaufzeichnungsanzugs integriert. Die Welt sieht der Nutzer in einer Videobrille. In der Spielewelt kann der Spieler verschiedenste Aktionen ausführen. Das Einsammeln von Münzen, das Umhertragen oder Umwerfen von Kisten gehört genauso zu möglichen Aktionen wie Brücken einstürzen zu lassen oder Schalter zu drücken um Gegnern zu entkommen. Als System zur Bewegungsaufzeichnung wird Xsens' MVN-Anzug verwendet. Anders als bei konventionellen Spielen, bei welchen der Spieler eine Person steuert, ist hier der Spieler selbst die Person. Somit kann sich der Spieler mit vollem Körpereinsatz ins Spiel integrieren und bekommt so das Gefühl er befinde sich im Spiel.

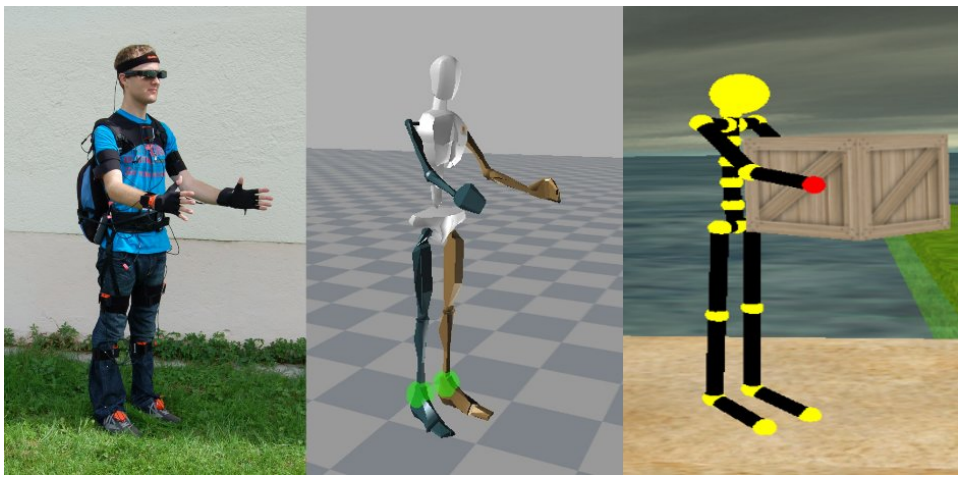

Abbildung 1.3: Drei Sichten auf das Framework: Links der Nutzer im Anzug, mittig die vom MVN-Studio erzeugten Körpersegmente, rechts die Person im Spiel.

Die weitere Arbeit gliedert sich wie folgt: Zuerst werden in Kapitel 2 Arbeiten vorgestellt, die sich mit ähnlichen Themen und Fragestellungen beschäftigt haben. In Kapitel 3 werden einige mathematische Grundlagen gelegt, welche für das Verständnis der Arbeit hilfreich sind. Im folgenden Kapitel 4 wird der Ansatz dieser Arbeit beschrieben. In Kapitel 5 wird die spezifische Implementierung sowie das in dieser Arbeit implementierte Framework vorgestellt. In Kapitel 6 werden die Implementierung und die verwendeten Methoden geprüft. Abschließend werden in Kapitel 7 die vorgestellten Verfahren und Experimente noch einmal zusammengefasst und mögliche Verbesserungen diskutiert.

2 Themenverwandte Arbeiten

Um Virtual Reality-Anwendungen nutzen zu können, bedarf es einer Form der Erkennung von Bewegungen und Aktionen, umso mit der Welt interagieren zu können. Dazu sind viele verschiedene Arten der Interaktion möglich.
Eine der einfachsten und stabilen Varianten zur Bewegungserkennung, beziehungsweise Lokalisierung von Objekten, ist die Verwendung von optischen Markern.
Schmalstieg u. a. [35] haben ein Framework entwickelt, welches mittels visuell erkannten Markern Positionen und Orientierungen im Raum detektieren kann. Sie benötigen lediglich eine Kamera und einige bedruckte Blätter um Objekte in einer dreidimensionalen, virtuellen Welt lokalisieren und bewegen zu können (siehe Abbildung 2.1).

Abbildung 2.1: Ein Versuchsaufbau für das Framework von Schmalstieg u. a. [35].

Eine andere Möglichkeit zur Lokalisierung und Verfolgung eines Objektes, beziehungsweise einer Person, ist eine Flugzeitkamera. So haben Ganapathi u. a. [15] ein Verfahren entwickelt, welches die gesamte Körperpose einer Person in Echtzeit aus einem solchen Kamerabild erkennt.
Ein anderer Ansatz ist die Lokalisierung von Personen oder Objekten ohne Marker,

direkt aus einem eindimensionalen, herkömmlichen Kamerabild. Freifeld u. a. [11] und Baak u. a. [2] haben jeweils eine Methode entwickelt, um aus einem simplen Kamerabild nicht nur die Position, sondern sogar die gesamte Körperpose zu erkennen und zu verfolgen.
In [14] wird ein Verfahren vorgestellt, um aus einem oder mehreren konventionellen Kamerabildern nicht nur die Pose von Menschen zu erkennen und zu verfolgen, sondern sogar auf die Konfiguration eines zugrundeliegenden kinematischen Skelettes von Menschen oder Tieren zu schließen.
Rosenhahn u. a. [33] untersuchten Bedingungen, welche die Bewegung des Menschen einschränken. In ihrer Arbeit studierten sie die Bewegung und die Restriktionen dieser anhand einer Person die verschiedene Sportgeräte nutzt. Hierzu wurde ein markerloses Verfahren mit Bildern einfacher Kameras aus verschiedenen Blickwinkeln, zur Erkennung der Person gewählt.
Die meisten solcher Verfahren können lediglich starre Objekte oder Personen mit sich nur leicht verformenden Kleidungsstücken erkennen und verfolgen. In den Arbeiten [13] und [34] werden Bewegungserfassungsverfahren präsentiert, welche auch auf sich verformende Körper oder Personen, deren Kleidungsstücke sich stark verformen, angewandt werden können.
Auch ist häufig in den Ansätzen eine statische Kamera gefordert. Hasler u. a. [18] stellen mit ihrer Arbeit (siehe Abbildung 2.2) eine Möglichkeit vor, Personen aus Bildern von unkalibrierten, sich bewegenden Kameras zu erkennen.

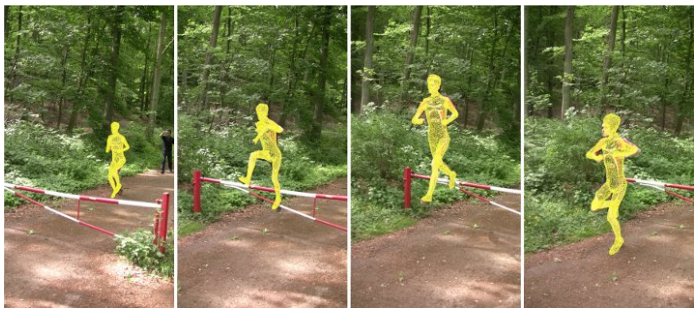

Abbildung 2.2: Bewegungserkennung aus unsynchronisierten, sich bewegenden Kameras aus der Arbeit von Hasler u. a. [18].

Nikolakis u. a. [28] haben eine ungewöhnliche Virtual Reality Anwendung geschaffen. Sie haben eine Anwendung für sehbehinderte Menschen vorgestellt. Sie ermöglichen es den Probanden mittels eines Bewegungserkennungshandschuhs, welcher gleichzeitig haptisches Feedback liefert, virtuell Objekte zu studieren und mit ihnen zu interagieren.

Park u. a. [29] beschränken ihre Interaktion mit ihrem Programm sogar auf die Bewegung eines Auges. Sie demonstrierten eine digitale Bildergalerie, welche mittels eines *Eyetrackers* (siehe Abbildung 2.3) gesteuert und in einem am Kopf befestigten Display betrachtet werden kann.

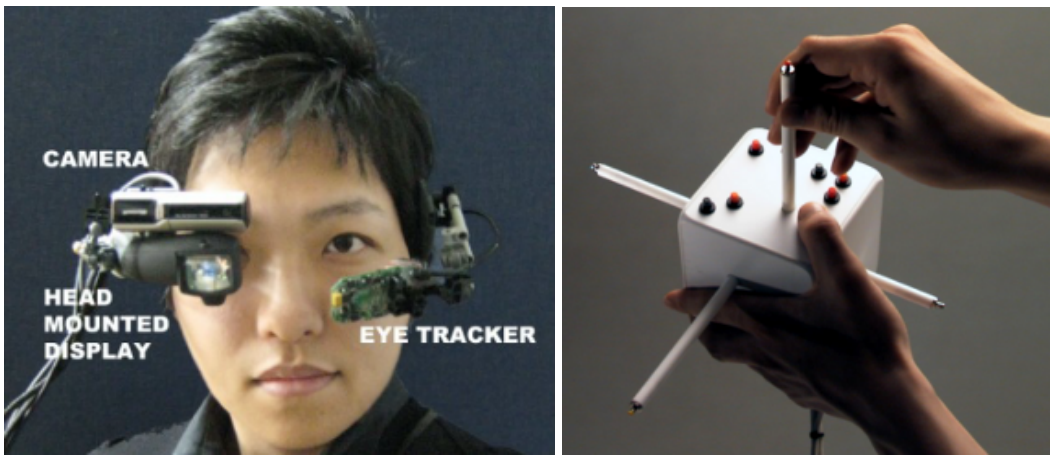

Abbildung 2.3: Links: Der in [29] verwendete Versuchsaufbau mit Eyetracker zur Steuerung einer Bildergalerie. Rechts: Die Cubic Mouse entwickelt von Fröhlich u. Plate [12] zur intuitiven Eingabe von Positionen in eine 3D-Anwendung.

Aber nicht nur die direkte Bewegungserkennung des menschlichen Körpers wird für die Steuerung von Virtual Reality Anwendungen genutzt.
In [12] wird eine andere Form der Eingabe in eine dreidimensionale Anwendung vorgestellt. Die Cubic Mouse (siehe Abbildung 2.3) ist ein Würfel mit drei herausstehenden Achsen. Diese können längs ihrer Achse verschoben werden, dadurch ist es dem Nutzer möglich intuitiver Positionen im Raum zu definieren.
Eine einfache Möglichkeit die komplette Körperpose einer Person zu ermitteln und in eine virtuelle Anwendung zu integrieren, sind Bewegungsaufzeichnungsanzüge. Diese arbeiten oft optisch und sind somit nur im Studio zu verwenden, aber es gibt auch auf Sensoren basierende Systeme.
Vlasic u. a. [36] haben einen Bewegungsaufzeichnungsanzug entwickelt, dieser besteht aus kleinen, tragbaren Ultraschallflugzeit-, Beschleunigungs- und Orientierungssensoren.
Pons-Moll u. a. [30] stellten einen Ansatz vor, der sowohl Daten eines Bewegungsaufzeichnungsanzugs, als auch Daten mehrerer Kameras nutzt, um so stabilere, unverrauschtere Posen einer Testperson zu berechnen.

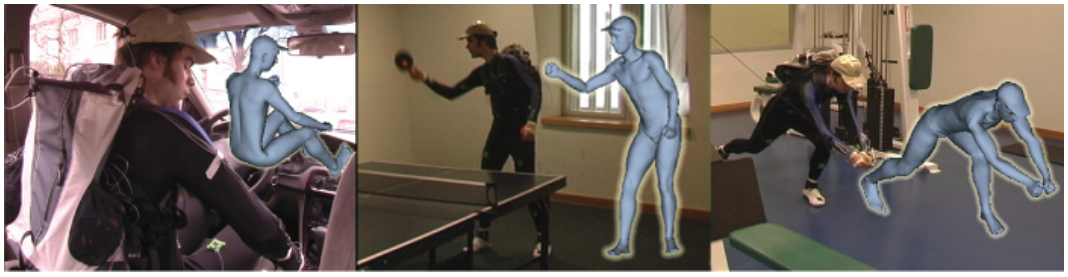

Abbildung 2.4: Der in [36] verwendete Bewegungsaufzeichnungsanzug.

Ein Bewegungsaufzeichnungsanzug, der ebenfalls auf Beschleunigungs- und Orientierungssensoren beruht und relativ frei einsetzbar ist, kommt auch in dieser Arbeit zum Einsatz.

Als Ausgabe einer Virtual Reality Anwendung kommen verschiedenste Techniken, von Beamern bis zu hochentwickelten Videobrillen, zum Einsatz.
Ein Beispiel für letztere ist die Arbeit von Liu u. a. [23]. In dieser wird eine halbdurchsichtige Brille präsentiert, bei welcher mit Hilfe einer flüssigen Linse verschiedene Fokusebenen angesteuert werden können. Damit wird das Scharfstellen verschiedener Tiefenebenen ermöglicht. Die Motivation dieser Arbeit ist, neben mehr Realitätsgefühl, die Beobachtung, dass normale Videobrillen, die lediglich eine Fokusebene besitzen, bei längerer Anwendung die Augen und ihre umgebende Muskulatur schädigen oder schwächen können.
Mohler u. a. [26] beschäftigten sich mit den Fragen, wie der Mensch Entfernungen wahrnimmt und wie sich seine Entfernungsschätzung verändert, wenn er sich in einer Umgebung bewegt, welche er als virtuelle Abbildung sieht, durch die er verbal geleitet wird oder durch die er sich blind bewegt.
De Luca u. a. [6] haben für die Bewegung in einer virtuellen Welt die Entwicklung *Cyberwalk* (siehe Abbildung 2.5) evaluiert. Der Cyberwalk bietet dem Spieler die Möglichkeit sich unbegrenzt in einem virtuellen Raum zu bewegen, ohne von Wänden beschränkt zu sein. Hierbei läuft der Spieler auf einer Rolle von Laufbändern, welche es ermöglicht den Boden in vertikaler Richtung beliebig so zu verschieben, dass sich der Spieler, ungeachtet seiner Aktionen in einer virtuellen Welt, immer auf dem Cyberwalk bewegt.

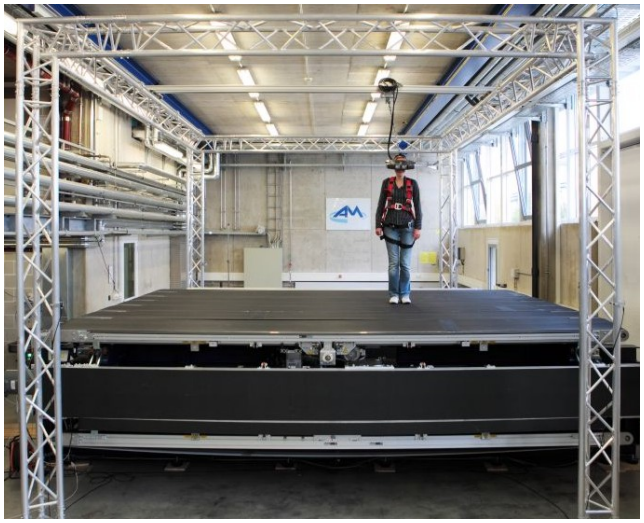

Abbildung 2.5: Der in [6] vorgestellte Cyberwalk.

