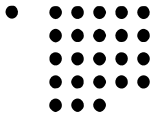

Fachhochschule Köln
Cologne University of Applied Sciences

Entwicklung und Vergleich von Verfahren zur Verbesserung der Gestenerkennung für den Einsatz in Natural User Interfaces

Masterthesis

ausgearbeitet von

Renée Schulz

zur Erlangung des akademischen Grades

Master of Science

vorgelegt an der

Fachhochschule Köln
Campus Gummersbach
Fakultät für Informatik und
Ingenieurwissenschaften

im Studiengang

Medieninformatik

Erster Prüfer: Prof. Dr. Wolfgang Konen
Fachhochschule Köln

Zweiter Prüfer: Prof. Dr. Gerhard Hartmann
Fachhochschule Köln

Gummersbach, im August 2013

Adressen: Renée Schulz
Sienhardtstraße 43a
51645 Gummersbach
renee@schulz52.de

Prof. Dr. Wolfgang Konen
Fachhochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
wolfgang.konen@fh-koeln.de

Prof. Dr. Gerhard Hartmann
Fachhochschule Köln
Institut für Informatik
Steinmüllerallee 1
51643 Gummersbach
gerhard.hartmann@fh-koeln.de

Kurzfassung

Die gestenbasierte Interaktion spielt in der Mensch-Computer Interaktion eine wichtige Rolle. Sie bietet alternative und natürliche Interaktionsmöglichkeiten zu den bisher genutzten Eingabemodalitäten. Da es bei der Nutzung von gestenbasierten Interfaces von maßgeblicher Bedeutung ist, dass die Gesten genaustmöglich erkannt werden, liegt die Realisierung einer guten Erkennungsrate, sowie einer geringen Rate von falsch erkannten oder falsch abgelehnten Gesten nahe. Im Folgenden wird die Erkennung von Gesten mittels Hidden Markov Modellen analysiert und auf Basis einer Datenerhebung und Analyse, Verbesserungsmöglichkeiten für die Erkennung von Gesten konzipiert.

Dafür werden verschiedene Ansätze entwickelt und geprüft. Verbesserungsmethoden, welche die aufgezeichneten Gestendaten im Hinblick auf eine bessere Erkennungsrate aufbereiten sollen, werden entwickelt und getestet. Dabei wird der Ansatz der „Action Plane", welcher von Richarz and Fink vorgestellt wurde, miteinbezogen. Die Erkennung auf Basis von den Positionsdaten differenzierter Featurevektoren wird ebenfalls entwickelt und getestet; Darunter Geschwindigkeitseigenschaften, eine Kombinationen aus Positionseigenschaften und Geschwindigkeitseigenschaften, ein vierdimensionaler Eigenschaftsvektor aus Ortskoordinaten und Absolutgeschwindigkeit, sowie der Phasenraum der Gesten.

Ziel der Arbeit ist es, aufzuzeigen welche der Methoden zur Verbesserung der Erkennungsleistung von Gesten verschiedener Personen eine tatsächliche Verbesserung liefert. Ein weiterer Ansatz, um die Erkennungsrate von Gesten zu verbessern, ist, optimale Parameter bei der Erstellung des Hidden Markov Modells zu gewährleisten. Für die verschiedenen Featurevektor-Ansätze werden die verwendeten Parameter des Hidden Markov Modells überprüft und angepasst. Zudem wird nicht nur die Erkennungsrate von personenübergreifenden Gesten getestet, sondern auch die Erkennung einer Person anhand ihrer Geste. Die Vermutung besteht darin, dass der Phasenraum einer Geste personenbezogene Charakteristika enthält, anhand derer eine Person eindeutig erkennbar ist.

Inhaltsverzeichnis

1 Einleitung

Derzeitige technische Entwicklungen eröffnen neue Möglichkeiten für alternative Interaktionstechniken in der Mensch-Computer Interaktion. Besonders die Natürlichkeit der Interaktion spielt eine immer größere Rolle in der Interaktionsgestaltung.

Viele neue Entwicklungen zielen darauf ab, eine Interaktion ohne weitere Hilfsmittel, wie Maus oder Tastatur, möglich zu machen. Die Verwendung von Gesten ist eine schon lang etablierte Form zwischenmenschlicher Kommunikation, weshalb einer dieser alternativen Interaktionsansätze eine Verwendung von Gestensteuerung darstellt. In der Gestensteuerung gibt es ebenfalls verschiedene Ansätze. Hierbei ist eine berührungsbasierte Gestensteuerung, beispielsweise bei einem Einsatz von Touch Screens, von einer berührungslosen Interaktion zu unterscheiden. Berührungslose Gesten werden auch dreidimensionale Gesten genannt, da sie von Nutzern frei ausgeführt werden und sich somit im natürlichen, dreidimensionalen Raum befinden.

Die große Herausforderung in der gestenbasierten Interaktion besteht somit darin, Gesten sicher und zuverlässig zu erkennen, damit dem Nutzer eine gebrauchstaugliche Interaktion gewährleistet werden kann und von dieser Art der Steuerung nicht frustriert wird, wenn seine Kommandos nicht ausgeführt oder fehlinterpretiert werden.

Gestenbasierte Interaktion hält in vielen Bereichen Einzug; Darunter nicht nur wissenschaftlich, methodische Arbeiten hinsichtlich der Verbesserung von algorithmischen oder mathematischen Verfahren [16][24], sondern auch arbeitsunterstützende Szenarien, wie medizinische Bereiche [29][28] und Augmented Reality Anwendungen [13]. Des Weiteren ist die Interaktion durch Gestensteuerung besonders in der Unterhaltungsbranche weit fortgeschritten. In Home-Entertainmentsystemen [22] und vor allem in Spielen werden diese häufig eingesetzt. Durch den Erfolg der Konsolen Nintendo Wii, Sony Move and Microsoft Kinect wurde gezeigt, dass eine gestenbasierte Interaktion von vielen Personen angenommen wird [4].

Gestenerkennung ermöglicht eine weitere Dimension der Interaktion mit Interfaces. Die gestenbasierte Interaktion eröffnet viele Möglichkeiten, aber ebenso werden in der Umsetzung der gestenbasierten Interaktion viele Herausforderungen aufgeworfen.

1.1 Problemstellung

Die Interaktion mit Systemen mittels Gesten bietet viel Potential. Um das Potential einer solchen Interaktion jedoch nutzen zu können, braucht es eine verlässliche Erkennung der eingesetzten Gesten, welche von verschiedenen Nutzern verschieden ausgeführt werden können [20]. Gestenerkennung wird erschwert durch individuelle Interpretationsspielräume in Bezug auf die Gesten und physische persönliche Merkmale, wie Größe und Armlänge der gestenausführenden Nutzer. Ferner spielt der Abstand im dreidimensionalen Raum zwischen Nutzer und System eine entscheidende Rolle. Sogar die Drehung der Geste kann bei verschiedenen Ausführungen derselben Geste, sowie bei unterschiedlichen Personen sehr differenziert ausfallen. Daraus folgt, dass ein Erkennungssystem anhand eines begrenzten Trainingsgestensatzes einzelner Trainings-Gesten in der Lage sein muss, auch personenübergreifend Gesten erkennen zu können.
Die Differenzen zwischen den ausgeführten Gesten müssen demnach sorgfältig analysiert werden, um die Besonderheiten der Gesten in der Erkennung zu beachten. Die Intention dahinter ist, die Interaktion mit dem Computer, auf Basis vom Menschen bereits erlernter Interaktionsparadigmen, natürlicher zu gestalten [10].

Das gewählte Verfahren zur Gestenerkennung, das Hidden Markov Modell (HMM), birgt ebenfalls einige Hürden. Die Gesten müssen in einer bestimmten Art und Weise vorbereitet werden, damit sie mittels des HMMs klassifiziert werden können.
Das HMM kann auf sehr diffusen Daten nur ein sehr ungenaues Modell bilden, weshalb es wichtig ist, dass die Gestendaten vorverarbeitet werden. Diese Vorverarbeitung muss dafür sorgen, dass die Gesten vergleichbar werden und ihre besonderen Eigenschaften nicht verlieren und somit unterscheidbar bleiben. Vorteile der HMMs sind im Gegensatz zur Klassifikation eines „langen" Featurevektors, wie beispielsweise in Koch et al. durchgeführt, die Invarianz gegenüber zeitlicher Dauer, sowie Start- und Endzustand der Geste.

1.2 Zielsetzung

Ein häufig verwendetes Werkzeug zur Erkennung und Klassifizierung von Gesten sind die Hidden Markov Modelle. Aufbauend auf den Grundlagen der HMMs von [23], werden in dieser Arbeit verschiedene Features in den Fokus gestellt. In dieser Arbeit wird das Verfahren der Hidden Markov Modelle zur Erkennung von Gesten angewendet und auf seine Erkennungsgenauigkeit von personenübergreifenden Gesten, sowie die Nutzbarkeit für personenbezogene Erkennung anhand ihrer Gesten geprüft. In dieser Arbeit geht es um die Erkennung von Handbewegungen.

Der erste Schritt im Versuchsaufbau besteht deswegen in einer zielorientierten Datenerhebung, da für das Erstellen, Trainieren und Testen von Hidden Markov Modellen geeignete Testdatensätze benötigt werden. Es stehen zwei verschiedene Testdatensätze von Hall und Milani et al. zur Verfügung, welche betrachtet und auf Brauchbarkeit hin analysiert werden müssen. Für den Fall, dass diese für die Nutzung in HMMs als nicht geeignet eingeschätzt werden, müssen eigene Datensätze aufgezeichnet werden. Für diese Aufzeichnung von Testdatensätzen muss zuvor ein Satz an Gesten festgelegt werden, anhand derer die Methode der HMMs getestet wird.

Nach der Erhebung der Testdatensätze, sowie des Testens der HMM an den erhobenen Gestendaten, sollen Schwachstellen dieser Methodik der Gestenerkennung anhand ihrer unveränderten Positionsdaten analysiert und mögliche Verbesserungsansätze, durch beispielsweise Variation der Features, eingesetzt werden. Der Verbesserungsansatz der „Action Plane" nach [24] soll ebenfalls vorgestellt und geprüft werden. In den meisten themenverwandten Arbeiten wird eine sehr hohe Anzahl an Eigenschaften der Geste zur Erkennung und Klassifikation verwendet. In dieser Arbeit soll sich die Erkennung anhand weniger Features, welche durch Verbesserungsansätze verfeinert werden, getestet werden.

Eine weitere Fragestellung und somit ein weiteres taktisches Ziel ist, ob Personen anhand ihrer besonderen Gestenmerkmale bestimmt werden können. Es müssen Ansätze konzipiert und implementiert werden, welche diese besonderen Eigenschaften von Gesten hinsichtlich der möglichst zuverlässigen Erkennung der Geste an sich und der potentiellen Erkennung der Person einer Geste testen und dessen Erfolg oder Misserfolg darstellen können. Die Hypothese, dass Personen anhand ihrer Trajektorie der Geste im Phasenraum erkannt werden können, soll geprüft werden. Zusammengefasst ergeben sich daraus folgende zentrale Forschungsfragen (operative und taktische Ziele):

- Wie müssen Daten aufgezeichnet werden, damit eine Erkennung mittels HMM darauf trainiert und getestet werden kann?
- Wie funktioniert die Erkennung mittels HMMs für Gesten?
- Etablierung des Ansatzes „HMM für Gesten" in der Zielsprache Python an der FH Köln und Auslotung dessen Reichweite.
- Welche Vorverarbeitungen der Gestendaten und Verbesserungen des Trainings- und Klassifizierungsprozesses sind wichtig, um eine verbesserte Erkennungsrate zu erzielen (insbesondere im Hinblick auf Personenunabhängigkeit)?
- Bringt die Verwendung der „Action Plane" einen Vorteil in der Erkennung?

- Können Personen anhand ihrer Gesten-Trajektorie im Phasenraum erkannt werden?

Danach werden die zentralen Erkenntnisse zusammengefasst und ein Fazit gezogen. Das strategische Ziel dieser Arbeit ist, wichtige Erkenntnisse für die Verbesserung der Gestenerkennungsverfahren zu gewissen und für den Einsatz in Natural User Interfaces zu verbessern.

1.3 Aufbau der Arbeit

Im weiteren Verlauf der Arbeit wird aufbauend auf den Thematiken aus Problemstellung und Zielsetzung ein Überblick über den Stand der Wissenschaft der verschiedenen Aspekte gegeben. Zu diesen Aspekten gehören der aktuelle Stand der Gestenerkennung, aktuelle Ansätze und Anwendungsmöglichkeiten mittels Hidden Markov Modellen und aktuelle Einsatzgebiete gestenbasierter Interaktion.

Nach dem Überblick des aktuellen Stands der Wissenschaft werden in Kapitel 3 die Grundlagen der wichtigen Aspekte dieser Arbeit beleuchtet. Zuerst werden in 3.1 die grundlegenden Eingenschaften und Besonderheiten von Gesten im Kontext des Natural User Interfaces geklärt. Danach wird in 3.2 auf die Anwendung von Hidden Markov Modellen im Kontext von Gestenerkennung eingegangen. In diesem Kapitel wird die allgemeine Vorgehensweise von HMMs vorgestellt und die einzelnen Bestandteile im Kontext von Gestenerkennung dargelegt.

Im Kapitel 4 wird auf den Grundlagen und dem Wissen, wie die Gestenerkennung mittels HMM abläuft, aufgebaut und zunächst in 4.1 eine Analyse bereits existenter erhobener Datensätze durchgeführt. Es soll bestimmt werden, welche Gesten für die Erkennungstests geeignet sind, sowie welche Features in den Daten enthalten sein müssen.
Danach wird vorgestellt, welche Vorbereitungen getroffen werden müssen, um Gesten zu verarbeiten. Dazu gehören die in 4.2 vorgestellten Aspekte. Darunter fallen die Auswahl der erhebbaren Daten, welche für die aufgenommenen Gesten gespeichert werden müssen, welche Datenaufbereitung dafür vorgenommen werden muss und eine für die Erkennung wichtige „Featuregeneration". Features sind die essentiellen Eigenschaften, anhand welcher die Gesten im Anschluss erkannt werden. Nach der Bestimmung und Diskussion wichtiger Features werden die Vorbedingungen der Daten beziehungsweise Eigenschaften von erhobenen Gestendaten zum Einsatz mittels HMMs diskutiert. Nach der Analyse und der darauf folgenden Auswahl an Gesten 4.3, welche im weiteren Verlauf aufgezeichnet werden sollen, wird das Vorgehen beim Aufzeichnen und Aufnehmen mit Testpersonen in 4.3.3 beschrieben. Zum Vorgehen gehört die Auswahl an

verwendeten Hardware- und Softwarekomponenten 4.3.2 und wie die Gesten von den Testpersonen aufgezeichnet wurden. Im Anschluss folgt eine Analyse, sowie Diskussion der aufgezeichneten Daten.
Im Kapitel der Gestenerkennung 5 werden die eingesetzten Mittel und Methoden, um die Gestendaten zu verarbeiten und sie für die Erkennung mittels HMM vorzubereiten in 5.1 dargelegt. Als Ausgangspunkt für die Erkennung werden die Originaldaten aus der Aufzeichnung mittels der Kinect für die HMM aufbereitet und ausgewertet. Anhand dieser Ergebnisse werden weitere Ansätze zur Verbesserung der Erkennung der Gesten entwickelt. In den folgenden Kapiteln werden diese Verbesserungsansätze vorgestellt und diskutiert. Diese Verbesserungsansätze werden auf den originalen Features, des Positionsdaten angewendet und die Ergebnisse dargestellt und diskutiert.

Darauf folgen weitere Featureansätze in den Kapiteln 5.5 und 5.6, welche auf den originalen, von der Kinect aufgezeichneten Daten aufbauen. Die in diesem Kapitel dargelegten Featureansätze zielen nicht nur auf die bessere Erkennung von personenübergreifenden Gesten, sondern auch auf die Erkennung einer Person anhand ihrer durchgeführten Geste. Aus dieser Überlegung resultiert vor allem die Betrachtung des in 5.10 dargelegten Phasenraums von Gesten. Unter den beschriebenen Featureansätzen befinden sich somit die Berechnung der Geschwindigkeiten 5.7 (eindimensionaler Raum), ein Ansatz zur Kombination der aus den Positionsdaten erstellten HMMs mit den aus den Geschwindigkeiten resultierenden HMMs 5.8, sowie die Bildung des Phasenraums einer Geste (Positions- und Geschwindigkeitsdaten einer Geste: sechs-dimensionaler Raum) 5.10. Über die verschiedenen Ansätze werden Hypothesen aufgestellt, welche im folgenden Kapitel getestet werden.

Im Kapitel 6 wird die Durchführung der Tests der vorangegangenen Hypothesen der verschiedenen Featureansätze dargestellt. Bei den Tests bezüglich des Phasenraums wird nicht nur die Erkennung personenübergreifender Gesten analysiert, sondern ebenfalls die Erkennungsrate einer Person anhand ihrer Geste. Eine der zentralen Hypothesen diesbezüglich ist, dass anhand des Phasenraums eine Person anhand ihrer Geste erkannt werden kann.
Im Anschluss werden die Ergebnisse in einem Fazit in Kapitel 7 zusammengefasst und ein Ausblick in 8 gegeben.

2 Stand der Wissenschaft

Gestenerkennung ist bereits ein weit verbreitetes Forschungsgebiet. Es existieren viele verschiedene Ansätze und Umsetzungen, Gesten in einem dreidimensionalen Kontext aufzuzeichnen und zu erkennen. Je nach Kontext der Gestenerkennung werden die Bewegungen des ganzen Körpers aufgezeichnet, oder nur bestimmte Gliedmaßen. Hand-Arm-Gestenerkennung wird häufig für das Erstellen oder Erkennen von Zeichensprache verwendet [26]. Ein anderes Anwendungsgebiet wird von Oh et al. beschrieben, in dem die berührungslose Gestensteuerung dazu genutzt werden soll, um den Komfort des alltäglichen Lebens zu erhöhen. Viele Anwendungsgebiete fallen in den Bereich der Unterhaltung. In der Arbeit von Soltani et al. wird dargestellt, wie der Einsatz von berührungslosen Gesten und der Kinect, bei der Entwicklung eines Spiels für taub-stumme Menschen verwendet werden kann. Zur Klassifizierung der Gesten werden verschiedene mathematische Ansätze verwendet. In der Arbeit von Koch et al. wird die Slow Feature Analyse benutzt, um Gesten mit wenigen Trainingsdaten personenunabhängig zu klassifizieren. Andere Ansätze gehen mit Dynamic Time Warping vor, um die Gemeinsamkeiten von zwei Signalen verschiedener Länge herauszufiltern [14]. In der Arbeit von Milani et al. werden dreidimensionale Gestendaten mittels Kinect aufgezeichnet, aus denen im weiteren Verlauf zusätzliche Features berechnet werden. Mit diesen Featurevektoren (90 Features pro Frame[1]) werden verschiedene Klassifizierer trainiert. Ziel ist die personenunabhängige Erkennung anhand eines limitierten Anfangsset an Gesten und einer Erweiterung durch vorher unbekannten Personen mit weiteren Trainingsgesten. In der Arbeit von Richarz and Fink wird ein Hidden Markov Modell-basiertes Gestenerkennungsverfahren vorgestellt, in dem die Featuregeneration mittels der Action Plane durchgeführt wird. Diese Action Plane bestimmt die Hauptebene, in der die Geste abläuft, um die einzelnen Gesten und ihre Erkennung personen- und ortsunanbhäniger zu gestalten. In [20] wird die Action Plane ebenfalls verwendet. Diese Action Plane soll in der Featuregeneration dieser Arbeit verwendet und getestet werden. Um ein HMM zu trainieren, ist die Wahl der Features besonders wichtig. Diese müssen in den Kontext der jeweiligen Erkennung passen. In der Arbeit von Caramiaux et al. werden beispielsweise als wichtige Eigenschaften die Geschwindigkeiten der Geste in Kombination mit Audioeigenschaften betrachtet. Die Arbeit zielte darauf ab,

[1] siehe „Framerate" im Abkürzungsverzeichnis.

Gemeinsamkeiten zwischen Gesten und auditiven Einflüssen zu untersuchen. In den meisten Arbeiten, die Hidden Markov Modelle verwenden, werden sehr viele Features zur Beschreibung der Gesten genutzt. Diese Arbeit grenzt sich davon ab, in dem sie auf nur wenigen Features eine Erkennung aufbaut. Es werden verschiedene Ansätze mit minimal drei bis maximal sechs beschreibenden Features (Kombinationen aus Ortskoordinaten und Variationen der Geschwindigkeiten) pro Modell getestet.

3 Grundlagen

Zuerst wird der Kontext der Mensch-Computer Interaktion unter dem Aspekt der Interaktionsgestaltung vorgestellt. Dieses interdisziplinäre Forschungsfeld beschäftigt sich mit der Gestaltung der Beziehung zwischen Mensch und Computer.
Danach werden in diesem Kapitel die Grundlagen und Vorbedingungen für eine Gestenerkennung, insbesondere durch Hidden Markov Modelle, erläutert. Eine wichtige Grundlage ist die Betrachtung der Ausprägungsmöglichkeiten und Eigenschaften von Gesten. Diese werden im Zusammenhang mit Natural User Interfaces betrachtet. Dazu gehört ebenfalls, wie Gesten ablaufen können und welche Unterschiede es dabei geben kann.
Des Weiteren ist die Erhebung beziehungsweise Nutzung von Gestendaten für ein Gestenerkennungsverfahren grundlegend, weshalb vor der Konzeption einer eigenen Datenerhebungen, beziehungsweise eines eigenen Erkennungsverfahrens, eine vorbereitende Datenanalyse durchgeführt wurde. Diese vorbereitende Datenanalyse beinhaltet das exemplarische Analysieren von vorhandenen Gesten-Datensätzen. Aufbauend auf diesen Erkenntnissen wird das Vorgehen der Erhebung von eigenen Daten, sowie die eigene Datenstruktur konzipiert und beschrieben.
Danach wird auf wichtige Aspekte der Datenvorverarbeitung für die Gestenerkennung eingegangen. Dazu gehören eine Datenaufbereitung, sowie Featuregeneration aus den aufgenommenen Gestendaten. Anschließend wird die Methode der Hidden Markov Modelle im Hinblick auf die Erkennung von Gesten erläutert.

3.1 Gesten im Natural User Interface

Wie bereits beschrieben geht es in dieser Arbeit um gestenbasierte Interaktion, welche im Bereich der Mensch-Computer Interaktion angesiedelt ist. In diesem Bereich geht es um die Ausgestaltung der Beziehung zwischen Mensch und Maschine, welche durch die Kommunikation, beziehungsweise Interaktion des Menschen mit dem Computer, erstellt und durchgeführt wird. Ein Ziel in der MCI ist es, eine möglichst gebrauchstaugliche Interaktion zur Verfügung zu stellen. Dies bedeutet, dass ein Mensch eine Anwendung effektiv und effizient nutzen kann und es an seine kognitiven Denkmuster angepasst ist. Ein Ansatz, diese Natürlichkeit einzubringen ist, möglichst natürliche Interaktionsstile

und -modi für eine Anwendung zu entwickeln.
Gesten gelten als natürliche Art der Interaktion, da Menschen verschiedene Arten von Gesten, beziehungsweise bedeutsame Handbewegungen, in ihrem alltäglichen Leben benutzen. Nach Billinghurst können Gesten anhand ihrer Funktion klassifiziert und wie folgt gruppiert werden:

semiotic Gesten dieser Gruppe werden verwendet, um bedeutende Informationen zu kommunizieren.

ergotic Diese Gesten werden benutzt, um die physische Umgebung zu manipulieren und Artefakte zu kreieren.

epistemic Epistemische Gesten werden genutzt, um von der Umgebung durch taktiles oder haptisches Erforschen.

Im Kontext der Natural User Interfaces ist es wichtig zu analysieren, welche Gesten zur Interaktion zwischen Menschen und Computern genutzt werden können, da NUIs auf die bereits erlernten Fähigkeiten von Menschen zurückgreifen sollen. Diese Art der kommunikativen Interaktion wird hauptsächlich durch semiotische Hand-Gesten abgebildet [3].
Zusätzlich zu der Hand-Arm-Gestenerkennung gibt es ebenfalls Ganzkörper-Gestenerkennung, welche auch „Aktionserkennung" genannt wird [26].
Diese Erkennungsarten sind vor allem durch die Möglichkeiten der rasanten Technologieentwicklung entstanden. Techniken zur Erfassung von Körperbewegungen können auf Hände, sowie auf andere Gliedmaßen gleichermaßen angewendet werden. Aus dem Spielebereich, in welchem komplexere Bewegungen ebenfalls sehr wichtig sind, wie beispielsweise sport-typische Bewegungen (Schwingen eines Golfschlägers, Treten eines Balls oder Rollen einer Bowlingkugel u.v.m.), rührt ebenfalls das Interesse sowie die Erkenntnis der Notwendigkeit, komplexe Gesten und Bewegungen erkennen zu können.
Im Gebiet der gestenbasierten Interaktion gibt es bisher kaum Standards. Bisher ist in der ISO/IEC 14754 ein grundlegendes Set von Gesten, beziehungsweise gestikulierbaren Kommandos, und Feedback für stiftbasierte Interfaces definiert. Dieses Set an Gesten beinhaltet „select", „delete", insert space", „split line", „move", „copy", „cut", „paste", „undo" und „scroll". Mitglieder der ISO erkannten die Wichtigkeit eines Standards für gestenbasierte Interfaces, weshalb Mitglieder des „Joint Technical Committee" bereits an einem allgemeineren Standard für gestenbasierte Interfaces arbeiten [6].

3.2 Hidden Markov Model für die Anwendung bei Gesten

Sollen Gesten aufgenommen werden, während diese ausgeführt werden, müssen dabei die beschreibenden Eigenschaften der Geste, auch Features genannt, aufgezeichnet werden. Diese Eigenschaften formen eine vektorbasierte Eigenschafts-Zeitserie, welche als Gestensample bezeichnet werden kann [14].
Das Hidden Markov Model ist ein häufig genutztes Werkzeug zur Gestenerkennung [21], da es vor allem eben diese zeitlich und eigenschaftsbasierten, abhängigen Informationen modellieren kann. Das Modell beschreibt eine Sammlung von Zuständen, welche durch bestimmte Übergänge miteinander verbunden sind. Jeder Übergang besitzt gewisse Wahrscheinlichkeiten; Darunter die Wahrscheinlichkeit in diesem Übergang zu bleiben, oder in den nächsten Zustand zu wechseln, sowie die Wahrscheinlichkeit, welches konkrete „Symbol", eines festgelegten Alphabets, in diesem gewissen Übergang beobachtet wird [16]. Zusammen ergeben diese Komponenten das Modell, welches eine vollständige Gestenbeschreibung, die eine Wiedererkennung, der von einem Menschen ausgeführten Bewegung, als eine Geste ermöglichen soll. Je nach Beschaffenheit der Zustände werden diskrete und kontinuierliche HMMs unterschieden. In dieser Arbeit werden diskrete Zustände betrachtet.

Die formale Definition des HMMs ist:

$$\lambda = (A, B, \pi) \tag{3.1}$$

Die Bestandteile des in 3.1 definierten Hidden Markov Modells sind:

Originale Beobachtungen Die originalen Beobachtungen sind die Rohdaten, welche beispielsweise mit der Kinect aufgezeichnet wurden und zur Gestenerkennung mittels HMMs vorverarbeitet werden müssen. Diese müssen in beobachtete Zustände eingeteilt werden, aus welchen die „Hidden States" gebildet werden [5]. Diese originalen Beobachtungen sind im Bezug auf die hier beschriebene Gestenerkennung beispielsweise Raumkoordinaten zu einer bestimmten Zeit. Mit einer Bildfrequenz von 30 fps werden durch die Kinect die Koordinaten der Gliedmaßen aufgezeichnet. Jeder Beobachtung muss einem Symbol aus einem festgelegten Alphabet an „Symbolen" zugewiesen werden [23]. Die Anzahl der beobachteten Zustände ändert sich dadurch nicht, lediglich die Komplexität der Beschreibung beobachteter Zustände (die Komplexität wird geringer). Die originalen Beobachtungen sind wie folgt aufgebaut und besitzen alle unterschiedliche Längen. Jeder Vektor enthält die zu diesem Zeitpunkt aufgenommenen Eigenschaften/Features einer Geste.

$$B_{timestamp} = [\vec{a}_1, \vec{a}_2, \vec{a}_3, ..., \vec{a}_z], z \in \mathbb{N} \tag{3.2}$$

Beobachtete Zustände Den originalen Beobachtungen wird ein festgelegtes Alphabet an Symbolen (auch „Output Symbol" genannt) mit der Anzahl N zugewiesen [23]. In einem mehrdimensionalen Feature-Raum muss demnach ein passendes Zustands-Alphabet definiert werden. In einem Beispiel mit Positionsdaten als originale Beobachtungen, wird das Alphabet mit Hilfe von Clusterzentren in der beobachteten Punktwolke festgelegt [11] (dies wird ebenfalls bei den in dieser Arbeit vorgestellten, vier- und sechsdimensionalen Feature-Räumen durchgeführt). Die Anzahl der Symbole wird durch die Anzahl der bestimmten Clusterzentren widergespiegelt und jeder original beobachtete Zustand wird durch sein nächstes Cluster Zentrum beschrieben. Diese Zentren haben nur noch eine abstrakte Referenz. Die übliche Vorgehensweise ist, die Zentren zu nummerieren. Sind die aufgenommenen Positionsdaten ihren nächsten Zentren zugewiesen, existiert nur noch eine Liste der Abfolge der Clusterzentren, wie diese im Verlauf der Geste durchlaufen wurden.

$$S = [s_1, s_2, s_3, ..., s_M], M = \text{Anzahl der Output Symbole} \tag{3.3}$$

Die beobachteten Zustände werden ihren zugehörigen Observation Symbols aus S zugeteilt (durch das Zuordnen zum nächsten Clusterzentrum), so dass diese aus einer eindimensionalen Liste O entsprechen.

$$O = [o_1, o_2, o_3, ..., o_m], m \in \mathbb{Z} \tag{3.4}$$

Versteckte Zustände (engl. „hidden states") sind eine abstrakte Darstellung des Ablaufs einer Geste, nachdem das HMM trainiert wurde. Ein „Hidden State" beinhaltet die Wahrscheinlichkeiten über das Auftreten eines bestimmtes „Symbols", bzw. beobachtbaren Zustandes, an einem bestimmten Hidden State.

$$V = [v_1, v_2, v_3, ..., v_N], N = \text{Anzahl Hidden States} \tag{3.5}$$

Initiale Zustandswahrscheinlichkeit π beschreibt mit welcher Wahrscheinlichkeit, welcher beobachtbare Zustand im ersten „hidden state" vorhanden ist. Die initiale Zustandswahrscheinlichkeit kann beliebig gewählt werden. Wird π jedoch von Anfang an optimal gewählt, erfordert es weniger Rechenzyklen, um sich der echten initialen Zustandswahrscheinlichkeit zu nähern.

Transitionsmatrix A beschreibt die Übergangswahrscheinlichkeiten der bestimmten Hidden States. Sie hat eine Größe von NxN.

$$A = \begin{pmatrix} a_{11} & a_{12} & ... & a_{m1} \\ a_{21} & a_{22} & ... & a_{2m} \\ ... & ... & ... & ... \\ a_{n1} & a_{n2} & ... & a_{nm} \end{pmatrix} \tag{3.6}$$

Beispielsweise beschreibt a_{12} die Wahrscheinlichkeit des Zustands a_1, in den Zustand a_2 zu wechseln.

Emissionsmatrix B beschreibt die Wahrscheinlichkeit, ein bestimmtes „Symbol" an einem bestimmten Hidden State aufzufinden. Deshalb hat die Matrix B eine Länge von NxM.

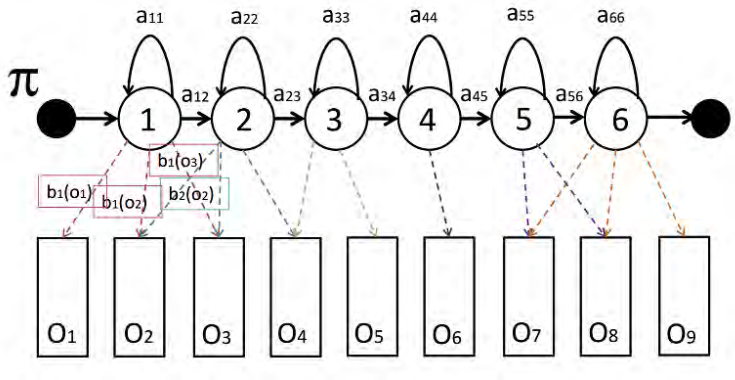

Abbildung 3.1: Visualisierung eines Left-to-Right Hidden Markov Modells.

Die Zustände a_{ij} sind, wie bereits beschrieben und in 3.1 dargestellt, die Hidden States des HMMs. Diese können entweder zum nächsten Zustand überwechseln, oder bestehen bleiben. Jedem Hidden State sind verschiedene beobachtete Zustände $O_M(M =$ Anzahl der festgelegten Output Symbole) zugeordnet. In der Emissionsmatrix B sind die Wahrscheinlichkeiten dieser beobachteten Zustände zu einem bestimmten Hidden State hinterlegt.

Die Transitionsmatrix A und die Emissionsmatrix B werden trainiert. Training bedeutet, dass man den Hidden State, auf den die beobachtbaren Werte mit einer gewissen Wahrscheinlichkeit zurückfallen, nicht kennt. Die beobachteten Zustände sind bekannt, jedoch nicht, mit welcher Wahrscheinlichkeit sie zu welchem Hidden State gehören. Diese beobachteten Zustände werden nun anhand ihrer Eigenschaften analysiert und versucht ein bestmögliches Modell für die Hidden States zu rekonstruieren, so dass alle beobachteten Werte anhand dieses Modells korrekt klassifiziert werden können.

Es gibt verschiedene Transitionsmodelle. Kann in der Transitionsmatrix jeder Zustand von jedem Zustand aus erreicht werden, wird das Modell „ergodic" genannt [23]. Die Darstellung in Abbildung 3.1 ist ein Left-to-Right Modell, in dem jeder Zustand nur in sich selbst verweilen oder zum nächsten Zustand wechseln kann.

Das Modell enthält zwei Annahmen. Die erste (Markov Annahme) besagt, dass der momentane Zustand nur vom vorhergegangenen Zustand abhängig ist [5].

$$P(q_t|q_1^{t-1}) = P(q_t|q_{t-1}) \tag{3.7}$$

Die zweite Annahme besagt, dass eine Beobachtung zum Zeitpunkt t, nur vom momentanen Zustand abhängig ist und somit unabhängig von vorherigen Zuständen oder Beobachtungen ist [5].

$$P(o_t|o_1^{t-1}, q_t^1) = P(o_t|q_t) \tag{3.8}$$

Es gibt nach Rabiner drei Hauptprobleme, welche bei der Nutzung eines HMMs gelöst werden müssen:

Evaluation Für eine gegebene Sequenz an beobachteten Symbolen und dem Modell $\lambda = (A, B, \pi)$ gilt es die Wahrscheinlichkeit, das Likelihood, von $P(O|\lambda)$ mit dem gegebenen Modell zu bestimmen. Zur Lösung dieses Problems wird der Forward-Algorithmus verwendet [23].

Decoding Mit einem gegebenen Modell $\lambda = (A, B, \pi)$ und einer beobachteten Sequenz O muss eine optimale Zustandssequenz für den darunterliegenden Markov-Prozess bestimmt werden. Anders ausgedrückt soll hiermit der „Hidden" Aspekt des Hidden Markov Modells aufgedeckt werden. Dieses Problem wird mit dem Viberi-Algoritmus angegangen [23].

Learning Dieses Problem kann als das „Trainieren" der HMM beschrieben werden. Mit der gegebenen Beobachtungssequenz O und der Anzahl der beobachteten und versteckten Zustände, soll ein Modell gefunden werden, welches die Wahrscheinlichkeit des Auftretens von O maximiert. Dies, der schwierigste Part der HMM, wird mit dem Baum-Welch Algorithmus gelöst [23].

Die Ausgabe des HMMs ist eine Log-Likelihood-Wahrscheinlichkeit darüber, wie gut die getestete Geste auf das trainierte Modell passt. Das trainierte Modell besitzt einen „Threshold", die Grenze, über der alle Werte als „erkannt" klassifiziert werden. Dieser Threshold ist ebenfalls durch eine Log-Likelihood-Funktion, bei dem die Trainingsgesten auf dem Trainings-HMM getestet werden, berechnet. Dadurch entsteht der LLH der

Trainingsgesten auf der Trainings-HMM, welcher mit einem Faktor t (welcher initial auf $t = 2$ gesetzt wird) multipliziert wird, um einen Threshold zu bestimmen, welcher mehr Erkennung zulässt. Dies ist notwendig, da andere Gesten eine kleinere LLH für das Modell haben, als die Trainingsgesten selbst, aber trotzdem die gleiche Geste darstellen.

$$\theta_t = t * \frac{1}{n} \sum_{j=1}^{n} lik(j) \tag{3.9}$$

Der genaue Lösungsvorgang der drei Hauptprobleme kann in der Arbeit von Rabiner oder Blunsom nachgelesen werden.

4 Vorbedingungen und Datenerhebung

Im folgenden Kapitel werden die notwendigen, vorbereitenden Schritte vor der Durchführung der Gestendatenaufnahmen zur Schaffung eines Datenpools für Trainings- und Testdaten, beschrieben. In die Vorbereitungen fließen Erkenntnisse aus der vorbereitenden Gestendatenanlyse 4.1 mit ein, von denen angenommen wird, dass die den Gesamtprozess der Gestenerkennung grundlegend verbessern können. Zum Schluss des Kapitels werden die Beobachtungen bei der Ausführung der Bewegungen durch die Testpersonen beschrieben und analysiert, um einen ersten Eindruck über die Verwendbarkeit der Daten zu erhalten.

4.1 Vorbereitende Gestendatenanalyse

Mit dem Thema der (personenunabhängigen) Gestenerkennung haben sich insbesondere zwei zentrale Arbeiten ([11], [20]) beschäftigt, welche Datensätze zur Verfügung stellen.
Die Datenformate beider Datensätze sind ähnlich, nicht identisch. Dazu ist anzumerken, dass die Testdaten von [11] ausschließlich Gesten einer einzigen Person umfassen, wohingegen der Datensatz von [20] verschiedene Gesten von 18 verschiedenen Personen erfasst hat.
Die Datenstruktur von Hall beinhalten in verschiedenen .csv-Dateien die X, Y und Z-Werte der Gesten. Da diese lediglich von einer Person aufgenommen wurden, sind diese nicht sehr aussagekräftig und repräsentativ, aber in Kombination mit anderen Gestendaten eventuell vergleichbar was die Ausführung und Besonderheiten einer einzelnen Geste betrifft. In [11] wurden die X, Y und Z-Werte (räumliche Koordinaten) einer einzigen Hand zu jeder Geste aufgezeichnet, wohingegen in den Daten von Milani et al. zusätzlich dazu jeweils noch die Koordinaten der Kopfposition gespeichert haben. Diese Koordinaten können dazu genutzt werden, um die Gesten ortsungebunden, relativ zum Kopf, auszuwerten.

In den Hall-Daten kann die Geste anhand der Visualisierung gut erkannt werden (siehe 4.3 und 4.2). Je nach Visualisierungsebene (Ebene auf welcher die dreidimensionalen

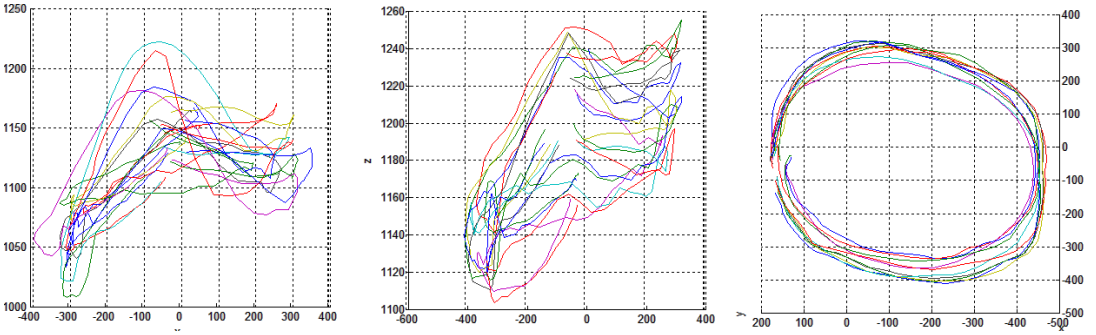

Abbildung 4.1: Hall Testdaten: 1. Alle verwendeten „Trainingsdaten" für die Geste „Circle" aus der Sicht auf die XZ-Ebene; 2. Verwendete Testdaten zur Erkennung der Geste „Circle" aus der Sicht auf die XZ-Ebene; 3. Testdaten „Circle" mit der Ansicht auf die XY-Ebene [11]

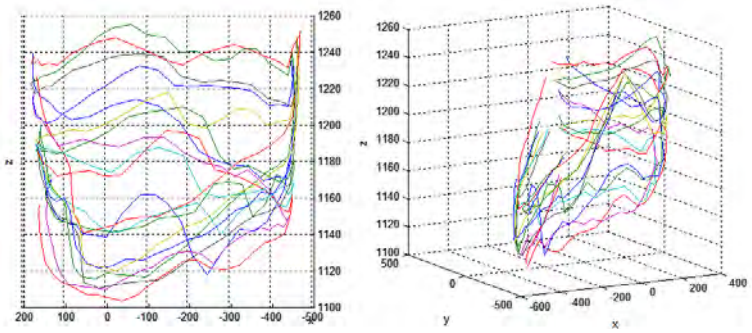

Abbildung 4.2: Hall Testdaten: 1. Ansicht auf die Testdaten „Circle" aus einer Sicht auf die XZ-Ebene; 2. Gedrehte Ansicht auf die Testdaten von „Circle" [11]

Daten zur zweidimensionalen Ausgabe auf dem Bildschirm umgesetzt werden[1]) kann wahrgenommen werden, welche Probleme beim Auswerten auftreten können. Durch die andere Skalierung der Z-Achse (weniger Werte auf der gleichen Achsenlänge), im Vergleich zu der X- und Y-Achse, wirken Gesten in der Visualisierung sehr ungenau, sind es jedoch nicht (siehe Abbildung 4.1). Aufgezeichnete Gestendaten von Personen liegen nicht perfekt übereinander; Es treten immer geringfügige bis starke Schwankungen auf.

Die Daten von [20] sind umfassender als die Hall-Daten. Die Eckdaten der Analyse sind:

[1] Das Umsetzen der Visualisierung von dreidimensionalen Punktwolken auf dem Bildschirm erfordert eine Transformation der Punkte auf eine reduzierte, zweidimensionale Ausgabeform, da der Bildschirm nur zweidimensional darstellen kann. Je nach umgesetztem „Blickwinkel" auf die dreidimensionale Punktwolke ergibt sich eine differenzierte, zweidimensionales Darstellung. Der Effekt der Dreidimensionalität kann durch Ermöglichen des Drehens der Visualisierung bereitgestellt werden.

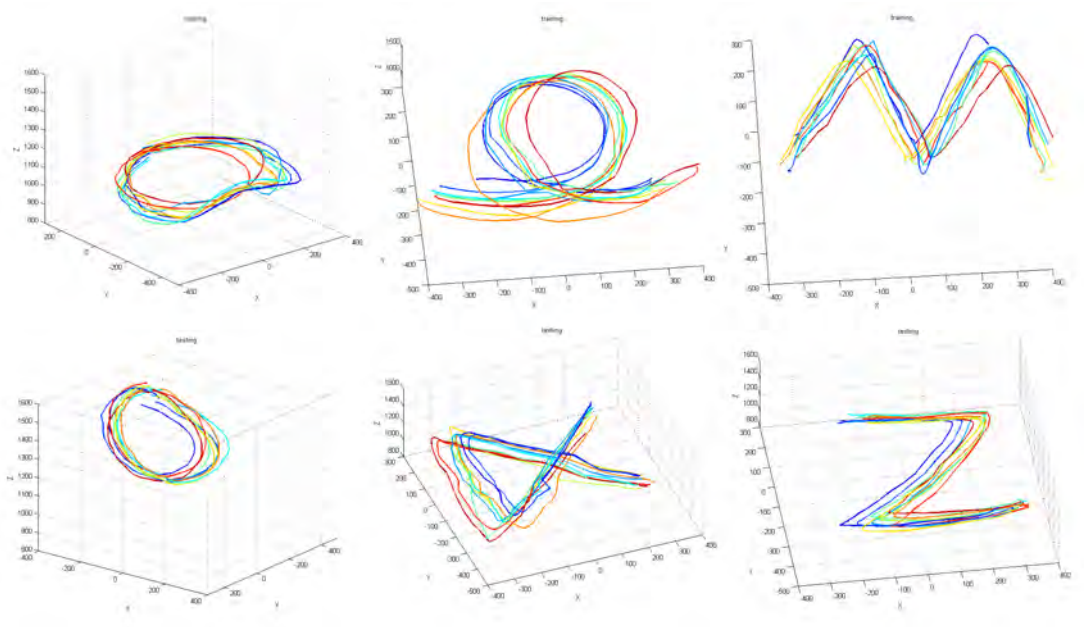

Abbildung 4.3: Alle von Hall verwendeten Gesten [11].

- Es wurden Daten von 18 Personen erfasst.
- Jede Person führte 9 Gesten jeweils mit Hand=„left" und „right" (circle, come here, down, go away, point, stop, up, vertical wave, wave) durch.
- Die aufgezeichneten Eigenschaften der Gesten beinhalten 6 Datenspalten mit den Ortskoordinaten der ausführenden Hand (x, y, z) und des Kopfes der Person (x, y, z).
- Eine Geste durfte mehrfach wiederholt werden.
- Mit der linken Hand gab es:
  - 927 Gesteninstanzen insgesamt.
  - Es wurden 1-24 Gesteninstanzen durchgeführt. Der Mittelwert liegt bei 5,7 Instanzen.
  - Die Anzahl an Frames[2] pro Geste variiert stark. Minimal wurden 9 Frames aufgezeichnet und maximal 151. Der Mittelwert der Gesten liegt zwischen 17,8 Frames (für die Geste „point") und 30 Frames (für die Geste „vertical wave").
  - Mittelwerte der Frames pro Person liegen zwischen 14,2 (Person 17) und 48,1 (Person 18).

[2] siehe „Framerate" im Abkürzungsverzeichnis

- Mit der rechten Hand gab es:
  - 936 Gesteninstanzen insgesamt
  - Es wurden 1-25 Gesteninstanzen durchgeführt. Der Mittelwert liegt bei 5,8 Instanzen.
  - Die Anzahl an Frames pro Geste variiert stark. Minimal wurden 9 Frames aufgezeichnet und maximal 64. Der Mittelwert der Gesten liegt zwischen 18,5 Frames (für die Geste „go away") und 27,9 Frames (für die Geste „vertical wave").
  - Mittelwerte der Frames pro Person liegen zwischen 15,4 (Person 17) und 35,9 (Person 18).

Durch die Analyse der Daten wird deutlich, dass diese mit einer geringen Frequenz an aufgezeichneten Bildern pro Sekunde aufgezeichnet wurden, so dass manche Gesten nicht vollständig aufgezeichnet wurden. Es fehlen Abschnitte innerhalb der Geste, sowie am Anfang oder Ende der Gestenausführung. Dazu kommt, dass bei manchen Gesten sogar zu viele Bilder am Anfang oder Ende der Geste aufgenommen wurden, so dass die Geste fälschlicher Weise in eine Richtung verlängert wurde. Der hohe Abstand zwischen den Trajektorienpunkten verursacht ebenfalls eine ungenaue Clusterbestimmung und verhindert das zuverlässige Berechnen von weiteren Features, wie beispielsweise der Geschwindigkeit (wäre eine Aufzeichnung der Zeit zu jedem Frame vorhanden). Anhand einer Visualisierung dieser Daten, kann teilweise nicht erkannt werden, um welche Geste es sich handelt. Diese geringe Frequenz beeinflusst somit ebenfalls maßgeblich den Erkennungsprozess, da die Gesten durch viele „Falschwerte" verzerrt sind und zu wenig Datenpunkte der eigentlichen Geste vorhanden sind. Dazu ist zu beachten, dass Gesten mehrfach durchlaufen werden durften und somit jede Gestenaufzeichnung sehr komplex ist.

In Abbildung 4.4 ist dargestellt, wie sich die geringe Bildfrequenz auf eine einzelne Geste (1.) auswirken kann. Die Geste „Circle" ist durch die geringe Anzahl der Messpunkte kaum erkennbar. Zudem wurde die Geste schlecht segmentiert. Dies bedeutet, dass noch Fragmente von „Nicht-Gesten"-Phasen der Ausführung in den Daten vorhanden sind, sowie eventuell zu wenige Daten, was bedeutet, dass die Geste nicht vollständig abgeschlossen wurde.
In der zweiten Darstellung in Abbildung 4.4 sind alle „Circles" aller Testpersonen dargestellt. An dieser Visualisierung kann deutlich erkannt werden, dass die Gestendaten ebenfalls sehr unterschiedlich ausfallen. Die Daten von Milani et al. werden deshalb im weiteren Verlauf dieser Arbeit nicht verwendet. Im weiteren Verlauf wurden eigene Gestendaten aufgezeichnet, da die Daten von [11] nicht repräsentativ sind und die Daten

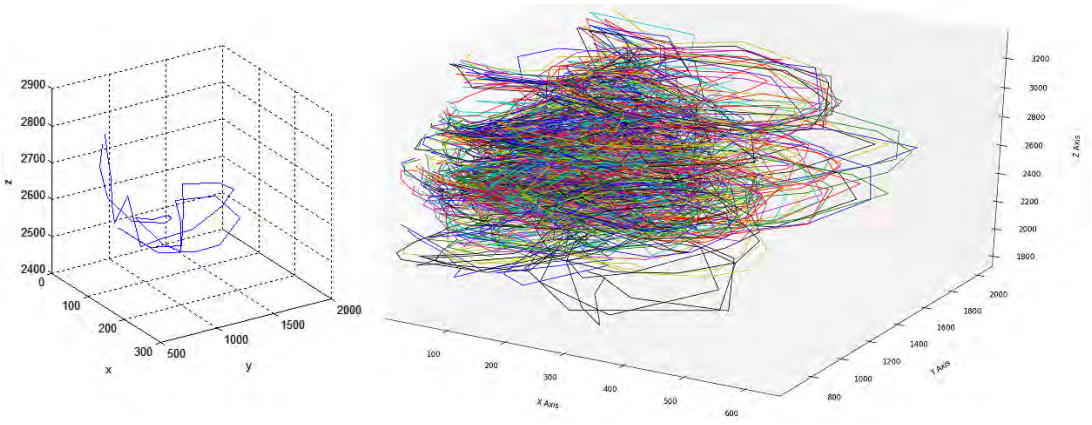

Abbildung 4.4: Milani Testdaten: Visualisierung eines „Circles" (links); Visualisierung aller „Circles" aller Testpersonen (rechts) [20]

von [20] eine sehr geringe Framerate[3] aufweisen und somit für die weitere Bearbeitung des Themas der Gestenerkennung verworfen werden.

4.2 Datenaufbereitung und Featuregeneration

Um das Modell einer Geste erstellen zu können, beziehungsweise die Geste anhand des Modell erkennen zu können, muss entschieden werden, welche Eigenschaften der Geste, in welcher Ausprägung und in welchem Format hierfür verwendet werden sollen. Ist entschieden, welche Daten und welche Form zu speichern sind, werden Gesten von echten Nutzern, auf diese Eigenschaften hin, aufgenommen. Geht es beispielsweise um die Bewegung der Hand während des Winkens, können die Positionen, der Winkel des Arms, die Entfernung zum Kopf, die Geschwindigkeit in den einzelnen Positionen und die Schnelligkeit des Umkehrens nach einem Schwung interessant sein. Ein geeignetes Aufnahmegerät wird gewählt und die Daten aufgenommen und in einem geeigneten Format gespeichert. Bei der Aufnahme können Stördaten, Schwankungen und andere Anomalien auftreten, die beim Gestenerkennungsprozess verzerrend oder behindernd sind.

Dazu kommt, dass manche interessanten Eigenschaften (engl. Feature) einer Geste nicht direkt durch das Aufnahmegerät aufgezeichnet werden können. Ist beispielweise die Geschwindigkeit der Geste in den einzelnen Ausführungspositionen zentral, so muss diese meist anhand der Positions und Zeitdaten errechnet werden. Die WiiMote liefert beispielsweise Beschleunigungsdaten, welche auch in Geschwindigkeiten umgerechnet

[3] Anzahl der Aufgezeichneten Frames (Featurevektoren) pro Sekunde

werden müssten (Dies ist allerdings numerisch nur sehr schwierig durchführbar. Noch schwieriger(praktisch fast unmöglich) ist es, auf die Positionsdaten zurückzukehren.), wenn diese für die Geste interessant wären. Dieser Vorgang wird Featuregeneration genannt. Im Folgenden werden Gründe und Ansätze der Datenaufbereitung und Featuregeneration vorgestellt.

Segmentierung der Gesten

Werden Datenmengen von Gesten aufgezeichnet, so müssen diese meist vorverarbeitet werden, um für einen Gestenerkennungsprozess verwendbar zu sein. Ein zu beachtender Aspekt ist, ob mehrere Gesten in einem Datensatz vorkommen, oder jede Geste ihren eigenen, gesonderten Datensatz besitzt. Es muss darauf geachtet werden, dass nur die reine Geste in einem Erkennungsprozess verwendet wird, da sonst eventuell die Ergebnisse verfälscht werden und somit die Erkennung beeinflussen. Werden die Gesten in einem Datensatz zusammengefasst, so müssen diese durch eine Verarbeitung oder durch manuelle Selektion segmentiert werden. In Abbildung 4.4 (1.) ist eine schlecht segmentierte Geste, welche zusätzlich mit einer sehr geringen Bildfrequenz aufgenommen wurde, dargestellt. Dies kann daran erkannt werden, dass nur wenige Punkte zur Verfügung standen, welche mit Linien miteinander verbunden wurden. Dadurch werden auch Gestenabläufe eckig und ungenau.
Das Risiko, aus geringen, falsch segmentierten Daten nur eine schlechte Erkennungsleistung hervorzubringen, ist sehr hoch. Deshalb sollte bei der Datenerhebung darauf geachtet werden, dass das Aufnahmegerät mindestens 30 Bilder pro Sekunde aufnehmen kann.

Die Segmentierung der Gesten kann entweder manuell oder durch einen Automatismus geschehen. Manuelles Segmentieren kann während der Aufnahme der Gesten stattfinden. Dies bedeutet, dass bei der Aufnahme nur Daten einer Geste aufgenommen werden sollen und darauf geachtet werden muss, dass keine Ruhepositionen oder mehrere Gesten hintereinander in einem Datensatz gespeichert werden. Ein Automatismus setzt meist nach der Aufnahme aller Gesten an. Dieser bestimmt beispielsweise anhand des Verlaufs der Daten, welche Datenpunkte eine Geste repräsentieren und welche „herausgefiltert" werden müssen. Bei beiden Ansätzen können Fehler auftreten. Die Fehlerquelle beim manuellen Aufzeichnen ist menschlich; der aufnehmende Nutzer beginnt die Aufnahme entweder zu früh, zu spät oder beendet das Aufnehmen der Geste falsch. Beim automatischen Segmentieren kommt es darauf an, wie genau der Algorithmus zum Trennen der Gesten auf den vorhandenen Daten arbeiten kann.

Berechnung von Features aus den aufgenommenen Daten

Die Features einer Geste sind die Eigenschaften, anhand derer die Geste erkannt werden soll. Diese sollten also möglichst prägnant für die Ausführung der Geste sein. Bei verschiedenen Gesten müssen verschiedene Features betrachtet werden, um eine Geste eindeutiger klassifizieren zu können.

Manche Eigenschaften einer Geste müssen anhand erhobener Daten bestimmt werden, da das Aufnahmegerät (in diesem Fall die Kinect), nur Positions- und Zeitdaten erfassen kann. Wird nur die Position einer Hand über die Zeit aufgezeichnet, kann die **Geschwindigkeit** in die Erkennung miteinbezogen werden, wenn diese aus den vorhandenen Werten berechnet wird.
Weitere Features wären:

Die Ebene in der die Geste ausgeführt wird Diese Eigenschaft ist nur für planare Gesten gültig. Planare Gesten werden in einer Ebne ausgeführt, jedoch ergibt sich durch die Aufzeichnung der Raumkoordinaten der Geste eine dreidimensionale Trajektorie. Ist über die Geste bekannt, dass diese anhand ihrer zweidimensionalen Trajektorie in der Ebene der Ausführung (die Ebene, in der die größte Positionsdifferenz der Trajektorie liegt) erkennbar ist, kann die Ebene der Ausführung, im Weiteren als „Action Plane" bezeichnet, bestimmt und zur Erkennung miteinbezogen werden. Wie diese Ebene bestimmt wird und auf welche Hindernisse und Probleme bei der Erkennung gestoßen werden kann, ist in Kapitel 5.6 nachzulesen. Die Features, die durch die Action Plane bestimmt werden, können verschiedene sein. Die Lage der Geste im Raum wird bekannt, wodurch die einzelnen Gesten durch eine Anpassung ihrer rotierten Lage im Raum, besser vergleichbar werden. Des Weiteren kann eine Reduktion der Dimension ins zweidimensionale vorgenommen werden. Da bekannt ist, dass die Personen beim Ausführen der Geste eine zweidimensionale Ausführungsabsicht hatten, können die Gesten auf ihre Action Plane projiziert werden, um den weiteren Erkennungsprozess nur noch mit den zweidimensionalen Gesten-Daten durchzuführen. Besonderheiten und den Vorgang der Projektion sind in Kapitel 5.6.3 zu finden.

Ableitung nach der Zeit Wie bereits beschrieben, kann die Geschwindigkeit aus den vorhandenen Positions- und Zeitdaten berechnet werden. Hierbei kann zwischen der Absolutgeschwindigkeit $v_{absolut}$, welche sich durch die Abstandberechnung zwischen zwei Punkten und deren zeitlicher Differenz berechnet und den Geschwindigkeiten der einzelnen Komponenten des Positionsfaturevektors ($(x, y, z)^T$) v_x, v_y und v_z unterschieden werden. Die Positionsdaten, bzw. Ortskoordinaten, kombiniert mit ihren partiellen Ableitungen, ergeben den Phasenraum mit dem

Featurevektor: $\vec{F} = (x, y, z, v_x, v_y, v_z)^T$. Die Berechnung und Integration der Geschwindigkeitskomponenten werden in den Kapiteln 5.7, 5.9 und 5.10 dargestellt.

4.3 Aufzeichnung und Auswahl der Gestendaten

Aufbauend auf den Erkenntnissen der Analyse vorhandener Datensätze und Überlegungen im Bezug auf das Erkennungsverfahren mit Hidden Markov Modellen (siehe Kapitel3.2), wurde die folgende Erhebung von Gestendaten durchgeführt.
Zuerst wird die Wahl der benutzen Hardware- und Softwarekomponenten zur Erhebung beschrieben und danach die Konzeption des Vorgehens, beziehungsweise den Aufbau des Datenformats inklusive Begründungen dieser.

4.3.1 Auswahl der verwendeten Gesten

Es sollen geeignete Gesten zur Erkennung gewählt werden. Es liegt nahe Gesten zu verwenden, sie bereits in anderen Arbeiten analysiert wurden, um vergleichbare Ergebnisse zu erhalten. Dementsprechend sollen folgende Gesten getestet werden:

- „Circle" - Kreisgeste (Ebenfalls eingesetzt von: [11][26][15][2][25])
- „Square" -Quadratgeste (Gewält aufgrund der Ähnlichkeit von Quadrat und Kreis; auch genutzt von: [25])
- „Z"- Z-Geste (Ebenfalls zu finden in:[11][15][9][25][2])
- „Heart" - Herz-Geste (Genutzt von: [2])

„Circle" ist eine oft verwendete Testgeste. Es ist eine Geste, die bei unterschiedlichen Anfangspunkten trotzdem gleich bleibt und eine Ausführung ohne (abrupte) Richtungsänderung beinhaltet. Bei der Aufzeichnung fiel auf, dass die Nutzer alle sehr ähnliche Kreise ausführten, was beispielsweise bei der Durchführung eines Quadrates nicht der Fall war. Bei einem „Square" kann der Nutzer wählen, ob er auf der Mitte einer Strecke oder an einer Kante anfängt, in welche Richtung er die Geste ausführt und wie abrupt die Ecken ausgeführt werden. Es besteht die Annahme, dass Square und Circle bei der Erkennung oft verwechselt werden, da sie positionsbezogen einen ähnlichen Ablauf beinhalten. Wird das HMM (aus Positionsdaten) auf einen Circle trainiert, könnten die beobachteten Zustände eines Squares ebenfalls auf die Hidden States eines Circles hindeuten und werden als ein solcher klassifiziert.

Hypothese 4.1 (Ähnlichkeit von Kreis und Quadrat)
Die Kreis- und Quadrat-Gesten werden aufgrund ihrer Ähnlichkeit der Ortskorrdinaten bei einer Positionsdaten-HMM häufig verwechselt.

Die Ausführung einer Z-Geste beinhaltet zwei Richtungsänderungen, welche in einem spitzeren Winkel ausgeführt werden, als bei einem Square. Zudem ist eine Z-Geste offen (der Endpunkt liegt nicht beim Startpunkt). Der Nutzer hat demnach die zusätzliche Schwierigkeit, die Hand nach einmaliger Ausführung wieder von neu im Raum zu positionieren, um die Geste erneut auszuführen. Dabei kann es unter anderem zu sichtbaren Positionsveränderungen kommen. Zusätzlich ist dem Nutzer nicht bekannt, an welcher Position er die vorherige Geste gestoppt hat, was bedeutet, dass die Größe, die Länge und Breite der Geste deutlicher variieren können als bei geschlossenen Gesten.

Hypothese 4.2 (Z-Geste)
Die Z-Gesten variieren bei den unterschiedlichen Personen stärker in ihrer Länge und Breite, als Kreis- und Quadrat-Gesten.

Die Heart-Geste ist wiederum dem Circle und dem Square in der Ausführung ähnlich. Der Nutzer hat den gleichen Start- und Endpunkt. Der Unterschied liegt hierbei in der Ausführung der Richtungsänderung zwischen den beiden Herzbögen. Im Gegensatz zu einem Quadrat oder Kreis, welche geometrisch festgelegte Formen sind, ist ein Herz jedoch interpretierbar. Es besteht die Annahme, da die Nutzer eine unterschiedliche Wahrnehmung von Herzen und vor allem von ihrer eigenen Ausführung eines Herzen haben, dass diese Gesten ebenfalls stärker variieren, als die Kreis- oder Quadrat-Gesten.

Hypothese 4.3 (Herz-Geste)
Die Herz-Gesten variieren bei den unterschiedlichen Personen stärker, als Kreis- und Quadrat-Gesten.

4.3.2 Auswahl der Hardware- und Softwarekomponenten

Bildbasierte Gestenerkennnung wird in der Entwicklung von NUIs für viele Anwendungen, wie Games, virtuelle Realität und beispielsweise Modeling-Werkzeuge, häufig verwendet [8]. Diese Art der Aufzeichnung ist herausfordernd aufgrund ihre Störanfälligkeit. Diese Art der Gestenaufzeichnung wurde beeinträchtigt von verschiedenen Lichtverhältnissen, schwierig zu erkennenden Hintergründen, welche nur schwer von den eigentlichen Bewegungen zu unterscheiden sind, und der Anschaffungskosten multibler Kamerasysteme [14].
Mit der Veröffentlichung der Xbox Kinect wurde ein kostengünstiges, infrarot-basiertes Werkzeug zur Gestenaufzeichung auf den Markt gebracht. Diese benutzt Tiefensensoren und kann einfach dazu genutzt werden, Bewegungen vor verschiedenen Hintergründen zu erkennen und aufzuzeichnen (siehe 4.5). Deshalb wird in dieser Arbeit die Kinect zur Aufnahme der Gesten verwendet. Genauer nutzt die Kinect eine Kombination aus Infrarot-Kamera und Infrarot-Projektor, um Bewegungen zu erkennen, welche nicht durch verschiedene Lichtverhältnisse beeinflusst werden. Ursprünglich für die Xbox entwickelt, wurden von Microsoft jedoch zeitnah nach der Veröffentlichung auch Gerätetreiber für den PC nachgeliefert. Dazu entwickelte Primesense ein unabhängiges Framework (OpenNI) für die Entwicklung mit der Kinect. Im Juni 2011 brachte Microsoft ebenfalls ein Software Development Kit (SDK) heraus, welches jedoch nicht frei verfügbar ist. Eine Einschränkung der Kinect ist die relativ geringe Reichweite des Infrarot-Sensors. Dadurch entsteht ein Aufnahmefeld von Gesten und Bewegungen von ungefähr $4m^2$ [19]. Die Kinect wurde gewählt,da sie die für diese Arbeit notwendige Funktion (bestimmte Eigenschaften von Gesten aufzuzeichnen) besitzt; jede andere Hardware mit derselben Funktionalität hätte ebenfalls benutzt werden können. Für die Aufnahme und Speicherung der Gesten wurden folgende Softwarekomponenten, beziehungsweise Bibliotheken , gewählt.

Open NI Das Open NI Framework ist eine Open Source SDK, welche für die Entwicklung von „3D Erfassungs-Middleware"-Libraries und Applications benutzt wird.

NITE NITE ist eine 3D Computer Vision Middleware, um auf Tiefen, Farb- und Audioinformationen zugreifen zu können. Diese erlaubt es Funktionen wie „Handlokalisierung und Tracking", oder „Nutzer von ihrem Hintergrund zu separieren" und viele andere, auszuführen.

Simple Open NI Open NI Library für Processing (nur für Processing <2.0) [4]

[4] Für eine genauere Installationsanleitung: https://code.google.com/p/simple-openni/wiki/Installation

Für das Aufzeichnen der Gesten wurde demnach eine Kombination aus Kinect und Processing gewählt. Die aufgezeichneten Daten werden, wie im folgenden Kapitel 4.3.3 beschrieben, aufgezeichnet und gespeichert.

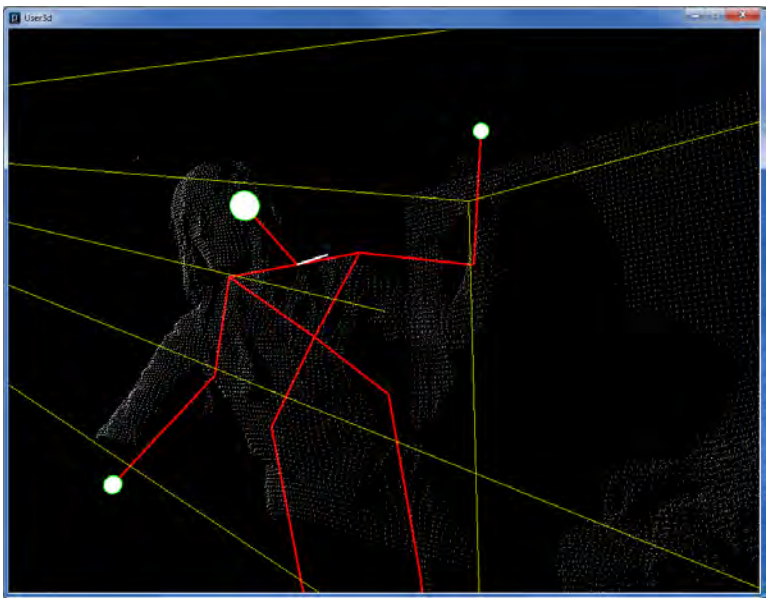

Abbildung 4.5: Ansicht der Ausgabe des Tiefenbilds der Kinect beim Aufzeichnen der Gestendaten.

4.3.3 Konzeption und Vorgehen

Um den Erkennungsprozess nicht durch Daten mit niedriger Auflösung (wenigen Messpunkten) negativ zu beeinflussen, sollen eigene Daten aufgenommen werden. Des Weiteren sollen keine Stördaten durch eine fehlerhafte Segmentierung der Gestendaten vorliegen, weshalb die „Segmentierung" in dieser Arbeit manuell vorgenommen wird. Dies bedeutet, dass beim Aufzeichnen der Gesten jeweils das Programm zur Aufzeichnung manuell gestartet und gestoppt wurde. Beim Start des Programms startet der Proband mit der vorher angekündigten Geste und bei Beendigung wird das Aufzeichnen manuell beendet. Genauer funktioniert dies wie folgt: Der Nutzer teilt mit, ab wann er bereit ist, die Geste zu starten, damit der Versuchsleiter die Aufnahme manuell beginnen kann. Dies bedeutet, dass der Proband seine Hand so positioniert hat, wie er starten möchte. Durch das Auslösen der Aufzeichnung, deutlich hörbar für den Probanden, beginnt dieser mit der Geste. Endet die Bewegung des Probanden, so beendet der Versuchsleiter die Aufnahme manuell. Auch mit diesem Vorgehen sind Stördaten vorhanden. Fängt der Nutzer zu früh mit der Geste an, also vor Programmstart oder beendet der Aufnehmende die Aufnahme zu früh, so gibt es ebenfalls Schwankungen

am Anfang und am Ende der verschiedenen Gesten, jedoch können komplett misslungene Versuche direkt aussortiert werden. Bei der Konzeption ist zu beachten, welche Komponenten der Geste, mit verfügbarer Hardware aufgenommen werden können und was benötigt wird, um keinen Datenüberschuss zu hinterlegen, der im weiteren Verlauf der Erkennung nicht gebraucht wird. Als eigenes Datenformat, zur initialen Erfassung der Daten, wurden die x,y,z-Koordinaten beider Hände und des Kopfes (unterschieden in der „tid": ID des getrackten Körperteils) gewählt, sowie die UserID[5] und die Zeit.

```
timestamp, uid, tid, pos_x, pos_y, pos_z
51072402793218, 1, 1, 37.004276, 467.838, 1680.4735
51072403013872, 1,15, -137.30783, 310.8352, 1563.6676
51072403211434, 1,9, 317.02515, -375.77258, 1703.0171
```

Abbildung 4.6: Beispieldatensatz zur Verdeutlichung der Struktur der hinterlegten Gestendaten

timestamp Die Aufzeichnung der Nanosekunden zum Zeitpunkt der Ausführung der Geste.

uid Darin wird eine Nummer zur genauen Identifizierung des Users hinterlegt, da die Kinect mehrere User gleichzeitig aufzeichnen kann. So können die Gestenanteile eines Users nachvollzogen werden, auch wenn mehrere User gleichzeitig vor der Kinect stehen.

tid Die eindeutige Identifizierung des aufgezeichneten Körperteils. Aufgezeichnet werden: Rechte Hand, Linke Hand und Kopf. Die verwendeten Identifizierungsnummern sind:
Rechte Hand: 15
Linke Hand: 9
Kopf: 1

pos_x Position des hinterlegten Körperteils auf der x-Achse, relativ zur Kinect.

pos_y Position des hinterlegten Körperteils auf der y-Achse, relativ zur Kinect.

pos_z Position des hinterlegten Körperteils auf der z-Achse, relativ zur Kinect.