Zusätzlich zur Erkennung einer Person und ihrer Pose ist oft nach der ausgeführten Aktion gefragt. Das Erkennen verschiedener Aktionen gliedert sich in vier Teile:
Die erste Aufgabe ist das Aufzeichnen von Bewegungsdaten mittels optischer Verfahren, siehe hierzu etwa [35], [15] und [14], oder mittels am Körper befestigter Sensoren. Diese können sowohl Geschwindigkeits- als auch Drehratensensoren beinhalten [16], [27] oder aber auch auf ausgefalleneren Techniken wie etwa Neigungs- [5] oder Winkelsensoren [21] beruhen.
Der zweite Schritt ist das Aufteilen der Aufzeichnungen in kurze Zeitabschnitte. Eine Technik, die sich für die Echtzeitunterteilung von Aufzeichnung eignet, ist die Technik eines gleitenden Zeitfensters fester Länge. Eine solche Technik wurde beispielsweise von Huynh u. Schiele [19] verwendet, um Aktionen zu erkennen und die Länge eines solchen Zeitfensters festzulegen.
Die dritte Aufgabe ist das Extrahieren von Features aus den gegebenen Zeitabschnitten. Hierbei sind verschiedenste Techniken möglich. Preece u. a. [31] vergleichen vierzehn verschiedene Features sowie deren Güte bei der Klassifikation von alltäglichen Aktionen anhand von Beschleunigungsmessungen.
Nach der Extraktion von Features muss aus diesen mittels verschiedener Klassifikatoren auf eine dazugehörige Aktion geschlossen werden. Hierbei wird eine Vielzahl von verschiedenen Möglichkeiten verwendet, diese reichen von Supportvektormaschinen [8] über neuronale Netze [24] bis hin zu Markov-Modellen [22].

Eines der bekanntesten, kommerziellen Bewegungserkennungssysteme ist die Spielkonsole Wii von Nintendo. Hier spielt der Nutzer mit einen Controller, welcher unter

anderem einen Beschleunigungssensor beinhaltet. So können Bewegung und Drehung einer Aktion erkannt werden. Durch dieses System ist es dem Spieler möglich, seine Armbewegungen im Spiel direkt zu steuern. Er erhält so ein realitätsnäheres Spielgefühl.
Ein marktreifes System einer Ganzkörperbewegungserfassung für Spiele hat Microsoft zusammen mit PrimeSense für seine Spielkonsole mit dem System Kinect entwickelt. Es erscheint Ende dieses Jahres. Hier wird der Spieler mittels einer 3D-Kamera erfasst und in verschiedenste Spiele integriert, in welchen er mit vollem Körpereinsatz spielen kann (siehe Abbildung 1.1).

Im Gegensatz zu den hier vorgestellten themenverwandten Arbeiten, welche Aktionen lediglich in der Realität erkennen und klassifizieren, werden im Rahmen dieser Bachelor-Arbeit Aktionen innerhalb eines Virtual Reality Spieles erlernt, erkannt und klassifiziert. Hierzu wird ein kommerzieller Ganzkörperbewegungsaufzeichnungsanzug vorausgesetzt und verwendet.

3 Mathematische Grundlagen

In diesem Kapitel werden zwei Methoden vorgestellt, um verrauschten Messungen zu rekonstruieren. Die erste dieser Methoden ist der Gauß-Glätter, die zweite der Kalman-Filter. Danach wird die multivariante Normalverteilung eingeführt, welche im weiteren Verlauf der Arbeit dazu verwendet wird, die Klassenverteilung der zu lernenden Aktionen zu repräsentieren. Anschließend werden zwei Klassifikatoren sowie eine Möglichkeit zur Fehlerabschätzung eines Klassifikators erklärt. Auch wird eine Möglichkeit vorgestellt Pfade durch einen Graphen zu suchen: Die Breitensuche. Weiter werden zwei Projektionen von OpenGL eingeführt. Zum Abschluss dieses Kapitels erfolgt die Definition eines Bewegungsaufzeichnungsanzug.

3.1 Gauß-Glätter

Datenquellen, welche Messungen einer physikalischen Größe liefern, sind in der Regel durch Rauschen überlagert, also nicht frei von Störungen. Unter der Annahme, dass es sich bei diesen um normalverteiltes Rauschen handelt, können durch Glätten der Funktionswerte, mittels eines Gauß-Glätters oder durch der Filterung mit Hilfe eines Kalman-Filters (siehe Kapitel 3.2) die zugrundeliegenden, ungestörten Werte rekonstruiert werden.
Der Vorteil eines Gauß-Glätters ist, dass dieser sich mittels eines intuitiven Parameters einstellen lässt. In welchem Maße die entstehende Funktion geglättet wird, kann durch einen einzelnen Parameter festgelegt werden. Glättet man zu wenig, enthält die resultierende Funktion noch zu viel Rauschen und ist sprunghaft, glättet man hingegen zu viel, so kann es dazu kommen, dass kleine, aber gewünschte Schwankungen so stark geglättet werden, dass diese verschwinden. Ein Nachteil eines Gauß-Glätters ist es, dass alle Daten oder zumindest ein gewisse Menge dieser benötigt werden, um einen Funktionswert vorherzusagen. Zur Glättung wird beim Gauß-Glätter die zu glättende Funktion mit einem Gauß-Kern gefaltet.
Für zwei Funktionen f und g: $\mathbb{R}^n \to \mathbb{R}$ ist die Faltung $(f * g)$ definiert als:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(y)\, g(t-y)\, \mathrm{d}y. \tag{3.1}$$

Die Faltung zweier Funktionen f und g liefert wieder eine Funktion h. Die so erhaltene Funktion h kann als ein anhand von g gewichtetes Mittel der Funktion f gesehen werden.
Ein möglicher Kern, der zur Glättung herangezogen werden kann, ist eine um Null zentrierte gauß'sche Normalverteilung. Diese hat somit einen Erwartungswert $\mu = 0$ und einen Parameter σ, mithilfe welches der Grad der Glättung reguliert werden kann. Um nun mit Hilfe eines Gauß-Glätters einen Datensatz zu glätten, wird für einen Punkt auf der entstehenden Funktion nicht nur der entsprechende Wert des Datensatzes verwendet, sondern eine Gewichtung über einige Nachbarn dieses Wertes vorgenommen. Wie stark ein Datenpunkt in den Funktionswert einfließt bestimmt der Kern. Wenn nun die Funktion $f(x)$ an der Stelle t approximiert werden soll so ist ihr Wert $\sum_{i=t-\Delta t}^{t+\Delta t} x_i k_i$. Hierbei ist Δt ein von σ abhängender Parameter, der bestimmt in welchem Bereich die Daten einfließen und k_i der Funktionswert des Kernes an der Stelle i.
Ableiten eines Gauß-Kernes und Verwendung des abgeleiteten Kernes als Kern eines Gauß-Glätters führt dazu, dass die Ableitung der Funktion direkt aus den Daten berechnet werden kann. Dies geht aus folgender Gleichung hervor:

$$\frac{\partial}{\partial t}(f * g) = \frac{\partial}{\partial t} f * g = f * \frac{\partial}{\partial t} g. \tag{3.2}$$

Hierbei ist $*$ die Faltung, $\frac{\partial}{\partial t}$ die partielle Ableitung nach der Zeit, g der Gauß-Kern und f ein Vektor von Daten. $(f * g)$ ist die Faltung des Kerns mit einem Datenvektor, also dessen Glättung. Um nun die Ableitung der geglätteten Daten zu erhalten, muss die Faltung der zwei Funktionen abgeleitet werden. Nach obiger Gleichung (3.2) genügt es, eine der beiden Funktionen abzuleiten. Somit ist es ausreichend den Kern g abzuleiten und ihn mit dem Vektor der Daten zu falten. Dadurch erhält man die Ableitung der geglätteten Daten.

3.2 Kalman-Filter

Ist die Rekonstruktion eines verrauschten Prozesses gewünscht, so muss dieser entweder geglättet (siehe Kapitel 3.1) oder gefiltert werden. Der Kalman-Filter [3] ist eine mathematische Methode um aus verrauschten Messungen die zugrundeliegenden Daten zu rekonstruieren. Das Verfahren ist ein Prädiktor-Korrektor-Verfahren, das heißt aus den Daten wird zunächst ein ungefährer Wert vorhergesagt und dieser nachfolgend korrigiert. Das Verfahren berechnet neue Werte anhand einer Markov-

Kette. Das heißt, nach der Markov-Annahme, welche besagt, dass bei Markov-Ketten ein Wert immer nur von seinem direkten Vorgänger abhängt, dass sich ein Funktionswert nur anhand von Werten des vorhergehenden Zeitschrittes berechnet. Somit ist es nicht nötig mehr als eine vorhergehende Messung sowie den aktuellen Zustand zu kennen um einen Funktionswert zu berechnen.
Der Kalman-Filter empfängt Daten einer Dimension und errechnet daraus Werte, die nicht zwingend die gleiche Dimension aufweisen. Somit führt der Kalman-Filter eine Transformation von einem Beobachtungsraum, in welchem er Messungen empfängt, in einen Merkmalsraum der Ausgabe durch.
Der Kalman-Filter wird angewandt um Funktionswerte $\mathbf{x} \in \mathbb{R}^n$ aus verrauschten Messungen $\mathbf{z} \in \mathbb{R}^m$ zu rekonstruieren. Dabei wird davon ausgegangen, dass die Werte $\mathbf{x}$ einem linearen System folgen:

$$\mathbf{x}_k = A\mathbf{x}_{k-1} + B\mathbf{u}_{k-1} + \mathbf{w}. \tag{3.3}$$

Das heißt ein Zustand $\mathbf{x}_k$ setzt sich aus mehreren Summanden zusammen. Einerseits aus der Veränderung des Zustandes A multipliziert mit dem Wert des vorhergehenden Zeitschrittes $\mathbf{x}_{k-1}$. Hinzu kommt eine Störung, welche sich aus einer Störung des vorhergehenden Zeitschrittes $\mathbf{u}_{k-1}$ sowie deren Änderung B zusammensetzt. $\mathbf{w}$ schlussendlich beschreibt ein zusätzliches normalverteiltes, mittelwertfreies, zeitunabhängiges Rauschen:

$$\mathbf{w} \sim \mathcal{N}(0, Q). \tag{3.4}$$

Hierbei ist Q eine zeitunabhängige Kovarianzmatrix des Rauschens.
Die Messungen $\mathbf{z} \in \mathbb{R}^m$ werden angenommen als:

$$\mathbf{z}_k = H_k\mathbf{x}_k + \mathbf{v}. \tag{3.5}$$

Das bedeutet die Messungen setzen sich zusammen aus einer Transformation H_k des zugrundeliegenden Zustandes $\mathbf{x}_k$ zum Zeitpunkt k, vom Werteraum in den Raum der Beobachtungen. Außerdem sind die Beobachtungen überlagert mit normalverteiltem, mittelwertfreiem, zeitunabhängigem Rauschen $\mathbf{v}$. Somit gilt auch für $\mathbf{v}$ und eine zeitunabhängige Kovarianzmatrix R:

$$\mathbf{v} \sim \mathcal{N}(0, R). \tag{3.6}$$

Um nun rückwärts von einer Messung $\mathbf{z}_k$ auf den dazugehörigen Zustand $\mathbf{x}_k$ zu schließen, wird wie folgt vorgegangen: Zuerst wird aus dem Zustand $\mathbf{x}_{k-1}$ des vor-

herigen Zeitschritt und der Störung $\mathbf{u}_{k-1}$ des vorangegangen Zeitschritt ein Wert für $\mathbf{x}_k$ vorhergesagt:

$$\hat{\mathbf{x}}_k = A\hat{\mathbf{x}}_{k-1} + B\mathbf{u}_{k-1}. \tag{3.7}$$

Danach wird eine Matrix P berechnet, welche ein Maß für den Schätzfehler ist:

$$\hat{P}_k = AP_{k-1}A^{\mathsf{T}} + Q. \tag{3.8}$$

Hierbei ist Q die Kovarianzmatrix des Rauschens der Daten (siehe (3.4)). Nach diesem ersten Prädiktor-Schritt wird diese Vorhersage korrigiert. Im ersten Schritt des Korrigierens wird die Kalman-Matrix K_k für den Zeitpunkt k berechnet. In sie fließen die Schätzung des Vorhersagefehlers P_k, die Projektionsmatrix H, sowie Matrix R (siehe (3.6)), welche das die Messdaten überlagernde Rauschen beschreibt, ein:

$$K_k = \hat{P}_k \cdot H^{\mathsf{T}}(H\hat{P}_k H^{\mathsf{T}} + R)^{-1}. \tag{3.9}$$

Danach wird der zu messende Wert $\mathbf{x}_k$ berechnet. Dies geschieht anhand seines geschätzten, mittels H in den Messraum projizierten Zustandes $\hat{\mathbf{x}}_k$ und der aktuellen, mit der Kalman-Matrix K_k gewichteten Messung $\mathbf{z}_k$:

$$\mathbf{x}_k = \hat{\mathbf{x}}_k + K_k(\mathbf{z}_k - H\hat{\mathbf{x}}_k). \tag{3.10}$$

Schlussendlich muss noch für den nächsten Zeitschritt die Fehlermatrix berechnet werden:

$$P_k = (I - K_k H)\hat{P}_k. \tag{3.11}$$

3.3 Multivariante Normalverteilung

Die Standardnormalverteilung ist eine Methode eine Zufallsvariable sowie ihre Verteilung zu repräsentieren. Die multivariante Normalverteilung ([17], S. 64ff) ist eine Erweiterung der einfachen Normalverteilung auf mehrere Dimensionen. Mit der multivarianten Normalverteilung lassen sich Verteilungen im mehrdimensionalen Raum repräsentieren. Ihre Dichtefunktion ist definiert als:

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma|^{\frac{1}{2}}} \exp^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})}. \tag{3.12}$$

Hierbei ist $\mathbf{x} \in \mathbb{R}^d$, $d \in \mathbb{N}$ die Dimension der Daten, $\boldsymbol{\mu} \in \mathbb{R}^d$ der Erwartungswert und $\Sigma \in \mathbb{R}^{d\times d}$ die Kovarianzmatrix. Um nun die Dichte (siehe Abbildung 3.1) der Nor-

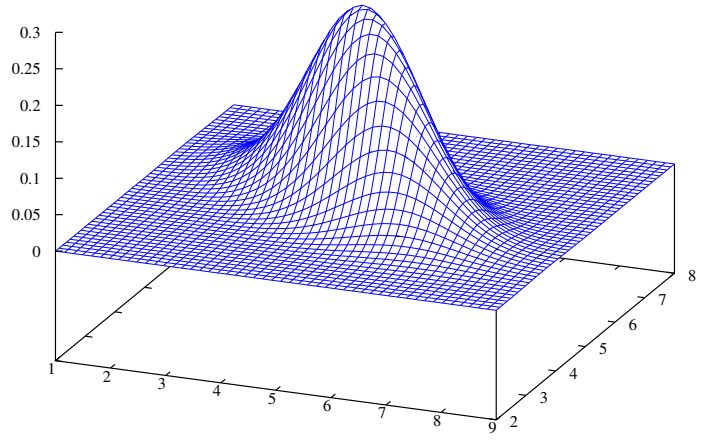

Abbildung 3.1: Visualisierung der Dichte der multidimensionalen Normalverteilung.

malverteilung $f(\mathbf{x})$ für ein gegebenes $\mathbf{x} \in \mathbb{R}^d$ errechnen zu können, müssen vorher $\boldsymbol{\mu}$ und Σ aus Trainingsdaten geschätzt werden:

$$\boldsymbol{\mu} = \frac{1}{n}\sum_{i=0}^{n}\mathbf{x}_i, \tag{3.13}$$

$$\Sigma = \frac{1}{n-1}\sum_{i=0}^{n}(\mathbf{x}_i - \boldsymbol{\mu})(\mathbf{x}_i - \boldsymbol{\mu})^{\mathsf{T}} \text{ mit } n > 1. \tag{3.14}$$

Für diese Schätzer ist es notwendig, dass alle Daten, aus denen gelernt werden soll, gleichzeitig vorhanden sind. Dies ist aber nicht zwingend der Fall (siehe Kapitel 5.7). Aus den obigen Formeln lassen sich jedoch leicht folgende rekursive Regeln ableiten:

$$\boldsymbol{\mu}_1 = \mathbf{x}_1, \tag{3.15}$$

$$\boldsymbol{\mu}_n = \frac{n-1}{n}\boldsymbol{\mu}_{n-1} + \frac{1}{n}\mathbf{x}_i, \tag{3.16}$$

$$\Sigma_1 = 0,\ \Sigma_2 = \boldsymbol{\mu}_2, \tag{3.17}$$

$$\Sigma_n = \frac{n-2}{n-1}\Sigma_{n-1} + \frac{1}{n-2}(\mathbf{x}_i - \boldsymbol{\mu}_n)(\mathbf{x}_i - \boldsymbol{\mu}_n)^T. \tag{3.18}$$

3.3.1 Quantile der multidimensionalen Normalverteilung

Ein p-Quantil der multidimensionalen Normalverteilung ist ein Funktion die den p-ten Teil der Masse der Verteilung abgrenzt. Liegt ein Wert innerhalb eines p-Quantils einer Verteilung, so trifft die Hypothese, welche durch die Verteilung repräsentiert wird, mit einer Wahrscheinlichkeit von p zu.

3.3.1.1 χ^2-Verteilung

Die χ^2-Verteilung mit n Freiheitsgraden ist definiert als Verteilung der Summe von n unabhängigen, quadrierten, normalverteilten Zufallsvariablen. Sie kann auch als eine Verteilung von quadrierten Abständen von n Zufallsvariablen zu deren Klassenmitte gesehen werden. Die χ^2-Verteilung mit n Freiheitsgraden hat folgende Dichte:

$$g_n(x) = \frac{x^{\frac{n}{2}-1}e^{-\frac{x}{2}}}{2^{\frac{n}{2}}\Gamma(\frac{n}{2})} \text{ für } x > 0. \tag{3.19}$$

Hierbei sind $x, n \in \mathbb{R}$ und $\Gamma(x)$ die Gamma-Funktion:

$$\Gamma(x) = \int_0^\infty t^{x-1}\mathrm{e}^{-t}\mathrm{d}t. \tag{3.20}$$

3.3.1.2 Mahalanobis-Distanz

Die Mahalanobis-Distanz ist ein Distanzmaß, welches einen, mittels einer Kovarianzmatrix gewichteten, Abstand im Raum berechnet. Für zwei Punkte $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ sowie eine Kovarianzmatrix $\Sigma \in \mathbb{R}^{n\times n}$ berechnet sich der Abstand folgendermaßen:

$$d_{Mahalanobis}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x}-\mathbf{y})^\mathsf{T}\Sigma^{-1}(\mathbf{x}-\mathbf{y})}. \tag{3.21}$$

Um nun für einen Punkt $\mathbf{x} \in \mathbb{R}^n$ zu prüfen ob er im p-Quantil einer Normalverteilung mit Kovarianzmatrix $\Sigma \in \mathbb{R}^{n\times n}$ und Klassenmitte $\boldsymbol{\mu} \in \mathbb{R}^n$ liegt, geht man wie folgt vor:
Zuerst wird der Mahalanobis-Abstand d_x zum Klassenmittelpunkt errechnet: $d_x = d_{Mahalanobis}(\mathbf{x}, \boldsymbol{\mu})$, hierbei fließt Σ ein. Danach wird geprüft ob d_x^2 kleiner ist als das p-Quantil der χ^2-Verteilung, also ob gilt: $d_x^2 < g_n^{-1}(p)$. Ist d_x^2 kleiner als das p-Quantil, so trifft die Aussage der Normalverteilung mit einer Sicherheit von p zu.

3.4 Klassifikatoren und Fehler

Klassifikatoren werden benötigt, um Objekte ihren zugehörigen Klassen zuzuordnen. Sie entscheiden anhand der die Klassen beschreibenden Verteilungen die Zugehörigkeit von Objekten zu einzelnen Klassen. In diesem Kapitel werden zwei fundamentale Klassifikatoren vorgestellt. Die Grundlage für die hier vorgestellten Klassifikatoren ist das Bayestheorem. Dieses liefert für zwei Ereignisse A und B die Wahrscheinlichkeit des Eintretens des Ereignisses A unter der Voraussetzung, dass bereits B eingetreten ist:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)} \text{ für } P(B) > 0. \tag{3.22}$$

Hierbei ist $P(X)$ die a-priori-Wahrscheinlichkeit des Ereignisses X und $P(X|Y)$ die bedingte Wahrscheinlichkeit, dass X eintritt unter der Voraussetzung, dass das Ereignis Y bereits eingetreten ist. Ein Klassifikator liefert für zwei Ereignisse oder Klassenzugehörigkeiten A und B sowie Vorwissen, beziehungsweise ein Modell, C ein Ergebnis der Form:

$$P(A|C) = \frac{P(C|A) \cdot P(A)}{P(C)} \leq \frac{P(C|B) \cdot P(B)}{P(C)} = P(B|C). \tag{3.23}$$

3.4.1 Maximum-Likelihood Klassifikator

Ein simpler Klassifikator ist der Maximum-Likelihood Klassifikator. Er stützt sich bei der Bestimmung der Klassenzugehörigkeit lediglich auf die Verteilung der einzelnen Klassen. Hierbei liefert er immer die Klasse zurück, deren Wert der Verteilungsfunktion an der spezifischen Stelle der größte ist. Der Klassifikator nimmt also in (3.23) gleiche a-priori-Wahrscheinlichkeiten der Klassen an. Sind Objekte einer bestimmten Klasse seltener als die einer andern, so wird dies vom Maximum-Likelihood Klassifikator nicht beachtet.
Die Bestimmung einer zum Objekt x gehörenden Klasse c geschieht mit folgender Formel:

$$MLE(x) = \underset{c_i}{\operatorname{argmax}} P(x|c_i). \tag{3.24}$$

Ein Beispiel im eindimensionalen Fall mit zwei Klassen ist in Abbildung 3.2[links] gegeben. Hier liefert der MLE-Klassifikator bis zur Stelle b Klasse rot und ab b Klasse blau.

3.4.2 Maximum-a-posteriori Klassifikator

Der Maximum-a-posteriori Klassifikator nutzt im Gegensatz zum Maximum-Likelihood Klassifikator zusätzlich Wissen über die Auftretenswahrscheinlichkeit, beziehungsweise a-priori-Wahrscheinlichkeiten der zu klassifizierenden Klassen. So ist es unwahrscheinlicher, dass zugunsten einer Klasse die selten auftritt entschieden wird. Die Wahl einer zum Objekt x gehörenden Klasse c geschieht folgendermaßen:

$$MAP(x) = \underset{c_i}{\operatorname{argmax}}\, P(x|c_i)P(c_i). \tag{3.25}$$

In Abbildung 3.2[links] ist ein Beispiel im eindimensionalen Fall mit zwei Klassen für die Klassenzugehörigkeitsentscheidung gegeben. Hier würde der MAP-Klassifikator bis zum Punkt a für Klasse rot entscheiden und somit im Bereich von b bis a die geringere Auftretenswahrscheinlichkeit der Klasse blau beachten.

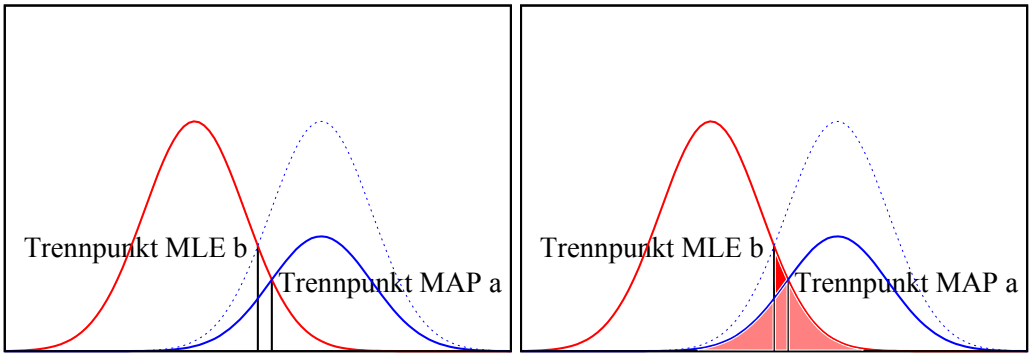

Abbildung 3.2: Links: Ein Klassifikationsbeispiel anhand von zwei Klassen rot und blau im eindimensionalen Fall mit unterschiedlichen Auftretenswahrscheinlichkeiten der Klassen. Rechts ist zusätzlich der Klassifikationsfehler verzeichnet, die hellrote Fläche ist der Klassifikationsfehler des MAP-Klassifikators, zum Fehler des MLE-Klassifikators kommt die dunkelrote Fläche hinzu.

3.4.3 Fehlerabschätzung

Um die Güte einer Klassifikation zu messen, kann der Fehler der Klassifikation berechnet werden. Der Fehler einer Klassifikation ist die Anzahl der Entscheidungen für Klasse c_2, obwohl das Objekt aus Klasse c_1 stammt. Dies tritt ein, wenn die Wahrscheinlichkeit für die Klasse c_2 höher ist als für die eigentliche Klasse c_1. Somit kann der Fehler grafisch als Fläche unter der Verteilungsfunktion der Klasse mit geringerer Wahrscheinlichkeit gesehen werden (siehe Abbildung 3.2[$rechts$]). Mathematisch

ist dies im eindimensionalen Fall für zwei Klassen c_1 und c_2, wobei der Erwartungswert von Klasse c_1 kleiner ist als der von c_2, das Integral:

$$\int_a^{\infty} P(c_1) + \int_{-\infty}^{a} P(c_2), \tag{3.26}$$

wobei a die Stelle ist, an welcher der Klassifikator die Klassenzugehörigkeitsentscheidung ändert.
Da der Maximum-Likelihood Klassifikator lediglich die Verteilung der Klassen c_1 und c_2 beachtet, nicht aber deren Auftretenswahrscheinlichkeit, steigt sein Fehler sobald sich die Auftretenswahrscheinlichkeiten unterscheiden. Ist $P(c_1) = P(c_2)$ so sind der Maximum-Likelihood und der Maximum-a-posteriori Klassifikator äquivalent. Je größer der Unterschied der Auftretenswahrscheinlichkeiten der Klassen ist, desto größer wird der Fehler des Maximum-Likelihood Klassifikators.

3.5 Breitensuche

Die Breitensuche ist eine Methode um den kürzesten Pfad zwischen zwei Elementen in einem Graphen zu finden. Dabei wird folgendermaßen vorgegangen:
Zunächst wird der Startknoten der Suche in eine Warteschlange gespeichert. Nachfolgend wird, solange die Warteschlange nicht leer ist, das erste Element aus dieser entnommen und für jeder Nachfolger dieses untersucht, ob es sich um den gesuchten Knoten handelt. Ist der Nachfolgerknoten der gesuchte, so wird der Pfad von diesem Element zur Wurzel zurückgegeben, anderenfalls wird der Nachfolgerknoten an das Ende der Warteschlange gespeichert. Sind alle Nachfolgerknoten untersucht, wird wieder das vorderste Element der Warteschlange entnommen und untersucht. Ist die Warteschlange leer, ist das gesuchte Element nicht im Graphen enthalten oder nicht vom Startknoten aus erreichbar. Durch die Warteschlange wird der Graph ebenenweise untersucht. Aufgrund dieses Vorgehens findet die Breitensuche den kürzesten Pfad vom Startknoten zum gesuchten und endet auch bei Zyklen nicht in Endlosschleifen. Hierbei hat das Verfahren eine Laufzeit von maximal $O(n + m)$, wobei n die Anzahl der Knoten und m die Anzahl der Kanten im Graphen ist.

3.6 Kameramodell

In OpenGL werden verschiedenste Transformationen dazu verwendet eine Szene auf die Bildschirmfläche zu projizieren. Es gibt jedoch zwei Transformationen, welche in dieser Arbeit von besonderem Interesse sind. Diese zwei Transformationen werden hier erläutert.

3.6.1 Betrachtertransformation

Die erste der hier vorgestellten Transformationen ist die Betrachtertransformation K. Diese projiziert die Welt, welche sich bereits in einem einheitlichen Koordinatensystem befindet, in ein Kamerakoordinatensystem und definiert somit die Sichtrichtung sowie den Standort der Kamera. Sie wird definiert durch eine vektorielle Position der Kamera $\mathbf{c}$, einen Betrachtungspunkt $\mathbf{p}$ der Szene sowie einen zur Sichtrichtung $(\mathbf{p} - \mathbf{c})$ senkrechten Vektor $\mathbf{u}$ (siehe Abbildung 3.3):

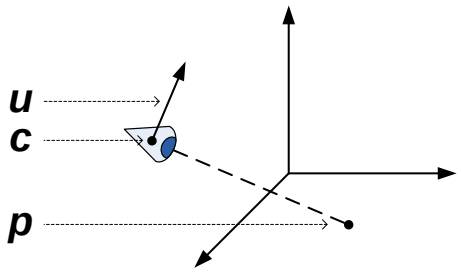

Abbildung 3.3: Die zu definierenden Vektoren der Betrachtertransformation.

Zur Berechnung der Betrachtertransformation K sei: $\mathbf{f} = \mathbf{p} - \mathbf{c}$, $\mathbf{g} = \frac{\mathbf{f}}{||\mathbf{f}||}$ sowie $\mathbf{v} = \frac{\mathbf{u}}{||\mathbf{u}||}$. Daraus ergeben sich $\mathbf{s}$ und $\mathbf{w}$ wie folgt: $\mathbf{s} = \mathbf{g} \times \mathbf{v}$, $\mathbf{w} = \mathbf{s} \times \mathbf{g}$.

$$K = \begin{pmatrix} \mathbf{s}_x & \mathbf{s}_y & \mathbf{s}_z & 0 \\ \mathbf{w}_x & \mathbf{w}_y & \mathbf{w}_z & 0 \\ -\mathbf{g}_x & -\mathbf{g}_y & -\mathbf{g}_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.27}$$

Der selbe Effekt kann auch durch eine Rotation sowie eine Translation erzeugt werden.