Im weiteren Verlauf werden ebenfalls Geschwindigkeitsfeatures, sowie der Phasen- beziehungsweise Konfigurationsraum der Gesten in die Erkennung mit einbezogen. Hierfür

[5] Eindeutige Zahl zur Identifizierung des getrackten Benutzers.

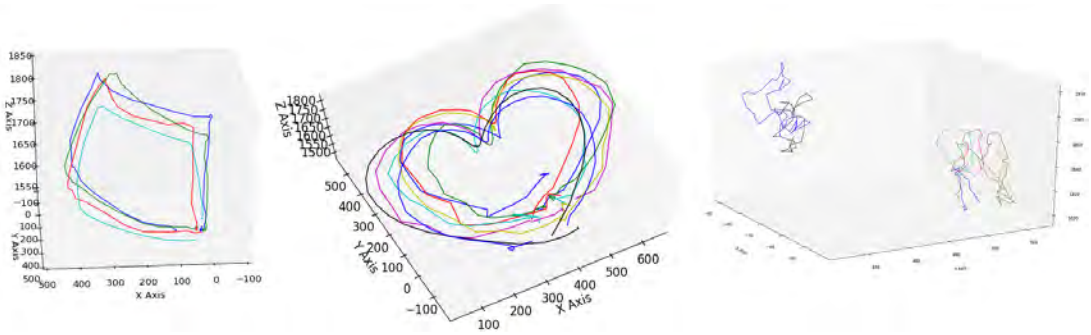

Abbildung 4.7: Eigene Testdaten: 1. Gestenbeispiel „Square"; 2. Gestenbeispiel „Heart"; 3. Beispielhafte Kopfbewegungsaufzeichnung zur Geste „Heart"

werden aus den bereits beschriebenen, aufgenommenen Komponenten weitere Features generiert, da mit der Kinect nicht direkt die Geschwindigkeit aufgezeichnet werden kann. Im Fall des Phasenraums ist dieses weitere Feature die Geschwindigkeit, in den einzelnen Punkten im Raum.
Wie die Geschwindigkeit und der Phasenraum mit einbezogen werden und welche Erkenntnisse daraus gewonnen werden können, werden in den Kapitelabschnitten 5.7, 5.8 und 5.10 des Kapitels „Gestenerkennung und Umsetzung" 5 näher beschrieben.

4.3.4 Analyse und Diskussion

Die Testdaten wurden von 20 zufälligen Personen erhoben, die sich in Alter, Geschlecht, Größe und Erfahrung mit Gestensteuerung unterschieden. Dabei ist zu beachten, dass in der aufgeführten Tabelle 4.1, Personen mehrfach aufgeführt werden, wenn diese Gesten mit der linken und der rechten Hand zur Verfügung stellen (ist bei einer Person vorgekommen: Person 1). Jede Person führte jede Geste mindestens fünf mal aus (entspricht mindestens 20 Testgesten pro Person), die durchschnittliche Durchführung pro Geste liegt jedoch höher. Insgesamt führten alle Testpersonen im Durchschnitt ca. 51 Gesten aus.

Unter den 20 Testpersonen sind 13 männlich und zwischen 23 bis 52 Jahre alt. Die 7 weiblichen Testpersonen sind zwischen 18 bis 51 Jahre alt.

Auffällig war, dass die Gesten „Circle" und „Square" von den Testpersonen mit einer vergleichbaren Startposition im oberen Teil der Geste, begonnen wurden. Der „Circle" wurde meist an seiner höchsten Position („ 12 Uhr") begonnen. Das „Square" hingegen wurde entweder an einer der beiden oberen Ecken, oder in der Mitte der oberen Linie begonnen. Das „ Heart" wurde von jeder Person unten in seiner Spitze begonnen. Bei dem „Z" wurde immer oben begonnen, was vermutlich auf die Schreibweise des erlernten

Personen	m/w	Left/Right	Circle	Square	Z-Geste	Heart	Summe
Person 0	m	L	21	15	15	17	68
Person 1	w	L	15	17	23	17	72
Person 1	w	R	18	0	0	0	18
Person 2	m	R	15	13	14	16	58
Person 3	m	R	5	5	5	6	21
Person 4	m	R	5	5	6	6	22
Person 5	w	R	5	5	5	5	20
Person 6	m	R	6	5	5	5	21
Person 7	m	R	5	5	5	5	20
Person 8	w	R	5	5	5	5	20
Person 9	m	R	6	5	5	5	21
Person 10	m	R	34	32	30	30	126
Person 11	m	R	14	12	13	17	56
Person 12	m	R	13	16	14	12	55
Person 13	m	R	10	11	11	11	43
Person 14	m	L	16	12	20	18	66
Person 15	m	R	16	12	12	14	54
Person 16	w	R	15	15	18	13	61
Person 17	w	L	17	16	18	16	67
Person 18	w	R	18	15	16	15	64
Person 19	m	L	14	15	15	16	60
Summe	Male: 13 Female: 7 Gesamt: 20	Left: 5 Right: 15	273	236	255	249	1013
Durchschnitt	-	-	13,65	11,75	12,7	12,45	50,55

Tabelle 4.1: Tabelle über Metadaten der Gestenaufzeichnung: Anzahlen der jeweiligen Personen und Gesamtanzahlen der Gesten

Buchstabens zurückzuführen ist. Nur wenige Personen fragten, ob das Z „zu mir oder zu Ihnen" ausgeführt werden soll. Den Testpersonen wurde mitgeteilt, dass sie das Z aus ihrer Perspektive ausführen sollen.
Die Größe und Position der Geste variiert merkbar. Es gibt Testpersonen, die die Geste neben ihrem Körper ausführten und welche, die vor ihrem Körper gestikulierten. Dies kann unter Umständen zu Fehlern beim Aufnahmeprozess mittels Kinect führen, da die Kinect bei Verdeckungen verschiedener Körperteile (beispielsweise Arm vor Kopf) Schwierigkeiten bekommt, die Körperteile auseinander zu halten. Dies ist jedoch bei der Aufnahme nicht vorgekommen.
Die Analyse der vorhandenen, sowie selbst aufgezeichneten Daten ergab, dass viele zu beachtende Randbedingungen für Gestenerkennung existieren. Die Nutzer sind unterschiedlich groß, haben unterschiedliche Armlängen und Winkel der Gestenausführung.

Personen	m	Left/Right	Alter	Summe
Person 0	m	L	25	68
Person 2	m	R	25	58
Person 3	m	R	25	21
Person 4	m	R	31	22
Person 6	m	R	52	20
Person 7	m	R	50	20
Person 10	m	R	24	126
Person 11	m	R	33	56
Person 12	m	R	23	55
Person 13	m	R	31	43
Person 14	m	L	23	66
Person 15	m	R	28	54
Person 19	m	L	25	60
Zusammenfassung	Male: 13	Left:3 Right: 10	ca. 30,38	668 (ca. 51,38 p.P.)

Tabelle 4.2: Tabelle über Metadaten der Gestenaufzeichnung: Anzahlen von Gesten männlicher Teilnehmer und Analyse des Durchschnittsalters

Personen	m	Left/Right	Alter	Summe
Person 1	w	L	23	72
Person 1	w	R	23	18
Person 5	w	R	51	20
Person 8	w	R	18	20
Person 9	w	R	51	21
Person 16	w	R	24	61
Person 17	w	L	24	67
Person 18	w	R	25	64
Zusammenfassung	Female: 7	Left:2 Right: 6	ca. 30,85	343 (ca. 49 p.P.)

Tabelle 4.3: Tabelle über Metadaten der Gestenaufzeichnung: Anzahlen der Gesten von weiblichen Teilnehmern und Analyse des Altersdurchschnitts

Dadurch müssen die Gestendaten voraussichtlich vor einer weiteren Verarbeitung in ihrer Positionierung und gegebenenfalls auch in ihrer Größe auf ein gemeinsames Format normalisiert werden.

Hypothese 4.4 (Normalisierung)
Anhand der aufgezeichneten Originaldaten, ohne Normalisierung in jeglicher Hinsicht, werden schlechtere Ergebnisse bei der Erkennung erzielt, als bei in Position und/oder Größe normierten Daten.

Ein weiteres Problem stellt die Segmentierung der Daten dar. Es ist wichtig, die Gestendaten möglichst gut von „nicht-Gesten"-Daten, wie Ruhezustand oder Folgegesten zu trennen. Ist die Segmentierung fehlerhaft oder ungenau, entstehen „Ausreißerpunkte" in den Aufzeichnungen einer Geste. Diese beeinflussen nicht nur die Erkennung einer Geste anhand ihrer Visualisierung negativ, sondern beeinflussen auch den „erlernten Algorithmus" der Hidden Markov Modelle (siehe Kapitel 3.2) und eventuelle Datenvorverarbeitungen (siehe Kapitel 4.2 und 5), die eigentlich für die Verbesserung der Erkennung zuständig sind. Bei der Bestimmung der Eigenwerte können Fehler auftreten, wenn die verwendete Datenmenge schlecht segmentiert wurde. Besonders negativ wirkt sich eine schlechte Segmentierung dann aus, wenn die Gesten unterschiedlich schlecht (manche zu kurz, andere zu lang (Bewegungen aufzeichnet, die nicht zur Geste gehören)) sind, da das HMM auf diesen Daten nur sehr unzuverlässig trainiert werden kann.

5 Gestenerkennung und Umsetzung

Nachdem die Gestendaten analysiert wurden, muss der Erkennungsprozess konzipiert werden. Gesten können im Bereich der Mensch-Computer Interaktion für viele Bereiche genutzt werden. Um diese Art der Interaktion für ein System auszugestalten, müssen die Bewegungen, welche als Gesten erkannt werden sollen, aufgenommen und für eine Anwendung erkennbar hinterlegt werden. Zur automatischen Erkennung werden meist mathematische Methoden, Modelle und Algorithmen aus dem Bereich der Mustererkennung verwendet.
Unterschieden werden berührungsbasierte und berührungslose Gesten. Im Kontext dieser Ausarbeitung geht es um berührungslose Gesten, welche jedoch mit ähnlichen mathematischen Grundlagen und Algorithmen bestimmt und erkannt werden können, wie berührungsbasierte Gesten. Die erhobenen Gestendaten unterscheiden sich je nach Aufnahmegerät in Form, Umfang und Ausführung, wie in Kapitel 4.3.3 beschrieben.
Für das Datenformat muss überlegt werden, welche messbaren Eigenschaften die Geste und ihre Ausführung ausmacht. Ein Ansatz zur Gestendatenerhebung ist das Aufzeichnen der Positionen im Raum des bewegten Körperteils über die Zeit. Soll eine Handbewegung erkannt werden, so muss über die Länge der ausgeführten Aktion mit einer bestimmten Framerate eine Liste aller Positionen gespeichert werden. Andere Ansätze sind beispielsweise das Aufzeichnen von beschleunigungsbasierten Daten oder Geschwindigkeiten (Konfigurationsraum).
Die Daten können über verschiedene Systeme aufgenommen werden. In der Spielindustrie werden beispielsweise die WiiMote zum Aufzeichnen der Beschleunigungsdaten und die Kinect zum Aufzeichnen von Positionsdaten und Tiefenbildern genutzt.

Gesten müssen hinterlegt, modelliert und anschließend anhand der gespeicherten Daten im Vergleich zu neuen Eingabedaten erkannt werden. Die Gesten verschiedener Nutzer müssen auf dieser Basis erkannt werden können, wodurch einige Herausforderungen entstehen. Gestenerkennung ist ein komplexer Prozess, welcher meist nicht ohne Kontextbezug gestaltet werden kann. Am Anfang steht die zu erledigende Aufgabe und die Herausforderung, eine Interaktionsmöglichkeit passend zu dieser zu entwickeln. Es sei angemerkt, dass Gestensteuerung nicht bei jeder Interaktion in jedem Kontext

praktikabel ist und demnach nur in Kontexten genutzt werden sollte, bei denen nach eingehender Nutzungskontextanalyse eine Gestensteuerung als sinnvoll erachtet wird.

In diesem Kapitel wird beschrieben, wie die Erkennung der Gesten realisiert wurde. Das Aufnehmen und Konzipieren der Gestendaten ist in Kapitel 4.3.2 zu finden.

5.1 Vorverarbeitungen vor dem eigentlichen Programmstart

Verschiedene Vorverarbeitungen werden angesetzt, um die Gestendaten der Testpersonen für die nachfolgende Erkennung vergleichbar zu gestalten.

5.1.1 Vereinheitlichung der aktiven Hand für die weitere Verarbeitung

Da sich unter den Testpersonen Rechts- sowie Linkshänder befinden, ist die aktive Hand, welche die Geste durchgeführt hat, unterschiedlich. Da allerdings beide Hände aufgezeichnet werden, muss für die Erkennung hinterlegt sein, welche Hand zur Verarbeitung die ausschlaggebende ist. Die ausgeführten Gesten sind, egal mit welcher Hand diese durchgeführt wurden, an sich gleich. Für die weitere Verarbeitung im Programm ist es praktikabel, die „verwendete Hand" zu vereinheitlichen, da diese mit verschiedenen IDs im Datenformat gespeichert wurden. Dies ist eine sehr projektbezogene Vereinheitlichung, die in diesem Kontext und diesen Datenformaten sinnvoll ist. Bei anderen Herangehensweisen wird diese Vereinheitlichung eventuell nicht benötigt.
Dazu werden die IDs der Hände für alle Linkshänder getauscht. Durch diese Veränderung werden neue, angepasste Gestendaten-Files gespeichert, welche in der folgenden Verarbeitung an Stelle der Originalfiles verwendet werden müssen. Der Kern dieses Anpassungs-Skripts lautet:

Folgende Daten müssen bekannt sein:

```
header = "timestamp,uid,tid,pos_x,pos_y,pos_z"
  LEFT_HAND = 9
  RIGHT_HAND = 15
  HEAD = 1
```

Um folgende Verarbeitung für die gesamte angegebene Ordnerstruktur durchzuführen:

```
for line in lines[1:] : #ignoring the header line
            seq = line.split(',')
            if int(seq[2]) == RIGHT_HAND :
                seq[2] = str(LEFT_HAND)
            elif int(seq[2]) == LEFT_HAND :
                seq[2] = str(RIGHT_HAND)
```

```
outputHandler.write(",".join(seq).strip() + "\n")
```

Dort werden lediglich die IDs der Hände für die jeweiligen Zeilen getauscht und somit die Geste nicht verändert oder verzerrt.

5.1.2 Drehrichtung der Gesten vereinheitlichen

Im weiteren Verlauf der Tests wird diese Vorverarbeitung, wenn verwendet, mit ***c*** gekennzeichnet.

Da eine Geste einen zeitlich linearen Ablauf besitzt, ist dieser Ablauf zentral bei der Erkennung und wird mit dem Left-to-Right Faktor im HMM modelliert. Dieser gibt an, wie weit von welchen zu welchem Zustand gesprungen werden darf. Bei gleichen Gesten sollte das Auslassen eines Zustandes nicht vorkommen, weshalb der Faktor auf zwei gesetzt wird. Dies bedeutet, dass der Zustand nur in sich verweilen oder zum nächsten Zustand übergehen kann. Sind jedoch Gesten mit verschiedenen Drehrichtungen in den Trainings- sowie Testdaten enthalten, gilt dieser LR-Faktor nicht mehr. Die Zustände werden bei verschiedenen Drehrichtungen in einer konträren Reihenfolge durchlaufen, weshalb eine Geste mit einem Faktor $LR = 2$ auf einem Trainingsdatensatz, der eine abweichende Drehrichtung beinhaltet, nicht erkannt werden kann. Diese Drehrichtung von Gesten, insofern sie eine besitzen, soll auf eine gleiche Richtung angepasst werden. Bei den hier verwendeten Gesten, können „Circle", „Square" und „Heart" mit oder gegen den Uhrzeigersinn ausgeführt werden. Das „Z" hat jedoch keine Drehrichtung, weshalb es nicht fälschlicher Weise „umgedreht" werden darf.
Das Vereinheitlichen der Gesten geschieht wie folgt:

- Zuerst wird geprüft, ob die Geste ein Z ist.
- Ist die Geste kein Z, wird ihre Drehrichtung festgestellt.
- Ist die Drehrichtung gegen den Uhrzeigersinn, so wird ihr Ablauf umgedreht, so dass die letzte Koordinate danach die erste Koordinate darstellt.

Hypothese 5.1 (Drehrichtung)
Eine einheitliche Drehrichtung der Gesten verbessert die Erkennungsrate.

Der genaue Ablauf und die verwendeten Methoden sind im Anhang 8 zu finden.

5.2 Initialisierung der HMM

Die Quellen [11] und [12] dienten als Vorlage für die Umsetzung des Hidden Markov Modells in Python.

Folgende Parameter müssen bei der Initialisierung der HMM gewählt werden:

- Anzahl der Output Symbols
- Anzahl der Hidden States
- LR-Parameter (ist dieser gleich der Anzahl Zustände, ist es ein semiotisches Modell)

Der „LR" ist ein Parameter, der kontextbezogen gewählt werden muss. Da Gesten für gewöhnlich einem linearen, zeitlichen Ablauf unterliegen und somit jeder beobachtbare Zustand hintereinander ausgeführt wird, bietet es sich an, eine Left-to-Right Schrittweite von 2 zu wählen. Jedoch wird angenommen, dass dies nicht für alle Features und alle Gesten, die optimale Lösung ist. Nach Hall werden initial 8 verschiedene Output Symbols und 12 Hidden States gewählt. Dadurch hat die Transitionsmatrix A die Dimension von 8x8. Die Emissionsmatrix B hat eine Dimension von 8x12.

Hypothese 5.2 (Left-to-Right Parameter)
Der Left-to-Right Parameter kann bei verschiedenen Kontexten, Gesten oder Features anders gewählt werden, um eine bessere Erkennung zu erhalten.

Hypothese 5.3 (Anzahl der Symbole und Hidden States)
Die Anzahl der Symbole und Hidden States ist kontextabhänig und ist nicht für alle Featurevektoren gleich.

5.3 Training

Für das Training des HMMs muss angegeben werden, in wie vielen Iterationen die HMM ihre Parameter anpassen soll. In Abschnitt 6.2 wird diese Anzahl getestet. Für das

Programm wurde eine Anzahl an 20 Iterationen gewählt, da die Parameter der HMM mit den hier verwendeten Trainingsgesten bereits zwischen fünf und 15 Iterationen stagnieren. Dazu müssen für das Training die verwendeten Trainingsdaten angegeben werden, aus denen das Modell/die Modelle gebildet werden sollen, welche im nächsten Schritt, dem Testen, geprüft werden.

5.4 Testen

Die durch das Training erstellten Modelle müssen nun für verschiedene Gesten getestet werden. Im Training wird das Set an Symbolen bestimmt, welche mittels Punktwolkenzentren in der Punktwolke der Geste, erstellt durch K-Means, abgebildet sind.

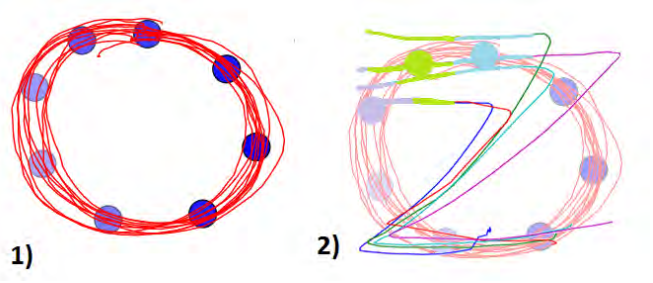

Abbildung 5.1: 1) Beim Training des Kreis-HMMs, werden die K-Means auf Kreisgesten gebildet. 2) Beim Testen der Z-Gesten auf das Kreis-HMM werden den beobachteten Punkten, die aus den Kreisen erlernten, diskreten Symbole zugewiesen.

Auf Basis des Log-Likelihoods des Modells wird eine Erkennungsschwelle festgelegt. Diese wird „Threshold" genannt. Der Threshold-Parameter aus der Gleichung 3.9 wird initial auf 2 gesetzt.

Hypothese 5.4 (Threshold Parameter)
Die Wahl eines statischen Threshold Parameters ist nicht für alle Featurevektoren, Kontexte und Gesten allgemeingültig. Der Threshold sollte in verschiedenen Kontexten anders gewählt werden, um eine optimale Erkennungsrate zu erzielen.

5.5 Normierungsansätze

Die vorhandenen Datensätze, sowie die aufgenommenen Datensätze mit der Kinect unterliegen bestimmten Beschränkungen. Sie beinhalten eine begrenzte Anzahl an Ei-

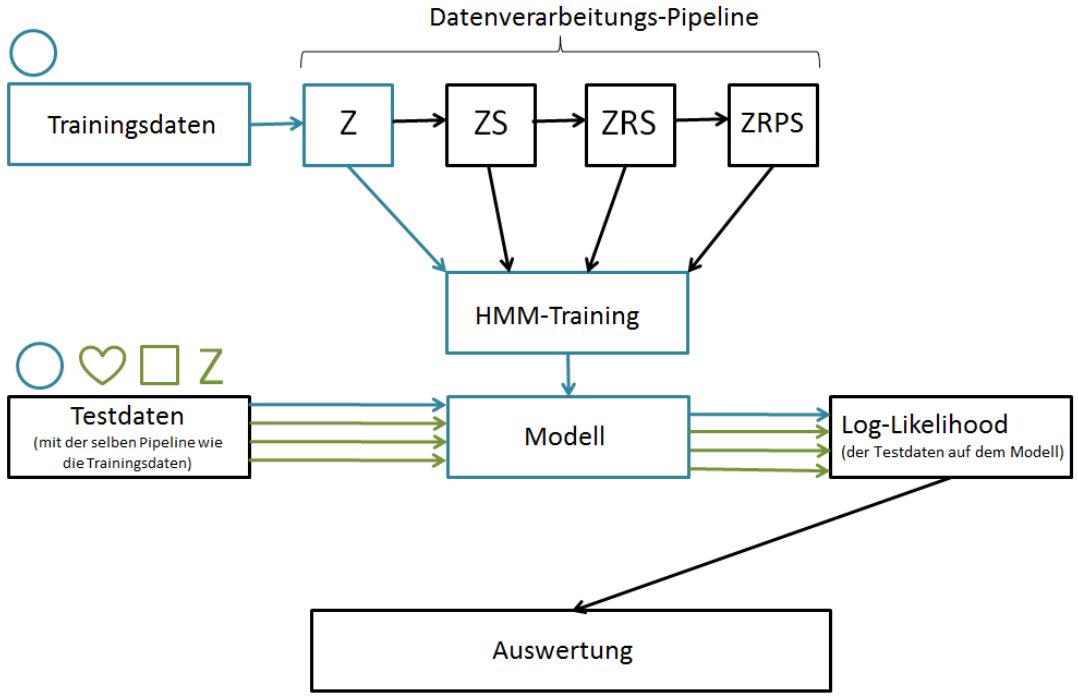

Abbildung 5.2: Beispiel des Ablaufs wie Trainingsdaten eines Kreises und Testdaten aller Gesten mit der HMM verarbeitet werden. Die Datenverarbeitungspipeline zeigt folgende aufeinander aufbauende Verarbeitungsschritte: Z (Zentrierung der Gesten), S (Skalierung der Gesten), R(Rotation der Gesten in eine gemeinsame Action Plane), P (Projektion in eine gemeinsame Action Plane); dies kann in den jeweiligen Kapiteln 5.5.1, 5.5.2, 5.6.2, 5.6.3 und 6.1 genauer nachgelesen werden.

genschaften. Bei allen Datensätzen handelt es sich hierbei hauptsächlich um Orts- und Zeitangaben (siehe Kapitel 4.3.3) der aufgezeichneten Gliedmaßen bei einer Geste.

Das Einzige, was die aufgenommenen Gesten ohne weitere Bearbeitung der Daten gemeinsam (außer die Absicht der Ausführung der gleichen Geste) haben ist, dass sie „irgendwo" vor der Kinect aufgenommen wurden. Dabei variieren Position, Entfernung und Größe der ausgeführten Geste. Bei genauerer Betrachtung der Originaldaten fällt ebenfalls auf, dass die Rotation der Gesten unterschiedlich ist und verschiedene Schwankungen durch Armlänge oder Körperdrehung entstehen können, wie man in Abbildung 5.3 sehen kann. In dieser Abbildungen sind die Gesten jedoch bereits übereinander gelegt, demnach ortsnormiert und auf die gleiche Größe skaliert, so dass diese möglichst vergleichbar sind. Durch eine Größennormalisierung fällt jedoch die Unterscheidung zwischen großen und kleinen Gesten der gleichen Art weg; ist dies für die

Erkennung interessant, so würde die Normierung der Größe weggelassen werden oder auf den Skalierungsparameter überprüft werden, um den Größenunterschied zwischen den verglichenen Gesten zu analysieren.

5.5.1 Ortsnormierung

Im Weiteren wird diese Vorverarbeitung der Gestendaten mit z beschrieben.

Die Gesten werden in einem Raum vor der Kinect aufgenommen. Die unterschiedliche Größe und Position der Personen verursacht, dass die aufgenommenen Gestendaten eine große Streuung in ihrer relativen Position zur Kinect besitzen. Dies ist ein Hindernis für die Erkennung, da die Gesten durch unterschiedliche Lage nicht gut miteinander verglichen werden können. Deshalb ist dafür zu sorgen, dass die Gesten in eine zu einander vergleichbare Lage im Raum gebracht werden. Der Ansatz für die Positions-/Ortsnormierung ist, den Mittelpunkt aller Gesten übereinander zu legen. Dafür wird der Mittelpunkt jeder Geste ermittelt, um die gesamte Geste in den Ursprung zu verschieben. Dafür wird jeder Wert jeder Ache um die Hälfte der Länge der Geste auf dieser Achse in Richtung des Ursprungs verschoben:

$$x_{neu} = x - \frac{x_{max} - x_{min}}{2} \tag{5.1}$$

$$y_{neu} = y - \frac{y_{max} - y_{min}}{2} \tag{5.2}$$

$$z_{neu} = z - \frac{z_{max} - z_{min}}{2} \tag{5.3}$$

Die Ortsnormierung wird wie beschrieben durchgeführt, da der zentrale Gedanke war, die Gesten „übereinander" zu legen, um eine bessere, aber möglichst simple und Vergleichbarkeit zu gewährleisten, die die Ausgangsdaten nicht verändert. Alle Gesten haben einen Mittelpunkt, deshalb wurden alle Gesten anhand ihres Mittelpunktes in den Ursprung verschoben. Ein anderer Ansatz wäre (statt $\frac{x_{max}-x_{min}}{2}$) die Mittelwerte der Geste auf den jeweiligen Achsen zu bestimmen und diesen von jedem Punkt abzuziehen, um die Gesten in den Mittelpunkt zu verschieben. Bei einer Mittelwertberechnung wird über alle Punkte addiert und durch ihre Anzahl geteilt. Bei dieser Berechnung werden die Maxima und Minima der Gesten bestimmt, um die Geste um die Hälfte der Länge auf der jeweiligen Achse in den Ursprung zu verschieben. Der Code ist im Anhang unter 8 zu finden.

5.5.2 Skalierung

Im Weiteren wird diese Vorverarbeitung der Gestendaten mit s beschrieben.

Um die Gesten auf eine gemeinsame, vergleichbare Größe zu skalieren, muss die Achse bestimmt werden, auf der die größte Ausdehnung vorhanden ist. Ist diese Achse bestimmt, wird anhand der Länge der Geste auf dieser, der Skalierungsfaktor bestimmt, um die Geste auf eine maximale Ausdehnung von 1 zu skalieren. Die jeweils kürzeren Ausdehnungen auf den anderen Achsen werden ebenfalls mit diesem Faktor skaliert, um die Geste nicht zu verzerren.

```
def toScale(data):
    tmpdata = []
    for dataset in data :
        x, y, z, t = toAxis(dataset)
        boundaries = [
                        (min(x), max(x)),
                        (min(y), max(y)),
                        (min(z), max(z)),
                        (min(t), max(t))
                      ]
        tmpaxis = []
        maxi = max([max(x) - min(x), max(y) - min(y), max(z) - min(z)])
        for ax, bound in zip([x, y, z, t], boundaries) :
            tmpitem = []
            for item in ax:
                if i == 3: #ignore time axis
                    tmpitem.append(item - bound[0])
                else :
                    tmpitem.append(item / maxi)
            tmpaxis.append(tmpitem)
            i += 1
        tmpdata.append(toPoints(tmpaxis[0], tmpaxis[1], tmpaxis[2],
            tmpaxis[3]))
    return tmpdata
```

Prinzipiell kann mit anderen Faktoren skaliert werden. Die Gesten könnten auch anhand der größten Standardabweichung auf den jeweiligen Achsen skaliert werden. Diese müssten im Vorfeld bestimmt werden. Die Minima und Maxima in den Daten sind bereits vorhanden.

5.6 Actionplane

5.6.1 Bestimmung der Action Plane

Ein weiterer Ansatz zur Featuregeneration ist, die Daten vor jeglicher Verarbeitung durch Gestenerkennungsalgorithmen, die Gesten möglichst simpel und vergleichbar darzustellen. Ein Ansatz hierfür ist die sogenannte „Action Plane". Die Action Plane beschreibt die Ebene, auf der die meiste Aktion einer Geste stattfindet. Diese Ebene ist in dem Fall von dreisimensionalen Raumkoordinaten, eine zweidimensionale Ebene, welche im Raum liegt. Diese Berechnung ist wichtig, da viele Gesten in ihrer originalen Aufzeichnung in verschiedenen Winkeln zueinander liegen, besonders, wenn diese von verschiedenen Personen ausgeführt werden. Nicht selten liegen die Gesten auch sehr verdreht zueinander, wie in Abbildung 5.3 gezeigt.

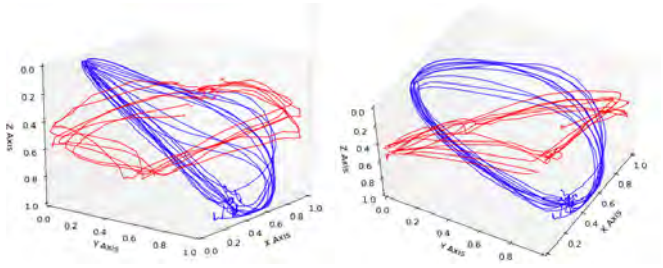

Abbildung 5.3: Visualisierung von verschiedenen Gesten und ihre Lage im Raum.

Für Gesten, die auf der Ebene direkt vor der Kamera ausgeführt werden wäre diese Ebene beispielsweise die X-Y-Ebene. In Abbildung 5.4 werden Gesten im Raum (blau) und ihre jeweilige die X-Y-Ebene rotierte und in die Action plane projizierte Darstellung visualisiert (rot).

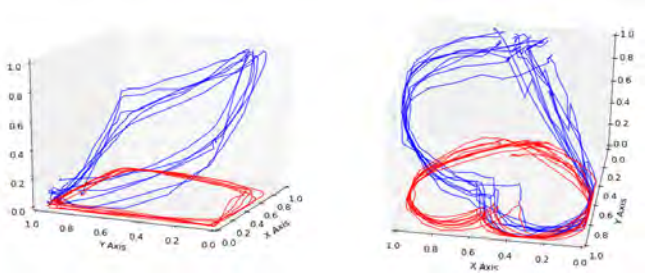

Abbildung 5.4: Visualisierung der X-Y-Z-Geste (blau) und ihre in die X-Y-Ebene rotierten und auf ihre Action Plane projizierten Gestenäquivalente (rot).

Um eine Action Plane zu bestimmen, können verschiedene Ansätze gewählt werden. Anhand der Punktwolke der Trainingsgesten muss eine Ebene bestimmt werden, die zu allen Punkten die geringste Abweichung aufweist, also die Punktwolke am besten

durch eine Ebene beschreibt. Diese Ebene kann auf mehrere Arten bestimmt werden. Eine Variante der Bestimmung einer möglichen Action Plane ist der Ansatz über die Methode der kleinsten Quadrate.
Diese Methode bestimmt die Ebene durch die Datenpunkte, die am nächsten zu den Datenpunkten verläuft, so dass die Summe der quadratischen Abweichungen der Ebene von den beobachteten Punkten minimiert wird (nach [18] und [1]).
Mit den gegebenen Werten $\{(x_i, y_i, z_i)\}_i^m = 1$ der Gestendatenpunkte werden die Parameter A, B und C einer Ebene $z = Ax + By + C$ so ermittelt, so dass die Summe der quadrierten Differenzen zwischen den Werten z_i und der Ebene $Ax_i + By_i + C$ minimal wird.

$$\min \sum_{i=1}^{n} [z_i - (Ax_i + By_i + C)]^2 \tag{5.4}$$

Das Problem dieses Vorgehens ist jedoch, dass nur die Abweichung in Z-Richtung betrachtet wird. Es müssen auch Action Planes erkannt werden können, welche hauptsächlich in der Z-X oder Z-Y-Ebene liegen. Deshalb wird die Bestimmung der Normalen zur Berechnung der Action Plane durch die Methode der Singulärwertzerlegung durchgeführt [17]. Das Produkt der Gestalt:

$$M = U\Sigma V^* \tag{5.5}$$

ist eine Singulärwertzerlegung einer komplexen $m \times n$-Matrix mit dem Rang r. U ist eine unitäre $m \times m$-Matrix, V^* ist die Adjungierte einer unitären $n \times n$-Matrix V und Σ ist eine $m \times n$-Diagonalmatrix. Die positiven Diagonaleinträge $\sigma_i, i = 1, \ldots, r$ von Σ heißen „Singulärwerte" von M, da $M^* \cdot M = \left(V\Sigma^T U^*\right) \cdot (U\Sigma V^*) = V\Sigma^T \Sigma V^*$ gilt. Die Singulärwerte der Matrix M sind gleich den Quadratwurzeln aus den positiven Eigenwerten von M^*M. Diese Berechnung wird auf jeder Geste einzeln angewendet, um die passende Ebene für jede einzelne Geste zu finden. Ist die Ebene gefunden, wird jede Geste auf diese Ebene projiziert.

5.6.2 Rotation der Gesten in eine gemeinsame Action Plane

Im Weiteren wird diese Vorverarbeitung der Gestendaten mit ***r*** beschrieben.

Nach der Bestimmung der Action Plane der einzelnen Gesten anhand ihrer aufgezeichneten Originaldaten, sollen die einzelnen Gesten auf eine gemeinsame Action Plane gedreht werden, um eine bessere Vergleichbarkeit zu gewährleisten. Dieser Vorgang dreht

alle Gesten-Punktwolken anhand ihrer jeweiligen Action Plane in eine gemeinsame Action Plane, so dass alle Gesten nach der Rotation dieselbe Action Plane haben. Um diese Ebene sind die Daten immer noch gestreut, die Lage der einzelnen Datenpunkte zur Action Plane der jeweiligen Geste wird in diesem Schritt nicht verändert.