3.6.2 Perspektivische Projektionsmatrix

Die zweite der hier vorgestellten Projektionen ist die perspektivische Projektionsmatrix. Diese Matrix T projiziert die Welt, oder einen Teil dieser: das Betrachtungssichtfeld, auf einen Einheits-Würfel. Bei ihrer Berechnung fließen folgende Werte ein: Der Öffnungswinkel des Kameramodells *f*, welcher den sichtbaren Bildausschnitt und dessen Abbildungsgröße beschreibt, die nahe Clippingebene *near* sowie die ferne Clippingebene *far*. Letztere definieren ab welchem *near*- und bis zu welchem *far*-Abstand zur Kamera Objekte der Welt sichtbar sind und somit den Beginn und das Ende des Betrachtungssichtfeldes. Außerdem fließt das Auflösungsverhältnis des Bildschirmes *aspect* ein. Hiermit wird bestimmt welches Format die Sicht der Welt erhält. Die Parameter, welche das Betrachtungssichtfeld beschreiben, sind in Abbildung 3.4 veranschaulicht. Nach der Anwendung der Projektion auf eine Szene befinden sich alle Objekte der Welt, welche sichtbar sind, welche sich also im Betrachtungssichtfeld befunden haben, in einem Einheits-Würfel. Objekte außerhalb der Sicht werden verworfen.

$$T = \begin{pmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{far+near}{far-near} & \frac{2 \cdot near \cdot far}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix} \tag{3.28}$$

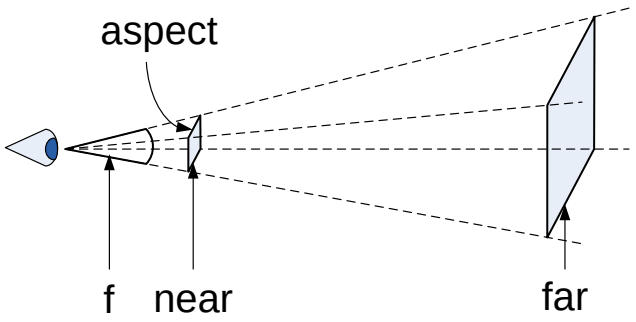

Abbildung 3.4: Visualisierung der Parameter der perspektivischen Projektionsmatrix.

3.7 Bewegungsaufzeichnungsanzug

Ein Bewegungsaufzeichnungsanzug ist ein System, welches die Posen einer Person erfasst und aufzeichnet. Er besteht aus am Körper befestigten Sensoren. An diesen werden verschiedenste Daten gemessen, wie zum Beispiel Geschwindigkeit und Orientierung. Mit Hilfe dieser Daten ist es möglich Position und Orientierung verschiedener Körpersegmente zu berechnen.

4 Ansatz

In diesem Kapitel wird das Vorgehen der Arbeit erläutert. Der grobe Ansatz dieser Arbeit ist in Abbildung 4.1 anhand des Datenflusses illustriert.

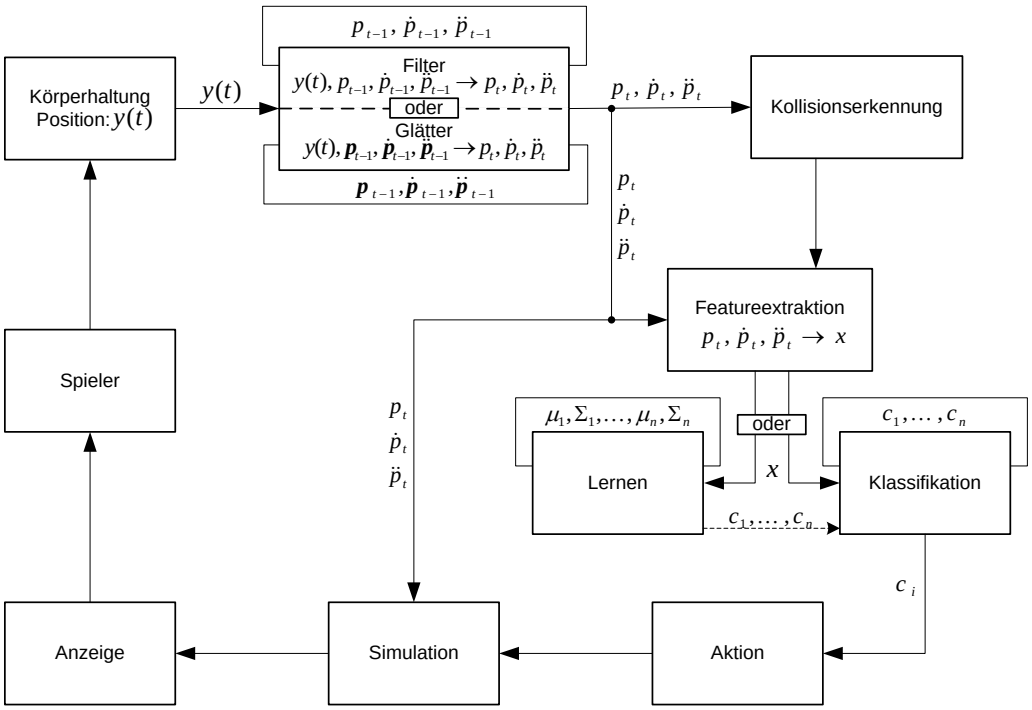

Abbildung 4.1: Der Ansatz dieser Arbeit im Überblick. Die Bewegungsdaten werden geglättet oder gefiltert und zur Kollisionserkennung, zur Featureextraktion sowie zur Simulation verwendet. Aus den extrahierten Features werden entweder Aktionsklassen erlernt oder Aktionen klassifiziert. Das Ergebnis der Simulation wird visualisiert, so dass der Nutzer reagieren kann.

Nach der Einführung einiger Notationen wird in diesem Kapitel gezeigt, wie ein Gauß-Glätter oder ein Kalman-Filter zur Berechnung geglätteter Positions- sowie Geschwindigkeits- und Beschleunigungsdaten verwendet werden kann. Nachfolgend

wird erläutert wie Kollisionen erkannt und wie die Featurevektoren extrahiert werden. Danach wird ein Aktionsmodell definiert und erklärt wie es bei der Klassifikation und beim Lernen von Aktionen eingeht. Zu den Klassifikatoren, deren Anwendung erläutert wird, wird auch die Fehlerabschätzung besprochen.

4.1 Notation

Zum besseren Verständnis dieser Arbeit werden hier einige Notationen festgelegt. Der MVN-Anzug (siehe Kapitel 3.7 und Kapitel 5.2) liefert zu 23 Körpersegmenten Positions- und Orientierungsdaten. Diese werden wie folgt benannt:

Symbol	Bedeutung
$\mathbf{p}_i(t) \in \mathbb{R}^3$	Positionsdaten zum Zeitpunkt t eines Körpersegmentes i $i \in \{1, \ldots, 23\}$
$R_i(t) \in \mathbb{R}^{3\times3}$	Orientierungsdaten zum Zeitpunkt t eines Körpersegmentes i $i \in \{1, \ldots, 23\}$
$\mathbf{x} \in \mathbb{R}^d$	Ein d-dimensionaler Featurevektor (siehe Kapitel 4.4)

4.2 Berechnen von Geschwindigkeit und Beschleunigung

Da das MVN-Protokoll (siehe Kapitel 5.2) lediglich Positions- und Orientierungsdaten sendet, für diese Arbeit aber auch Geschwindigkeits- und Beschleunigungswerte notwendig sind, müssen diese gesondert berechnet werden. In diesem Abschnitt wird erläutert wie ein Kalman-Filter (siehe Kapitel 3.2) oder ein Gauß-Glätter (siehe Kapitel 3.1) verwendet werden können, um die Positionsdaten zu glätten, beziehungsweise zu filtern und Geschwindigkeits- und Beschleunigungswerte aus den verrauschten Positionsdaten des Anzugs zu errechnen.

4.2.1 Glättung mittels eines Kalman-Filters

In dieser Arbeit wird pro Dimension der Eingangsdaten ein Kalman-Filter eingesetzt. Als Eingabe für die Berechnung der Geschwindigkeits- und Beschleunigungswerte

eignen sich von den gegebenen Positions- und Orientierungsdaten lediglich Positionsdaten. Somit ergibt sich als Beobachtungsraum ein Raum lediglich einer Dimension. Die Positionsmessungen des Beobachtungsraums werden durch den Kalman-Filter in einen Merkmalsraum projiziert. Dieser ist in diesem Fall dreidimensional, er enthält Positions-, Geschwindigkeits- und Beschleunigungswerte. Somit ergibt sich die Projektionsmatrix H vom Werteraum in den Raum der Beobachtungen als:

$$H = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$$

In jedem Zeitschritt Δt ändert sich die Position anhand der Geschwindigkeit des vorhergehenden Zeitpunktes und die Geschwindigkeit proportional zur Beschleunigung des letzten Zeitschrittes. Außerdem wird angenommen, dass die Beschleunigung etwa der Beschleunigung des vorhergehenden Wertes entspricht. Somit ergibt sich die Matrix der Änderung A:

$$A = \begin{pmatrix} 1 & \Delta t & 0 \\ 0 & 1 & \Delta t \\ 0 & 0 & 1 \end{pmatrix}$$

Als Ergebnis des Kalman-Filters erhält man zusätzlich zu den gefilterten Positionswerten Geschwindigkeits- und Beschleunigungswerte.

4.3 Kollisionserkennung

Zur Erkennung von Aktionen bedarf es einer Kollisionserkennung. Diese übernimmt in dieser Arbeit die Physik-Engine *Bullet* (siehe Kapitel 5.4). Sobald ein Körperteil ein Objekt berührt, liefert *Bullet* zurück welches Objekt und welches Körperteil involviert waren. Außerdem wird in dieser Arbeit eine Aktion immer nur bei ihrer ersten Berührung mit einem Objekt als Kollision gemeldet. Während der weiteren Berührung bis zum Verlassen des Objektes wird für dieses spezielle Körperteil und das spezifische Objekt nichts weiter unternommen.

4.4 Features

Um Aktionen unterscheiden zu können, bedarf es verschiedener *Features*, welche die Aktion charakterisieren. Diese müssen zum Zeitpunkt der Kollision des Körperteiles mit dem Objekt aus den Positions-, Geschwindigkeits- und Beschleunigungswerten des Körperteiles extrahiert werden. Dieser Vektor an Features $\mathbf{x}$ wird anschließend dazu verwendet, die Aktion zu klassifizieren oder um mit seiner Hilfe eine Klasse von Aktionen zu erlernen. Hierbei können verschiedenste Werte verwendet werden. Welche spezifischen Features und wie viele extrahiert werden, hängt davon ab welche und wie viele Aktionen unterschieden werden sollen. Zum Zeitpunkt $\hat{t}$ der Kollision eines dafür bestimmten Körperteiles j mit der Oberfläche eines dafür bestimmten Objektes k (siehe Kapitel 5.7), welche von der Physikengine (siehe Kapitel 5.4) geliefert wird, wird ein Vektor von Features $\mathbf{x}$ extrahiert. Der Featurevektor setzt sich zusammen aus Werten der Geschwindigkeit und der Beschleunigung aus einem kleinen Zeitfenster von τ Sekunden vor und nach der Kollision.

In dieser Arbeit wird als erster Ansatz jeweils die Geschwindigkeit und Beschleunigung im exakten Zeitpunkt der Kollision als Features verwendet:

$$\mathbf{x}_j = (\dot{\mathbf{p}}_j(\hat{t}), \ddot{\mathbf{p}}_j(\hat{t})), \tag{4.1}$$

also $\dot{\mathbf{p}}_j(\hat{t})$ die Geschwindigkeit sowie $\ddot{\mathbf{p}}_j(\hat{t})$ die Beschleunigung des entsprechenden Körperteiles j zum Zeitpunkt der Kollision. Somit können insbesondere verschiedene Bremsbewegungen von andersartigen Bewegungen unterschieden werden.

Als zweiter Ansatz wird ein dreidimensionaler Featurevektor verwendet. Hierzu werden im Zeitfenster zuerst die gewichteten Maximalwerte für Geschwindigkeit und Beschleunigung ermittelt. Diese Werte sind stabiler bezüglich Abweichungen vom exakten Kollisionszeitpunkt. Um ein lokales Maximum in der Mitte des Zeitfensters globalen Maxima am Rande vorzuziehen, wurden für die Ermittlung die Werte mit $r(t) = cos(a \cdot (\hat{t} - t))$ multipliziert:

$$\hat{t}_V = \underset{t}{\operatorname{argmax}}\, r(t)\dot{\mathbf{p}}_j(\hat{t}+t) \quad \text{für } t \in [-10, 10]\,, \tag{4.2}$$

$$\hat{t}_A = \underset{t}{\operatorname{argmax}}\, r(t)\ddot{\mathbf{p}}_j(\hat{t}+t) \quad \text{für } t \in [-10, 10]\,, \tag{4.3}$$

$$\mathbf{x}_j = (\ddot{\mathbf{p}}_j(\hat{t}+\hat{t}_A), \ddot{\mathbf{p}}_j(\hat{t}+\hat{t}_A)_{V'}, \hat{t}_V). \tag{4.4}$$

Hierbei ist $\hat{t}_V$ der Zeitpunkt der gewichteten maximalen Geschwindigkeit, $\hat{t}_A$ der Zeitpunkt der gewichteten maximalen Beschleunigung sowie V' der normierte gewichtete maximale Geschwindigkeitsvektor:

$$V' = \frac{\dot{\mathbf{p}}_j(\hat{t} + \hat{t}_V)}{\|\dot{\mathbf{p}}_j(\hat{t} + \hat{t}_V)\|}. \tag{4.5}$$

Des Weiteren ist $\ddot{\mathbf{p}}_j(\hat{t} + \hat{t}_A)_{V'}$ die Projektion des gewichteten maximalen Beschleunigungsvektors auf den normierten gewichteten maximalen Geschwindigkeitsvektor. Durch die Projektion der Beschleunigung auf die Geschwindigkeit kann auf die Richtung der Beschleunigung geschlossen werden.