Hypothese 5.5 (Action Plane - Rotation)
Die Klassifikation der Gestendaten verbessert sich durch die vorherige Rotation anhand der Action Plane der Daten in eine gemeinsame Ebene.

Anhand des Normalenvektors der Ebene, die durch die Methode der Singulärwertzerlegung errechnet wurde, wird die erstellte Ebene auf die [1,1,1]-Ebene gedreht, damit alle Gesten die gleiche Lage im Raum besitzen.

Der Normalenvektor:

$$\hat{n} = (n_1, n_2, n_3)^T \tag{5.6}$$

wird in der Drehmatrix:

$$R_{\hat{n}}(\alpha) = \begin{pmatrix} n_1^2(1-\cos\alpha)+\cos\alpha & n_1 n_2(1-\cos\alpha)-n_3\sin\alpha & n_1 n_3(1-\cos\alpha)+n_2\sin\alpha \\ n_2 n_1(1-\cos\alpha)+n_3\sin\alpha & n_2^2(1-\cos\alpha)+\cos\alpha & n_2 n_3(1-\cos\alpha)-n_1\sin\alpha \\ n_3 n_1(1-\cos\alpha)-n_2\sin\alpha & n_3 n_2(1-\cos\alpha)+n_1\sin\alpha & n_3^2(1-\cos\alpha)+\cos\alpha \end{pmatrix} \tag{5.7}$$

verwendet, um die Ebene um den Winkel zwischen der gewünschten Zielebene (in diesem Fall die X-Y-Ebene) und der Action Plane der jeweiligen Geste, zu drehen.

5.6.3 Projektion der Geste auf die Action Plane

Im Weiteren wird diese Vorverarbeitung der Gestendaten mit $\boldsymbol{p}$ beschrieben.

Wurde die Action Plane bestimmt, so können alle Punkte der Geste mit der folgenden Gleichung 5.8 auf diese Ebene projiziert werden. $\hat{n}$ ist der Normalenvektor der Action Plane und $\vec{x}$ der zu projizierende Punkt.

$$\vec{x}_p = \vec{x} - \frac{\vec{x}\cdot\hat{n}}{\hat{n}\cdot\hat{n}}\,\hat{n} \tag{5.8}$$

Das Ergebnis ist eine Geste, deren Koordinaten alle in einer Ebene liegen. Ziel der Projektion ist es, die Gesten möglichst vergleichbar zu gestalten, ohne ihre formgebenden Eigenschaften zu vernachlässigen.

Hypothese 5.6 (Action Plane - Projektion)
Die Erkennungsrate verbessert sich durch das Projizieren auf die (gemeinsame) Action Plane, da die Gesten vergleichbarer werden.

5.7 Verwendung der Geschwindigkeitsdaten

Ein weiterer Featureansatz ist die Verwendung der Geschwindigkeitsdaten, welche aus den aufgenommenen Positionsdaten berechnet werden müssen. Der für diesen Ansatz entwickelte Featurevektor besteht aus den Geschwindigkeiten in den einzelnen Ortskoordinatenkomponenten (siehe 5.9).

$$\vec{F} = (v_x, v_y, v_z)^T \tag{5.9}$$

Hypothese 5.7 (Geschwindigkeitsfeatures)
Die Gesten lassen sich anhand ihrer dreidimensionalen Geschwindigkeitsdaten klassifizieren.

Anhand der Geschwindigkeitseigenschaften einer Geste, welche aus den Positionsdaten und Zeitdifferenzen gebildet werden, lässt sich ein Hidden Markov Modell trainieren und auswerten. Die Symbole sind, wie bei den Positionsfeatures ebenfalls, Clusterzentren in der Punktwolke. In diesem Fall demnach Positionen im dreidimensionalen Raum, wobei auf den Koordinatenraumachsen die Geschwindigkeiten der Geste aus ihren Positionskomponenten aufgetragen sind.

Bei vielen ähnlichen Gesten (beispielsweise Ellipse und Kreis) wird die Unterscheidung allein durch ihre Geschwindigkeitsprofile vermutlich nicht möglich sein. Für die hier verwendeten Gesten ist dies jedoch möglich (Hypothese 5.7).

5.8 Kombination der Ergebnisse der Positions- und Geschwindigkeits-Modelle

Da manche Gesten in ihren Positionsdaten sehr ähnlich sind, soll die Eigenschaft der Gesten hinzugezogen werden, welche diese unterscheidet. Die Geschwindigkeiten der formähnlichen Gesten (Kreis und Quadrat), wie in Abbildung 5.5 dargestellt, unterscheidet sich deutlich. Damit die Absolutgeschwindigkeit, welches nur ein weiteres Feature ist, in der Kombination mit den anderen drei Features (x, y, z)-Koordinaten, nicht untergeht, sollen zwei HMMs gebildet und ihre Ergebnisse im Anschluss kombiniert werden. Es werden demnach zwei Modelle gebildet mit den Featurevektoren:

$$F_1 = (x, y, z)^T \tag{5.10}$$

$$F_2 = (v_{absolut}) \tag{5.11}$$

Die Zeitkomponente in 5.11 ist die Zeit in Nanosekunden normiert auf eine Länge von 100 Zeiteinheiten, um die Geschwindigkeiten der verschiedenen Gesten vergleichbar zu visualisieren. In der Erstellung und Auswertung der HMMs wird die Zeit nicht berücksichtigt.

Hypothese 5.8 (Kombination der Position- und Geschw.-Ergebnisse)
Kann durch das Hidden Markov Modell über die Positionsdaten keine eindeutige Entscheidung getroffen werden, hilft die Kombination mit den Ergebnissen des Geschwidigkeitsmodells, um die Geste eindeutiger bestimmen zu können.

Die einzelnen HMMs werden, wie in den vorherigen Kapiteln beschrieben, benutzt. Allein die Auswertung ist in diesem Fall eine andere. Die Auswertung erfolgt anhand der Kombination beider Ergebnisse. Von jeder Geste auf jeder HMM werden die jeweils zusammengehörigen Ergebnisse (Log-Likelihoods der Geste auf den entsprechenden HMMs und die dazugehörigen Thresholds) addiert. Dadurch entsteht:

$$Threshold_{new} = Threshold_{positions} + Threshold_{speeds} \tag{5.12}$$

$$LLH_{new} = LLH_{positions} + LLH_{speeds} \tag{5.13}$$

Wie bei allen vorherigen vorgestellten Modellen, gilt eine Geste als erkannt, wenn:

$$LLH_{new} > Threshold_{new} \tag{5.14}$$

gilt.

Diese simple Verknüpfung wurde gewählt, da sich im Verlauf der Tests zu den Positions-, sowie Geschwindigkeits-Modellen ergab, dass der Vergleich der LLHs, wenn diese überschwellig sind, die beste Erkennungsrate liefert. Bei diesem Vergleich ist die Geste auf der HMM mit dem größeren LLH somit besser erkannt worden. Durch diese Kombination können ebenfalls Gesten erkannt werden, welche auf einer HMM allein „nicht erkannt" wurden, wenn diese auf der jeweils anderen HMM einen hohen LLH-Wert (LLH in der Nähe von 0) erzielten. Die Ergebnisse dieser Kombination sind in Kapitel 6.5 zu finden.

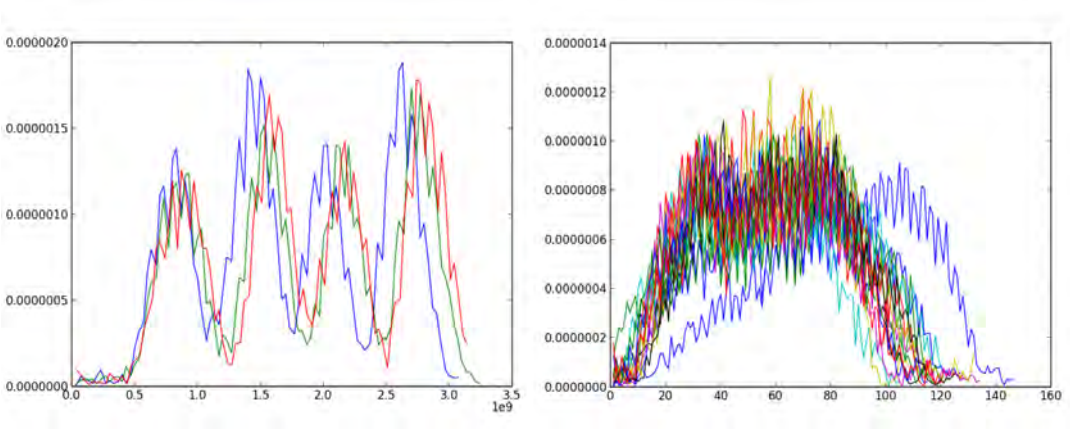

Abbildung 5.5: Visualisierung der absoluten Geschwindigkeiten von „Squares" (links) und „Circles" (rechts)

5.9 Kombination der Ortskoordinaten mit der Absolutgeschwindigkeit

Das Hidden Markov Modell, gebildet auf den Positionsdaten der Gesten, unterliegt gewissen Beschränkungen. Gesten, welche in ihrer Ausführung ähnliche Positionen beinhalten, unterliegen einer hohen Verwechslungsgefahr.
Die Problematik, Gesten anhand ihrer Geschwindigkeiten zu verwechseln ist ebenfalls bei ähnlichen Abläufen vorhanden, jedoch unterscheiden sich formähnliche Gesten oftmals in ihren Geschwindigkeiten in den einzelnen Abschnitten einer Geste. Ein Kreis ist beispielsweise in seiner Form (und vor allem in der Art seiner trainierten Observable Symbols) dem Quadrat sehr ähnlich. Wird die Ausführung der beiden Gesten beobachtet, kann festgestellt werden, dass sich das Geschwindigkeitsprofil maßgeblich unterscheidet. Sind Ecken oder Richtungsänderungen in Gesten beziehungsweise Formen vorhanden, so bremst der Nutzer bei der Ausführung der Geste ab, um ihre Hand

in eine andere Richtung weiter zu bewegen.

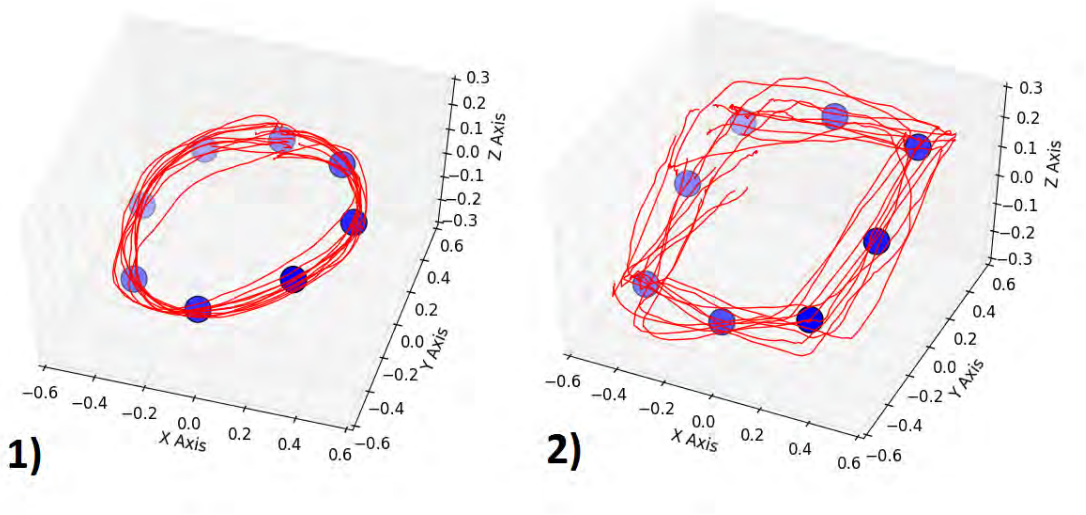

Abbildung 5.6: Verdeutlichung der Verwechslungsgefahr von Kreisen und Quadraten anhand der Visualisierung der K-Means bzw. Observable Symbols eines „Circles"(1) und eines „Squares" (2)

Ein weiteres wichtiges Feature könnte demnach die absolute Geschwindigkeit in den Ortskoordinaten sein. Um dies zu analysieren, wird in diesem Test der vierdimensionale Featurevektor:

$$\vec{F} = (x, y, z, v_{absolut})^T \tag{5.15}$$

gebildet und für das Erstellen und Testen des HMMs genutzt.
Das Vorgehen in diesem Test entspricht dem Vorgang des Positionsdaten-Feature-Tests. Auch im vierdimensionalen Raum können K-Means zur Bildung der Output Symbole verwendet werden. Damit $v_{absolut}$ im Vergleich zu den X, Y, und Z-Koordinaten nicht untergeht, oder überdimensioniert wirkt, wird die Länge der $v_{absolut}$-Achse auf die Länge der längsten der drei anderen Achsen skaliert. Der zugehörige Code ist im Anhang unter 8 zu finden.

5.10 Phasenraum

Durch die Hinzunahme der partiellen Ableitungen in den Ortskoordinaten nach der Zeit, wird der Featurevektor um drei Dimensionen erweitert. Wie bereits in den vorherigen Kapiteln beschrieben, ähneln sich manche Gesten in ihrer Form, jedoch un-

terscheiden sich die Geschwindigkeiten formähnlicher Gesten durchaus. Durch die einzelnen Geschwindigkeitskomponenten werden feingranularere Aussagen bezüglich der Geschwindigkeiten, im Gegensatz zum Featurevektor des vierdimensionalen Raums, getroffen. Der Featurevektor des Phasenraums lautet:

$$\vec{F} = (x, y, z, v_x, v_y, v_z)^T \tag{5.16}$$

Die Clusterzentren werden ebenfalls durch den Algorithmus der K-Means für den sechsdimensionalen Raum bestimmt, um das Symbolalphabet zu erstellen. Es wird angenommen, dass durch diese Verfeinerung der enthaltenen Informationen im Featurevektor, nicht nur die einzelnen Gesten besser unterscheidbar machen, sondern ebenfalls personenbezogene Gestencharakteristika durch die Beschreibung der Trajektorie durch den Featurevektor des Phasenraums auftreten.

Hypothese 5.9 (personenbez. Erkennung im Phasenraum)
Die Person kann anhand ihrer charakteristischen Trajektorie im Phasenraum wiedererkannt werden.

6 Durchführung von Tests

Die Wirksamkeit der in den vorherigen Kapiteln dargestellten, auf die Forschungsfragen hin entworfene, Kombination aus Features mit passenden Verarbeitungsmethoden werden in diesem Kapitel getestet.

Dabei gilt es die Anforderungen, die mit der Untersuchung von personenübergreifenden (Wird die Geste einer Person auf den gleichen Gestendaten anderer Personen erkannt?) und personenbezogenen (Wird die Geste einer Person anhand von eigenen gleichen, aber nicht denselben, Gestendaten erkannt?) Aspekten verbunden sind, zu berücksichtigen. Bei personenbezogenen Aspekten kann eine Testung nur vorgenommen werden, wenn mindestens 15 Durchführungen pro Geste zur Verfügung gestellt wurden, so dass diese Daten in Trainings- und Testdaten aufgeteilt werden können. Es müssen mindestens 10 Datensätze vorhanden sein, um ein HMM zu trainieren. Des Weiteren werden ausreichend Testdaten benötigt (mindestens 5, welche von den 10 Trainingsdaten verschieden sein müssen), um das trainierte Modell in einem ausreichenden Umfang zu testen.

6.1 Testaufbau

Es werden Tests mit den eigenen aufgenommenen Daten, aufgeteilt in Trainings- und Testdaten, sowie mit den Daten von Hall durchgeführt, um Referenzwerte zu erhalten.

In einem ersten Schritt wird festgelegt, welche Daten für das Training der HMM genutzt und welche für das abschließende Testen der Erkennungswirkung der trainierten HMMs verwendet werden. Die strikte Trennung dieser Gestendaten ist für die Aussagekraft über die Qualität der Erkennung maßgeblich. Die Daten von [11] beinhalten jeweils Daten derselben Person, was bedeutet, dass in den Daten von Hall keine Trennung der Trainings- und Testperson vorliegt. Die Training- und Testdaten sind sich sehr ähnlich, jedoch voneinander unterschiedlich, sodass die Ergebnisse dieser Daten dennoch als Referenzauswertung des HMMs betrachtet werden können.

Aufteilung der Daten für personenübergreifende Tests:

Training Die Daten der Personen 10-19 werden verwendet. Von jeder Person werden 5 Gesten genutzt, so dass der Trainingsdatensatz insgesamt 50 Datensätze pro Geste enthält.

Testen Die Daten der Personen 0-9 werden verwendet. Insgesamt sollen 50 Datensätze pro Geste verwendet werden. Aus den Daten der Personen 0-9 werden dazu 5 Gestendaten pro Person verwendet.

Pro Geste sollen mindestens 10 Ausführungen der gleichen Geste für das Training der HMM verwendet werden. Da die Hälfte aller Personen zum Trainieren, sowie Testen, verwendet werden kann, sind weitaus mehr als 10 Ausführungen der Trainingsdaten vorhanden. Pro Person werden 5 Gestendatensätze pro Geste verwendet, wo eine Gesamtmenge von 50 Trainings- und Testgesten pro Geste entsteht.

Bei der Aufteilung der personenbezogenen Tests ist zu beachten, dass pro Person in jedem Durchlauf 15 Gestendaten der gleichen Geste verwendet werden. Jeder Durchlauf wird drei mal mit anders ausgewählten Trainings und Testdaten durchgeführt.
Aufteilung der Daten für personenbezogene Tests:

Training Jeweils 10 Gestendaten der Personen 0, 1, 10 und 18 werden verwendet. Pro Person wird ein eigenes HMM auf den jeweiligen Trainingsdaten gebildet.

Testen Von den Personen 0, 1, 10 und 18 werden jeweils 5 Gestendaten verwendet.

> Anmerkung: Um ein personenbezogenes Hidden Markov Modell zu erstellen, müssen genügend (mindestens 10 Datensätze einer Geste zum Trainieren des Modells und davon unterschiedliche zum Testen) Testdaten einer Person vorhanden sein.

Damit bei den personenbezogenen Tests trotz geringer Anzahl an Gestendaten ein verlässlicherer Wert für die Qualität des Modells entsteht, werden bei diesen Tests einfache Kreuzvalidierungen durchgeführt. Dabei wird die pro Person zur Verfügung stehende Datenmenge, bestehend aus N Elementen pro Geste, aufgeteilt in k gleich große Mengen. Da bei diesen Tests jeweils $N = 15$ Gestendaten pro Person pro Geste genutzt werden, liegt die Aufteilung in drei Mengen mit jeweils $k = 5$ Elementen nahe. Diese Einteilung ist besonders praktisch, da mindestens 10 Gestendaten zum Training verwendet werden sollen, was durch diese Aufteilung gegeben ist. Fünf Gestendaten werden pro Durchlauf zum Testen verwendet.

In Abbildung 6.2 ist die Aufteilung der Daten für eine Kreuzvalidierung einer Person für drei Durchläufe dargestellt. Diese Aufteilung wird für jede Geste mit ihren entsprechenden Vorverarbeitungsschritten pro Person durchgeführt. Für vier Gesten mit

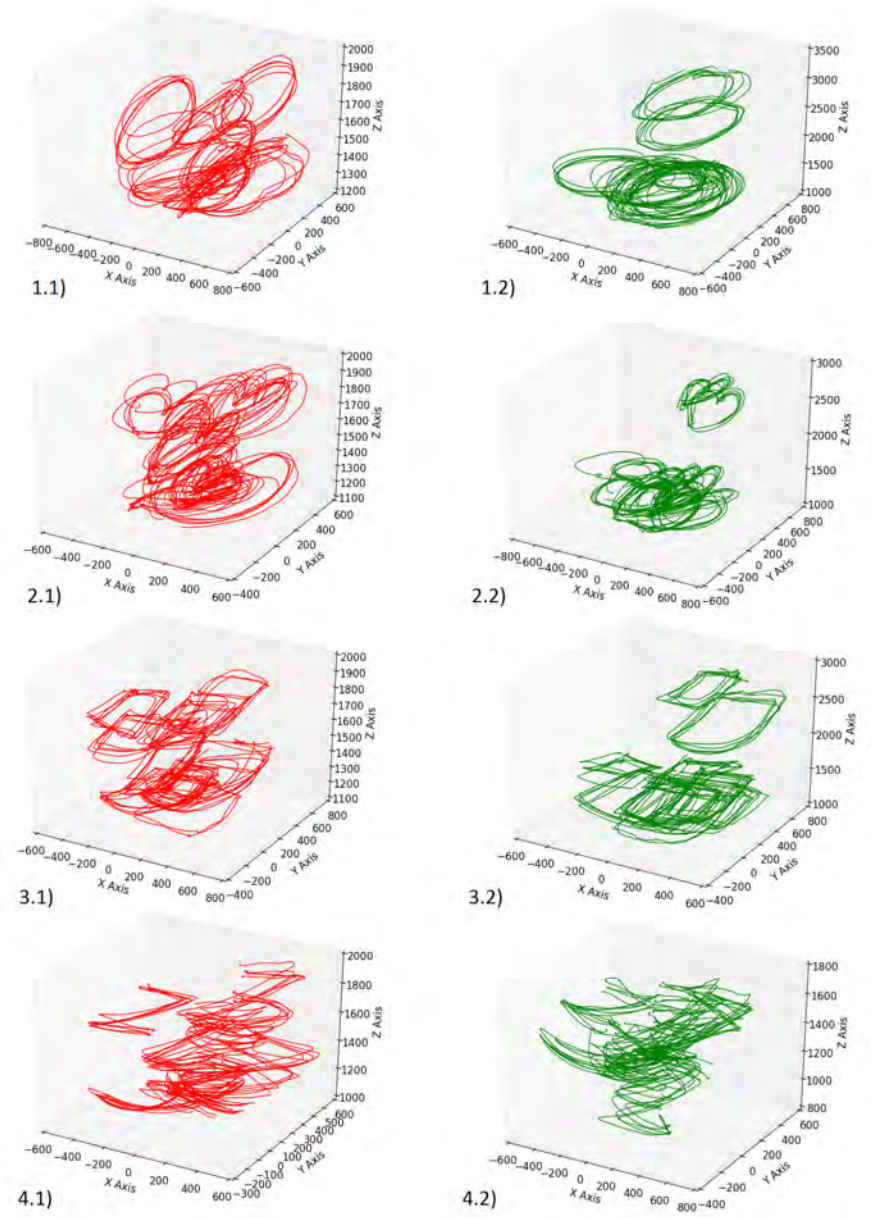

Abbildung 6.1: Ansicht aller personenübergreifenden Trainigsdaten (rot) von Person 10-19 und Testdaten (grün) von Person 0-9. 1) „Circle"; 2) „Heart"; 3) „Square"; 4) „Z"

jeweils drei Kreuzvalidierungsdurchläufen und fünf verschiedenen Vorverarbeitungs-Varianten, ergibt dies 60 Testdurchläufe mit jeweils 10 Result-Files (150 Result-Files pro personenbezogenem Gesten-Test).

Variablen, die jeweils für die folgenden Tests festgelegt werden müssen:

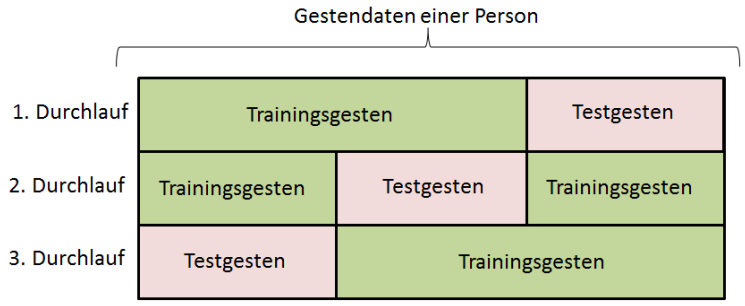

Abbildung 6.2: Verdeutlichung der Anwendung der Kreuzvalidierung für die Gestendaten der personenbezogenen Tests.

1. Welche Featurevektoren verwendet werden (Positionsfeatures, Positions- und Geschwindigkeitsfeatures, $(x, y, z, v_{absolut})$-Featurevektor, Phasenraumfeatures).
2. Welche Daten für das Training und Testen verwendet werden.
3. Anzahl der Output Symbols.
4. Anzahl der Hidden States.
5. Welcher LR-Parameter wird gewählt (ist dieser = Anzahl Zustände, ist es ein semiotisches Modell).
6. Welcher Threshold-Parameter t gewählt wird; Initial ist dieser auf $t = 2$ gesetzt.
7. Welche Verarbeitungsschritte vorher auf die Daten angewendet werden. Die einzelnen Vorverarbeitungsschritte (genauer beschrieben in Kapitel 5 werden NUR auf die Positionsdaten angewendet aus denen ggf. im Folgenden Geschwindigkeitsdaten berechnet werden) werden wie folgt abgekürzt:

   nichts Die Daten werden unverändert verwendet.

   c Die Daten werden vorher auf ihre Drehrichtung angepasst. Ist c in der Vorverarbeitung enthalten, sind die Gesten danach alle mit dem Uhrzeigersinn gedreht (siehe 5.1.2).

   zc Aufbauend auf c werden die Daten in diesem Schritt ebenfalls auf einen gemeinsamen Mittelpunkt (den Koordinatenursprung) zentriert (siehe 5.5.1).

   zcs Ebenfalls aufbauend zu der Verarbeitung zc werden die Daten zusätzlich auf eine gemeinsame Größe skaliert (siehe 5.5.2).

   zcrs Die Daten werden zusätzlich in eine gemeinsame Action Plane rotiert (siehe 5.6.2).

zcrps Der letzte hier verwendete Vorverarbeitungsschritt ist das Projizieren der Daten in ihre Action Plane (siehe 5.6.3).

8. Wie viele Durchläufe desselben Tests durchgeführt werden, um über die Ergebnisse zu mitteln.

Die trainierten HMMs werden für den Durchlauf des Tests gespeichert und pro Testdurchlauf „Result-Files" (siehe 6.1) mit den Ergebnissen angelegt. In den Result-Files werden pro Zeile folgende Ergebnisse gespeichert:

Spalte 1+2 Angabe darüber, mit welchen Gestendaten was getan wurde. In Spalte 1+2 ist die Angabe der Trainingsgeste vorgesehen.

Spalte 3+4 Die Angabe welche Geste getestet wurde, befindet sich in dieser Spalte.

Spalte 5 Angabe der Likelihoods der getesteten Geste auf der HMM der trainierten Geste.

Spalte 6 Angabe des Thresholds der verwendeten HMM.

Spalte 7 Index der Testgeste des Testdatensatzes der gleichen Geste. Werden 10 Test-Kreise angegeben, wird von $0-9$ inkrementiert.

Spalte 8 REC oder NOT: Angabe ob die Testgeste erkannt wurde, oder nicht.

Spalte 9 Angabe des „einfachen" Thresholds θ_1 der HMM, für einen Threshold-Parameter von $(t = 2)$.

```
Training,Circle,Testing,Circle,-29.469502,-86.513243,0,REC,-43.25662
Training,Circle,Testing,Circle,-48.866654,-86.513243,1,REC,-43.25662
Training,Circle,Testing,Circle,-41.383365,-86.513243,2,REC,-43.25662
...
Training,Circle,Testing,Heart,-3499.9293,-86.5132436,0,NOT,-43.25662
Training,Circle,Testing,Heart,-4006.4980,-86.5132436,1,NOT,-43.25662
Training,Circle,Testing,Heart,-3776.2395,-86.5132436,2,NOT,-43.25662
...
```

Abbildung 6.3: Beispiel für ein Result-File. Die Genauigkeit des LLHs und des Thresholds, bzw. die Anzahl der Nachkommastellen, ist in den genutzten Result-Files höher.

Die Result-Files werden im Anschluss genutzt, um die Ergebnisse zu analysieren. Die Analyse wird mittels „Recognition-Matrizen"(6.1) und „Confusion-Matrizen" durchgeführt. Die Recognition Matrix ist, wie in Tabelle 6.1 angegeben, aufgebaut.

	Test-Geste 1	Test-Geste 2	Test-Geste 3	Test-Geste 4
Train-Geste 1	1.0	0	0	0
Train-Geste 2	0	1.0	0	0
Train-Geste 3	0	0	1.0	0
Train-Geste 4	0	0	0	1.0

Tabelle 6.1: Aufbau der Recognition-Matrix

In der Recognition Matrix wird der Prozentsatz, normiert auf 1, der überschwelligen Erkennungen notiert. Die Zeilen stehen für die verwendete HMM der trainierten Geste. Die Reihen zeigen die auf die HMMs getestete Geste. In einem optimalen Erkennungsfall sollten nur Einträge auf der Diagonalen sichtbar sein. Werden Testgesten auf anderen Trainings-Gesten erkannt so würde auch in der entsprechenden Zeile und Spalte die Zahl der „falsch" erkannten Gesten eingetragen.

Aufbauend auf den Result-Files werden ebenfalls drei verschiedene Varianten von Konfusionsmatrizen gebildet:

Variante 1 Bestimmt für jedes Paar, bestehend aus Trainings- und Testgeste, die absolut größte Log-Likelihood.

Variante 2 Bestimmt für jedes Paar, bestehend aus Trainings- und Testgeste, das mit dem größten Abstand zwischen LLH der Testgeste und Threshold θ_t des HMM.

Variante 3 Bestimmt für jedes Paar, bestehend aus Trainings- und Testgeste, das mit dem größten Ratio $\frac{Abstand}{|Threshold|}$

Ist der LLH der Testgeste $> \theta_t$, wird der entsprechende Eintrag, für den LLH nach Variante 1, 2 oder 3 am Größten ist, in der jeweiligen Konfusionsmatrix um 1 inkrementiert. Ist die LLH kleiner als θ_t jeder Trainingsgeste, so wird in der Zeile „unknown" die entsprechende Spalte inkrementiert.
Zum Auswerten der Konfusionsmatrizen werden im Anschluss noch die Genauigkeit (Accuracy), die „false unknown rate" und die „false positive rate" bestimmt. Die Accuracy bildet sich durch die Addition der Diagonalelemente der Konfusionsmatrix, dividiert durch die Anzahl der Gesten (vier). Die false unknown rate ist die Addition der als „unbekannt" klassifizierten Gesten ebenfalls dividiert durch die Anzahl an verwendeten Gesten. Die Bestimmung der false positive rate läuft ähnlich ab. Alle Elemente der Matrix, welche nicht auf der Diagonalen oder in der Spalte „unknown" liegen, werden addiert und durch die Anzahl der Gesten dividiert. So erhält man eine Aussage über die Qualität der Erkennung.

6.2 Anzahl der Iterationen der HMM

Bei der Erstellung des HMM werden mehrere Durchläufe ausgeführt, um die Parameter des Modells zu verfeinern. Das Modell kann mit festgelegten Anzahl an Durchläufen trainiert oder durch eine Abbruchbedingung, welche dann abbricht, wenn die Parameter der HMM konvergieren. Die Tests haben ergeben, dass für die in dieser Arbeit verwendeten Trainingsdaten eine Anzahl von 20 Iterationen ausreichend ist, da die Parameter bereits zwischen 5 und 15 Iterationen konvergieren, was in Abbildung 6.4 deutlich wird.

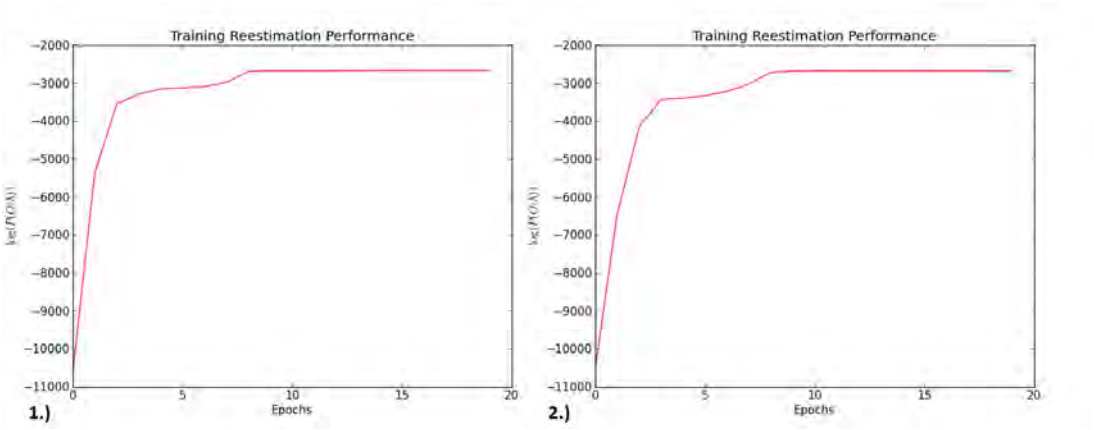

Abbildung 6.4: Beispielhafte Visualisierung der Iterationsanzahl des HMM. 1.) „Square" für die Verarbeitung: zcs; 2.) „Z" für den Phasenraum ohne Vorverarbeitung.

6.3 Positionsfeature Tests

Die Positionsfeature-Tests werden auf dem dreidimensionalen Feature-Vektor, bestehend aus Positionsdaten (x, y, z), welche von der Kinect aufgenommen wurden, durchgeführt. Die folgenden Tests werden mit den eigens aufgenommenen Gestendaten, sowie mit den von [11] zur Verfügung gestellten Gestendaten durchgeführt (allerdings ohne die „L"-Geste, da von dieser Geste Daten fehlen).

6.3.1 Test zum Left-to-Right Parameter

Der Test zum Left-to-Right Parameter bezieht sich auf die Hypothese 5.2. Es wird angenommen, dass bei unterschiedlichen Features, ein angepasster LR-Parameter eine höhere Genauigkeit bei der Erkennung liefert, als für alle Featurevektoren einen $LR = 2$ zu verwenden. In diesem Test wird der LR-Parameter bezüglich der Positionsdaten-Features analysiert.

	Accuracy(STD±AVG)	False unknown rate	False positive rate
LR=2	**78%±4,57602**	**2,50%**	**19,60%**
LR=3	73,80%±9,1554	4,20%	22,10%
LR=5	71,80%±5,08945	4,60%	23,70%
LR=8	74,20%±6,7247	5,60%	20,20%

Tabelle 6.2: Positionfeatures Left-to-Right-Tests: (zcs)

In diesem Test wurde die Vorverarbeitung ***zcs*** gewählt, da diese eine gewisse Vergleichbarkeit der Daten erzeugt(und wie sich in weitere Test herausgestellt hat, ebenfalls gute Ergebnisse liefert) und somit erwartet wird, dass die Ergebnisse zuverlässige Aussagen zu den verschiedenen verwendeten LR-Parametern liefern. In Tabelle 6.2 wird dargestellt, wie sich die Accuracy, FUR[1] und FPR[2] für die verschiedenen LR-Parameter verhalten. Auffällig ist, dass der Unterschied nicht besonders groß, aber dennoch vorhanden ist. Zwischen den Ergebnissen der LR-Faktoren größer als zwei, ist kein großer Unterschied festzustellen, liefern jedoch alle eine schlechtere Erkennungsgenauigkeit, als $LR = 2$. Dementsprechend wird in den weiteren Tests der LR-Parameter auf 2 gesetzt.

6.3.2 Variation der Anzahl an Observable Symbols und Hidden States

> Anmerkung: Im Folgenden werden die Anzahl der Hidden States mit n und die Anzahl der Observable Symbols mit k abgekürzt.

Die Anzahl der Observable Symbols und Hidden States kann variiert werden. In Hypothese 5.3 wurde angemerkt, dass für verschiedene Kontexte eventuell andere Anzahlen an Observable Symbols k und Hidden States n angemessen sind. In diesem Test wird die Variation der Symbole und Hidden States für den Positionsfeaturevektor geprüft. Eine weitere Überlegung ist, dass durch die Vorverarbeitung eine gewisse Vergleichbarkeit geschaffen sein muss, damit die Symbole, angelegt durch die Clusterzentren, überhaupt eine Aussagekraft bekommen. Wie in Abbildung 6.5 dargestellt, ist für $k = 4$ und $n = 6$, angewendet auf den unverarbeiteten Rohdaten, keine deutliche Aussagekraft über gestenspezifische Eigenschaften erkennbar, da die Clusterzentren zwischen, inmitten und um die Gesten herum verteilt sind, da diese nicht auf einen gemeinsamen Mittelpunkt normiert sind. Dementsprechend werden zur Auswertung des Einflusses von k und n die Ergebnisse ab der Vorverarbeitung *zc* betrachtet.

In Tabelle 6.3 wird dargestellt, wie sich die Ergebnisse für die Veränderungen der Hidden States und Observable Symbols verändern. Die besten Ergebnisse sind hervor-

[1] false unknown rate
[2] false positive rate

	n/k	Accuracy(AVG±STD)	FUR	FPR
nichts	4/6	47,00%±3,7178%	6,80%	46,30%
	6/9	41,90%±3,868139%	8%	50,20%
	8/12	40,35%±3,03356%	19%	40,65%
	12/18	46,10%±3,2878%	13,80%	40,20%
c	4/6	41,10%±7,49733%	8,40%	50,50%
	6/9	42,30%±2,8178%	11,50%	46,30%
	8/12	44%±4,49471%	10,50%	45,50%
	12/18	44,90%±2,82842%	13,10%	42,10%
zc	4/6	68,50%±8,2674%	2,80%	28,70%
	6/9	74,20%±4,8518%	3%	22,80%
	8/12	75%±5,30117%	3,80%	21,30%
	12/18	74,30%±4,388622%	4,50%	21,20%
zcs	4/6	71%±6,682%	3%	26%
	6/9	74,10%±5,82666%	4,50%	21,50%
	8/12	**78%±4,57602%**	**2,50%**	**19,60%**
	12/18	76,00%±3,62077%	5,20%	18,90%
zcrs	4/6	68,20%±12,12322%	3,20%	28,70%
	6/9	78,20%±6,403%	4,10%	17,70%
	8/12	76,30%±1,59765%	3,60%	20,10%
	12/18	**79,30%±3,2821%**	**5%**	**15,80%**
zcrps	4/6	74,20%±6,97872%	2,80%	23,10%
	6/9	73,70%±4,4508%	4,10%	22,20%
	8/12	**75,40%±5,07050%**	**2,40%**	**22,20%**
	12/18	74,90%±4,86029%	5%	20,10%

Tabelle 6.3: Vergleichstest mit verschiedenen Observable Symbols und Hidden States

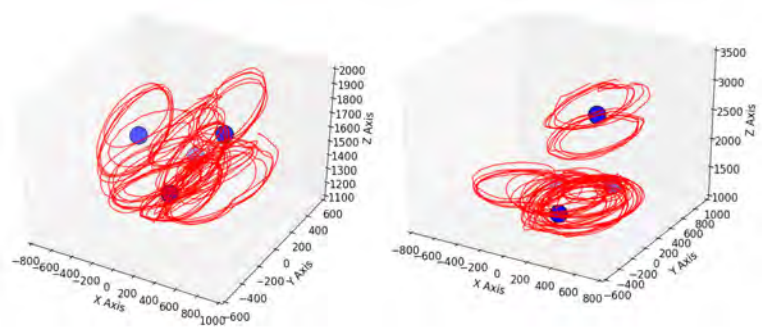

Abbildung 6.5: Unveränderte Trainingsdaten (links) und Testdaten (rechts) der „Circle"-Geste mit ihren Clusterzentren bzw. Observable Symbols ($k = 4$, $n = 6$).(Anmerkung: Es sind jeweils vier Zentren (blau) visualisiert, jedoch werden manche sehr stark verdeckt.)

gehoben. Es ist auffällig, dass die besten Ergebnisse bei der Vorverarbeitung *zcs*, *zcrs* und *zcrps* entstehen. Für die Anzahlen $k = 8$ und $n = 12$, welche am häufigsten die

höchste Accuracy aufweisen, ist das Ergebnis der Verarbeitung mit *zcs* das mit der höchsten Genauigkeit. Da die Ergebnisse gewissen Schwankungen unterliegen (erkennbar an der Standardabweichung) wurde für den weiteren Verlauf der Positionsfeature Tests $k = 8$ und $n = 12$ gewählt, obwohl $k = 12$ und $n = 18$ für *zcrs* das gesamt beste Ergebnis lieferte. Dieses Ergebnis wird bei den Threshold-Tests unter 6.8.1 noch einmal aufgegriffen.

6.3.3 Personenübergreifende Tests

Bei den personenübergreifenden Tests werden zu den wie in Kapitel 6.1 beschriebenen Trainings- und Testdatensätzen ebenfalls die Gestendaten von Hall-Daten hinzugezogen, so wie sie von Hall zusammengestellt und wie diese bereits in Kapitel 4.1 beschrieben wurden.

Zu aller erst, wird die aufgestellte Hypothese 5.1 über die Wichtigkeit der vergleichbar gedrehten Gesten getestet, da alle weiteren Tests darauf aufbauen sollen. In Tabelle 6.5 wird gezeigt, dass die Drehrichtung der Gesten tatsächlich einen Einfluss auf die Erkennung hat. Da jedoch nur wenige Gesten gegen den Uhrzeigersinn im Testdatensatz vorhanden sind (31 von 200), ist die Auswirkung nicht besonders hoch. Im Trainingsgestendatensatz sind:

- 5 Square CCW[3]
- 0 Circle CCW
- 0 Heart CCW

Im Testdatensatz sind:

- 15 Square CCW
- 5 Circle CCW
- 11 Heart CCW

In der Tabelle 6.4 wird dargestellt, wie viele Gesten auf wie vielen HMMs als überschwellig erkannt werden. Dabei wird deutlich, dass Circle und Square häufig verwechselt werden. Die beiden Gesten werden auf dem eigenen, sowie auf dem jeweils anderen HMM häufig als überschwellig klassifiziert. Dazu kommt dass die Log-Likelihood der Kreise auf den Quadrat-HMMs, sowie der Quadrate auf den Kreis-HMMs oft besser ist, als auf der eigenen HMM. Die Gesten sind in ihren Ortskoordinaten sehr ähnlich, was zu ähnlichen Clusterzentren bei der Bildung der Observable Symbols führt. Dadurch ist

[3] CCW: counter clockwise

c	Circle	Heart	Square	Z
Circle	0,718	0,314	0,7	0,45
Heart	0,246	0,71	0,372	0,46
Square	0,656	0,484	0,684	0,692
Z	0,104	0,242	0,25	0,564

zc	Circle	Heart	Square	Z
Circle	0,874	0,022	0,716	0,018
Heart	0,036	0,918	0,04	0,004
Square	0,988	0,098	0,898	0,434
Z	0	0	0	0,95

zcs	Circle	Heart	Square	Z
Circle	0,976	0,008	0,73	0,008
Heart	0	0,87	0	0
Square	0,988	0,15	0,872	0,528
Z	0	0	0	0,886

zcrs	Circle	Heart	Square	Z
Circle	0,946	0,006	0,738	0,01
Heart	0	0,882	0	0
Square	0,97	0,128	0,862	0,46
Z	0	0	0	0,952

zcrps	Circle	Heart	Square	Z
Circle	0,934	0,004	0,73	0
Heart	0	0,892	0	0,024
Square	0,97	0,17	0,826	0,266
Z	0	0	0	0,96

Tabelle 6.4: Recognition Matritzen der Positionfeature-HMMs für alle Gesten auf allen HMMs

eine hohe Verwechslungsgefahr gegeben. Positive Erkennungen werden beispielhaft in der Tabelle 6.4 mit den Vorverarbeitungsschritten *zc* grün dargestellt, schlechte rot und der nur sehr knappe Unterschied zwischen der Häufigkeit von überschwelligen Squares auf der Square-HMM und auf der Circle-HMM wird magentafarben dargestellt. Heart und Z werden weniger häufig auf anderen HMMs als überschwellig klassifiziert. In der Tabelle 6.6 und der zugehörigen Abbildung 6.6 wird der Verlauf der „Recoginition" für die einzelnen Gesten aufgezeigt.

In Tabelle 6.5 werden die verschiedenen Varianten der Konfusionsmatrixbildung, beispielhaft für die Vorverarbeitung mit *zcs* dargestellt. Die ausführlicheren Variantentests-Ergebnisse für alle Vorverarbeitungsschritte sind im Anhang unter 8 in den Tabellen 8.2, 8.3 und 8.4 zu finden. Wie sich die Klassifikation der Varianten zusammensetzt wird in Kapitel 6.1 beschrieben. Durch diese Auswertung wird deutlich, dass Variante 1, der Vergleich der LLH auf dem Threshold θ_t der HMM, die beste Genauigkeit, sowie niedrigste false positive rate liefert. Die false unknown rate bleibt bei den verschiedenen Varianten auf einem vergleichbaren Niveau. Deshalb wird diese Klassifikationsmethode nach Variante 1 für die folgenden Tests verwendet.

Trotz Anpassung der Parameter und verschiedener Vorverarbeitungsschritte wird für den Test mit ähnlichen Gesten (Kreis und Quadrat) maximal eine Erkennungsrate von 78% erzielt. Für die Unterscheidung von sehr ähnlichen Gesten, bezogen auf ihre Ortskoordinaten, können demnach weitere Features hinzugezogen werden, um die Genauigkeit zu verbessern. Bei ausreichend unterschiedlichen Gesten, wie bei Hall, ist

		Accuracy	FUR	FPR
nichts	Variante 1	40,35%	19%	40,65%
	Variante 2	40%	7,30%	52,70%
	Variante 3	41%	7,30%	51,70%
c	Variante 1	44%	10,50%	45,50%
	Variante 2	36,20%	9,60%	54,20%
	Variante 3	39,10%	9,60%	51,30%
zc	Variante 1	75%	3,80%	21,30%
	Variante 2	62,20%	3,60%	34,20%
	Variante 3	66,90%	3,60%	29,50%
zcs	**Variante 1**	**78%**	**2,50%**	**19,60%**
	Variante 2	61,20%	2,30%	36,50%
	Variante 3	67,50%	2,30%	30,30%
zcrs	Variante 1	76,30%	3,60%	20,10%
	Variante 2	60,90%	3,30%	35,80%
	Variante 3	67,60%	3,30%	29,10%
zcrps	Variante 1	75,40%	2,40%	22,20%
	Variante 2	64,50%	2,20%	33,30%
	Variante 3	67,70%	2,20%	30,10%

Tabelle 6.5: Test der Varianten zur Auswertung

Circle	c	zc	zcs	zcrs	zcps	Square	c	zc	zcs	zcrs	zcps
Circle	0,718	0,874	0,976	0,946	0,934	Circle	0,7	0,716	0,73	0,738	0,73
Heart	0,246	0,036	0	0	0	Heart	0,372	0,04	0	0	0
Square	0,656	0,988	0,988	0,97	0,97	Square	0,684	0,898	0,872	0,862	0,826
Z	0,104	0	0	0	0	Z	0,25	0	0	0	0
Heart	c	zc	zcs	zcrs	zcps	Z	c	zc	zcs	zcrs	zcps
Circle	0,314	0,022	0,008	0,006	0,004	Circle	0,45	0,018	0,008	0,01	0
Heart	0,71	0,918	0,87	0,882	0,892	Heart	0,46	0,004	0	0	0,024
Square	0,484	0,098	0,15	0,128	0,17	Square	0,692	0,434	0,528	0,46	0,266
Z	0,242	0	0	0	0	Z	0,564	0,95	0,886	0,952	0,96

Tabelle 6.6: Erkennungsverlauf der verschiedenen Gesten bei unterschiedlicher Vorverarbeitung; die dazugehörige Visualisierung ist: 6.6.

Verarbeitung	Accuracy (AVG±STD)	False unknown rate	False positiv rate
c	44%±4,4947%	10,50%	45,50%
zc	75%±5,3012%	3,80%	21,30%
zcs	78%±4,57602%	2,50%	19,60%
zcrs	76,30%±1,5976%	3,60%	20,10%
zcrps	75,40%±5,0705%	2,40%	22,20%

Tabelle 6.7: Positionfeatures, Variante 1

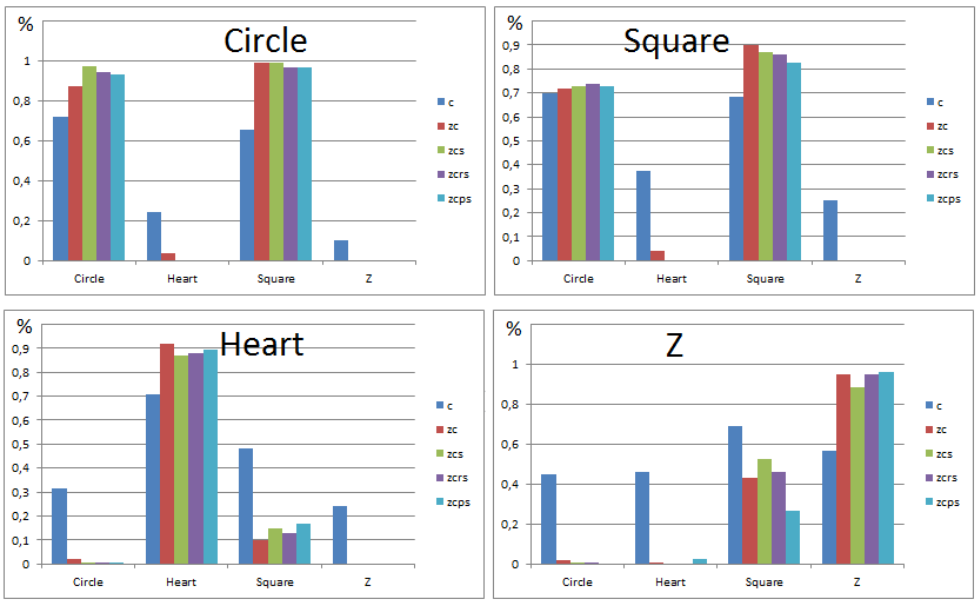

Abbildung 6.6: Visualisierung zur Tabelle 6.6, zeigt die Recognition Matrix Werte verschiedener Gesten, welche im Graph angezeigt sind, auf den unterschiedlichen HMMs mit verschiedenen Verarbeitungsschritten.

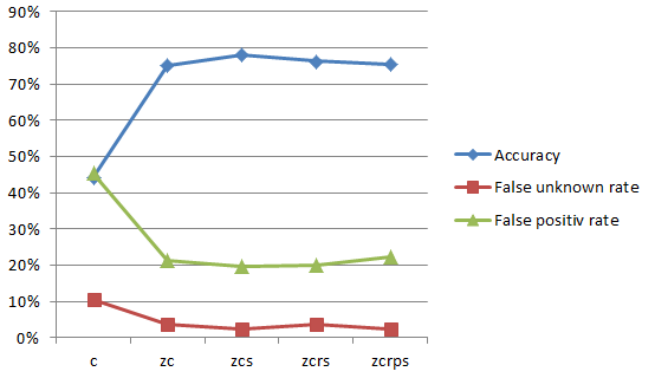

Abbildung 6.7: Darstellung der Erkennungsrate, False unknown rate und false positive rate des Modells für die unterschiedlichen Verarbeitungsschritte. (LR=2, k=8, n=12)

die Erkennungsgenauigkeit deutlich über 90% (siehe Kapitel 6.3.4). Des Weiteren wird deutlich, dass die Thresholds von Kreis und Quadrat doppelt bis drei mal so niedrig sind, wie die Thresholds von Herz und Z. Dies ist darauf zurückzuführen, dass die Gesten „im oberen Teil" begonnen wurden, aber durchaus viele verschiedene Startpositionen gewählt wurden (siehe Kapitel 4.3.4). Kreise, welche an ihrer höchsten Position begonnen wurden, werden auf der Quadrat-HMM besser erkannt, wenn diese mit Da-

ten, welche diesen Startpunkt gemeinsam haben, trainiert wurde. Die Quadrate der Trainingsgesten wurden häufiger in ihren Eckpunkten gestartet, als die Testgesten der Quadrate. Diese wurden unter anderem ebenfalls häufig in der Mitte der oberen Seite begonnen. Diese Testgesten werden somit mit den Trainingskreisen, welche häufiger in ihrer höchsten Position begonnen wurden, verwechselt.

6.3.4 Vergleichstests mit den Hall-Gestendaten

Um Vergleichswerte auf Basis anderer Gestendaten zu erhalten, werden die Gestendaten von Hall genutzt. Die Trainings- und Testdaten sind vor ihrer weiteren Verarbeitung bereits sehr ähnlich, weshalb die Genauigkeit der Erkennung für die verschiedenen Vorverarbeitungsschritte kaum steigt (bzw. steigen kann). Dieser Vergleichstest ist jedoch wichtig, um zu zeigen, dass die verwendeten Algorithmen die annähernd die gleiche Genauigkeit aufweisen, wie die von Hall vorgestellten Ergebnisse.

	Accuracy (AVG±STD)	FUR	FPR
nichts	94,60%±5%	4,20%	1,20%
c	96,80%±1,83303%	1,80%	1,40%
zc	98%±6%	2%	0%
zcs	100%±0%	0,00%	0,00%
zcrs	98,60%±2,44131%	1%	0,40%
zcrps	96%±5,21919%	4%	0%

Tabelle 6.8: Vergleichswerte für Accuracy, FUR und FPR berechnet auf den Daten von Hall

	circle	M	round	x	z
circle	0,9	0	0	0	0
M	0	0,99	0	0	0
round	0	0	1	0	0
x	0	0	0	0,91	0
z	0	0	0	0	1
unknown	0,1	0,01	0	0,09	0

Tabelle 6.9: Konfusionsmatrix für die Daten von Hall mit der Verarbeitung: *zcrps*

Auffällig ist, dass durch die Rotation und Projektion auf die Action Plane, die Genauigkeit um 1,4-4% abnimmt (siehe Tabelle 6.8 und Abbildung 6.8). Dies liegt vor allem daran, dass die Geste „X" fälschlicher Weise als „unknown" klassifiziert wird (siehe Tabelle 6.9). Wie jedoch an der Standardabweichung in Tabelle 6.8 erkennbar, liegt die Genauigkeit der Erkennung dennoch in einem Rahmen von 90% - 100%.

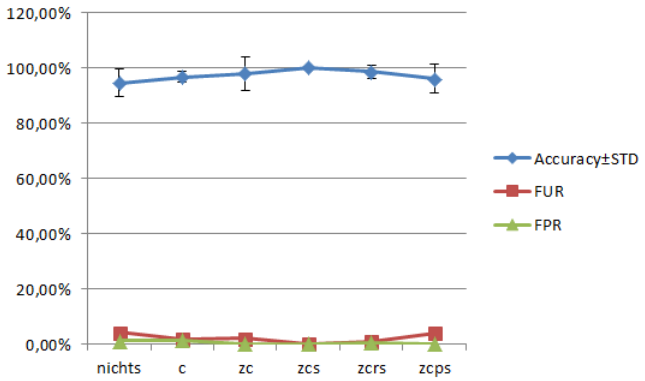

Abbildung 6.8: Darstellung der Erkennungsrate, False unknown rate und false positive rate der Tests mit den Hall-Gestendaten. (LR=2, k=8, n=12)

6.3.5 Personenbezogene Tests

Für die personenbezogenen Tests werden, wie in Kapitel 6.1 beschrieben, die Daten der Personen 0, 1, 10 und 18 verwendet. Für die Gesten der Nutzer soll bestimmt werden, wie wirksam der Einsatz von Positionsfeature-HMMs ist im Bezug auf die personenbezogene Wiedererkennung. Wie bei den personenübergreifenden Tests, wird der Einsatz der verschiedenen Vorverarbeitungsschritte auf eine Verbesserung der Accuracy, FUR und FPR hin analysiert. Bei allen Gesten wird deutlich, dass die Accuracy bei c besonders hoch ist und mit den verschiedenen aufeinander aufbauenden Vorverarbeitungsschritten sinkt. Die Erkennung bei c ist so hoch, da die Nutzer bei der Gestendatenaufzeichnung nie ihren Standpunkt, relativ zur Kinect, verändert haben. Dadurch liegen alle Gesten einer Person an der gleichen Position im Raum. Die Ortskoordinaten sind demnach fast identisch, was eine besonders gute Log-Likelihood auf Basis der Ähnlichkeit der Features mit sich bringt. Dieses Ergebnis ist jedoch aussagelos, da Personen anhand ihrer Gesten und nicht anhand ihrer Standorte vor der Kamera erkannt werden sollen. Die Erkennung sollte ebenfalls funktionieren, wenn sich die Nutzer im Raum bewegen oder zu einem späteren Zeitpunkt, in einem anderen Ort, die gleiche Geste noch einmal ausführen. Daraus folgt, dass bei der personenbezogenen Erkennung nur Daten mit den minimalen Vorverarbeitungsschritten z und c in Betracht gezogen werden dürfen. Wie in den Tabellen 6.10, 6.11, 6.12 und 6.13 zu erkennen ist, ist die Erkennungsrate, bzw. Accuracy, bei der Anwendung von Vorverarbeitungsschritten sehr gering. Die FUR und FPR steigt hingegen an.

	Accuracy (AVG±STD)	False unknown rate	False positiv rate
c	84,50%±8,124%	2,30%	13,20%
zc	56,80%±7,433%	3,20%	40%
zcs	51,80%±8,3066%	4,80%	43,30%
zcrs	50,80%±9,069%	7,50%	41,70%
zcrps	51,30%±5,099%	3%	45,70%

Tabelle 6.10: Personenbezogene Auswertung zu den Kreis-Gesten, Positionsfeatures, Variante 1.

	Accuracy(AVG±STD)	False unknown rate	False positiv rate
c	81,70%±9%	4,80%	13,50%
zc	60,80%±8,5%	0,80%	38,30%
zcs	42,50%±8,888%	0,20%	57,30%
zcrs	42,50%±6,63325%	0,20%	57,30%
zcrps	46,80%±9%	0,20%	53%

Tabelle 6.11: Personenbezogene Auswertung zu den Herz-Gesten, Positionsfeatures, Variante 1.

	Accuracy(AVG±STD)	False unknown rate	False positiv rate
c	70,50%±11,6619%	0,50%	29%
zc	47,70%±13,6381%	0%	52,30%
zcs	41,70%±9,0138%	0,70%	57,70%
zcrs	39,80%±6,7268%	0,40%	59,80%
zcrps	44,70%±8,5%	0%	55,30%

Tabelle 6.12: Personenbezogene Auswertung zu den Quadrat-Gesten, Positionsfeatures, Variante 1.

	Accuracy(AVG±STD)	False unknown rate	False positiv rate
c	78,80%±11,8848%	0,70%	20,50%
zc	49,20%±13,6839%	0%	50,80%
zcs	40,50%±11,8427%	0%	59,50%
zcrs	36,30%±8%	0,20%	63,50%
zcrps	35%±12,0519%	0,30%	64,70%

Tabelle 6.13: Personenbezogene Auswertung zu den Z-Gesten, Positionsfeatures, Variante 1.

6.4 Geschwindigkeitsfeature Tests

Wie bei den Positionsdaten auch, können bei den Tests mit den Geschwindigkeitsdaten einige Parameter variiert werden. Der LR-Parameter wird mit $LR = 2$ festgelegt. Aus den Erfahrungen der vorherigen Tests wurde abgeleitet, dass ein Testdurchlauf mit

8 Observable Symbols und 12 Hidden States für die verschiedenen Vorverarbeitungsschritte Ergebnisse erzeugt wird, die darauf hinweisen, welche Vorverarbeitung die beste Genauigkeit liefert. Des Weiteren wird angenommen, dass auf der Basis der so gefundenen „besten Vorverarbeitungen" weiter nach den „besten" Anzahlen für Observable Symbols und Hidden States analysiert werden kann. Um einen anschaulichen Überblick über den Inhalt der Geschwindigkeitsdaten zu erlangen, wurden die Geschwindigkeiten v_x, v_y und v_z im dreidimensionalen Raum visualisiert (siehe 6.9).

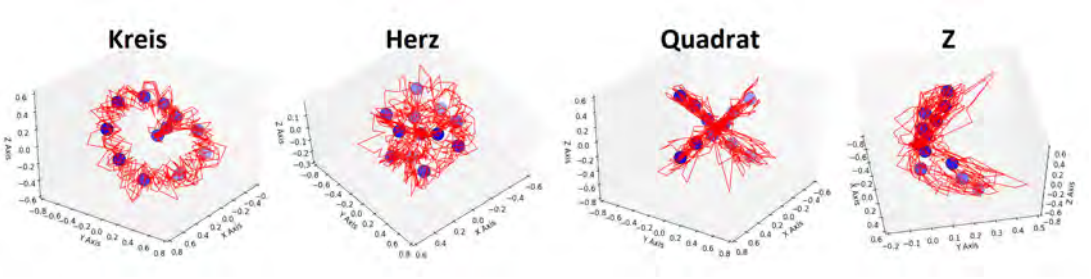

Abbildung 6.9: Darstellung der Geschwindigkeitfeatures im dreidimensionalen Raum. X-Achse: v_x, Y-Achse: v_y, Z-Achse: v_z. Vorverarbeitung: zcrs; $k = 12$, $n = 18$.

	Accuracy (AVG±STD)	False unknown rate	False positiv rate
c	57,10%±8,7694%	6,80%	36,10%
zc	74,00%±7,0505%	8,10%	17,90%
zcs	74,00%±6,7683%	8,10%	17,90%
zcrs	71,20%±9,794%	7,20%	21,60%
zcrps	75,50%±7,46257%	7,70%	16,80%

Tabelle 6.14: Geschwindigkeitsfeatures, Variante 1, k=8, n=12, alle Verarbeitungsschritte

Bei den Ergebnissen in Tabelle 6.14 ist auffällig, dass zwischen zc und zcs kein wesentlicher Unterschied besteht. Dies ist darauf zurückzuführen, dass die Geschwindigkeiten anhand der Distanz zwischen den Koordinaten und der Anzahl an Nanosekunden zwischen den Frames berechnet werden, welche durch ein Scaling ebenfalls nur skaliert werden. Die Unterschiede in den Geschwindigkeiten bleiben proportional zueinander erhalten, egal wie groß die Gesten sind. Die Zeit wird nicht skaliert, oder in die Berechnung der HMMs miteinbezogen, diese wird nur zur Berechnung der Geschwindigkeiten verwendet. Da festgestellt wurde, dass bei den Geschwindigkeitsfeatures die Vorverarbeitungen zcs und zcrps die besseren Raten für Genauigkeit, false unknwon und false positive lieferten (siehe Tabelle 6.14), wurde für den besten Fall (zcrps) eine Analyse der Variationen der Anzahl von Observable Symbols und Hidden States vorgenom-

men. Das Ergebnis in Tabelle 6.15 zeigt, dass im Gegensatz zu den Positionsfeature-HMMs, eine Anzahl von $k = 12$ und $n = 18$ bessere Ergebnisse liefert als die bei den Positionsfeature-HMM beste Anzahl $k = 8$ und $n = 12$. Dies wurde für alle Vorverarbeitungsschritte genauer analysiert und mit den alternativen Ergebnissen mit $k = 8$ und $n = 12$ verglichen, was in Tabelle 6.16 dargestellt wird. Dadurch wird Hypothese 5.3 belegt.

	n/k	Accuracy(AVG±STD)	FUR	FPR
zcrps	4 6	52,10%±6,85638%	9,40%	38,60%
	6 9	70,30%±9,5268%	6,70%	23%
	8 12	75,50%±7,46257%	7,70%	16,80%
	12 18	79,40%±3,64828%	9%	11,70%
	16 24	77,80%±2,26715%	12,40%	9,90%
	24 36	72,00%±5,08945%	18,60%	9,40%

Tabelle 6.15: Verschiedene n/k für die Geschwindigkeitsfeatures

	n/k	Accuracy(AVG±STD)	FUR	FPR
c	8 12	57,10%±8,7694%	6,80%	36,10%
	12 18	63%±7,685%	6,60%	30,40%
zc	8 12	74,00%±7,0505%	8,10%	17,90%
	12 18	81,20%±2,25222%	9,70%	9,10%
zcs	8 12	74,00%±6,7683%	8,10%	17,90%
	12 18	85,90%±2,1189%	8,20%	5,90%
zcrs	8 12	71,20%±9,794%	7,20%	21,60%
	12 18	84,20%±1,8901%	7,50%	8,50%
zcrps	8 12	75,50%±7,46257%	7,70%	16,80%
	12 18	79,40%±3,64828%	9%	11,70%

Tabelle 6.16: Accuracy, FUR, FPR und AVG±STD für $k = 8$; $n = 12$ und $k = 12$; $n = 18$

6.4.1 Personenbezogene Tests

Da es bei der personenbezogenen Erkennung im Kapitel 6.3.5 grundsätzlich daran scheiterte, dass die Gesten ohne Vorverarbeitung nicht vergleichbar genug sind, damit die Ergebnisse gültig wären und ab der Vorverarbeitung durch Zentrierung die Gesten zu vergleichbar wurden, wird nach Features gesucht, die unabhängig von den Ortskoordinaten sind und genug Unterschiede bezüglich der verschiedenen Ausführungen der Personen beinhalten. Ob eine personenbezogene Erkennung mittels Geschwindigkeitsfeatures möglich ist, soll in diesem Kapitel getestet werden.

	Accuracy(AVG±STD)	False unknown rate	False positiv rate
c	**76,80%±10,5%**	**2,50%**	**20,70%**
zc	75,80%±11,19%	3,30%	20,80%
zcs	73,30%±10,54%	1%	25%
zcrs	68%±9,43%	0,80%	31,20%
zcrps	71,70%±11,84%	0,80%	27,50%

Tabelle 6.17: Personenbezogener Test für Geschwindigkeitsfeatures (Kreis)

Wie in Tabelle 6.17 dargestellt, werden zuerst mit den Parametern, welche im personenübergreifenden Test die besten Werte für Genauigkeit, FUR und FPR lieferten, die gesamten Vorverarbeitungsschritte getestet. Von diesen Parametern wird erwartet, dass sie den optimalen Parametern des personenbezogenen Tests ähnlich sind, weshalb anhand der Ergebnisse, dargestellt in 6.17, die Parameter angepasst werden können. Dadurch, dass die Vorverarbeitung c die besten Ergebnisse liefert, wird anhand dieser Vorverarbeitung in Tabelle 6.18 analysiert, welche Anzahl Hidden States und Observable Symbols die höchste Genauigkeit und geringsten FUR und FPR-Werte liefert.

n/k	Accuracy(AVG±STD)	False unknown rate	False positiv rate
8/12	65%±12,69%	2,70%	32,30%
12/18	76,80%±10,5%	2,50%	20,70%
16/24	**79%±13,64%**	**7,30%**	**13,70%**
20/30	78,80%±16%	8,30%	10,80%
24/36	63,30%±14,62%	17,20%	19,50%

Tabelle 6.18: Personenbezogener Test für Geschwindigkeitsfeatures mit variablen Hidden States und Observable Symbols für die Kreis-Geste (c)

Es wurde festgestellt, dass zwischen den Ergebnissen $k = 12/n = 18$ und $k = 20/n = 30$ kein großer Unterschied besteht, $k = 16/n = 24$ jedoch sehr knapp das höchste Ergebnis für AVG+STD erreicht. Diese Kombination an Hidden States und Observable Symbols wird für den folgenden Threshold-Test (Tabelle 6.19) verwendet. Es wird jedoch angenommen, dass die beiden anderen Kombinationen ($k = 12/n = 18$ und $k = 20/n = 30$) ähnliche Ergebnisse liefern.

t	Accuracy	FUR	FPR
2	83%	9%	8%
4	84%	7%	9%
4,7	85%	6%	9%
9	85%	6%	9%

Tabelle 6.19: Thresholdvariation für den personenbezogenen Test mit Kreis-Gesten für Geschwindigkeitsfeatures (c; $k = 16$ und $n = 24$)

Durch die Variation des Thresholds, dargestellt in Tabelle 6.19, wird festgestellt, dass die höchste Accuracy mit einem Threshold-Parameter $t = 4,7$ erreicht wird.

6.5 Kombinierter Test aus Positionsfeature-HMM und Geschwindigkeitsfeature-HMM

In diesem Test werden die aus Kapitel 5.8 beschriebene Kombination der HMM aus Positionsfeatures und HMM der Absolutgeschwindigkeit (aufgetragen auf eine Zeitachse) genutzt. Wie in Hypothese 5.8 beschrieben, ist die Annahme, dass formähnliche Gesten anhand ihrer Geschwindigkeiten unterscheidbar sind. In Abbildung 6.10 sind die Absolutgeschwindigkeiten der einzelnen Gesten (einer Person: P0) beispielhaft visualisiert. In dieser Abbildung ist erkennbar, dass die Geschwindigkeiten deutliche Unterschiede aufweisen, weshalb die in Kapitel 5.8 beschriebene Vergleichsmethode entwickelt wurde.

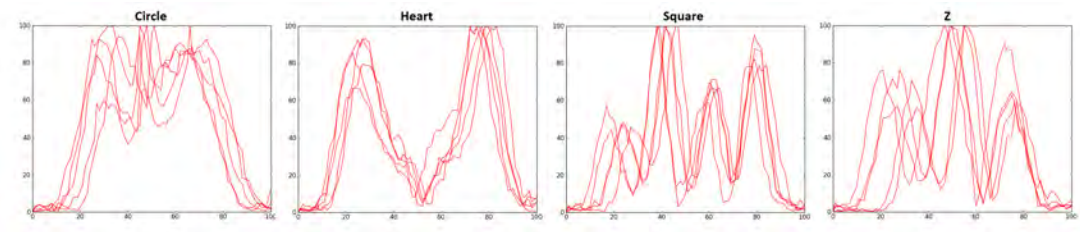

Abbildung 6.10: Darstellung der eindimensionalen Geschwindigkeitsfeatures. X-Achse: t normiert auf 100 Zeiteinheiten, Y-Achse: $v_{absolut}$

Für die Tests wurden die Parameter $LR = 2$, $k = 8$ und $n = 12$ gewählt. Bereits nach einigen Testdurchläufen stellte sich heraus, dass die Kombination der HMMs nur sehr niedrige Genauigkeiten lieferte. Die Genauigkeit liegt sogar ein wenig unter der Genauigkeit, die eine zufällige Verteilung liefern würde (Zufällig müsste die Genauigkeit bei 25% liegen.). Durch die Anwendung der Verarbeitungsschritte verbesserte sich die Genauigkeit nicht. Werden die Ergebnisse der Tabellen im Anhang unter 8.7 und 8.8 betrachtet, wird deutlich, dass die Kombination des Thresholds θ_t und der LLHs der beiden HMMs zu erheblichen Verwechslungen führt. Die Thresholds verschiedener HMMs sind nur schwer bis gar nicht vergleichbar, weshalb die simple Addition der beiden Ergebnisse zu schlechten Genauigkeiten führt. Durch die Vorverarbeitungsschritte werden die Gesten anhand der Positions-HMM genauer erkannt, was bedeutet, dass diese eine größere LLH auf der zugehörigen Threshold aufweisen. Wird dies mit einer knapp unterschwelligen LLH der Geschwindigkeits-HMM kombiniert, gilt die Geste insgesamt als erkannt. Die Verwechselungen zwischen Kreis und Quadrat nehmen ab, wohingegen Verwechslungen zwischen Quadrat und Herz, sowie Z und Square zunehmen.