4.5 Aktionsmodell

Ein Klassifikator braucht ein Modell anhand dessen er seine Zugehörigkeitsaussage treffen kann. In dieser Arbeit ist dies eine Menge von normalverteilten Klassen. Diese Normalverteilungen werden von einem Modell $\mathcal{M}$ beschrieben. Das Modell hat je zwei Parameter pro Klasse. Diese Parameter $\boldsymbol{\mu}_i$ sowie Σ_i pro Klasse c_i werden zusammengefasst im Parametersatz θ:

$$\mathcal{M}, \theta \text{ wobei } \theta = (\boldsymbol{\mu}_1, \Sigma_1, \ldots, \boldsymbol{\mu}_n, \Sigma_n). \tag{4.6}$$

Jede Klasse c_i ist normalverteilt mit den Parametern $\boldsymbol{\mu}_i$ und Σ_i:

$$\mathcal{M} : \mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_i, \Sigma_i)\big|_{i=c_i}. \tag{4.7}$$

Während des Lernprozesses werden jeweils die Parameter für jede Klasse $c_1, \ldots, c_m$ gelernt. Das heißt es werden die zugehörigen $\boldsymbol{\mu}_i$ und Σ_i bestimmt. Da es sich hier um überwachtes Lernen handelt, das heißt zu jedem Featurevektor ist die Klassenzugehörigkeit gegeben, wird der Parametersatz θ bestmöglich optimiert:

$$\hat{\theta} = \underset{\theta}{\operatorname{argmax}}\, P(\theta | \mathbf{x}_1, \ldots, \mathbf{x}_n, c_1, \ldots, c_m). \tag{4.8}$$

Beim Klassifizieren einer Aktion wählt der hier verwendete MLE-Klassifikator immer die Klasse, welche die a-posteriori Wahrscheinlichkeit maximiert:

$$\hat{c} = \underset{c_i}{\operatorname{argmax}}\, P(c_i | \mathcal{M}, \mathbf{x}). \tag{4.9}$$

4.6 Erkennung und Klassifikation von Aktionen

Bei einer Kollision eines Körperteiles mit einem speziellen Objekt muss ermittelt werden um welchen Typ von Aktion es sich handelt. Zum Zeitpunkt der Kollision wird hierzu der Featurevektor $\mathbf{x}$ extrahiert. Mithilfe dieses Vektors soll nun die Erkennung und Klassifikation der Aktionen vorgenommen werden. Hierzu werden pro Aktion Normalverteilungen (siehe Kapitel 3.3) gelernt (siehe Kapitel 5.7). Sind nach einem Lernprozess pro Klasse $\boldsymbol{\mu}$ und Σ bekannt, ist also das Modell $\mathcal{M}$ gelernt, werden für einen gegebenen Featurevektor $\mathbf{x}$ die Dichten der Normalverteilungen der einzelnen Aktionen berechnet. Die soeben erkannte Kollision wird nun mit Hilfe eines Klassifikators und $\mathbf{x}$ der Aktion zugeordnet, deren Wahrscheinlichkeit an der Stelle des Featurevektors am höchsten ist, also der Klasse c_i für welche $c = \text{argmax}_{c_i} P(c_i|\mathcal{M}, \mathbf{x})$ gilt. Hierbei werden gleiche Auftretenswahrscheinlichkeiten angenommen, somit entspricht dieses Vorgehen dem MLE-Klassifikator (siehe Kapitel 3.4.1).
Ist das Modell $\mathcal{M}, \theta$ anfangs nicht bekannt, so kann dieses aus einer Reihe von Demonstrationen gelernt werden. Dieser Prozess wird in Kapitel 5.7 beschrieben. Außerdem ist davon auszugehen, dass verschiedene Spieler unterschiedliche Parametersätze θ benötigen.

4.7 Fehlerarten und Fehlerabschätzung

Wurde im Spiel eine Aktion klassifiziert so ist auch der Fehler, beziehungsweise die Sicherheit der Zugehörigkeit zur Aktionsklasse von Interesse. Hierbei werden zwei Arten von Fehlern unterschieden. Einerseits die allgemeine Unsicherheit und andererseits ein zu geringer Unterschied zwischen zwei Dichten von Verteilungen. Ein Klassifikationsfehler tritt auf, wenn sich die vom Klassifikator gewählte Klasse $\hat{c}$ und die wirkliche Klasse c_i unterscheiden. Dies ist möglich, wenn die Dichte der Normalverteilung an der gegebenen Stelle für $\hat{c}$ größer ist als für c_i: $P(\hat{c}_i|\theta) > P(c_i|\theta)$.

4.7.1 Allgemeine Unsicherheit

Da eine Normalverteilung unendlich ausgedehnt ist, liefert sie auch noch weit entfernt von ihrem Zentrum Werte größer Null. Je weiter entfernt der Featurevektor vom Zentrum der Normalverteilung liegt, desto geringer ist die Wahrscheinlichkeit, dass eine Demonstration der Aktion diesen Featurevektor liefert. Somit sollten nur

Aktionen als solche klassifiziert werden, welche in einem gewissen Prozentsatz des Volumens der Normalverteilung liegen (siehe Kapitel 3.3.1). Außerhalb dieses Volumens ist die Sicherheit der Klassifikation zu gering. Hier sollte der Spieler die Klassenzugehörigkeit entscheiden. Anschließend sollten $\boldsymbol{\mu}$ und Σ der entsprechenden Verteilung anhand von $\mathbf{x}$ angepasst werden.

4.7.2 Zu geringer Unterschied

Ist der Unterschied der Dichte zweier Normalverteilungen zu gering, so kann keine sichere Zuordnung geschehen. Dies ist der Fall sobald die Wahrscheinlichkeit der Zugehörigkeit der Aktion zu zwei Klassen c_i und c_j ähnliche Werte liefert:

$$|\log P(c_i) - \log P(c_j)| \leq 2. \tag{4.10}$$

Ist also die Wahrscheinlichkeit der Zugehörigkeit einer Aktion zu einer Klasse nicht wenigstens doppelt so hoch als zur anderen Klasse, so sollte nachgefragt und nachgelernt werden.

4.8 Lernen der Aktionen

Um die Aktionen einer Person anzupassen wird für jede Aktion eine Normalverteilung geschätzt. Hierzu werden einige Ausführungen jeder Aktion gesammelt. Aus diesen Ausführungen werde die Featurevektoren $\mathbf{x}$ extrahiert. Diese werden nun dazu verwendet $\boldsymbol{\mu}$ und Σ der jeweiligen Verteilungen zu errechnen. Wird während der Ausführung eine Zuordnung zu einer Aktionsklasse als mit zu geringer Sicherheit zurückgewiesen, so kann nach einer Nachfrage zu welcher Klasse die Aktion gehört, deren Verteilung angepasst werden.

5 Implementierung

In diesem Abschnitt wird das erarbeitete Framework vorgestellt. Es gliedert sich in sieben Teile. Hierzu gehört das implementierte Spiel, die Eingabe mittels eines Bewegungsaufzeichnungsanzuges, die Ausgabe mit Hilfe einer Videobrille, die Physik der Objekte, die Agenten im Spiel sowie die Berechnung von Geschwindigkeit und Position und schlussendlich die Anwendung des Lernens. Abbildung 5.1 zeigt eine Versuchsperson mit vollständigen Versuchsaufbau. Auf der Abbildung sind der Bewegungsaufzeichnungsanzug, sowie die Videobrille zu erkennen. Im Rucksack befinden sich zwei Notebooks, eines für den Empfang der Anzugsdaten und eines für das Framework sowie die Ansteuerung der Brille. Seitlich am Rucksack befinden sich die Empfänger des Anzuges.

Abbildung 5.1: Eine Testperson im Bewegungsaufzeichnungsanzug mit Videobrille und mit zwei Notebooks im Rucksack.

5.1 Spiel

Das Spiel wird mittels Daten eines Bewegungsaufzeichnungsanzugs gesteuert. Diese können direkt über Netzwerk in das Framework eingebracht werden. Dort können die Daten entweder direkt verwendet oder abgespeichert werden. Ein Bild des Frameworks ist in Abbildung 5.2 gezeigt.

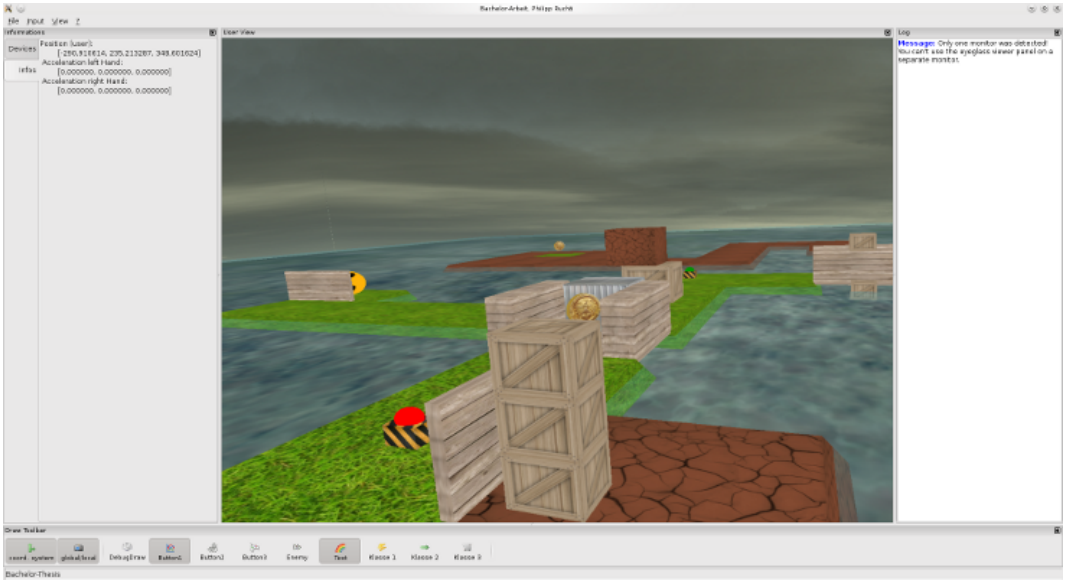

Abbildung 5.2: Ein Blick auf das erarbeitete Framework mit Bildfläche, Steuerelementen, Menü und Infoleisten.

Ziel des implementierten Spieles ist es, die im Level verteilten Münzen einzusammeln. Dazu ist es nötig, Schalter zu drücken oder Kisten auf spezielle Schalter zu platzieren um Tore zu öffnen, offen zu halten oder zu schließen. Letzteres ist nötig um Gegnern zu entkommen. Auch ist es möglich Brücken zu zerstören (siehe Abbildung 5.3) um den Gegnern den Weg zu versperren oder diesen mit Kisten zu verbauen. Spezielle Kisten können nicht nur umgestoßen oder umhergehoben, sondern auch eingesammelt werden, um sie aus dem Weg zu schaffen. Um ein Lernen dieser Aktionen auch während des Spielens im Anzug zu ermöglichen, existiert ein im-Spiel-Menü, welches, wenn die Unsicherheit der Klassifikation zu hoch ist, den Spieler bestimmen lässt zu welcher Aktionenklasse die nicht klassifizierte Aktion gehört. So ist es möglich das Spiel auch während der Nutzung an den Spieler anzupassen.

Abbildung 5.3: Links eine funktionstüchtige Brücke, rechts eine zerstörte.

5.2 Eingabe

Um den Spieler in das Framework zu integrieren wird ein XSens MVN Bewegungsaufzeichnungsanzug verwendet. Dieser liefert anhand von 17 am Körper platzierten Sensoren Daten. Die Sensoren beinhalten neben Drehraten- und Beschleunigungssensoren auch sogenannte Hall-Sensoren um das Erdmagnetfeld zu messen. Um aus den Sensordaten Körperteilpositionen zu berechnen, verwendet das mitgelieferte MVN-Studio ein kinematisches Modell und Skalierungswerte des Spielers. Die Software liefert Daten von 23 Körpersegmenten mit einer Datenrate von bis zu 120Hz. Diese Daten können dann mittels UDP über Netzwerk verbreitet und in das hier vorgestellte Framework eingespielt werden. Obwohl das MVN-Studio neben Position und Orientierung der Körpersegmente auch Geschwindigkeits- und Beschleunigungsdaten liefert, ist es nicht möglich diese über das im MVN-Studio verwendete Protokoll zu verschicken.
Bevor der Anzug genutzt werden kann, muss er mittels vier vordefinierten Posen kalibriert werden. Durch die Kalibrierung ist es der Software möglich die genaue Position der Sensoren am Körper zu bestimmen, so dass eine ungenaue Platzierung eines Sensors die Daten nicht verfälscht. Für eine genauere Beschreibung des Anzuges und seiner Kalibrierung siehe [32] und [25].

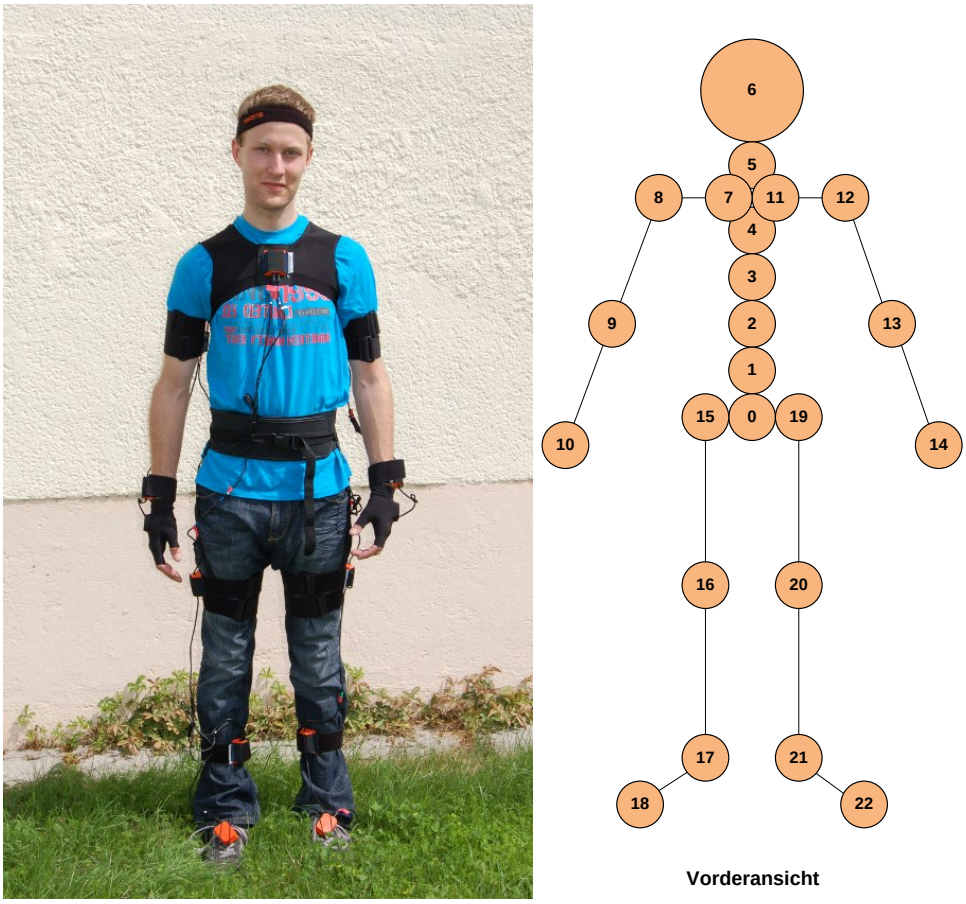

Abbildung 5.4: Links: Der XSens-Datenanzug. Rechts: Die 23 Körpersegmente, welche das MVN-Studio zurückliefert.

5.3 Ausgabe

Damit sich der Spieler während des Spiels frei bewegen kann und trotzdem immer die virtuelle Welt sieht, trägt er eine Videobrille. Hierbei handelt es sich um eine Zeiss Cinemizer Plus 3D-Videobrille. Sie besitzt zwei Displays mit 640x480 Pixeln und einer integrierten Optik. Die Optik bietet einen Öffnungswinkel von 32°, so entsteht der Eindruck eines 45 Zoll großen Fernsehers, der aus zwei Meter Entfernung betrachtet wird.

Abbildung 5.5: Zeiss Cinemizer Plus 3D-Videobrille - die verwendete Videobrille.