Dies bedeutet, dass der Gedanke die formähnlichen Gesten anhand ihrer Geschwindigkeiten zu unterscheiden, ein richtiger Gedanke ist, die hier beschriebene Umsetzung in der Gesamterkennung nicht zielführend ist.

	Accuracy (AVG±STD)	False unknown rate	False positiv rate
c	26,35%±4,945%	0,4%	73,25%
zc	24,6%±2,385%	2,4%	73%
zcs	24,35%±3,248%	6,65%	69%
zcrs	23,95%±2,285%	2,4%	73,65%
zcrps	22,9%±2,508%	3,35%	73,75%

Tabelle 6.20: Ergebnisse für Accuracy, FUR und FPR für die Kombination der Positions-HMM mit der Absolutgeschwindigkeits-HMM; Variante 1

6.6 Tests mit vierdimensionalem Featurevektor aus Position und absoluter Geschwindigkeit

Im vierdimensionalen Featurevektor befinden sich die Informationen über die Ortskoordinaten sowie deren Absolutgeschwindigkeit. Diese Absolutgeschwindigkeit wird auf die Größe der längsten anderen Achse skaliert, damit die Geschwindigkeiten im Vergleich zu den anderen Achsen die gleiche Aussagekraft besitzen. In Abbildung 6.11 ist dies in einer dimensionsreduzierten Variante dargestellt, indem die Ortskoordinaten auf ihre Action Plane rotiert und projiziert wurden. So ist auf der Z-Achse die Geschwindigkeit $v_{absolut}$ aufgetragen.
In Tabelle 6.23 wird beschrieben, welche Ergebnisse bei verschiedenen Anzahlen an Hidden States und Output Symbolen bei diesem Test resultierten. Im Gegensatz zu anderen Tests bleiben die Ergebnisse für $k = 6$, $n = 9$ bis $k = 16$, $n = 24$ fast gleich. Die „Problemgesten" Kreis und Quadrat können jedoch, wie in Tabelle 6.21 und 6.22 dargestellt, nicht besser unterschieden werden, als bei den Positionsfeatures.

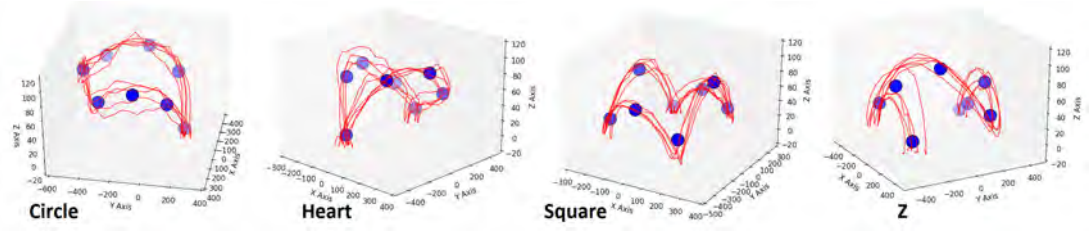

Abbildung 6.11: Darstellung der Geschwindigkeit aufgetragen auf der Actionplane der Gesten im dreidimensionalen Raum. X-Achse: x, Y-Achse: y, Z-Achse: $v_{absolut}$. Vorverarbeitung: nichts.

	c	zc	zcs	zcrs	zcrps
Circle	0,828	0,944	0,946	0,912	0,91
Heart	0,474	0,134	0	0	0
Square	0,972	0,914	0,942	0,958	0,956
Z	0,484	0	0	0	0

Tabelle 6.21: Beispielhafte Veränderung der Recognition für die verschiedenen Verarbeitungsschritte der Kreis-Geste. Die Recognition Matrizen der Positionsfeatures (zum Vergleich) sind in Tabelle 6.4 zu finden. ($k = 8$, $n = 12$)

	c	zc	zcs	zcrs	zcrps
Circle	0,628	0,586	0,806	0,768	0,53
Heart	0,016	0,008	0	0	0
Square	0,23	0,402	0,188	0,184	0,45
Z	0,1	0	0	0	0
unknown	0,026	0,004	0,006	0,048	0,02

Tabelle 6.22: Konfusionsmatrizen-Veränderung für die verschiedenen Verarbeitungsschritte der Kreis-Geste. Variante 1.

Werden die Recognition-Ergebnisse und die Konfusions-Matrizen-Ergebnisse aus den Positions-HMMs betrachtet (Tabelle 6.4), wird deutlich, dass der Kreis dort nur auf sich oder auf dem Quadrat erkannt wird. Bei der Recognition Matrix 6.21 und den Konfusions-Matrizen-Ergebnissen 6.22 des vierdimensionalen Vektors wird das gleiche festgestellt. Durch die Hinzunahme der Absolutgeschwindigkeit ändert sich an diesem Phänomen demnach nichts.

	n/k	Accuracy(AVG±STD)	False unknown rate	False positive rate
zcs	4 6	67,15%±3,65%	6,9%	25,95%
	6 9	74,4%±10,68%	7,15%	18,45%
	8 12	73,25%±5,21%	9,95%	16,8%
	12 18	73,55%±3,02%	12%	14,45%
	16 24	73,6%±6,47%	14,55%	11,85%

Tabelle 6.23: Tests mit verschiedenen Anzahlen an Hidden States und Observable Symbols für den vierdimensionalen Featurevektor $\vec{F} = (x, y, z, v_{absolut})$.

Der vierdimensionale Featurevektor eignet sich demnach nicht besser für die hier verwendeten Gesten und Verarbeitungen als die Positionsfeatures. Wie bereits im vorangegangenen Test der kombinierten HMMs festgestellt, eignet sich die Absolutgeschwindigkeit nicht, um die Genauigkeit der Gestenerkennung zu verbessern; es verschlechtert die Ergebnisse aber nicht deutlich. Im Vergleich zu den Positionsfeatures schwanken diese mit einem kleinen Vorsprung um etwa die gleiche Accuracy wie die vierdimensio-

	Accuracy (AVG±STD)	False unknown rate	False positive rate
c	56,1%±5,16%	10,25%	33,65%
zc	68,35%±7,94%	11,9%	19,75%
zcs	73,25%±5,21%	9,95%	16,8%
zcrs	73,6%±5,95%	10%	16,4%
zcrps	71,2%±5,21%	12,05%	16,75%

Tabelle 6.24: Ergebnisse der Konfusionsmatrizen der Variante 1 für $k = 8$ und $n = 12$ des vierdimensionalen Featurevektors über alle Vorverarbeitungsschritte.

nalen Features. Werden die Absolutgeschwindigkeiten mehrerer Personen für die gleiche Geste betrachtet, wie in der Abbildung 6.12 dargestellt, ist erkennbar, weshalb diese keinen besonderen Mehrwert für de Erkennung liefern. Die Absolutgeschwindigkeit der gleichen Geste ist sehr unterschiedlich.

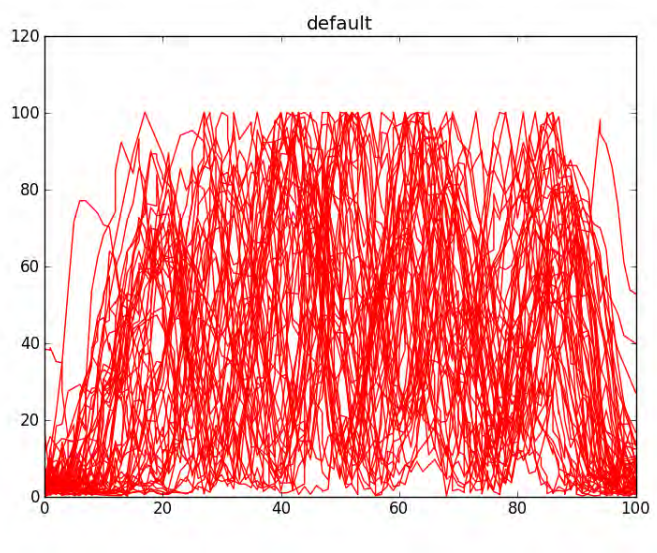

Abbildung 6.12: Darstellung der Absolutgeschwindigkeiten aller Trainings-Squares. X-Achse: t, Y-Achse: $v_{absolut}$. Beide Achsen sind für jede Geste zur besseren Visualisierung auf 100 Einheiten normiert.

6.7 Phasenraumfeature Tests

Der Phasenraum wird, wie er in Kapitel 5.10 beschrieben wird mit dem Featurevektor $\vec{F} = (x, y, z, v_x, v_y, v_z)^T$ aus Gleichung 5.16 verwendet. Zuerst muss die Anzahl der optimalen Hidden States und Observable Symbols festgestellt werden. Dazu wurde ein Test, dessen Ergebnisse in Tabelle 6.25 dargestellt sind, durchgeführt. Dafür wurde

aus Basis der guten Ergebnisse aus den vorherigen Tests, eine Vorverarbeitung von *zcs* gewählt. Durch diesen initialisierenden Test wurde das beste Ergebnis für $k = 12$ und $n = 18$ festgestellt. Dementsprechend werden in den folgenden Tests diese Anzahlen an Hidden States und Symbols genutzt.

	n/k	Accuracy(AVG±STD)	False unknown rate	False positiv rate
zcs	6/9	73,30%±6,1703%	4,70%	22%
	8/12	80,50%±3,9752%	5,00%	14,50%
	12/18	**85,50%±1,9678%**	**5,60%**	**8,90%**
	16/24	78,20%±10,6817%	14%	7,90%

Tabelle 6.25: Phasenraum; Initialtest zu der Anzahl an Hidden States und Observable Symbols; mit der Vorverarbeitung: zcs

In Tabelle 6.26 sind für $k = 12$ und $n = 18$ alle Varianten der Vorverarbeitungen aufgezeigt. Es ist auffällig, dass zwischen allen Ergebnissen, welche Zentrierung beinhalten, kaum Unterschiede in der Genauigkeit der Erkennung auftreten. Zwischen den Verarbeitungen mit *zc* und *zcrps* schwanken die Genauigkeiten maximal um ca. 4%. Das beste Ergebnis liefert, vergleichbar mit den Ergebnissen der Positionsdaten, die Verarbeitung mit *zcs*. Die Ergebnisse, wie in Tabelle 8.20 dargelegt, werden maßgeblich von der Geste „Square" beeinflusst. Die Ergebnisse der Genauigkeit dieser Geste verschlechtern sich mit den Verarbeitungen von *zc* bis *zcrps* und verschlechtern somit das Gesamtergebnis der Erkennung. Square wird deutlich häufiger mit einer anderen Geste, in diesem Fall Circle, verwechselt, als alle anderen Gesten. Dazu ist das häufigste Auftreten von als „unknown" klassifizierten Gesten, das der Quadrat-Geste. Wie auch im Positionsfeature-Test, zeigen sich die Gesten Kreis und Quadrat als schwierig auseinanderzuhalten. Im Gegensatz zu den Positionsfeatures ist die Verwechslung jedoch einseitig; Quadrate werden als Kreise erkannt, Kreise jedoch nicht so häufig als Quadrate.

	Accuracy(AVG±STD)	False unknown rate	False positiv rate
c	45,40%±4,5945%	13,70%	41%
zc	81,50%±8,5599%	7,10%	11,40%
zcs	**85,50%±1,9678%**	**5,60%**	**8,90%**
zcrs	83,70%±4,382%	8%	8,30%
zcrps	84,20%±3,908%	7,10%	8,80%

Tabelle 6.26: Ergebnisse des Phasenraums für alle Vorverarbeitungsschritte für $k = 12$ und $n = 18$

6.7.1 Personenbezogene Tests

Die in Hypothese 5.9 angemerkte Erkennung von Personen anhand ihrer Gesten soll in diesem Abschnitt getestet werden. Dafür werden, wie in Kapitel „Testaufbau" 6.1 beschrieben, Daten der Personen 0, 1, 10 und 18 verwendet. Von jeder Person 15 Datensätze pro Geste was mit einer dreifachen Kreuzvalidierung zu 15 Result-Files des Testdurchlaufs führt.

Für den Phasenraum wurde die Hypothese 5.9 aufgestellt, dass anhand dieses sechsdimensionalen Featurevektors besonders gut personenbezogene Gesten erkannt werden können, da die Kombination aus Positionsdaten und Geschwindigkeiten in den einzelnen Positionskomponenten pro Person besonders unterschiedlich ausfallen kann. Das soll anhand dieses Tests geprüft werden. Werden die Ortskoordinaten unverändert betrachtet, stimmt diese Hypothese mit einer Genauigkeit von ca. 70%, wie in der Tabelle 6.27 in der ersten Zeile dargestellt. Personen bewegen sich jedoch vor der Kamera, weshalb die Daten in dieser Form nicht betrachtet werden können. Die Gesten müssen mindestens in ihrem Ort normiert sein, um auch Gesten verschiedener Zeitpunkte und verschiedener Ausführungsorte derselben Person erkennen zu können. Wie in Tabelle 6.27 aufgezeigt, verschlechtert sich die Genauigkeit der Erkennung aber zunehmend mit der Normierung der Ortskoordinaten. Um die Ortskoordinaten sinnvoll in einer Erkennung miteinbeziehen zu können, müssen diese einer gewissen Normierung unterliegen, die Geschwindigkeitsdaten jedoch nicht. Diese verlieren bei einer zu starken Vereinheitlichung zusätzlich ihre personenbezogenen Charakteristika. Zusammen ergeben diese Features, mit den hier verwendeten Verarbeitungsmethoden, eine eher geringe Genauigkeit in der Erkennung. In Abbildung 6.13 ist ebenfalls dargestellt, dass die Phasenraumkomponenten zwar kleine Unterschiede bei verschiedenen Personen aufweisen, jedoch mehr Ähnlichkeiten besitzen.

	Accuracy(AVG±STD)	False unknown rate	False positiv rate
c	68,70%±9,7467%	0,80%	30,50%
zc	57,20%±9,81%	0,80%	42%
zcs	52,20%±11,576%	4,70%	43,10%
zcrs	50%±7,81%	3,50%	46,50%
zcrps	50,80%±12,884%	1%	48,10%

Tabelle 6.27: Personenbezogene Phasenraumergebnisse aller Vorverarbeitungsschritte für die Kreis-Geste

Durchgehend werden die Gesten mit einer Accuracy von 50% erkannt, war zwar immerhin doppelt so hoch ist, wie eine zufällige Verteilung (diese müsste bei vier Testgesten bei 25% liegen), aber deutlich schlechter ist, als die Ergebnisse auf den Geschwindigkeitsfeatures, welche eine Accuracy von bis zu 80% erreichten (siehe Tabelle

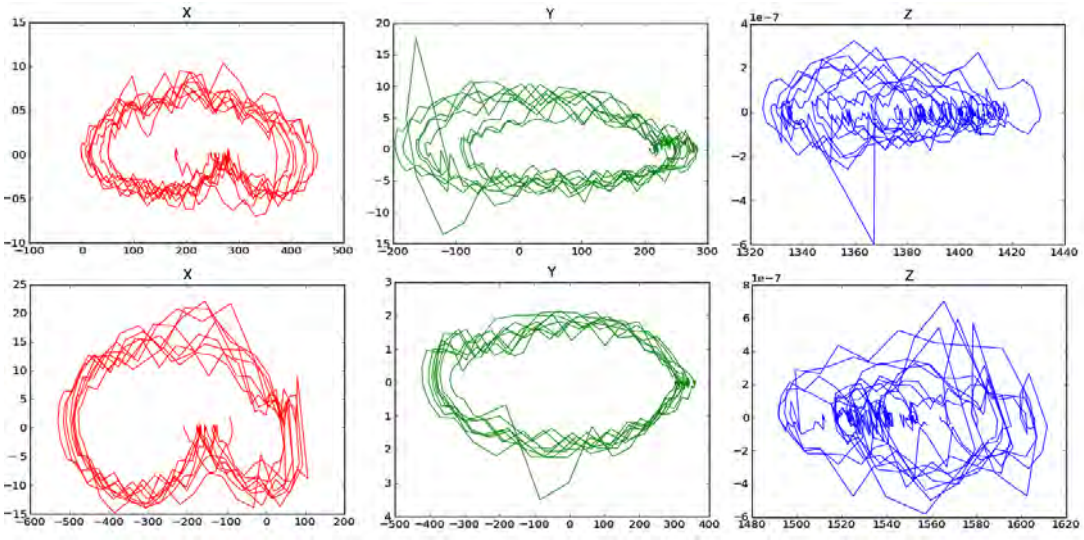

Abbildung 6.13: Darstellung des Phasenraums in drei Graphen für zwei verschiedene Personen. Rot: X-Achse: x, Y-Achse: v_x; Grün: X-Achse: y, Y-Achse: v_y; Blau: X-Achse: z, Y-Achse: v_z. Ohne Vorverarbeitung.

6.18 und 6.17). Auch eine Variation des Thresholds (testweise bis zu einem $t = 30$) konnte keine Verbesserung der Erkennungsrate erzielen.

6.8 Variation des Thresholds

In den Tests wurden viele Variablen ausgetauscht, um eine bessere Erkennungsrate zu erhalten. Unter anderem wurde der Left-to-Right Faktor (LR) von einem Minumum von zwei bis zur maximalen Anzahl (Anzahl der Hidden States) variiert. Die Anzahl der Observable Symbols, sowie der Hidden States, wird bei den Tests ebenfalls variiert. Eine Variable, die bisher nicht verändert wurde, ist der Parameter t zur Berechnung der Thresholds. Der Threshold θ_t setzt sich, wie in 3.2 beschrieben, aus dem errechneten Log-Likelihood der trainierten HMM, sowie einem Faktor nach [11] (der initial $t = 2$ gewählt wird) zusammen.

Um die Hypothese 5.4 zu testen, werden aus den vorangegangenen Tests die mit den höchsten Erkennungsraten herausgesucht, um bei diesen den Threshold-Faktor zu variieren. Der Faktor bestimmt die Verschiebung der Erkennungsschwelle, welche Log-Likelihoods von Gesten auf der trainierten HMM angenommen oder abgewiesen werden. Liegt das Log-Likelihood der zu erkennenden Geste überschwellig zum Threshold ($LL > Threshold$), wird die Geste als erkannt ausgegeben. Dieser Parameter kann $t = 1$ gewählt werden, da er minimal dem errechneten Log-Likelihood der Trainigsdaten auf der HMM selbst entsprechen muss. Wird der Wert erhöht, erhöht sich ebenfalls

der Spielraum für Gesten, überschwellig zu sein und als „erkannt" bewertet zu werden. In dieser Arbeit werden drei verschiedene Ansätze zur Gestenerkennung verwendet, die jeweils mit anders aufbereiteten Daten arbeiten. Die Annahme ist, dass der Threshold bei diesen unterschiedlichen Ansätzen andere Auswirkungen hat und eventuell mit dem bisher verwendeten Faktor nicht optimal ist.
Im Folgenden wird dargelegt, was sich mit der Veränderung des Faktors ändert und welche Schlüsse daraus gezogen werden.

6.8.1 Thresholdvariation bei der Erkennung auf Basis der Positionsdaten von Gesten

Für die Erkennung anhand von Positionsdaten (Bezug zum Kapitel 6.3, Featurevektor: $\vec{F} = (x, y, z)^T$) wird das Doppelte der Log-Likelihood des trainierten HMMs empfohlen [11]. Um diese Empfehlung zu prüfen, werden bei den vorangegangenen Tests, welche die beste Erkennungsrate lieferten, verschiedene Tests mit variablem Threshold durchgeführt.

	Accuracy	FUR	FPR
1	52,60%	34,10%	13,30%
1,1	60%	23,60%	16,40%
1,2	64,50%	18,40%	17,10%
1,3	67%	14,90%	18,10%
1,4	70,40%	10,60%	19%
1,5	71,80%	9,10%	19,10%
1,6	73,10%	7,80%	19,10%
1,7	74,50%	6,20%	19,30%
1,8	76,30%	4,40%	19,30%
1,9	76,70%	4%	19,30%
2	76,90%	3,80%	19,30%
2,2	77,70%	3%	19,30%
2,4	78,40%	2,30%	19,30%
2,6	78,80%	1,90%	19,30%
2,8	79,40%	1,30%	19,30%
4	80%	0,30%	19,70%
4,3	80%	0%	19,70%

Tabelle 6.28: Threshold Test für zcs $n = 8$ und $k = 12$

Eine Thresholdvariation wird auf den Daten des Tests mit der Vorverarbeitung *zcs*, 8 Output Symbolen und 12 Hidden States ausgeführt, da dieser, wie in Tabelle 6.3 beschrieben, sich unter den drei besten Ergebnissen der Erkennung auf den Positionsdaten befindet.

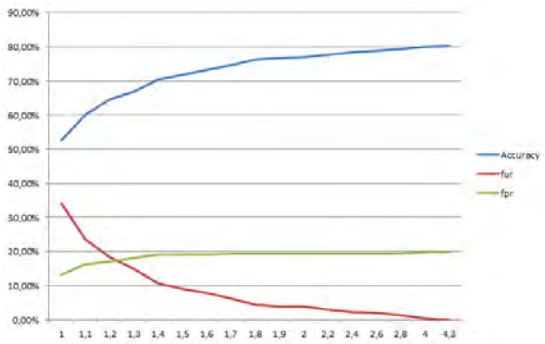

Abbildung 6.14: Darstellung der Auswirkung des Thresholds auf die Accuracy, FUR und FPR (zcs; $LR = 2$; $k = 8/n = 12$).

	Accuracy	fur	fpr
1	52,90%	39,20%	8,00%
1,1	59%	31,20%	9,60%
1,2	63,20%	24,60%	12,20%
1,3	67%	18,80%	14,30%
1,4	68,80%	16,70%	14,50%
1,5	70,60%	14,80%	14,60%
1,6	71,70%	13,50%	14,80%
1,7	72,70%	12,10%	15,20%
1,8	74,30%	10,50%	15,20%
1,9	76,60%	8%	15,40%
2	77,90%	6,70%	15,40%
2,2	78,20%	6%	15,40%
2,4	79,20%	5,40%	15,40%
2,6	79,60%	5,00%	15,40%
2,8	80,40%	4,10%	15,50%
4	**82%**	**1,50%**	**16,10%**
5	82%	1,50%	16,10%

Tabelle 6.29: Threshold Test für zcrs $n = 12$ und $k = 18$

6.8.2 Thresholdvariation bei der Erkennung auf Basis der Geschwindigkeitsfeatures von Gesten

Bei der Erkennung mittels Positionsdaten (vorheriges Kapitel) wird ein Parameter von $t = 2$ empfohlen, um den Threshold zu wählen. In diesem Abschnitt wird dieser Faktor für die Erkennung mittels Geschwindigkeitsfeatures (Bezug zum Kapitel 6.4, Featurevektor: $\vec{F} = (v_x, v_y, v_z)^T$) überprüft und analysiert welches t für Geschwindigkeitsdaten optimal ist.

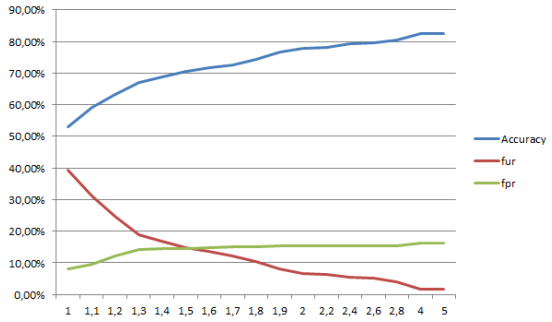

Abbildung 6.15: Darstellung der Auswirkung des Thresholds auf die Accuracy, FUR und FPR (zcrs; $LR = 2$; $k = 12/n = 18$).

t	Accuracy	FUR	FPR
1	52,90%	46,60%	0,50%
1,1	60%	38,70%	1,10%
1,2	66,70%	31,30%	2,00%
1,3	71%	26,30%	2,50%
1,4	75,80%	21,00%	3%
1,5	78,90%	17,40%	3,70%
1,6	80,40%	15,50%	4,10%
1,7	81,30%	14,50%	4,20%
1,8	83,10%	12,40%	4,50%
1,9	85,00%	10%	4,80%
2	85,60%	9,30%	5,10%
2,2	86,40%	8%	5,50%
2,4	86,80%	7,10%	6,10%
2,6	86,90%	6,70%	6,40%
2,8	87,50%	6,00%	6,50%
4	88,90%	4,10%	7,00%
5	89,70%	3,20%	7,10%
8	90,60%	2%	7,40%
12,1	**91,20%**	**1,40%**	**7,40%**
13	91,20%	1,40%	7,40%
31	91,20%	1,00%	7,80%

Tabelle 6.30: Darstellung der Auswirkung des Thresholds auf die Accuracy, FUR und FPR (zcs; $k = 12$ und $n = 18$) für die Geschwindigkeitsfeatures.

Es wird festgestellt, dass Accuracy, FUR und FPR bei $t = 12,1$ stagnieren. Die Genauigkeit bei $t = 12,1$ nimmt im Gegensatz zur Genauigkeit bei $t = 2$ um 5,6% zu, allerdings steigt ebenfalls die FPR um 2,3% an. Die FUR sinkt um 7,9%. Durch die Verschiebung des Thresholds werden mehr Gesten als „erkannt" angenommen. Diese Vergrößerung des Bereichs, in dem Gesten als angenommen gelten, begünstigt vor allem

das falsche erkennen ähnlicher Gesten. Dies kommt dadurch zustande, dass die LLH von ähnlichen Gesten, auch wenn diese sehr klein ist, oft nicht bei $-\infty$ liegt, wie es bei deutlich verschiedenen Gesten vorkommt.

6.8.3 Tresholdvariation bei der Erkennung auf Basis des Phasenraums von Gesten

Genauso wie bei den Positions- Geschwindigkeitsfeatures zuvor, soll der von [11] empfohlene Faktor $t = 2$ zur Bestimmung des Thresholds ebenfalls für den Phasenraum geprüft werden.

t	Accuracy	FUR	FPR
1	56,10%	40,40%	3,50%
1,1	63%	32,10%	4,60%
1,2	69,10%	25,00%	5,90%
1,3	74%	19,30%	6,60%
1,4	77,30%	15,50%	7%
1,5	78,60%	13,10%	8,30%
1,6	79,10%	12,10%	8,80%
1,7	80,20%	10,40%	9,40%
1,8	81,50%	8,80%	9,70%
1,9	82,00%	7,90%	10,10%
2	83,10%	6,70%	10,20%
3	85,50%	3,60%	10,90%
7,2	87,70%	1,30%	11%
7,8	**87,90%**	**1,10%**	**11%**
10	87,90%	1%	11,00%
16	87,90%	0,80%	11,30%

Tabelle 6.31: Darstellung der Auswirkung des Thresholds auf die Accuracy, FUR und FPR (zcs; $k = 12$ und $n = 18$) im Phasenraum.

In Tabelle 6.31 ist dargestellt, dass die Accuracy bei einem $t = 7,8$ stagniert. Die Accuracy steigt im Vergleich zu $t = 2$ um 4,8% an. Gleichzeitig sinkt die FUR um 5,6%. Allerdings steigt die FPR dezent um 0,8%.

6.9 Zusammenfassung der Testergebnisse

Die besten Testergebnisse für eine hohe Genauigkeit der Erkennungsrate (personenübergreifend) lieferten die zentrierten, skalierten und rotierten Positionsdaten mit $LR = 2$, $t = 4$, $k = 12$ und $n = 18$ (82%), die zentrierten und skalierten Phasenraumdaten mit $LR = 2$, $t = 7,8$, $k = 12$ und $n = 18$ (87,9%), sowie die aus den zentrierten und skalierten Positionsdaten errechneten Geschwindigkeitsdaten mit $LR = 2$, $t = 12,1$, $k = 12$ und $n = 18$ (ca. 91,2%, was die Hypothese 5.7 bestätigt). Der vierdimensionale Featurevektor konnte keine Verbesserungen im Vergleich zu den Positionsfeatures hinsichtlich der besseren Unterscheidbarkeit von formähnlichen Gesten liefern. Der Test mit der Kombination zweier Hidden Markov Modelle zur besseren Genauigkeit für die Erkennung lieferte die schlechtesten Ergebnisse, welche teilweise unter der zu erwartenden Zufallserkennung von 25% (bei einem Vergleich von vier HMMs) lagen.
Die höchste Genauigkeit für eine personenbezogene Erkennung lieferten, gegen die Vermutung aus Hypothese 5.9, die Ergebnisse auf Basis der Geschwindigkeitsfeatures mit den Parametern: $LR = 2$, $t = 4,7$, $k = 12$ und $n = 18$ (85%). Somit liefern die Geschwindigkeitsfeatures die gesamt besten Ergebnisse für personenbezogene und personenübergreifende Erkennung von Gesten.
Die Parameter der optimalen Werte sind jedoch nicht allgemeingültig. Viel wichtiger als die Ergebniswerte ist die Erkenntnis über die Verbesserungsmöglichkeiten durch Datenaufbereitung und Variation der HMM-Parameter.
In fast allen Tests, außer dem Test mit den Geschwindigkeitsfeatures, wurden Positionsdaten der Geste miteinbezogen. Es wurde festgestellt, dass Ortskoordinaten nicht ohne weitere Vorverarbeitung für die Erkennung und Klassifizierung mittels HMMs geeignet sind. Durch den technischen Umstand, dass Nutzer an verschiedenen Orten mit verschiedenen Entfernungen zur Kinect stehen können, müssen diese Gesten mindestens ortsnormiert werden, bevor eine Erkennung darauf angewendet werden kann, was Hypothese 4.4 für Positionsdaten bestätigt (siehe Verbesserung der Accuracy von *c* zu *zsc* in allen Tests mit Positionsdaten). Bei Gesten, welche in verschiedene Drehrichtungen ausgeführt werden können, muss eine Vergleichbarkeit geschaffen werden. Eine Möglichkeit zeigt die Hypothese der verbesserten Vergleichbarkeit nach einer angepassten Drehrichtung 5.1 der Geste auf, welche in Kapitel 6.3.3 bestätigt wird. Die Vereinheitlichung der Drehrichtung wird vor allem dadurch bedingt, dass die Gesten zur eindeutigen Präsentierung eine immer wiederkehrende Abfolge an Zuständen einhalten sollen; weshalb auch der LR-Parameter auf 2 gesetzt wird. Werden verschieden gedrehte Abläufe in die Modellbildung integriert, entsteht kein verlässliches Modell, um eine Geste zu beschreiben. Die Annahme, dass für manche Gesten oder Situationen (beispielsweise schnell durchgeführte Gesten) ein höherer LR-Parameter bessere Ergeb-

nisse für Accuracy, FUR und FPR liefert, bestätigte sich nicht. In realen Situationen, in denen Gestenerkennung in Echtzeit durchgeführt wird, könnte die Veränderung des Parameters jedoch Relevanz bekommen, da Abschnitte von Gesten kurz verdeckt oder sehr schnell ausgeführt werden können. Das führt dazu, dass eventuell ein Abschnitt der Geste zu einem bestimmten Clusterzentrum fehlt, bzw. fälschlicherweise einem anderen Zentrum zugewiesen wird und der LR-Parameter von 2 nicht mehr ausreicht, um diese Geste zu beschreiben. Deshalb kann über die allgemeine Gültigkeit der Hypothese 5.2 keine Aussage getroffen werden. Im Rahmen dieser Arbeit gilt sie jedoch. Um den LR-Parameter auch bei unterschiedlich gedrehten Gesten gering zu halten, wurde in dieser Arbeit die Drehrichtungen aller Gesten angepasst. Dies ist jedoch nicht der einzig mögliche Lösungsansatz. Ein weiterer denkbarer Ansatz wäre, für beide Drehrichtungen ein Hidden Markov Modelle zu trainieren. Jedoch steigt mit der Anzahl an verglichenen Modellen nicht nur die Rechenzeit des Programms, sondern auch die Wahrscheinlichkeit, Gesten falsch zu klassifizieren.
Durch die Hinzunahme von Features gibt es nicht immer einen sinnvollen Informationsgewinn über die Eigenschaften der Geste. Features müssen nutzungskontextbezogen abgewogen und entwickelt werden. Zum Kontext gehört hierbei die Überlegung, welche Gesten verwendet werden, wie diese ausgeführt werden und ob beispielsweise Ähnlichkeiten der genutzten Gesten (wie beispielsweise Kreis und Quadrat) in diesem Nutzungskontext auftreten. Sind Gesten, wie in dieser Arbeit Kreis und Quadrat, sehr ähnlich, müssen andere Features betrachtet werden als bei Gesten, die sich in ihrer Form stark unterscheiden. Es können verschiedene Features gewählt werden, um diese Unterscheidung der ähnlichen Gesten durchzuführen. Ein Ansatz, der in dieser Arbeit vorgestellt wird, ist die Geschwindigkeiten der einzelnen Koordinaten miteinzubeziehen und den Phasenraum der Geste zu analysieren. Die in dieser Arbeit verwendeten Positions-HMMs verwechseln Kreise und Quadrate, so wie es in Hypothese 4.1 angenommen wurde. Bei HMMs, welche auf Positionsdaten agieren, muss dies auch so sein. Wie in Abbildung 6.16 dargestellt, unterscheiden sich die Clusterzentren der beiden Gesten (fast) gar nicht. Werden Clusterzentren über eine Mischung aus Kreisen und Quadraten gebildet, so wird an den gebildeten Clusterzentren deutlich, dass diese auf beide Gesten gleichzeitig passen. Nicht nur die Clusterzentren bei Kreisen und Quadraten werden ähnlich gelegt, sondern ihre Reihenfolge wird ebenfalls gleich durchlaufen. Für das HMM ist die LLH eines Kreises ein Quadrat zu sein demnach zwangsweise ähnlich hoch, wie für ein Quadrat ein Quadrat zu sein. Das Hidden Markov Modell müsste, um diese Gesten unterscheiden zu können, Informationen über prägnante Unterschiede der Gesten besitzen. Durch die Hinzunahme der Geschwindigkeiten (v_x, v_y, v_z) sind solche Informationen gegeben. Die Erkennung verbessert sich um ca. 6%. Diese geringe Verbesserung liegt daran, dass auch nach Einbezug des Phasenraums ca. 25% der

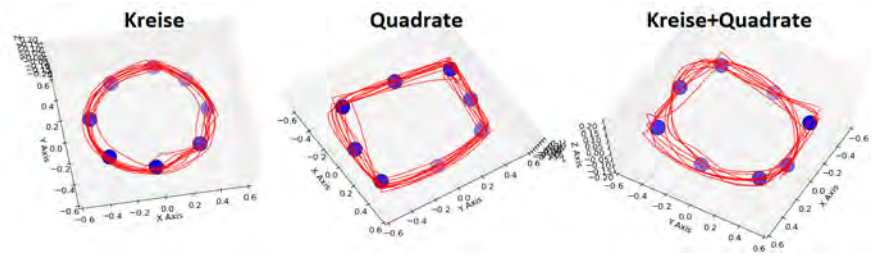

Abbildung 6.16: Visualisierung der Clusterzentren von Kreisen (1) von Quadraten (2) und einer Mischung aus Kreisen und Quadraten (3).