Die Sicht der Brille wird anhand der Kopfposition und Rotation, welche der Anzug liefert, immer so gewählt, dass der Spieler den Eindruck erhält er befände sich in der virtuellen Welt, dem Spiel. Die entsprechende Transformation setzt sich zusammen aus: $M_t = \mathit{Projektion} \cdot T_{\mathit{Kopf}\to\mathit{Brille}} \cdot T_{\mathit{Welt}\to\mathit{Kopf}}(t)$.
Hierbei ist $T_{\mathit{Welt}\to\mathit{Kopf}}$ äquivalent zu einer Betrachtertransformation. Im speziellen sieht die hier verwendete Matrix wie folgt aus:

$$M_t = \begin{pmatrix} \frac{f}{aspect} & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & \frac{far+near}{far-near} & \frac{2 \cdot near \cdot far}{far-near} \\ 0 & 0 & -1 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & -10 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} \begin{bmatrix} & & \\ & R_6(t) & \\ & & \end{bmatrix} & \begin{bmatrix} \\ \mathbf{p}_6(t) \\ \\ \end{bmatrix} \\ \begin{matrix} 0 & 0 & 0 \end{matrix} & 1 \end{pmatrix} \tag{5.1}$$

Hierbei ist R_6 die Rotationsmatrix des Kopfsegmentes, $\mathbf{p}_6$ die Position des Kopfsegmentes, $f = 32°$ der Öffnungswinkel der Brille, $near = 10$, beziehungsweise $far = 10000$, die Entfernung zur nahen, beziehungsweise fernen Clippingebene. $aspect = \frac{640}{480} = 1{,}3$ beschreibt das Auflösungsverhältnis der Brille. Aufgrund der Kalibrierung des Anzuges kann als Transformation zwischen Kopfsegment und Brille die Identität sowie eine Translation von 10 cm in Blickrichtung angenommen werden. Dies ist möglich, da das MVN-Studio annimmt, dass die Person beim Kalibrieren geradeaus schaut und diese Kopfpose als Blick Richtung Horizont kalibriert. Des Weiteren sitzt die Brille auf Höhe der Augenmitte, so dass auch hier nichts gegen diese Annahme spricht. Die Translation kommt dadurch zustande, dass sich die Position und Orientierung, die das MVN-Studio für das Kopfsegment liefert, auf den Kopfmittelpunkt beziehen, die Brille aber 10 cm von diesem entfernt auf der Nase sitzt.

5.4 Physik

Damit im Spiel die Gegenstände physikalisch korrekt reagieren und Kollisionen erkannt werden, wird eine Physikengine verwendet. In diesem Fall ist dies *Bullet Physics Libary*, siehe hierzu [4]. Bullet ist eine professionelle, quelloffene Kollisionserkennungs-, Starrkörper- und Weichkörperdynamikbibliothek. Sie bietet neben simpler Starrkörperdynamik und einfacher Kollisionserkennung noch viele weitere interessante Funktionen, zum Beispiel die Integration von weichen Objekten wie Stoffen oder Raycasting. In dieser Arbeit werden lediglich Starrkörper sowie Kollisionserkennung und -behandlung dieser verwendet. So hat jedes Objekt der Spielwelt seinen eigenen, ihm angepassten Starrkörper und kann dadurch mit anderen Objekten kollidieren. Auch das Lernen wird durch die Kollisionserkennung des Spielers mit seiner Umgebung erheblich vereinfacht.

5.5 Agenten

Das implementierte Spiel beinhaltet Computer-Gegner, welche dem Spieler folgen und diesen zum Handeln zwingen. Kommt der Spieler einem der Computer-Agenten zu nahe, beginnt dieser einen Weg zum Spieler zu planen. Hierbei beachtet der Computer-Agent zerstörte Brücken, verschlossene Tore und Wasser. Hierzu läuft er nicht naiv soweit er kann in Richtung des Spielers, was ihn öfters in eine Sackgasse führen könnte. Stattdessen plant der Computer-Agent seinen Weg. Zur Wegfindung wurde eine Breitensuche mit einer Liste der bereits besuchten Felder implementiert. Durch die Breitensuche wird ein optimaler, in diesem Fall kürzester, Weg zum Spieler geplant. Die Breitensuche findet auch in einem Graphen, beziehungsweise Spielfeld, mit Zyklen einen Pfad. Zur Optimierung der Suche wurde ein zusätzlicher Test eingeführt. Vor Beginn der Wegplanung prüft der Computer-Agent, ob die Manhattan-Distanz zum Spieler kleiner ist als eine gewisse Mindestdistanz d_S, also ob $|x_{Player} - x_{Agent}| + |y_{Player} - y_{Agent}| \leq d_S$ gilt. Man kann diesen Wert d_S als die Sichtweite des Gegners auffassen. Nachdem der Computer-Agent eine kurze Wegstrecke zurückgelegt hat, sucht er erneut einen Pfad, da es mittlerweile möglich ist, dass sein derzeit geplanter Weg nicht mehr zum Spieler führt oder durch den Spieler verbaut wurde. Diese Suchmethode muss nicht für alle Gegebenheiten die einfachste sein, da das Level aber aus Wegen im Wasser besteht, entartet die Suche an diesen Stellen zu einem linearen Weg und ermöglicht dadurch eine simple Wegfindung.

5.6 Berechnung von Geschwindigkeit und Beschleunigung

Zur Berechnung von Geschwindigkeits- und Beschleunigungswerten aus Positionen wurden in dieser Arbeit zwei Verfahren implementiert und getestet. Einerseits die Faltung eines Vektors von Positionsdaten mit einem Kern eines Gauß-Glätters für die Berrechnung der Geschwindigkeit und mit einem zweiten Kern zur Berechnung der Beschleunigung, andererseits die Verwendung eines Kalman-Filters. Beide Verfahren haben Vor- und Nachteile offenbart. Die Faltung mit entsprechenden Kernen benötigt, da Daten aus der Zukunft herangezogen werden müssen, das Zwischenspeichern von Daten. Dadurch kommt es zwischen Aktion und Ausführung, beziehungsweise Visualisierung, zu einer Zeitverzögerung, die je nach Grad der Glättung bis zu zwei Sekunden betragen kann. Auf der anderen Seite liefert der Kalman-Filter zwar Daten in Echtzeit, diese sind aber wesentlich verrauschter und haben eine Verzögerung *innerhalb* der Daten. Das heißt die zur Positionsänderung zum Zeitpunkt t gehörige Geschwindigkeit liefert der Kalman-Filter mit einer Verzögerung von mehreren Frames. Die Daten der Beschleunigung haben ihrerseits eine Verzögerung von mehreren Frames bezüglich der Geschwindigkeit. Da für das Lernen der Aktionen (siehe Kapitel 5.7) zusammenpassende Werte von Position, Geschwindigkeit und Beschleunigung essenziell sind, wurde hierzu ein Gauß-Glätter verwendet. Zum Spielen in Echtzeit mit fest programmierten Aktionen empfiehlt sich jedoch die Verwendung des Kalman-Filters, da hier die Verzögerung geringer ausfällt.

Um mittels eines Kalman-Filters (siehe Kapitel 3.2) aus den Positionsdaten Geschwindigkeits- und Beschleunigungswerte zu erhalten, müssen einige Parameter gesetzt werden. Die meisten der Parameter wurden bereits in Kapitel 4.2.1 festgelegt. Eine allgemeine Annahme über das Rauschen der Messdaten kann jedoch nicht getroffen werden. Die Daten der Matrix **B** des Kalman-Filters wurden empirisch bestimmt:

$$B = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.1 & 0 \\ 0 & 0 & 0.01 \end{pmatrix}$$

Die Einheiten der Matrix B sind $1\,[\mathrm{cm}]$, $0.1\left[\frac{\mathrm{cm}}{\frac{1}{60}\cdot \mathrm{s}}\right]$, beziehungsweise $0.01\left[\frac{\mathrm{cm}}{\frac{1}{60}\cdot \mathrm{s}^2}\right]$. Auch die Annahme des Grades des Rauschens der Daten v wurde empirisch ermittelt: $v = 1\,[\mathrm{cm}]$.

5.7 Lernen von Aktionen

In diesem Kapitel wird beschrieben wie verschiedenste Aktionen erlernt und später klassifiziert werden.
Um Aktionen erlernen zu können, wurde ein Spielobjekt definiert, welches mehrere Aktionen zulässt, im Speziellen ist dies die Kiste. An ihnen wird stellvertretend Aufheben, Einsammeln und Wegschlagen erlernt und klassifiziert.
Mit Hilfe eines Featurevektors, welcher aus den Anzugsdaten der Hand, die das Objekt während der Aktion berührt, extrahiert wird, wird eine multidimensionale Normalverteilung geschätzt. Hierzu werden vorerst zwei Features verwendet, der Betrag der Geschwindigkeit und der Betrag der Beschleunigung im exakten Zeitpunkt des Auftreffens. Da diese Werte sehr empfindlich auf den exakten Zeitpunkt der Kollision mit dem Objekt sind, sind sie anfällig gegenüber Misskalibrierungen des Anzuges.
In einem zweiten Ansatz werden Features aus einem kleinen Zeitfenster berechnet: $\tau = \frac{1}{6}$, $a = 0.1$. Diese Werte sind somit weniger anfällig auf den exakten Zeitpunkt der Kollision und erhöhen die Erkennungsrate.
Nach einer anfänglichen Lernphase ist es möglich, zu berechnen wie gut die erlernten Aktionen bezüglich Unterscheidbarkeit sind. Hierzu wird der Klassifikationsfehler des Maximum-Likelihood Klassifikators errechnet. Dieser wird hier verwendet, da von einer gleichmäßigen Auftretenswahrscheinlichkeit der Aktionen ausgegangen wird. Der Spieler sollte alle Aktionen etwa gleich oft demonstriert haben, so dass alle Klassen gleich gut erlernt werden können. Sollte ein Spieler ein Level nach längerem spielen noch einmal laden, würde sich zur Angabe der Güte der Maximum-a-posteriori Klassifikator eignen, da in diesem Fall nicht mehr gegeben ist, dass der Spieler alle Aktionen gleich oft ausgeführt hat.

6 Experimente

In diesem Kapitel werden vier Experimente beschrieben, um die hier vorgestellten Verfahren zu prüfen und zu verifizieren. Zunächst wird die Glättung, beziehungsweise Filterung, der Positionsmessungen sowie die Berechnung von Geschwindigkeits- und Beschleunigungswerten mittels eines Gauß-Glätters sowie eines Kalman-Filters analysiert. Danach wird die Verwendung fest programmierter Schwellwerte zur Klassifikation von Aktionen untersucht. Im Vergleich dazu wird das hier implementierte Lernverfahren getestet. Die erlernten Klassen aus diesem Experiment werden nachfolgend dazu verwendet, die Übertragbarkeit von Aktionsklassen auf andere Personen zu prüfen.

6.1 Experiment: Berechnung geglätteter Position, Geschwindigkeit und Beschleunigung

Um die Berechnung der Geschwindigkeits- und Beschleunigungswerte zu prüfen, wird folgendes Experiment durchgeführt. Eine Testperson bewegt ihren Arm in einer sich gleichmäßig wiederholenden Bewegung hin und her. Dabei wird die Wiederholung der Bewegung mittels eines Taktgebers zeitlich festgelegt. In dem hier evaluierten Experiment bewegte die Testperson ihre rechte Hand einen Meter nach rechts und links. Dabei wurde die Bewegung mit einem Metronom auf 1Hz festgelegt.
Die resultierende Geschwindigkeits- und Beschleunigungskurve einer Periode ist in Abbildung 6.1 gezeigt. Es ist zu sehen, dass die Werte für die Geschwindigkeit, bei ausreichender Glättung (siehe Abbildung 6.1$[mitte]$), während der ersten halben Periode in einer glatten, nach oben geöffneten, parabelförmigen Kurve verlaufen, während die Beschleunigungswerte gleichzeitig eine entgegengesetzte, nach unten geöffnete Parabel beschreiben. In der zweiten Hälfte der Aktion wiederholen sich die Kurven für Geschwindigkeit und Beschleunigung. Dies tritt aufgrund der Betragsbildung ein, sonst verliefen die Kurven im Negativen, da sich nun die Hand in

entgegengesetzter Richtung bewegt. Diese Kurven decken sich mit den Überlegungen, die sich aus dem Aufbau des Experimentes ergeben.
Dass sich die Hand nicht genau einen Meter von rechts nach links bewegt, liegt an der Montage des Sensors auf dem Handrücken. Dadurch befindet sich jeweils die Hand zwischen Sensor und Start- beziehungsweise Zielposition.
In Abbildung 6.1$[links]$ sind Kurven gezeigt, welche mit einem Gauß-Glätter und $\sigma = 2$ erzeugt wurden. Die Kurven in Abbildung 6.1$[mitte]$ wurden ebenfalls mit einem Gauß-Glätter, jedoch mit $\sigma = 12$ erzeugt. Im Vergleich fällt auf, dass die Kurven in Abbildung 6.1$[links]$ wesentlich mehr Rauschen enthalten als in Abbildung 6.1$[mitte]$. Somit ist ein Gauß-Glätter mit $\sigma = 12$ eine gute Wahl zur Berechnung von Geschwindigkeits- und Beschleunigungswerten.
Die Geschwindigkeits- und Beschleunigungskurven in Abbildung 6.1$[rechts]$ sind das Ergebnis eines Kalman-Filters. Vergleicht man dieses mit Abbildung 6.1$[mitte]$, so ist deutlich zu erkennen, dass die Positionskurven übereinstimmen, aber die Geschwindigkeitskurve in Abbildung 6.1$[rechts]$ um fast eine $\frac{1}{4}$ Sekunde verschoben ist. Auch die Beschleunigungskurve ist verschoben, diese sogar um fast $\frac{2}{3}$ Sekunden. Somit ist der Kalman-Filter bezüglich der Übereinstimmung der Kurven eine weniger gute Wahl für die Berechnung von Geschwindigkeits- und Beschleunigungswerten.

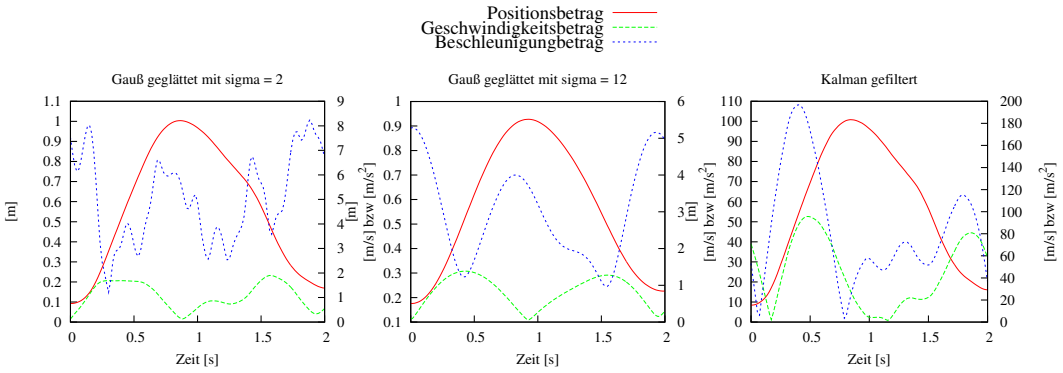

Abbildung 6.1: Geschwindigkeits- und Beschleunigungskurve während einer Periode der Aktion. Links: geglättet, mit einem Gauß-Glätter und $\sigma = 2$, mitte: $\sigma = 12$. Rechts: Geschwindigkeits- und Beschleunigungskurve während einer Periode der Aktion, Kalman gefiltert.