Quadrate falsch als Kreise erkannt werden und 10% als unbekannt klassifiziert werden. Dies liegt unter anderem an den verschiedenen Startpunkten der Square-Geste. Andere Ansätze, welche eventuell eine größere Verbesserung hervorrufen, sind denkbar. Es könnte die Länge der Distanzvektoren der originalen Beobachtungspunkte zu ihren Clusterzentren betrachtet werden, um festzustellen, ob es ein gleichbleibender Abstand ist oder diese in einer ganz bestimmten Signatur Längenunterschiede aufweisen. Wird ein Kreis auf die Clusterzentren eines Quadrats zugewiesen, so liegen einige Zentren sehr passend zur Geste, andere aber ein Stück außerhalb der Geste, wie in Abbildung 6.16 (3) dargestellt ist. Die Ecken des Quadrats ragen deutlich über die äußere Trajektorie des Kreises hervor, wenn diese auf die gleiche Größe skaliert sind. Wären diese Unterschiede feststellbar, könnten Kreise und Quadrate ebenfalls auseinander gehalten werden. Ein Ansatz diese Unterschiede zu analysieren, wäre über die Radien der Kreise, die entstehen, wenn man vom Mittelpunkt der Geste zu jedem beobachteten Featurepunkt den Kreisradius ermittelt.

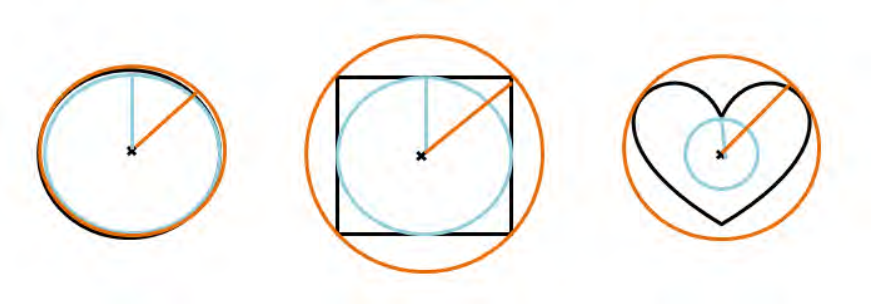

Abbildung 6.17: Visualisierung der Kreisradien der verschiedenen Gesten: (1) von Kreisen (2) von Quadraten (3) von Herzen.

Sind die Gesten der Nutzer durch den Nutzungskontext sehr stark im Raum gedreht, laufen aber auf einer Ebene ab, so kann die Action Plane (Rotation sowie Projektion) die Genauigkeit der Erkennung maßgeblich beeinflussen. Bei Gesten, welche von ihrer

Drehung bzw. Ausrichtung sehr ähnlich ausfallen, kann die Action Plane im Zweifelsfall sogar eine Verschlechterung der Erkennung herbeiführen.
Generell treten in den Tests der verschiedenen Features ähnliche Probleme, ausgelöst durch immer verschiedene Faktoren, auf. Die Verwechslung von Gesten ist der am häufigsten aufgetretene Fall, welcher die Genauigkeit der Erkennung mindert. Durch die verschiedenen besten Ergebnisse für Accuracy, FUR und FPR für die verschiedenen Tests wurde die Hypothese 5.3 bestätigt. Die Anzahl an Hidden States und Observable Symbols muss für verschiedene Gesten, Features und ggf. weitere Aspekte passend gewählt werden. Zwar sind alle besten Ergebnisse mit $k = 12$ und $n = 18$ erzielt worden, in den vorangegangenen Tests konnte jedoch beobachtet werden, dass bei verschiedenen LR-Parametern oder Verarbeitungsschritten andere Anzahlen für Hidden States und Observable Symbols bessere Ergebnisse lieferten, wie es in Kapitel 6.3.2 und Tabelle 6.3 dargestellt wird. Ein weiterer Ansatz ist, die Art der Klassifikation zu verändern. Eine häufige Verwechslung kann auch durch das Prüfen auf mehrere Klassifikatoren verhindert werden, wenn für diese eine sinnvolle Art der Verknüpfung der Ergebnisse besteht. Ein Negativbeispiel ist der in dieser Arbeit entworfene Ansatz der Kombination der Ergebnisse zwei verschiedener Hidden Markov Modelle 6.5 für dieselbe Geste. Die Hypothese 5.8 bestätigte sich somit durch diese Arbeit nicht. Durch eine andere Kombination der Ergebnisse, beziehungsweise andere Kombination an Modellen (beispielsweise das Modell der Geschwindigkeitfeatures (v_x, v_y, v_z) mit dem Modell der Positionsfeatures), könnte dieser Ansatz bessere Ergebnisse im Hinblick auf Unterscheidbarkeit formähnlicher Gesten liefern. Die in Kapitel 6.8 dargestellte Thresholdvariation wurde unter der Annahme durchgeführt, dass der Threshold-Parameter $t = 2$ nicht für alle hier verwendeten Featurevektoren die optimale Wahl darstellt. Es wurde gezeigt, dass für alle getesteten Varianten die Accuracy durch die Verringerung des Thresholds steigt, jedoch zu Gunsten der ebenfalls steigenden False positive rate. Durch die Verschiebung des Thresholds bis zur Konvergenz der Accuracy wird die Grenze der Erkennung zwar deutlich „niedriger" angesetzt, fällt aber nicht weg. Wird der Threshold über diesen Wert hinaus verschoben, siehe beispielsweise Tabelle 6.31, verändert sich die Accuracy nicht mehr, die FPR steigt jedoch weiter an, bis keine Geste mehr als „unknown" klassifiziert wird. Demnach kann über die Hypothese des Thresholds 5.4 bestätigt werden, dass de Accuracy durch die Veränderung des Thresholds steigen kann. Diese Verschiebung der Erkennungsgrenze beinhaltet jedoch ebenfalls die Erhöhung der FPR, was es in den meisten Fällen zu vermeiden gilt. Ob diese angestiegene FPR im Vergleich zur ebenfalls höheren Accuracy für die Gestenerkennung zuträglich oder gar schädlich ist, muss in dem Kontext, in dem die Gestenerkennung eingesetzt werden soll, analysiert werden. In dieser Arbeit wurde wertneutral festgestellt, dass die Tresholdveränderung ebenfalls die Ergebnisse der Accuracy, FUR und FPR beeinflusst. Wie bereits erwähnt,

wurde die Hypothese 5.9 über die personenbezogene Erkennung im Phasenraum nicht bestätigt. Wie in Kapitel 6.7.1 beschrieben, liegt dies mitunter an den ungeeigneten Normierungsverfahren für die Gesten. Um Gesten derselben Person immer wieder zu erkennen, darf die Erkennung nicht ortsbezogen sein, weshalb die Ortsnormierung zwingend ist. Im Phasenraum sind die Ortskoordinaten der Geste enthalten, welche durch die Verarbeitungen zentriert und skaliert sehr ähnlich werden. Dies ist für die Erkennung des Gestentyps, wenn unter anderem Positionsfeatures verwendet werden, sehr wichtig. Die Positionsdaten der Kreise und Quadrate werden bei der Erkennung anhand von Positions-HMMs verwechselt (siehe 8.2). Bei den Geschwindigkeitsdaten-HMMs werden vor allem einige Kreise als Quadrate und einige Zs als Quadrate erkannt (siehe 8.6). Im Phasenraum wurde festgestellt, dass vor allem ca. 30% der Quadrate als Kreis und ca. 10% der Quadrate als „unknown" erkannt werden. Kreise, Herzen und Zs werden weniger als 10% gesamt verwechselt (siehe 8.20). Da Quadrat-Gesten, wie bereits beschrieben, an verschiedenen Stellen begonnen wurden, sind die Positionsdaten mancher Quadrate verschieden. Zu den verschiedenen Anfangskoordinaten, was bereits Unsicherheiten in der HMM erzeugt, kommen die zwangsweise ebenfalls anderen Geschwindigkeiten in den einzelnen Ortskoordinatenkomponenten hinzu. Diese sind zwangsweise anders, da die Geschwindigkeiten der Gesten differieren, werden diese in einer Ecke oder in der Mitte einer Geraden begonnen. Wird die Geste in der Mitte der Geraden begonnen, so beschleunigt der Nutzer fünf mal pro Geste. Anfangs- und Endgeschwindigkeiten sind geringer, als in den durchgängigen Geraden des Quadrates. Wird ein Quadrat in einer Ecke begonnen (Startpunkte in beiden oberen Ecken sind vorhanden) existieren nur vier Beschleunigungen, welche in der Regel ähnliche Geschwindigkeiten erzielten (Bei manchen Nutzern waren die horizontalen Geschwindigkeiten höher, als die vertikalen Geschwindigkeiten.). Demnach ist der Phasenraum anfällig für die beschriebenen Differenzen in der Gestenausführung. Werden nur die Gesten Kreis, Herz und Z betrachtet, wäre die Erkennungsgenauigkeit des Phasenraums für personenübergreifende Gesten bei über 90%. Alle in dieser Arbeit verwendeten kurzen Featurevektoren haben unter bestimmten Voraussetzungen Verwechslungspotential. Das Z ist einem Quadrat in seinen Geschwindigkeiten ähnlich, der Kreis dem Quadrat in der Form und somit entstehen ebenfalls Verwechslungen zwischen Quadrat, Z und Kreis im Phasenraum. Da die HMMs deutliche Einschränkungen im Bezug auf die Erkennung von Gesten mit verschiedenen Start- und Endpunkten aufweist, wäre es ggf. sinnvoll die Start- und Endpunkte der Gesten auf eine gemeinsame Stelle zu drehen bzw. verschieben. Die Geste könnte hierfür den Startpunkt und die Lücke zwischen Start und Endpunkt so verschieben, dass beispielsweise alle Quadrate in der gleichen Ecke beginnen, wie in Abbildung 6.18 dargestellt.

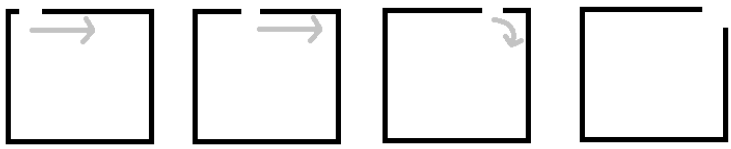

Abbildung 6.18: Darstellung wie der Anfangspunkt (von „irgendwo in der oberen Gerade") eines Quadrats in die obere rechts Ecke gedreht bzw. verschoben werden könnte.

Diese Einschränkungen und die Eigenschaften der Hidden Markov Modelle müssen bei einer Umsetzung bedacht werden. Sind die Parameter der HMM gewählt, so gelten diese für alle Gesten. Alle diese Faktoren (Wie ähnlich sind sich Gesten und in welchen Features drückt sich diese Ähnlichkeit aus?; Welche Parameter der HMM sind für den Nutzungskontext sinnvoll[4]?) müssen in einem realen Nutzungskontext bei der Umsetzung bedacht und entsprechend ausgestaltet werden.

[4]Ein höherer LR-Parameter kann beispielsweise bei potentiellen Verdeckungen von Teilen der Geste sinnvoll sein oder bei schnell ausgeführten Gesten.

7 Fazit

Die Erhebung eigener Gestendaten wurde für den Kontext dieser Arbeit erfolgreich durchgeführt. Aufbauend auf einer Analyse bereits existenter Datensätze konnte ein Datenspeicherungsformat gewählt werden, welches die wichtigen, mit der Kinect aufzeichenbaren Features, beinhaltet. Als besonders wichtig stellte sich dabei eine hohe Framerate heraus, um die Gestendaten nicht durch eine schlechte Abdeckung der Gestendatenpunkte zu verzerren. Die Daten müssen zur Verwendung für ein Hidden Markov Modell so aufbereitet werden, dass die einzelnen Datenpunkte durch ein vorher definiertes Set an Symbolen beschrieben werden. Die Bildung der Symbole mittels des K-Means-Algorithmus, stellte sich als effektiv heraus, da es in dieser Arbeit um positionsbezogene Daten, oder deren Ableitungen, handelt. Wie die HMMs funktionieren und wie die beschriebenen Features verwendet werden, wird in den Kapiteln 3 und 5 ausgeführt.

Ganz ohne Vorverarbeitung können die Daten nicht sinnvoll verwendet werden, da die Erkennung von Gesten unabhängig von ihrem Ausführungsort ablaufen sollte. Bei Ortskoordinaten als Features bietet sich hierfür eine Zentrierung der Gesten in einen gemeinsamen Mittelpunkt an. Für eine bessere Vergleichbarkeit der Gesten bewährte sich vor allem die Skalierung auf eine gemeinsame Größe. Die Verwendung der Action Plane, sowohl „Rotation in die Action Plane" als auch „Projektion und Rotation in die Action Plane" bringen nicht immer eine Verbesserung der Genauigkeit der Erkennung. Die Anwendung so vieler Methoden zur Vereinheitlichung birgt das Risiko, die Gesten zu ähnlich zu machen und erhöht das Verwechslungsrisiko vor allem bei der personenbezogenen Erkennung von Gesten. Durch die Verarbeitung r und p nimmt bei vielen Gesten die FUR ein wenig zu. In der Tabelle 8.2 ist ebenfalls zu erkennen, dass Kreise mit einer Verarbeitung von *zcrps* weniger genau erkannt werden, als bei *zcs* oder *zcrs*. Dies kann jedoch auch daran liegen, dass die HMMs für jeden Durchlauf neu gebildet werden und die gebildeten Zentroiden für Squares, besonders gut auf die Kreis-Testgesten passten. Die Verwendung der Action Plane brachte in dieser Arbeit nur eine geringe Verbesserung bei Gesten mit sehr unterschiedlicher Lage zueinander. Die Verwendung der Action Plane wird für personenübergreifende Gesten vorgeschlagen, welche kontextbezogen sehr unterschiedliche Lagen im Raum haben können oder horizontale wie vertikale Haupt-Aktionsebenen beinhalten (beispielsweise: Kreisgeste

horizontal über dem Kopf ausführen oder Kreisgeste vertikal vor dem Körper ausführen).

Die Erkennung von Personen anhand ihrer Gesten-Trajektorie im Phasenraum erzielte in den Tests in dieser Arbeit eine Genauigkeit von ca. 52% für die Vorverarbeitung von *zcs* (LR=2, k=8, n=12). Die Erkennungsgenauigkeit anhand der Geschwindigkeiten fiel höher aus. Somit können Personen nur bedingt an ihrer Phasenraumtrajektorie erkannt werden, da die Genauigkeit zwar über einer zufälligen Verteilung liegt (diese liegt bei 25%), aber mit einer Genauigkeit von ca. 52% nur jede zweite Geste der richtigen Person zuordnet.

Insgesamt konnten trotz der kurzen Featurevektoren Erkennungsgenauigkeiten von bis zu 80-90% auf realen Nutzerdaten erreicht werden. Abzüge in der Genauigkeit kamen durch ähnliche Gesten zustande, welche bei der Erkennung verwechselt wurden. Auf ähnlichen Daten derselben Geste, wie beispielsweise den Daten von Hall, wurde eine Erkennungsgenauigkeit von 100% erreicht. Kontextbezogen (abhängig von Verwendungszweck, Anforderungen, Nutzungskontext, Umgebungseinflüssen, Nutzern, Aufgaben und verwendeter Hard- und Software u.v.m.) ist demnach abzuwägen, ob solch kurze Featurevektoren, wie in dieser Arbeit verwendet, für eine Gestenerkennung ausreichen. Ein Test, um dies zu prüfen, wäre mehr und verschiedene Probanden hinzuzuziehen. Da die realen Nutzer alle erwachsen waren und in diesen Tests keine Kinder miteinbezogen wurden, wird vermutet, dass für die Erkennung der Gesten von Kindern ein robusterer Algorithmus verwendet werden muss, welcher auch bei nur sehr ungenauer Ausführung beispielsweise Kreis und Quadrat verlässlich unterscheiden kann, um die Frustration der Nutzer gering zu halten.

Der Gestenerkennungsprozess ist demnach eine Balance zwischen der Schaffung von Vergleichbarkeit versus der Gefahr, Gesten durch eine Vereinheitlichung einer zu hohen Verwechslungsgefahr durch Ähnlichkeiten auszusetzen.

8 Ausblick

Obwohl die Erkennung der realen Nutzergesten eine Genauigkeit von bis zu 91,2% angenommen hat, können und sollten noch viele Verbesserungen entwickelt werden. Diese hohe Erkennungsgenauigkeit ist vor allem darum mit Vorsicht zu genießen, weil die Erkennung falscher Gesten bei fast 10% liegt. Eine Falscherkennung von 10% kann zu einer großen Frustration führen. Bei der Entwicklung von gestenbasierten Natural User Interfaces sollte darauf geachtet werden, dass sich das System dem Nutzer anpasst und nicht der Nutzer versucht die Fehlleistung des Systems durch das Erlernen der Vermeidung solcher Falscherkennungen auszugleichen. Eine besondere Schwachstelle sind formähnliche Gesten, wie durch die durchgeführten Tests dargelegt wurde. Allerdings sollten auch solche Gesten mit einer hohen Genauigkeit unterschieden werden können. Dazu können weitere Features hinzugezogen oder andere Verarbeitungsschritte vor der Bildung der HMM mittels der Gestendaten angesetzt werden. Ein solcher Ansatz zur Erweiterung des Featurevektors wäre die Gestenlage zum Kopf der ausführenden Person auszuwerten, welche bereits bei der Datenerhebung bedacht wurde. Dies könnte sich vor allem in Hinblick auf eine Verbesserung der personenbezogenen Erkennungsgenauigkeit auswirken.

Des Weiteren sollten komplexere Gesten miteinbezogen werden, um die Wirkung der Action Plane weiter zu testen. Für Gesten, welche mehrere Hauptaktionsebenen besitzen (beispielsweise Tanzschritte oder Bewegungstechniken aus Gymnastik oder Kampfsport), kann die Verwendung mehrerer Action Planes angedacht werden, um die Geste zu beschreiben. Für sehr komplexe Gesten sollten bei der Aufzeichnung mehrere Aufnahmegeräte verwendet werden, um mögliche falsche Aufzeichnungen durch Verdeckungen zu minimieren. Dazu kommt, dass durch das Testen auf verschiedene Aufzeichnungen eine höhere Genauigkeit, ähnlich der Kreuzvalidierung, entsteht.

Die Parameter der HMM, die in den Tests angepasst wurden, um eine optimale Erkennung zu erzielen, könnten ebenfalls automatisch variiert werden. Der Threshold könnte beispielsweise so lange verkleinert werden, bis die Genauigkeit der Erkennung konvergiert (so wie es in den beschriebenen Tests manuell durchgeführt wurde).

Für den realen Einsatz muss ebenfalls eine Echtzeiterkennung entwickelt werden. Für diese ist die zuverlässige Segmentierung der Gesten ein sehr zentraler Aspekt. Ein möglicher Ansatz wäre, ebenfalls für Ruhegesten HMMs zu entwickeln und auf

diese zu testen. Durchgehend müsste ein gewisser Abschnitt der aktuellen Bewegung des Nutzers, ähnlich einem Sliding Window Verfahren, auf alle verfügbaren Modelle getestet werden. Andere Ansätze sind denkbar.

Insgesamt hat diese Arbeit gezeigt, dass die Hidden Markov Modelle für die Erkennung von Gesten eingesetzt werden können, welche Herausforderungen und Probleme auftreten können und wie damit umzugehen ist. Für das strategische Ziel, die Verbesserung von Gestenerkennung für den Einsatz in Natural User Interfaces, wurden einige Erkenntnisse durch die hier getesteten Features und Verbesserungsansätze aufgezeigt.

Hypothesen

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[1] Hervé Abdi. Partial least squares regression (PLS-regression), 2003. 45

[2] Daniel Bertram. Untersuchungen zur Varianzreduktion beschleunigungsbasierter 3D-Gestendaten, 2012. 27

[3] M Billinghurst. Chapter 14: Gesture-based Interaction. *Human Input to Computer Systems: Theories, Techniques and Technologies*, 2011. 14

[4] Azfar Bin Tomi and DayangRohayaAwang Rambli. A Conceptual Design for Augmented Reality Games Using Motion Detection as User Interface and Interaction. In Halimah Badioze Zaman, Peter Robinson, Maria Petrou, Patrick Olivier, Timothy K. Shih, Sergio Velastin, and Ingela Nyström, editors, *Visual Informatics: Sustaining Research and Innovations*, volume 7067 of *Lecture Notes in Computer Science*, pages 305–315. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25199-3. doi: 10.1007/978-3-642-25200-6_29. URL http://dx.doi.org/10.1007/978-3-642-25200-6_29. 6

[5] Phil Blunsom. Hidden Markov models. *Lecture notes, August*, 2004. 15, 18, 19

[6] Paloma Cantón, Ángel L. González, Gonzalo Mariscal, and Carlos Ruiz. Applying New Interaction Paradigms to the Education of Children with Special Educational Needs. In Klaus Miesenberger, Arthur Karshmer, Petr Penaz, and Wolfgang Zagler, editors, *Computers Helping People with Special Needs*, volume 7382 of *Lecture Notes in Computer Science*, pages 65–72. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-31521-3. doi: 10.1007/978-3-642-31522-0_10. URL http://dx.doi.org/10.1007/978-3-642-31522-0_10. 14

[7] Baptiste Caramiaux, Frédéric Bevilacqua, and Norbert Schnell. Analysing gesture and sound similarities with a HMM-based divergence measure. In *Proceedings of the Sound and Music Computing Conference, SMC*, 2010. 11

[8] Paul Doliotis, Alexandra Stefan, Christopher McMurrough, David Eckhard, and Vassilis Athitsos. Comparing gesture recognition accuracy using color and depth information. In *Proceedings of the 4th International Conference on PErvasive Technologies Related to Assistive Environments*, PETRA '11, pages 20:1–20:7, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0772-7. doi: 10.1145/2141622.2141647. URL http://doi.acm.org/10.1145/2141622.2141647. 29

[9] Abdallah El Ali, Johan Kildal, and Vuokko Lantz. Fishing or a Z?: investigating the effects of error on mimetic and alphabet device-based gesture interaction. In *Proceedings of the 14th ACM international conference on Multimodal interaction*,

ICMI '12, pages 93–100, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1467-1. doi: 10.1145/2388676.2388701. URL http://doi.acm.org/10.1145/2388676.2388701. 27

[10] Rita Francese, Ignazio Passero, and Genoveffa Tortora. Wiimote and Kinect: gestural user interfaces add a natural third dimension to HCI. In *Proceedings of the International Working Conference on Advanced Visual Interfaces*, AVI '12, pages 116–123, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1287-5. doi: 10.1145/2254556.2254580. URL http://doi.acm.org/10.1145/2254556.2254580. 7

[11] Jonathan C. Hall. How to Do Gesture Recognition With Kinect Using Hidden Markov Models (HMMs). 2011. URL http://www.creativedistraction.com/-demos/gesture-recognition-kinect-with-hidden-markov-models-hmms/. 8, 16, 20, 21, 22, 23, 27, 39, 52, 58, 62, 65, 77, 78, 81, 89, 93, 96, 113

[12] Michael Hamilton. Python Hidden Markov Model, http://www.cs.colostate.edu/~hamiltom/code.html. 2011. 39, 104

[13] Wolfgang Hürst and Casper Wezel. Gesture-based interaction via finger tracking for mobile augmented reality. *Multimedia Tools and Applications*, 62(1):233–258, 2013. ISSN 1380-7501. doi: 10.1007/s11042-011-0983-y. URL http://dx.doi.org/10.1007/s11042-011-0983-y. 6

[14] Cem Keskin, AliTaylan Cemgil, and Lale Akarun. DTW Based Clustering to Improve Hand Gesture Recognition. In AlbertAli Salah and Bruno Lepri, editors, *Human Behavior Understanding*, volume 7065 of *Lecture Notes in Computer Science*, pages 72–81. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25445-1. doi: 10.1007/978-3-642-25446-8_8. URL http://dx.doi.org/10.1007/978-3-642-25446-8_8. 11, 15, 29

[15] Patrick Koch, Wolfgang Konen, and Kristine Hein. Gesture recognition on few training data using slow feature analysis and parametric bootstrap. In *Neural Networks (IJCNN), The 2010 International Joint Conference on*, pages 1–8. IEEE, 2010. 7, 11, 27

[16] Hyeon-Kyu Lee and J.H. Kim. An HMM-based threshold model approach for gesture recognition. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(10):961–973, 1999. ISSN 0162-8828. doi: 10.1109/34.799904. 6, 15

[17] Jörg Liesen and Volker Mehrmann. Die Singulärwertzerlegung. In *Lineare Algebra*, Bachelorkurs Mathematik, pages 273–280. Vieweg+Teubner Verlag, 2012. ISBN 978-3-8348-0081-7. doi: 10.1007/978-3-8348-8290-5_19. URL http://dx.doi.org/10.1007/978-3-8348-8290-5_19. 45

[18] D. Marquardt. An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi: 10.1137/0111030. URL http://epubs.siam.org/doi/abs/10.1137/0111030. 45

[19] 2013 Microsoft. Kinect for Windows Sensor Components and Specifications. *http://msdn.microsoft.com/en-us/library/jj131033.aspx*, 2013. 29

[20] Narges S Milani, Denijel Sakic, Arne Grumpe, Christian Wöhler, and Gernot Fink. Partially Supervised Gesture Recognition. In *Proceedings. 22. Workshop Computational Intelligence, Dortmund, 6.-7. Dezember 2012*, page 259. 7, 8, 11, 20, 21, 23, 24, 93

[21] Nhan Nguyen-Duc-Thanh, Sungyoung Lee, and Donghan Kim. Two-stage Hidden Markov Model in Gesture Recognition for Human Robot Interaction. *Int J Adv Robotic Sy*, 9(39), 2012. 15

[22] JongHwan Oh, Yerhyun Jung, Yongseok Cho, Chaewoon Hahm, Hyeyoung Sin, and Joonhwan Lee. Hands-up: motion recognition using kinect and a ceiling to improve the convenience of human life. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '12, pages 1655–1660, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1016-1. doi: 10.1145/2212776.2223688. URL http://doi.acm.org/10.1145/2212776.2223688. 6, 11

[23] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989. ISSN 0018-9219. doi: 10.1109/5.18626. 7, 15, 16, 18, 19

[24] Jan Richarz and Gernot A Fink. Visual recognition of 3D emblematic gestures in an HMM framework. *Journal of Ambient Intelligence and Smart Environments*, 3 (3):193–211, 2011. 3, 6, 8, 11

[25] Thomas Schlömer, Benjamin Poppinga, Niels Henze, and Susanne Boll. Gesture recognition with a Wii controller. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, pages 11–14, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-004-3. doi: 10.1145/1347390.1347395. URL http://doi.acm.org/10.1145/1347390.1347395. 27

[26] J Schumacher, D Sakič, A Grumpe, Gernot A Fink, and Christian Wöhler. Active Learning of Ensemble Classifiers for Gesture Recognition. In *Pattern Recognition*, pages 498–507. Springer, 2012. 11, 14, 27

[27] F. Soltani, F. Eskandari, and S. Golestan. Developing a Gesture-Based Game for Deaf/Mute People Using Microsoft Kinect. In *Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference on*, pages 491–495, 2012. doi: 10.1109/CISIS.2012.55. 11

[28] S. Soutschek, J. Penne, J. Hornegger, and J. Kornhuber. 3-d gesture-based scene navigation in medical imaging applications using time-of-flight cameras. In *Computer Vision and Pattern Recognition Workshops, 2008. CVPRW '08. IEEE Computer Society Conference on*, pages 1–6, 2008. doi: 10.1109/CVPRW.2008.4563162. 6

[29] Juan P Wachs, Helman I Stern, Yael Edan, Michael Gillam, Jon Handler, Craig Feied, and Mark Smith. A Gesture-based Tool for Sterile Browsing of Radiology Images. *Journal of the American Medical Informatics Association*, 15(3):321–323, 2008. doi: 10.1197/jamia.M2410. URL http://jamia.bmj.com/content/15/3/321.abstract. 6

Anhang

Quellcode

In diesem Kapitel werden wichtige Quellcode Abschnitte dargestellt. Die Methoden zur Erstellung des Hidden Markov Modells (hmm.py) sind von Hamilton erstellt und werden als Grundlage für die hier verwendeten HMM-Verbesserungsansätze und Tests verwendet.

Make Clockwise

```
def determineOrientationDataset(dataset):
    xs, ys = toAxis(dataset)[0:2] # dont need the other axis
    count = 0;
    if len(xs) != len(ys) : raise Exception("Axis must have the same
        length")
    n = len(xs)
    for i in xrange(n) :
        j = (i + 1) % n
        k = (i + 2) % n

        z = (xs[j] - xs[i]) * (ys[k] - ys[j])
        z -= (ys[j] - ys[i]) * (xs[k] - xs[j])

        if z < 0 : count -= 1
        if z > 0 : count += 1
    if count > 0 :
        return "CCW"
    elif count < 0 :
        return "CW"
    else :
        return "ERR"
```

Bestimmt die Drehrichtung der Geste. Es bekommt das Datenarray aus Punkten übergeben und gibt eine Zahl größer als 0 zurück, wenn die Geste gegen den Uhrzeigersinn gedreht ist, kleiner als 0, wenn die Geste mit dem Uhrzeigersinn gedreht ist und für count = 0 gibt diese „ERR" (zur Ausgabe, dass ein Fehler vorliegt) zurück.

```
def determineOrientation(data):
    orientations = []
    for dataset in data:
        orientations.append(determineOrientationDataset(dataset))
    return orientations
```

Diese Methode wendet „determineOrientationDataset" auf allen Gestendatensätzen an.

```
def makeCW (data):
    orientations = determineOrientation(data)
    tData = []
    for dataset, orientation in zip(data, orientations):
```

```
        if orientation == 'CCW' :
            tData.append(reverseDataset(dataset))
        else :
            tData.append(dataset)
    return tData
```

Diese Methode passt die Orientierung bzw. Drehrichtung der Geste an, wenn diese CCW gedreht ist und gibt die gesamten Daten CW zurück.

```
''' this method checks if data contains a z gesture
    a perfect centered and scaled z gesture will have a distance
    of sqrt(2) from start to end point while circle, square and heart
    will have a distance of 0 '''
def isZGesture(data):
    data = centerData(data)
    data = toScale(data)
    mean = 0
    for dataset in data:
        mean += distance(dataset[0], dataset[-1],N=3)
    mean /= len(data)
    if mean < 0.7 : return False
    return True
```

Diese Methode bestimmt, ob die jeweilige Geste ein Z ist, damit „makeClockwise" nicht fälschlicherweise auf ein Z angewendet wird.

Action Plane Rotation und Projektion

```
def angle_between(v1, v2):
    v1_u = unit_vector(v1)
    v2_u = unit_vector(v2)
    ang = numpy.arccos(numpy.dot(v1_u, v2_u))
    if math.isnan(ang):
        if (v1_u == v2_u).all():
            return 0.0
        else:
            return numpy.pi
    return ang
```

Bestimmt den Winkel zwischen zwei Vektoren. Die Methode bekommt zwei Vektoren übergeben und gibt ihren Winkel zurück.

```
def createRotMat (n,a):
    mat = {}
    mat[(0,0)] = (n[0] ** 2 ) * (1 - math.cos(a)) + math.cos(a)
    mat[(0,1)] = (n[0] * n[1]) * (1 - math.cos(a)) - n[2] * math.sin(a)
    mat[(0,2)] = (n[0] * n[2]) * (1 - math.cos(a)) + n[1] * math.sin(a)

    mat[(1,0)] = (n[1] * n[0]) * (1 - math.cos(a)) + n[2] * math.sin(a)
    mat[(1,1)] = (n[1] ** 2) * (1 - math.cos(a)) + math.cos(a)
    mat[(1,2)] = (n[1] * n[2]) * (1 - math.cos(a)) - n[0] * math.sin(a)

    mat[(2,0)] = (n[2] * n[0]) * (1 - math.cos(a)) - n[1] * math.sin(a)
    mat[(2,1)] = (n[2] * n[1]) * (1 - math.cos(a)) + n[0] * math.sin(a)
    mat[(2,2)] = (n[2] ** 2 ) * (1 - math.cos(a)) + math.cos(a)
    return mat
```

Erstellt die Rotationsmatrix aus dem Normalenvektor n und dem Winkel a.

```
def toPlane(data, normals, doProject=True, newNormal=array([1, 1, 1])):
    """ rotate all gestures from their own action plane
        to the plane with normal newNormal
    """
    proj_data = []
    i = 0
    for dataset in data:
        proj_points = []

        rotAxis = cross(normals[i], newNormal)

        rotAxis = rotAxis / numpy.linalg.norm(rotAxis)
        assert(abs(numpy.linalg.norm(rotAxis)-1)<1e-8)
        ang = angle_between(normals[i], newNormal)
        mat = createRotMat(rotAxis, ang)
        td = numpy.transpose(dataset)

        mu = array([mean(td[0]), mean(td[1]), mean(td[2])])
        for point in dataset:
            tmp = [point[0] - mu[0], point[1] - mu[1], point[2] - mu[2]]

            if (doProject == True) :

                tmp = projectToPlane(normals[i], tmp)
                assert(abs(numpy.dot(normals[i], tmp))<1e-8)
            tmp2 = multMatVec(mat, tmp)
            tmp = tmp2
            if doProject:
                assert(abs(numpy.dot(newNormal, tmp))<1e-8)

            proj_points.append([tmp[0], tmp[1], smoothFloat(tmp[2]),
                point[3]])

        proj_data.append(proj_points)
        i += 1
    return proj_data
```

Rotiert die Daten aller Gesten anhand ihrer Normalen in eine gemeinsame Action Plane, die über ihre Normale definiert wird. Wenn doProject = True gesetzt ist, werden diese zusätzlich ebenfalls in diese gemeinsame Action Plane projiziert. Zurückgegeben werden die rotierten/projizierten Daten.

```
def projectToPlane(n, x):
    return subVec(x, multSVec((dotProd(n, x) / dotProd(n, n)) , n))
```

Helfermethode zum Projizieren der Daten anhand ihres Normalenvektors.