6.2 Die drei verwendeten Beispielaktionen

In diesem Abschnitt werden die drei zum Lernen und Klassifizieren verwendeten Aktionen beschrieben. Für jede Aktion wird ein Diagramm mit je einer Ausführung der Aktion gezeigt. Im diesem wird der Verlauf von Geschwindigkeits- und Beschleunigungswerten der Hand sowie ihr Abstand zur Kistenmitten gezeigt. Zusätzlich ist die Kollisionsdauer verzeichnet und die Werte von Abstand, Geschwindigkeit und Beschleunigung zum Beginn der Kollision hervorgehoben (blaue Kreise).

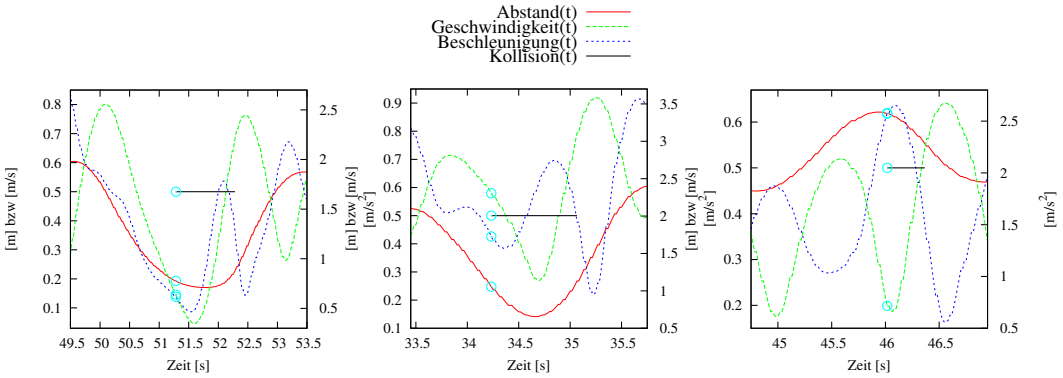

Abbildung 6.2: Je eine beispielhafte Aktion. Von links nach rechts: Aufheben, Einsammeln, Wegschlagen. Zusätzlich zum Abstand der Hand zur Kistenmitte, Kollisionsdauer, Geschwindigkeit und Beschleunigung der Hand sind die Werte zum Zeitpunkt des Beginns der Kollision hervorgehoben.

6.2.1 Aufheben

Bei der ersten der drei Aktionen handelt es sich um das Aufheben einer Kiste. Hierbei wird die Kiste mit beiden Händen seitlich gegriffen.
Die Kurven für Geschwindigkeit und Beschleunigung in Abbildung 6.2[links] weisen im Moment der beginnenden Kollision Werte nahe Null auf. Dies resultiert daraus, dass die Hand im Moment der Kollision an der Oberfläche der Kiste zum Stillstand kommt.

6.2.2 Einsammeln

Die zweite der Beispielaktionen ist das Einsammeln einer Kiste. Hierbei wird eine oder werden beide Hände durch die Kiste bewegt.

An den Kurven in Abbildung 6.2[mitte] ist erkennbar, dass im Moment der ersten Kollision die Werte für Geschwindigkeit und Beschleunigung mittlere Werte aufweisen. Die Beschleunigung hat in diesem Moment zusätzlich einen Tiefpunkt. Die ausführende Person hat ihre Hand bis zur Kistenmitte beschleunigt und ab dort wieder abgebremst.

6.2.3 Wegschlagen

Das Wegschlagen einer Kiste ist die letzte der drei Beispielaktionen. Hierbei schlägt der Spieler auf die Oberfläche der Kiste und bremst seine Hand in diesem Moment, da bei einer Schlagbewegung in der Realität die Hand ebenfalls zum Stillstand kommen würde.
Diese Bremsbewegung spiegelt sich in einem Hochpunkt der Beschleunigung sowie einem Tiefpunkt der Geschwindigkeit zum Beginn der Kollision wider (siehe Abbildung 6.2[rechts]). Nach der Kollision hat die Person ihre Hand wieder von der Kiste entfernt und somit die Geschwindigkeit wieder erhöht.

6.3 Experimente: Fest programmierte Aktionen

In diesem Experiment wird das konventionelle Vorgehen des Programmierens anhand von fest gewählten Schwellwerten evaluiert. Zur Erkennung der Aufhebeaktion wurde der Abstand der Hände im Vergleich zur Kistengröße sowie dessen Mitte zum Mittelpunkt der Kiste gemessen. Die Einsammelbewegung wurden als solche erkannt, sobald sich eine der Hände nahe genug an der Kistenmitte befand. Wegschlagen wurde anhand des Absolutbetrages der Beschleunigung der mit der Kiste kollidierenden Hand erkannt. Um die fest programmierten Aktionen zu testen, wurden drei Versuchspersonen gebeten mehrmalig die drei implementierten Aktionen an einer Kiste im Spiel auszuführen. Hierbei erhielten die Personen keine Rückmeldung vom Spiel, ob ihre Aktionen erfolgreich waren. So wurde garantiert, dass sich die Personen nicht an die Aktionen anpassen konnten, sondern lediglich die unverfälschte Güte der Ausführung bewertet wurde. Als Referenz, wie viele Aktionen erkannt werden sollen, wurde die Gesamtzahl der Kollisionen zwischen Händen und Kiste gezählt.

Aktion	Person 1	Person 2[1]	Person 2[2]	Person 3
Aufheben	80,645%	84,375%	75%	100%
Einsammeln	60%	83,333%	68,966%	70,370%
Wegschlagen	77,778%	0%	0%	0%
Gesamt	72,152%	55,914%	48,352%	57,5%

Tabelle 6.1: Korrekt erkannte Aktionen in Prozent pro Versuchsperson beziehungsweise Ausführung ([1], [2]) bei fest programmierten Aktionen.

Im Vergleich zu den nachfolgenden Versuchspersonen weisen die Aktionen von Versuchsperson 1 eine hohe Erkennungsrate auf. Dies ist darauf zurückzuführen, dass diese Person im Gegensatz zu den weiteren Versuchspersonen bereits Übung mit den hier implementierten, fest programmierten Aktionen hatte und so mit diesen ausreichend zurechtkam.
Versuchsperson 2 hat in ihrer ersten Ausführung der drei Aktionen sehr ähnliche Schlag- und Hebeaktionen verwendet. Sowohl sie als auch Versuchsperson 3 führten ihre Schlagaktionen so aus, dass es zu einer zu geringeren Bremsbewegung im Moment des Auftreffens kam. So wird bei beiden Versuchspersonen keine Schlagbewegung als solche erkannt. Begünstigt durch die zu geringen Beschleunigungswerte beim Schlag werden ein Großteil der Hebe- und Einsammelbewegungen korrekt erkannt. Die Kiste wird also vorher nicht ungewollter Weise weggeschlagen.
Da Versuchsperson 2 die Hebe- und Schlagaktionen in ihrer ersten Ausführung sehr ähnlich ausgeführt hat, wurde sie gebeten eine zweite, unterscheidbarere Demonstration für jede Aktion durchzuführen. Im zweiten Durchlauf hat die Testperson ihre Einsammelaktionen zum Teil mit beiden Händen vorgenommen, somit wurden diese zum Teil als Hebebewegungen erkannt und nicht als Wegschlagen einer Kiste. Somit haben sich für diese Versuchsperson bei den fest programmierten Aktionen die Ergebnisse lediglich verschlechtert.

Ein System, welches auf hart programmierten Schwellwerten beruht, ist nicht für jeden Spieler intuitiv geeignet. Eine Möglichkeit hier Verbesserung zu schaffen, wäre es dem Spieler während seiner Ausführung Rückmeldung zu geben, ob seine soeben ausgeführte Aktion erfolgreich war und als die gewünschte erkannt wurde. Hierbei würde sich der Spieler dem System anpassen. Dies war jedoch in diesem Experiment nicht gewünscht. Eine weitere Möglichkeit wäre, die Schwellwerte auf jeden Spieler

persönlich anzupassen. Hierbei würde es sich bereits um einen simplen Lernprozess handeln und nicht mehr um eine reine fest programmierte Aktionserkennung.

6.4 Experimente: Lernen von Aktionen

Mit Hilfe der Ausführungen, an welchen die fest programmierten Schwellwerte getestet wurden (siehe Kapitel 6.3), wurden Aktionsklassen erlernt, um die Güte des Lernens der verschiedenen Aktionen zu bestimmen. Diese an den einzelnen Spieler angepassten Aktionsklassen wurden nachfolgend evaluiert. Hierzu wurde gezählt wie oft einer Testperson eine Aktion gelang und wie oft sie eine andere als die gewünschte Aktion ausführte. Also wie oft es zu korrekten und wie oft es zu einer Fehlklassifikation kam.
Für jede der drei Aktionen wurde gezählt wie viele der Kollisionen der Hände mit der Kiste zur richtigen Aktion zugeordnet wurden.
Zum Test der erlernten Klassen wurde das Kreuzvalidierungsverfahren verwendet. Hierbei werden die Aktionsklassen nicht aus allen Demonstrationen gelernt. In jedem Durchgang wird eine Demonstration ausgelassen. Nach dem Lernen der Klassenverteilungen aus den restlichen Ausführungen wird die Zugehörigkeit der ausgelassenen Demonstration ermittelt.

6.4.1 Erster Ansatz

Mit dem ersten Ansatz für den Featurevektor (siehe (4.1)) waren keine zufriedenstellenden Ergebnisse erreichbar. Somit werden diese hier lediglich kurz erwähnt und im weiteren Verlauf der Arbeit der dreidimensionale Featurevektor (siehe (4.4)) verwendet. Es war mit dem zweidimensionalen Featurevektor kein Ergebnis möglich, bei welchem für die meisten der Testpersonen mehr als 90% der Aktionen richtig klassifiziert wurden. Für die drei Testpersonen wurden folgende Ergebnisse erzielt:

Aktion	Person 1	Person $2^{[1]}$	Person $2^{[2]}$	Person 3
Aufheben	100,0%	45,2%	93,3%	100,0%
Einsammeln	100,0%	31,3%	84,4%	96,3%
Wegschlagen	96,7%	46,7%	89,7%	74,1%
Gesamt	**98,7%**	**44,1%**	**89,0%**	**90,0%**

Tabelle 6.2: Ergebnisse des Lernen mit Hilfe des zweidimensionalen Featurevektors pro Versuchsperson, beziehungsweise Ausführung ($^{[1]}, ^{[2]}$).

6.4.2 Zweiter Ansatz

Mit den dreidimensionalen Featurevektoren (siehe Kapitel 4.4) ergeben sich für die verschiedenen Testpersonen folgende Ergebnisse:

	Person 1	Person $2^{[1]}$	Person $2^{[2]}$	Person 3
Aufheben	100,0%	65,6%	90,6%	100,0%
Einsammeln	93,3%	86,7%	93,1%	85,2%
Wegschlagen	100,0%	80,6%	93,3%	96,2%
Gesamt	**97,5%**	**77,4%**	**92,3%**	**93,8%**
vorherges. Güte	97,8%	77,1%	93,8%	97,8%

Tabelle 6.3: Ergebnisse der verschiedenen Versuchsperson, beziehungsweise Ausführungen ($^{[1]}, ^{[2]}$), bei eigens erlernten Aktionen sowie die vorhergesagte Gesamtgüte.

Zusätzlich zu den Klassenverteilungen der Aktionen wurde eine Vorhersage des eintretenden Fehlers berechnet. Hierzu wurde numerisch über den entsprechenden Teil des Verteilungsraumes integriert und für jede Klasse gemessen wie oft für diese entschieden werden würde und wie oft für andere Klassen. Diese Güte beschreibt die Trennbarkeit der erlernten Aktionsklassen.

Wie man in Tabelle 6.3 sieht werden für Versuchsperson 1 mit den eigens erlernten Aktionen fast alle der ausgeführten Aktionen korrekt klassifiziert. Dies Ergebnis

überrascht nicht, da die selbe Versuchsperson bereits bei den fest programmierten Aktionen (siehe Tabelle 6.1) eine hohe Erkennungsrate vorzuweisen hatte. Die Erkennungsrate konnte im Vergleich zu den fest programmierten Aktionen noch einmal erheblich verbessert werden.
Versuchsperson 2 wählte für ihre Demonstrationen zuerst eine ähnliche Ausführung der Aktionen Aufheben und Wegschlagen. Somit können diese nur schwer unterschieden werden, was sich auch in den Werten der Erkennungsrate widerspiegelt (siehe Tabelle 6.3). Hier versagt sowohl die feste Programmierung als auch der lernende Ansatz. Diesem Spieler sagte die Berechnung der Güte des Klassifikators ein eher dürftiges Ergebnis vorher, deshalb wurde die Versuchsperson angehalten ihre Aktionen in einem zweiten Durchgang unterscheidbarer zu präsentieren.
Mit Hilfe der zweiten Demonstration konnten Klassen erlernt werden, welche sich deutlich besser unterscheiden lassen. Dies spiegelt sich direkt im Ergebnis der Kreuzvalidierung wider (siehe Tabelle 6.3). Auch die vorhergesagte Güte des Klassifikators lässt auf eine bessere Trennbarkeit der Klassen schließen.
Mit den fest programmierten Aktionen konnte Versuchsperson 3 keine akzeptablen Ergebnisse erzielen, da sie ihre Aktionen sehr langsam ausführte. Somit kam sie während der Ausführung nicht auf die von der festen Programmierung erwarteten absoluten Beschleunigungswerte. Mit den eigens angepassten Klassen ist es ihr möglich die meisten ihrer Aktionen erfolgreich auszuführen (siehe Tabelle 6.3).

Um die Verwechselungshäufigkeit zwischen bestimmten Klassen zu untersuchen und um herauszufinden welche der Aktionen am häufigsten verwechselt werden, wurde eine Tabelle der Zuordnungen von Aktionen jeder Aktionsklasse zu den drei Beispielaktionen aufgestellt.

Aktionsklasse \ erkannt als	Aufheben	Einsammeln	Wegschlagen
Aufheben	96,6%	2,2%	1,1%
Einsammeln	1,4%	96,0%	2,7%
Wegschlagen	4,7%	4,7%	90,7%

Tabelle 6.4: Zuordnungsrate von Ausführungen jeder Beispielaktion zu den verschiedenen Aktionsklassen für alle Versuchspersonen (bei Versuchsperson 2 nur die zweite Demonstration).

Als Ergebnis dieses Experimentes kann festgestellt werden, dass keine der Aktionen auffallend oft mit einer bestimmten anderen Aktion verwechselt wurde.

6.5 Übertragbarkeit von erlernten Aktionsklassen

Als letztes wurde getestet wie gut sich die gelernten Aktionen von einer auf eine andere Person übertragen lassen. Für dieses Experiment wurden die Aktionsklassen einer der obigen Testpersonen gelernt. Nachfolgend wurde für jede Ausführung einer Aktion einer anderen Versuchspersonen geprüft, ob sie korrekt klassifiziert wurde. Auch in diesem Experiment wurden zur Überprüfung der Güte die Anzahl der richtigen und falschen Klassifikationen gezählt.

erlernt für / verwendet von	Person 1	Person $2^{[1]}$	Person $2^{[2]}$	Person 3
Person 1	97,5%	10,1%	59,5%	38,0%
Person $2^{[1]}$	20,4%	77,4%	33,3%	30,1%
Person $2^{[2]}$	57,1%	22,0%	92,3%	28,6%
Person 3	55,0%	30,0%	52,5%	93,8%

Tabelle 6.5: Korrekt klassifizierte Aktionen gesamt der verschiedenen Versuchspersonen, beziehungsweise Ausführungen ($^{[1]}, ^{[2]}$), bei fremden, beziehungsweise eigen, Verteilungen.

erlernt für / verwendet von	Person 1	Person $2^{[1]}$	Person $2^{[2]}$	Person 3
Person 1	100,0%	0,0%	16,1%	0,0%
Person $2^{[1]}$	3,1%	65,6%	6,3%	0,0%
Person $2^{[2]}$	28,1%	3,1%	90,6%	0,0%
Person 3	100,0%	0,0%	14,8%	100,0%

Tabelle 6.6: Korrekt klassifizierte Aufhebeaktionen der verschiedenen Versuchspersonen, beziehungsweise Ausführungen ($^{[1]}, ^{[2]}$), bei fremden, beziehungsweise eigen, Verteilungen.

verwendet von \ erlernt für	Person 1	Person 2[1]	Person 2[2]	Person 3
Person 1	93,3%	23,3%	86,7%	100,0%
Person 2[1]	50,0%	86,7%	36,7%	90,0%
Person 2[2]	75,9%	62,1%	93,1%	82,8%
Person 3	59,3%	40,7%	88,9%	85,2%

Tabelle 6.7: Korrekt klassifizierte Einsammelaktionen der verschiedenen Versuchspersonen, beziehungsweise Ausführungen ([1], [2]), bei fremden, beziehungsweise eigen, Verteilungen.

verwendet von \ erlernt für	Person 1	Person 2[1]	Person 2[2]	Person 3
Person 1	100,0%	5,6%	88,9%	0,0%
Person 2[1]	9,7%	80,6%	58,1%	3,2%
Person 2[2]	70,0%	3,3%	93,3%	6,7%
Person 3	3,8%	50,0%	53,8%	96,2%

Tabelle 6.8: Korrekt klassifizierte Wegschlagaktionen der verschiedenen Versuchspersonen, beziehungsweise Ausführungen ([1], [2]), bei fremden, beziehungsweise eigen, Verteilungen.

Es ist erkennbar, dass Versuchsperson 2 ein erheblich besseres Ergebnis mit den Aktionsklassen von Versuchsperson 1 erzielt und umgekehrt, als Versuchsperson 3. Dies ist der Fall, da Versuchsperson 2 und Versuchsperson 1 eine ähnlichere Ausführung der Aktionen gewählt haben. Versuchsperson 3, welche eine gemächliche Ausführung der Aktionen gewählt hat, kann kaum eine Aktion mit fremden Aktionsklassen erfolgreich ausführen.
Als Ergebnis des Experimentes lässt sich festhalten, dass auf eine Person erlernte Klassen nicht übertragbar sind. Dies ist auch der Fall, wenn zwei Personen eine ähnliche Ausführung der Aktionen wählen.

6.6 Ergebnis der Aktionsklassifikations-Experimente

Als Ergebnis der Experimente kann festgestellt werden, dass ein lernender Ansatz einem fest programmierten vorzuziehen ist. Dieser ermöglicht es auch Eigenarten eines Spielers zu erkennen. So ist es bei den hier implementierten fest programmierten Aktionen nicht möglich, mit zwei Händen eine Kiste einzusammeln, ohne dass diese Aktion als Aufheben erkannt wird. Ebenfalls ist es Spielern mit einer langsamen Ausführung der Aktionen nicht möglich Schlagaktionen erfolgreich auszuführen. Bei fest programmierten Aktionen kann nicht jeder Spieler ohne sich an das Spiel anzupassen Ergebnisse erzielen, mit welchen sich flüssig spielen lässt. Erlernte Aktionsklassen sind nicht übertragbar. Auch für Personen die eine ähnliche Ausführung von Aktionen wählen, ist es notwendig die Aktionsklassen erneut zu lernen.
Das Verfahren zur Ermittlung der Güte der gelernten Aktionsklassen liefert Ergebnisse, welche nur wenig von den eintretenden Ergebnissen abweichen.

7 Schluss und Diskussion

In dieser Arbeit wurde ein Virtual Reality Spiel mit Ganzkörperbewegungserkennung implementiert. Es wurden zwei Verfahren vorgestellt, um aus verrauschten Positionsmessungen geglättete Positionsdaten sowie Geschwindigkeits- und Beschleunigungswerte zu erhalten. Beide Verfahren haben Vor- und Nachteile offenbart. Es wurden weiterhin zwei Methoden zur Erkennung von Aktionen implementiert und geprüft. Als erster Ansatz wurde hierzu das Vorgehen des festen Programmierens anhand von Schwellwerten gewählt. Im Vergleich dazu wurde ein auf Normalverteilungen beruhender Lernalgorithmus implementiert. In den Experimenten wurde gezeigt, dass sich Aktionen einerseits zwar fest programmieren lassen, jedoch ein lernender Ansatz bevorzugt werden sollte. So ist es auch langsamen Spielern oder solchen mit Eigenarten in der Ausführung der Aktionen möglich, flüssig zu spielen. Es wurde gezeigt, dass sich erlernte Aktionsklassen nicht von einer auf eine andere Person übertragen lassen. Somit muss das Spiel auf jeden Spieler persönlich angepasst werden. Auch wurde eine Möglichkeit gezeigt, die Erkennungsraten eines Parametersatzes vorherzusagen.
Das Verfahren zur Berechnung geglätteter Positionsdaten sowie Geschwindigkeits- und Beschleunigungswerte ließe sich noch verbessern, wenn die Parameter des Kalman-Filters mittels eines Gradientenalgorithmus sowie den Daten des Gauß-Glätters als Referenz optimiert werden würden. Auch ist es möglich noch weitere Aktionen, Objekte oder Körperteile in den Lernprozess zu integrieren. Pro Aktion und Körperteil müssten hier Verteilungen gelernt werden, da jedes Körperteil seine eigene Dynamik aufweist. Auch ist es möglich mehr und andere Features zu verwenden um so die Klassifikationsgüte zu heben.

Literaturverzeichnis

[1] ANDRILUKA, Mykhaylo ; ROTH, Stefan ; SCHIELE, Bernt: Monocular 3D Pose Estimation and Tracking by Detection. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[2] BAAK, Andreas ; ROSENHAHN, Bodo ; MÜLLER, Meinard ; SEIDEL, Hans-Peter: Stabilizing Motion Tracking Using Retrieved Motion Priors. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[3] BISHOP, Gary ; WELCH, Greg: An Introduction to the Kalman Filter. 2006

[4] COUMANS, Erwin: Bullet 2.76 Physics SDK Manual. `http://bulletphysics.com/ftp/pub/test/physics/Bullet_User_Manual.pdf`. Version: 2010

[5] DAI, Rongching ; STEIN, R.B. ; ANDREWS, B.J. ; JAMES, K.B. ; WIELER, M.: Application of tilt sensors in functional electrical stimulation. In: Rehabilitation Engineering, IEEE Transactions on 4 (1996), jun., Nr. 2

[6] DE LUCA, Alessandro ; MATTONE, Raffaella ; GIORDANO, Paolo R. ; BÜLTHOFF, Heinrich H.: Control design and experimental evaluation of the 2D cyber walk platform. In: IROS'09: Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems. Piscataway, NJ, USA : IEEE Press, 2009

[7] DIETRICH, Nira-Corina: Mixed Reality, Frameworks zur Entwicklung von AR-Systemen. 2004

[8] DOUKAS, C. ; MAGLOGIANNIS, I.: Advanced patient or elder fall detection based on movement and sound data. In: Pervasive Computing Technologies for Healthcare, 2008. PervasiveHealth 2008. Second International Conference on, 2008

[9] DUETSCHER, J. ; BLAKE, A. ; REID, I.: Articulated Body Motion Capture by Annealed Particle Filtering. In: Computer Vision and Pattern Recognition, IEEE Computer Society Conference on 2 (2000)

[10] FREDERICK P. BROOKS, Jr.: What's Real About Virtual Reality? In: IEEE Computer Graphics and Applications (1999)

[11] FREIFELD, Oren ; WEISS, Alex ; ZUFFI, Silvia ; BLACK, Michael: Contour People: A Parameterized Model of 2D Articulated Human Shape. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[12] FRÖHLICH, Bernd ; PLATE, John: The cubic mouse: a new device for three-dimensional input. In: CHI '00: Proceedings of the SIGCHI conference on Human factors in computing systems. New York, NY, USA : ACM, 2000

[13] FURUKAWA, Yasutaka ; PONCE, Jean: Dense 3D Motion Capture from Synchronized Video Streams. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[14] GALL, Juergen ; STOLL, Carsten ; AGUIAR, Edilson de ; THEOBALT, Christian ; ROSENHAHN, Bodo ; SEIDEL, Hans-Peter: Motion Capture Using Joint Skeleton Tracking and Surface Estimation. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[15] GANAPATHI, Varun ; PLAGEMANN, Christian ; THRUN, Sebastian ; KOLLER, Daphne: Real Time Motion Capture Using a Single Time-Of-Flight Camera. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[16] H. J. LUINGE, P. H. V.: Measuring orientation of human body segments using miniature gyroscopes and accelerometers. In: Medical & Biological Engineering & Computing 43 (2005)

[17] HARTUNG, Joachim ; ELPELT, Bärbel: Multivariate Statistik: Lehr- u. Handbuch d. angewandten Statistik. München/Wien : Oldenbourg, 1984

[18] HASLER, Nils ; ROSENHAHN, Bodo ; THORMÄHLEN, Thorsten ; WAND, Michael ; GALL, Juergen ; SEIDEL, Hans-Peter: Markerless Motion Capture with Unsynchronized Moving Cameras. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[19] HUYNH, Tâm ; SCHIELE, Bernt: Analyzing features for activity recognition. In: sOc-EUSAI '05: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence. New York, NY, USA : ACM, 2005

[20] Microsoft: Werbevideo Kinect. `http://www.xbox.com/de-DE/kinect/default.htm`. Version: 2010

[21] KOSTOV, A. ; ANDREWS, B.J. ; POPOVIC, D.B. ; STEIN, R.B. ; ARMSTRONG, W.W.: Machine learning in control of functional electrical stimulation systems for locomotion. In: Biomedical Engineering, IEEE Transactions on 42 (1995), jun., Nr. 6

[22] LESTER, Jonathan ; CHOUDHURY, Tanzeem ; KERN, Nicky ; BORRIELLO, Gaetano ; HANNAFORD, Blake: A hybrid discriminative/generative approach for modeling human activities. In: In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI, 2005

[23] LIU, Sheng ; CHENG, Dewen ; HUA, Hong: An optical see-through head mounted display with addressable focal planes. In: Mixed and Augmented Reality, IEEE/ACM International Symposium on 0 (2008)

[24] LOVELL, N.H. ; WANG, Ning ; AMBIKAIRAJAH, E. ; CELLER, B.G.: Accelerometry Based Classification of Walking Patterns Using Time-frequency Analysis. In: Proceedings of the 29th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2007

[25] MAIER, Christian ; KUHNER, Daniel ; RUCHTI, Philipp: Computergestütztes Bewegungstraining: Erkennung und Korrektur von Körperhaltung und Bewegungsabläufen. Freiburg, Baden-Württemberg, Deutschland, 2010

[26] MOHLER, Betty J. ; CREEM-REGEHR, Sarah H. ; THOMPSON, William B.: The influence of feedback on egocentric distance judgments in real and virtual environments. In: APGV '06: Proceedings of the 3rd symposium on Applied perception in graphics and visualization. New York, NY, USA : ACM, 2006

[27] NAJAFI, B. ; AMINIAN, K. ; LOEW, F. ; BLANC, Y. ; ROBERT, Ph.: Measurement of stand-sit and sit-stand transitions using a miniature gyroscope and its application in fall risk evaluation in the elderly. In: IEEE Transactions on Biomedical Engineering 49 (2002), Nr. 8

[28] NIKOLAKIS, Georgios ; TZOVARAS, Dimitrios ; MOUSTAKIDIS, Serafim ; STRINTZIS, Michael G.: CyberGrasp and PHANTOM Integration: Enhanced Haptic Access for Visually Impaired Users. In: Proceedings of the 9th International Conference SSpeech and Computer"SPECOM 2004. St. Petersburg, Russia : Moskow State Linguistic University, September 2004

[29] PARK ; MIN, Hyung ; LEE, Seok H. ; CHOI, Jong S.: Wearable augmented reality system using gaze interaction. In: ISMAR '08: Proceedings of the 7th

IEEE/ACM International Symposium on Mixed and Augmented Reality. Washington, DC, USA : IEEE Computer Society, 2008

[30] PONS-MOLL, Gerard ; BAAK, Andreas ; HELTEN, Thomas ; MUELLER, Meinard ; SEIDEL, Hans-Peter ; ROSENHAHN, Bodo: Multisensor-Fusion for 3D Full-Body Human Motion Capture. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[31] PREECE, S.J. ; GOULERMAS, J.Y. ; KENNEY, L.P.J. ; HOWARD, D.: A Comparison of Feature Extraction Methods for the Classification of Dynamic Activities From Accelerometer Data. In: Biomedical Engineering, IEEE Transactions on (2009)

[32] ROETENBERG, Daniel ; LUINGE, Henk ; SLYCKE, Per: Xsens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors. 8. April 2009. Enschede, Niederlande: XSENS TECHNOLOGIES, 2009. `http://www.xsens.com/images/stories/PDF/MVN_white_paper.pdf`

[33] ROSENHAHN, Bodo ; SCHMALTZ, Christian ; BROX, Thomas ; WEICKERT, Joachim ; CREMERS, Daniel ; SEIDEL, Hans-Peter: Markerless Motion Capture of Man-Machine Interaction. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[34] SALZMANN, Mathieu ; URTASUN, Raquel: Combining Discriminative and Generative Methods for 3D Deformable Surface and Articulated Pose Reconstruction. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010

[35] SCHMALSTIEG, Dieter ; FUHRMANN, Anton ; SZALAVÁRI, Gerd Hesina Z. ; ENCARNACÇÃO, L. M. ; GERVAUTZ, Michael ; PURGATHOFER, Werner: The studierstube augmented reality project. In: PRESENCE - Teleoperators and Virtual Environments 11(1) (2002)

[36] VLASIC, Daniel ; ADELSBERGER, Rolf ; VANNUCCI, Giovanni ; BARNWELL, John ; GROSS, Markus ; MATUSIK, Wojciech ; POPOVIĆ, Jovan: Practical motion capture in everyday surroundings. In: SIGGRAPH '07: ACM SIGGRAPH 2007 papers. New York, NY, USA : ACM, 2007, S. 35

[37] ZHOU, Mingcai ; LIANG, Lin ; SUN, Jian ; WANG, Yangsheng: AAM based Face Tracking with Temporal Matching and Face Segmentation. In: IEEE Conference on Computer Vision & Pattern Recognition, 2010