Normierungsmethoden

```
def determineCenters (data):
    centers = []
    for dataset in data:
        centers.append(determineCenter(dataset))
    return centers

def determineCenter(dataset):
    x, y, z, t = toAxis(dataset)
```

```
    return [
                max(x) - (max(x) - min(x)) / 2 ,  # center x
                max(y) - (max(y) - min(y)) / 2 ,  # center y
                max(z) - (max(z) - min(z)) / 2 ,  # center z
                min(t)  # beginning of measurement
            ]

def centerData(data):
    centers = determineCenters(data)
    tmpdata = []
    for center , dataset in zip(centers, data) :
        tmpdataset = []
        for item in dataset:
            tmpdataset.append([
                               (item[0] - center[0]),
                               (item[1] - center[1]),
                               (item[2] - center[2]),
                               (item[3] - center[3]),
                               ]
                              )
        tmpdata.append(tmpdataset)
    return tmpdata
```

Die Methoden, um die Gesten anhand ihres Mittelpunktes in den Ursprung zu verschieben. Es werden die Mittelpunkte der Gesten bestimmt, in „centers" gespeichert und in „centerData" wird von jedem Punkt von jeder Geste der Mittelpunkt abgezogen, um die Geste zu verschieben. Optional hätte man diese Transition auch mit einer Matrixoperation durchführen können.

```
def toScale(data):
    tmpdata = []
    for dataset in data :
        x, y, z, t = toAxis(dataset)

        boundaries = [
                      (min(x), max(x)),
                      (min(y), max(y)),
                      (min(z), max(z)),
                      (min(t), max(t))
                     ]  # using the t axis here is wrong
        # print boundaries
        i = 0
        tmpaxis = []
        maxi = max([max(x) - min(x), max(y) - min(y), max(z) - min(z)])
        for ax, bound in zip([x, y, z, t], boundaries) :
            tmpitem = []
            for item in ax:
                if i == 3:
                    tmpitem.append(item - bound[0])
                else :
                    tmpitem.append(item / maxi)
            tmpaxis.append(tmpitem)
            i += 1

        tmpdata.append(toPoints(tmpaxis[0], tmpaxis[1], tmpaxis[2],
            tmpaxis[3]))
```

```
    # print tmpdata
    return tmpdata
```

Um die Gesten auf eine gemeinsame, vergleichbare Größe zu skalieren, muss die Achse bestimmt werden, auf der die größte Ausdehnung vorhanden ist. Ist diese Achse bestimmt, wird anhand der Länge der Geste auf dieser, der Skalierungsfaktor bestimmt, um die Geste auf eine maximale Ausdehnung von 1 zu skalieren. Die jeweils kürzeren Ausdehnungen auf den anderen Achsen werden ebenfalls mit diesem Faktor skaliert, um die Geste nicht zu verzerren.

Berechnung der Geschwindigkeiten

```
def createSpeeds(data,stripT=False):
    speeds = []
    for dataset in data:
        x, y, z, t = toAxis(dataset)

        maxi = max([max(x) - min(x), max(y) - min(y), max(z) - min(z)])
        old_point = [0, 0, 0]
        old_time = 0
        speed = []
        size = len(dataset)
        for item in dataset:
            delta_t = item[3] - old_time
            s = distance([item[0], item[1], item[2]],old_point,N=3)
            speed.append(s / delta_t if delta_t != 0 else 1) # catching
                a numerical error
            old_point = [ item[0], item[1], item[2]]
            old_time = item[3]
        maxp = max([max(x) - min(x), max(y) - min(y), max(z) - min(z)])
        fp = maxp
        fa = maxi
        speed = numpy.array(speed) / fp * fa
        speed[0] = 0

        # remapping stuff
        tmax = max(t)
        tmin = min(t)
        tn = []
        for item in t: tn.append(remap(item, tmin, tmax, 0, 100))
        # remapping t

        smax = max(speed)
        smin = min(speed)
        sn = []
        for item in speed : sn.append(remap(item,smin,smax, 0,100))
        if stripT :
            speeds.append(sn)
        else :
            speeds.append([sn,tn])
    return speeds
```

Diese Methode bestimmt die Absolutgeschwindigkeiten zwischen den Punkten. Damit die Zeitachse nicht beachtet wird, muss hier stripT=True übergeben werden. Das Remappen von 0 bis 100 ist für einen schönen Plot erstellt worden.

```
# scales the speed component to match the dimensions of the position
    component
def scaleA4 ( a4array ):
    rData = []
    for dataset in a4array :
        x,y,z,v = toAxis(dataset)
        maxi = max([max(x), max(y), max(z)])
        maxv = max(v)
        v = numpy.array(v)
        #v = ( v / maxv ) * maxi
        v = divideArr(v, maxv)
        v = multiplyArr(v, maxi)
        rData.append(toPoints(x, y, z, v))
    return rData
```

Diese Methode skaliert die Absolutgeschwindigkeit auf die Länge der längsten anderen Achse, damit das Verhältnis zwischen den Werten im Gleichgewicht ist. Ein Wert sollte bei der Bildung der K-Means keinen geringeren Einfluss haben als die anderen Werte.

```
def mergeTo4D(data_one, data_two) :
    data_r = []
    for i, j in zip(data_one, data_two) :
        dataset_r = []
        for k,l in zip(i,j) :
            item_r = []
            item_r.append(k[0])
            item_r.append(k[1])
            item_r.append(k[2])
            item_r.append(l)
            dataset_r.append(item_r)
        data_r.append(dataset_r)
    return data_r
```

Diese Methode fügt die Positionsdaten und die Absolutgeschwindigkeit zusammen in einen vierdimensionalen Vektor.

Plot Methoden

```
''' creates a plot where gesture is made flat and adds the speed
    component as
    z axis '''
def createCanyon(data):
    #rotate data to 0,0,1
    normals = getNormalsSVD(data)
    data = toPlane(data, normals, doProject=True, newNormal=array([0,0,1
        ]))
    #get speeds
    speeds = createSpeeds(data)
    #remove timetag
    #replace the z axis
    n_data = [] # new data
    for dataset, speed in zip (data, speeds):
        x,y,z,t = toAxis(dataset)
        z = speed[0]
        n_data.append(toPoints(x, y, z, t))
    return n_data
```

Diese Plot Methode ist zur besseren Visualisierung des vierdimensionalen Featurevektors. Die Darstellung ist in einer Dimension reduziert. Die Geste wird auf eine Action Plane von $[0, 0, 1]$ projiziert und die Z-Achse stellt die Geschwindigkeit dar.

```
def plot3D(data, c='r', kmeanss=None):
    figure = pyplot.figure()
    ax = figure.gca(projection="3d")
    xs = []
    ys = []
    zs = []
    for dataset in data:

        txs = []
        tys = []
        tzs = []
        txs, tys, tzs, tts = toAxis(dataset)
        xs.append(txs)
        ys.append(tys)
        zs.append(tzs)

    for i in range(0, len(xs)):
        ax.plot(xs[i], ys[i], zs[i], c=c)
    if kmeanss != None :
        xs,ys,zs = toAxis(kmeanss)[0:3]
        ax.scatter(xs,ys,zs,c='b',s=400,marker='o',zorder=9)

    ax.set_xlabel("X Axis")
    ax.set_ylabel("Y Axis")
    ax.set_zlabel("Z Axis")
```

Diese Plotmethode plottet dreidimensional. Sie bekommt die Daten, eine Farbe (hier Rot) und ob K-Means-Daten vorhanden sind, um diese mit auszugeben.

```
def plot2D (data, c='r', label='default'):
    figure = pyplot.figure()
    ax = figure.add_subplot(111)
    ax.set_title(label)
    for dataset in data:
        ds = dataset[0]
        dt = dataset[1]
        ax.plot(dt, ds,c=c)
```

Zweidimensionaler Plot. Bekommt die Daten, eine Farbe und ein Label übergeben.

Phasenraummethoden

```
def createPhasespace(data):
    speeds = []

    for dataset in data:

        x, y, z = toAxis(dataset)[0:3]
        maxi = max([max(x) - min(x), max(y) - min(y), max(z) - min(z)])

        old_point = [0, 0, 0]
        old_time = 0
        speed = []
```

```
        time = []
        size = len(dataset)
        for item in dataset:
            delta_t = item[3] - old_time

            delta_p = distance3darr ([ item[0], item[1], item[2]],
                old_point)
            vektor = divide3d(delta_p, delta_t) if delta_t != 0 else [1,
                 1, 1]
            vektor.append(item[3])  # time
            speed.append(vektor)
            time.append(item[3])
            old_point = [ item[0], item[1], item[2]]
            old_time = item[3]

        speed[0] = [0, 0, 0, 0]

        x,y,z,t = toAxis(speed)
        maxp = max([max(x) - min(x), max(y) - min(y), max(z) - min(z)])


        fp = maxp
        fa = maxi

        x = numpy.array(x) / fp * fa
        y = numpy.array(y) / fp * fa
        z = numpy.array(z) / fp * fa

        speed = toPoints(x,y,z,t)

        speeds.append(speed)


    return speeds
```

In dieser Methode werden die Geschwindigkeiten in den einzelnen Positions-Komponenten berechnet. Die Zeit ist hier noch enthalten, da sie ggf. im weiteren Verlauf noch gebraucht wird. Die Zeit wird bei der Berechnung der K-Means etc. nicht beachtet.

```
def mergeTo6D (data_one, data_two):
    data_return = []
    for i, j in zip(data_one, data_two) :
        dataset_return = []
        for k, l in zip(i, j) :
            item_return = []
            item_return.append(k[0])
            item_return.append(k[1])
            item_return.append(k[2])
            item_return.append(l[0])
            item_return.append(l[1])
            item_return.append(l[2])
            dataset_return.append(item_return)
        data_return.append(numpy.array(dataset_return))
    return numpy.array(data_return)
```

In dieser Methode werden die Ortskoordinaten-Daten und die Geschwindigkeitsdaten aus „createPhasespace" zu einem sechsdimensionalen Vektor zusammengeführt.

HMM Methoden

```
def createObservations(data, kMeans):
    YXClustered = getPointClusters(data, kMeans)
    robservations = []
    for dataset in YXClustered:
        robservation = []
        for tp in dataset:
            robservation.append(tp[1])
        robservations.append(robservation)
    return robservations
```

Die Methode bekommt die Daten sowie die Clusterzentren übergeben. YXClustered enthält die Information über den von jedem Punkt nächsten Zentroiden, ausgerechnet über den Vergleich der Minimalabstände in getPointClusters, und dessen Index. Die Methode gibt eine Liste an Observations zurück; das bedeutet, dass jeder Punkt einem Zentroiden zugeordnet wurde und anstelle der Originaldaten der zugeordnete Index enthalten ist. (Die Indizes stellen das Alphabet dar.)

```
def createObservationsND(data, kMeans):
    XYClustered = getPointClustersND(data, kMeans)
    robservations = []
    for dataset in  XYClustered :
        robservation = []
        for item in dataset :
            robservation.append(item[1])
        robservations.append(robservation)
    return robservations
```

Diese Methode macht das gleiche wie „createObservations(data, kMeans)" für den n-dimensionalen Raum. Der einzige Unterschied ist die Verwendung der Methode „getPointClustersND" statt „getPointClusters".

```
'''
data 3 dimensional array of dataset, point and element
returns A List with arrays of all minimal distances between a point and
    its centroids
'''
def getPointClusters(data, centroids):
    YXClustered = []
    for dataset in data:  # matlab n
        tmp = []
        for point in dataset:  # matlab i
            f = []
            for cen_point in centroids:  # matlab j
                # calculates the distance from any point to any centroid
                f.append(math.sqrt((cen_point[0] - point[0]) ** 2.0 + (
                    cen_point[1] - point[1]) ** 2.0 + (cen_point[2] -
                    point[2]) ** 2.0))

            tmp.append((min(f), f.index(min(f))))
        YXClustered.append(tmp)
    return YXClustered
```

Erstellt nach Hall. Die Methode berechnet für jeden Punkt den Abstand zu allen Zentroiden und bestimmt daraus, welcher der Zentroid mit dem geringsten Abstand ist. Die Methode gibt ein Array aus Arrays für jeden Datensatz zurück, welche den Abstand des Punktes zum nächsten Zentroiden und den dazugehörigen Index des Zentroiden enthält.

```
def getPointClustersND (data, centroids):
    YXClustered = []
    for dataset in data:  # matlab n
        tmp = []
        for point in dataset:  # matlab i
            f = []
            for cen_point in centroids:  # matlab j
                # calculates the distance from any point to any centroid
                f.append(distance(cen_point, point,len(cen_point)))

            tmp.append((min(f), f.index(min(f))))
        YXClustered.append(tmp)
    return YXClustered
```

Diese Methode macht das gleiche wie „getPointClusters" für den n-dimensionalen raum.

```
'''
outputsymbols => matlab K
LeftRightTransitions => matlab LR
returns priorTansitionMatrix
'''
def priorTransitionMatrix(K, LR):
    matrix = numpy.matrix(numpy.identity(K))
    matrix = multiply(matrix, (1.0 / LR))

    for i in range(0, K - (LR - 1)):
        for j in range(0, LR):
            matrix[i, i + j] = (1.0 / LR)


    for i in range(K - (LR - 2) - 1, K):
        for j in range(1, K - i + 1):
            matrix[i, i + (j - 1)] = (1.0 / (K - i))

    return matrix
```

Erstellt die initiale Transitionsmatrix anhand der Anzahl der Outputsymbols und des LR-Parameters.

```
def HMMtraining(observations, numberOfHiddenStates,
    numberOfObservableSymbols, scaling, LR=2) :
    assert(type(LR) is int)
    transitionMatrix = numpy.array(priorTransitionMatrix(
        numberOfHiddenStates, LR))

    btmp = numpy.random.random_sample((numberOfHiddenStates,
        numberOfObservableSymbols))
    row_sums = btmp.sum(axis=1)
    b = btmp / row_sums[:, numpy.newaxis]
    V = []
    for i in xrange(numberOfObservableSymbols):
```

```
        V.append(i)

    ''' TRAINING '''
    hmm = HMM(numberOfHiddenStates, V=V, B=b, A=transitionMatrix)
    hmm = baum_welch(hmm, observations, epochs=20, scaling=scaling,
        verbose=False)

    hmm.V = V
    classifier = HMM_Classifier()
    classifier.add_pos_hmm(hmm)
    summ = 0
    count = 0
    for observation in observations:
        summ += classifier.classify_pos_ll(observation, scaling=scaling)
        count += 1
    summ /= count
    hmm.summ = summ

    return hmm
```

Diese Methode ist für das Training der HM zuständig. Sie bekommt die Observations, die Anzahl der Observations und Hidden States einen Scaling-Wert und den LR-Parameter übergeben. Das Skaling=True (True ist default) sorgt für eine logarithmische Skalierung der Wahrscheinlichkeit. Die Methode gibt eine trainierte HMM zurück.

```
def HMMdiscrete(observations, numberOfHiddenStates,
    numberOfObservableSymbols, robservations, thresh=2.0, kmeans=None,
    inputhmm=None,LR=2):

    transitionMatrix = numpy.array(priorTransitionMatrix(
        numberOfHiddenStates, LR))
    btmp = numpy.random.random_sample((numberOfHiddenStates,
        numberOfObservableSymbols))
    row_sums = btmp.sum(axis=1)
    b = btmp / row_sums[:, numpy.newaxis]
    scaling = True
    V = []
    threshold = None
    if (inputhmm == None) :
        for i in xrange(numberOfObservableSymbols):
            V.append(i)

        ''' TRAINING '''

        hmm = HMM(numberOfHiddenStates, V=V, B=b, A=transitionMatrix)
        hmm = baum_welch(hmm, observations, epochs=20, scaling=scaling,
            verbose=False, graph=True)
    else :
        hmm = inputhmm
        summ = hmm.summ
        threshold = hmm.summ * thresh
        print hmm
    classifier = HMM_Classifier()
    classifier.add_pos_hmm(hmm)

    if threshold == None:
```

```
        summ = 0
        count = 0
        for observation in observations:
            summ += classifier.classify_pos_ll(observation, scaling=
                scaling)
            count += 1
        summ /= count
        threshold = summ * thresh

    result = []
    for observation in robservations:
        llh = classifier.classify_pos_ll(observation, scaling=scaling)
        if (math.isnan(llh)) :    llh = -1e10  #symptomatic: code it as
            a REJECT
        result.append([llh, threshold, summ])
    return result
```

Diese Methode kann entweder eine HMM übergeben bekommen oder eine HMM erstellen, wenn es keine übergeben bekommt. Wenn es keine übergeben bekommt, macht diese Methode das gleiche, wie die Methode HMMtraining, gibt die erstellte HMM jedoch nicht zurück, sondern testet diese direkt gegen andere übergebene Gestendaten. Dazu vergleicht diese Methode die (trainierte)HMM mit den Observations weiterer Gesten und gibt die LLH, den Threshold und summ (LLH der trainierten HMM) für jede Geste zurück.

Variantentests der Positionsfeatures

Die Recognition Matrizen der Positionsfeatures sind in Tabelle 6.4 zu finden.

c	Circle	Heart	Square	Z
Circle	0,554	0,126	0,408	0,208
Heart	0,01	0,476	0,056	0,074
Square	0,306	0,268	0,32	0,25
Z	0,016	0,036	0,062	0,41
unknown	0,114	0,094	0,154	0,058

zc	Circle	Heart	Square	Z
Circle	0,582	0,006	0,382	0
Heart	0,004	0,924	0,002	0
Square	0,414	0,006	0,54	0,036
Z	0	0	0	0,954
unknown	0	0,064	0,076	0,01

zcs	Circle	Heart	Square	Z
Circle	0,792	0,004	0,518	0
Heart	0	0,948	0	0
Square	0,206	0,022	0,426	0,032
Z	0	0	0	0,954
unknown	0,002	0,026	0,056	0,014

zcrs	Circle	Heart	Square	Z
Circle	0,836	0,006	0,604	0,002
Heart	0	0,93	0	0
Square	0,158	0,028	0,316	0,006
Z	0	0	0	0,97
unknown	0,006	0,036	0,08	0,022

zcrps	Circle	Heart	Square	Z
Circle	0,68	0,006	0,496	0
Heart	0	0,91	0	0
Square	0,32	0,062	0,43	0,004
Z	0	0	0	0,996
unknown	0	0,022	0,074	0

Tabelle 8.2: Konfusionsmatrizen der Variante 1 auf den Positionsfeatures

c	Circle	Heart	Square	Z
Circle	0,314	0,056	0,224	0,182
Heart	0,036	0,498	0,11	0,134
Square	0,534	0,368	0,51	0,512
Z	0	0,002	0,008	0,126
unknown	0,116	0,076	0,148	0,046

zc	Circle	Heart	Square	Z
Circle	0,012	0,006	0,006	0
Heart	0,006	0,908	0,004	0
Square	0,982	0,022	0,918	0,34
Z	0	0	0	0,65
unknown	0	0,064	0,072	0,01

zcs	Circle	Heart	Square	Z
Circle	0,076	0,004	0,04	0,006
Heart	0	0,896	0	0
Square	0,924	0,072	0,908	0,414
Z	0	0	0	0,568
unknown	0	0,028	0,052	0,012

zcrs	Circle	Heart	Square	Z
Circle	0,05	0,002	0,02	0
Heart	0	0,862	0	0
Square	0,944	0,102	0,904	0,364
Z	0	0	0	0,62
unknown	0,006	0,034	0,076	0,016

zcrps	Circle	Heart	Square	Z
Circle	0,04	0,004	0,016	0
Heart	0	0,84	0	0
Square	0,96	0,142	0,91	0,212
Z	0	0	0	0,788
unknown	0	0,014	0,074	0

Tabelle 8.3: Konfusionsmatrizen der Variante 2 auf den Positionsfeatures

c	Circle	Heart	Square	Z
Circle	0,468	0,104	0,336	0,232
Heart	0,012	0,482	0,076	0,13
Square	0,39	0,334	0,406	0,386
Z	0,012	0,006	0,034	0,206
unknown	0,118	0,074	0,148	0,046

zc	Circle	Heart	Square	Z
Circle	0,06	0,006	0,058	0
Heart	0,004	0,92	0,004	0
Square	0,936	0,01	0,866	0,162
Z	0	0	0	0,828
unknown	0	0,064	0,072	0,01

zcs	Circle	Heart	Square	Z
Circle	0,16	0,004	0,064	0,002
Heart	0	0,92	0	0
Square	0,84	0,048	0,884	0,252
Z	0	0	0	0,734
unknown	0	0,028	0,052	0,012

zcrs	Circle	Heart	Square	Z
Circle	0,242	0,004	0,124	0
Heart	0	0,896	0	0
Square	0,752	0,064	0,804	0,222
Z	0	0	0	0,762
unknown	0,006	0,036	0,072	0,016

zcrps	Circle	Heart	Square	Z
Circle	0,134	0,002	0,104	0
Heart	0	0,854	0	0
Square	0,866	0,13	0,824	0,106
Z	0	0	0	0,894
unknown	0	0,014	0,072	0

Tabelle 8.4: Konfusionsmatrizen der Variante 3 auf den Positionsfeatures

Auswertung der Geschwindigkeitsfeatures

c	Circle	Heart	Square	Z
Circle	0,914	0,396	0,5	0,516
Heart	0,926	0,892	0,742	0,902
Square	0,998	0,826	0,878	0,974
Z	0,686	0,38	0,446	0,832

zc	Circle	Heart	Square	Z
Circle	0,776	0	0,144	0,126
Heart	0,512	0,852	0,248	0,2
Square	0,96	0,374	0,89	0,886
Z	0,704	0,102	0,376	0,764

zcs	Circle	Heart	Square	Z
Circle	0,79	0,008	0,126	0,034
Heart	0,556	0,842	0,362	0,424
Square	0,984	0,432	0,878	0,87
Z	0,622	0,082	0,368	0,802

zcrs	Circle	Heart	Square	Z
Circle	0,8	0,016	0,136	0,028
Heart	0,216	0,834	0,098	0,086
Square	0,986	0,522	0,884	0,93
Z	0,666	0,144	0,384	0,8

zcrps	Circle	Heart	Square	Z
Circle	0,816	0,02	0,146	0,062
Heart	0,412	0,856	0,182	0,246
Square	0,99	0,544	0,898	0,892
Z	0,676	0,202	0,34	0,764

Tabelle 8.5: Recognition Matrizen der Geschwindigkeitsfeature-HMMs

c	Circle	Heart	Square	Z
Circle	0,548	0,038	0,21	0,136
Heart	0,024	0,774	0,048	0,042
Square	0,396	0,054	0,572	0,402
Z	0,014	0,022	0,058	0,39
unknown	0,018	0,112	0,112	0,03

zc	Circle	Heart	Square	Z
Circle	0,644	0	0,014	0,006
Heart	0,014	0,868	0,006	0,004
Square	0,22	0,004	0,746	0,228
Z	0,102	0,008	0,112	0,7
unknown	0,02	0,12	0,122	0,062

zcs	Circle	Heart	Square	Z
Circle	0,656	0	0,03	0,002
Heart	0,008	0,862	0,004	0,018
Square	0,282	0,004	0,758	0,252
Z	0,028	0,008	0,082	0,682
unknown	0,026	0,126	0,126	0,046

zcrs	Circle	Heart	Square	Z
Circle	0,6	0	0,046	0
Heart	0	0,814	0	0,002
Square	0,338	0,026	0,742	0,276
Z	0,038	0,03	0,11	0,69
unknown	0,024	0,13	0,102	0,032

zcrps	Circle	Heart	Square	Z
Circle	0,668	0	0,04	0,01
Heart	0,004	0,844	0	0,008
Square	0,266	0,022	0,786	0,192
Z	0,04	0,014	0,078	0,72
unknown	0,022	0,12	0,096	0,07

Tabelle 8.6: Konfusionsmatrizen der Variante 1 der Geschwindigkeitsfeature-HMMs

Auswertung der Kombination aus Positionsfeature-HMM und Geschwindigkeitsfeature-HMM

c	Circle	Heart	Square	Z
Circle	0,894	0,864	0,85	0,842
Heart	0,958	0,91	0,902	0,914
Square	0,958	0,88	0,89	0,958
Z	0,87	0,812	0,804	0,858

zc	Circle	Heart	Square	Z
Circle	0,944	0,86	0,894	0,96
Heart	0,968	0,938	0,95	0,99
Square	0,974	0,912	0,924	0,99
Z	0,9	0,828	0,798	0,912

zcs	Circle	Heart	Square	Z
Circle	0,852	0,768	0,8	0,868
Heart	0,954	0,906	0,932	0,99
Square	0,982	0,938	0,936	0,994
Z	0,878	0,798	0,77	0,888

zcrs	Circle	Heart	Square	Z
Circle	0,952	0,896	0,91	0,972
Heart	0,872	0,834	0,836	0,892
Square	0,988	0,924	0,934	0,998
Z	0,918	0,832	0,826	0,924

zcrps	Circle	Heart	Square	Z
Circle	0,96	0,886	0,904	0,974
Heart	0,97	0,93	0,932	0,99
Square	0,968	0,91	0,93	0,994
Z	0,942	0,86	0,86	0,952

Tabelle 8.7: Recognition Matrizen der kombinierten Positions- und Geschwindigkeits-HMM-Auswertung

c	Circle	Heart	Square	Z
Circle	0,16	0,138	0,124	0,164
Heart	0,35	0,352	0,38	0,298
Square	0,094	0,112	0,14	0,134
Z	0,396	0,398	0,356	0,404
unknown	0	0	0	0

zc	Circle	Heart	Square	Z
Circle	0,19	0,192	0,186	0,18
Heart	0,198	0,24	0,236	0,19
Square	0,43	0,368	0,382	0,452
Z	0,182	0,2	0,174	0,178
unknown	0	0	0,022	0

zcs	Circle	Heart	Square	Z
Circle	0,06	0,048	0,042	0,07
Heart	0,296	0,308	0,31	0,292
Square	0,352	0,34	0,34	0,332
Z	0,274	0,254	0,236	0,29
unknown	0,018	0,05	0,072	0,016

zcrs	Circle	Heart	Square	Z
Circle	0,324	0,332	0,338	0,312
Heart	0,184	0,162	0,18	0,206
Square	0,278	0,276	0,278	0,274
Z	0,208	0,218	0,184	0,206
unknown	0,006	0,012	0,02	0,002

zcrps	Circle	Heart	Square	Z
Circle	0,01	0,022	0,014	0,006
Heart	0,208	0,2	0,192	0,204
Square	0,472	0,424	0,432	0,512
Z	0,31	0,33	0,316	0,278
unknown	0	0,024	0,046	0

Tabelle 8.8: Variante 1. Kombinierte Positions- und Geschwindigkeits-HMM Auswertung

Auswertungen der Gesten des vierdimensionalen Featurevektors

c	Circle	Heart	Square	Z
Circle	0,828	0,148	0,164	0,284
Heart	0,474	0,81	0,362	0,564
Square	0,972	0,34	0,842	0,83
Z	0,484	0,242	0,314	0,764

zc	Circle	Heart	Square	Z
Circle	0,944	0,054	0,714	0,026
Heart	0,134	0,878	0,09	0,046
Square	0,914	0,118	0,816	0,49
Z	0	0	0	0,832

zcs	Circle	Heart	Square	Z
Circle	0,946	0,016	0,698	0,014
Heart	0	0,794	0	0
Square	0,942	0,202	0,876	0,62
Z	0	0	0	0,902

zcrs	Circle	Heart	Square	Z
Circle	0,912	0,048	0,674	0,054
Heart	0	0,896	0	0
Square	0,958	0,286	0,85	0,282
Z	0	0	0	0,894

zcrps	Circle	Heart	Square	Z
Circle	0,91	0,018	0,692	0,004
Heart	0	0,868	0	0
Square	0,956	0,15	0,834	0,242
Z	0	0	0	0,806

Tabelle 8.9: Recognition Matrizen des 4D Featurevektors $(x, y, z, v_{absolut})^T$

c	Circle	Heart	Square	Z
Circle	0,628	0,038	0,046	0,008
Heart	0,016	0,594	0,034	0,076
Square	0,23	0,12	0,612	0,42
Z	0,1	0,118	0,176	0,488
unknown	0,026	0,13	0,132	0,008

zc	Circle	Heart	Square	Z
Circle	0,586	0,016	0,246	0
Heart	0,008	0,896	0	0
Square	0,402	0,058	0,596	0,068
Z	0	0	0	0,878
unknown	0,004	0,03	0,158	0,054

zcs	Circle	Heart	Square	Z
Circle	0,806	0,004	0,4	0
Heart	0	0,822	0	0
Square	0,188	0,078	0,512	0,048
Z	0	0	0	0,948
unknown	0,006	0,096	0,088	0,004

zcrs	Circle	Heart	Square	Z
Circle	0,768	0,02	0,426	0,004
Heart	0	0,872	0	0
Square	0,184	0,05	0,482	0,028
Z	0	0	0	0,94
unknown	0,048	0,058	0,092	0,028

zcrps	Circle	Heart	Square	Z
Circle	0,53	0,024	0,166	0,004
Heart	0	0,88	0	0
Square	0,45	0,024	0,72	0,062
Z	0	0	0	0,834
unknown	0,02	0,072	0,114	0,1

Tabelle 8.10: Konfusionsmatrizen des 4D Featurevektors $(x, y, z, v_{absolut})^T$

Personenbezogene Auswertungen mittels Positionsfeatures

c	P0	P1	P10	P18
P0	1	0	0	0,0266
P1	0	0,86	0,02	0,2866
P10	0	0	0,88	0
P18	0	0,12	0,0733	0,64
unknown	0	0,02	0,0266	0,0466

zc	P0	P1	P10	P18
P0	0,5066	0,4466	0	0,12
P1	0,22	0,2733	0	0,3
P10	0	0,0333	0,9266	0
P18	0,2733	0,2066	0	0,5666
unknown	0	0,04	0,0733	0,0133

zcs	P0	P1	P10	P18
P0	0,58	0,4466	0	0,4066
P1	0,1866	0,2466	0	0,2133
P10	0	0	0,88	0
P18	0,2333	0,2466	0	0,3666
unknown	0,0000	0,0600	0,1200	0,0133

zcrs	P0	P1	P10	P18
P0	0,5866	0,3333	0	0,22
P1	0,24	0,3733	0	0,3933
P10	0	0	0,8333	0,0133
P18	0,1733	0,2933	0	0,24
unknown	0,0000	0,0000	0,1667	0,1333

zcrps	P0	P1	P10	P18
P0	0,66	0,4533	0	0,3666
P1	0,2333	0,3733	0	0,3066
P10	0	0,0333	0,8733	0,18
P18	0,1066	0,14	0,0066	0,1466
unknown	0	0	0,12	0

Tabelle 8.11: Personenbezogene Auswertung der Konfusionsmatrizen (Variante 1) der Kreis-Geste

c	P0	P1	P10	P18
P0	0,8666	0	0	0
P1	0,0933	0,8333	0	0,0066
P10	0,0333	0,08	0,8866	0,2133
P18	0,0066	0,0666	0,04	0,68
unknown	0	0,02	0,0733	0,1

zc	P0	P1	P10	P18
P0	0,4733	0,0266	0,0466	0
P1	0,2666	0,6533	0,2733	0,2266
P10	0,24	0,1866	0,5666	0,0066
P18	0,02	0,1266	0,1133	0,74
unknown	0	0,0066	0	0,0266

zcs	P0	P1	P10	P18
P0	0,3066	0,12	0,0933	0,08666
P1	0,4866	0,6333	0,3133	0,4133
P10	0,1133	0,08666	0,3933	0,1266
P18	0,0933	0,16	0,2	0,3666
unknown	0	0	0	0,0066

zcrs	P0	P1	P10	P18
P0	0,3066	0,12	0,0933	0,0866
P1	0,4866	0,633	0,3133	0,4133
P10	0,1133	0,0866	0,3933	0,1266
P18	0,0933	0,16	0,2	0,3666
unknown	0	0	0	0,0066

zcrps	P0	P1	P10	P18
P0	0,5466	0,3333	0,2533	0
P1	0,2266	0,4133	0,3667	0,2066
P10	0,1866	0,1733	0,3	0,1733
P18	0,04	0,08	0,08	0,6133
unknown	0	0	0	0,0066

Tabelle 8.12: Personenbezogene Auswertung der Konfusionsmatrizen (Variante 1) der Herz-Geste.

c	P0	P1	P10	P18
P0	1	0	0	0,03333
P1	0	0,98	0,18	0,6466
P10	0	0	0,8666	0
P18	0,06	0,4333	0,1866	0,8267

zc	P0	P1	P10	P18
P0	0,9333	0,8333	0	0,2533
P1	0,9933	0,9333	0	0,66
P10	0,0667	0,1	0,86	0,0466
P18	1	0,6133	0	0,8

zcs	P0	P1	P10	P18
P0	0,94	0,8533	0	0,7067
P1	0,96	0,9333	0	0,7667
P10	0	0	0,8133	0
P18	0,9666	0,88	0	0,8866

zcrs	P0	P1	P10	P18
P0	0,9933	0,9333	0	0,7666
P1	1	1	0	0,8666
P10	0	0	0,8333	0,04666
P18	1	0,9933	0	0,8

zcrps	P0	P1	P10	P18
P0	0,9933	0,9333	0	0,8133
P1	1	1	0	0,8666
P10	0,0733	0,08	0,8666	0,1
P18	1	0,96	0,0266	0,8

Tabelle 8.13: Recognitionmatrizen der personenbezogenen Erkennung der Kreis-Geste (Positionsdaten).

c	P0	P1	P10	P18
P0	0,9133	0	0	0
P1	0,44	0,8533	0,0133	0,0067
P10	0,2533	0,2066	0,88	0,2067
P18	0,0733	0,08	0,1	0,5533

zc	P0	P1	P10	P18
P0	0,9666	0,1066	0,1933	0,0066
P1	0,8866	1	0,72	0,6066
P10	0,9866	0,4733	1	0,06
P18	0,3333	0,42	0,3466	0,9266

zcs	P0	P1	P10	P18
P0	0,9866	0,4933	0,5267	0,2533
P1	0,98	1	0,6533	0,7267
P10	0,96	0,6933	1	0,7267
P18	0,82	0,8733	0,8533	0,9133

zcrs	P0	P1	P10	P18
P0	0,9866	0,4933	0,5267	0,2533
P1	0,98	1	0,6533	0,7267
P10	0,96	0,6933	1	0,7267
P18	0,82	0,8733	0,8533	0,9133

zcrps	P0	P1	P10	P18
P0	0,9933	0,7066	0,54	0
P1	0,96	0,9266	0,92	0,5066
P10	0,9066	0,6466	0,8933	0,44
P18	0,6733	0,4266	0,74	0,9266

Tabelle 8.14: Recognitionmatrizen der personenbezogenen Erkennung der Herz-Geste (Positionsdaten).

c	P0	P1	P10	P18
P0	0,8	0	0	0
P1	0,08	0,6933	0,1333	0,08
P10	0,06	0,2467	0,5933	0,1867
P18	0,06	0,0533	0,26	0,7333
unknown	0	0,0067	0,0133	0

zc	P0	P1	P10	P18
P0	0,5067	0,0067	0,0067	0,0467
P1	0,1	0,3667	0,2467	0,18
P10	0,1	0,16	0,4867	0,2267
P18	0,2933	0,4667	0,26	0,5467
unknown	0	0	0	0

zcs	P0	P1	P10	P18
P0	0,5933	0,3533	0,0733	0,48
P1	0,0733	0,2	0,2	0,1067
P10	0,0067	0,1933	0,4867	0,0267
P18	0,3267	0,2533	0,2133	0,3867
unknown	0	0	0,0267	0

zcrs	P0	P1	P10	P18
P0	0,6067	0,4333	0,2	0,5933
P1	0,1533	0,34	0,2067	0,1467
P10	0,0333	0,12	0,5	0,1133
P18	0,2	0,1	0,0933	0,1467
unknown	0,0067	0,0067	0	0

zcrps	P0	P1	P10	P18
P0	0,6067	0,38	0,16	0,5
P1	0,12	0,4133	0,1667	0,0933
P10	0,08	0,1	0,56	0,2
P18	0,1933	0,1067	0,1133	0,2067
unknown	0	0	0	0

Tabelle 8.15: Personenbezogene Auswertung der Konfusionsmatrizen (Variante 1) der Quadrat-Geste.

c	P0	P1	P10	P18
P0	1	0,0133	0,0067	0,0000
P1	0,64	0,7667	0,3333	0,4400
P10	0,6933	0,4800	0,7600	0,5200
P18	0,42	0,2067	0,5733	0,9067

zc	P0	P1	P10	P18
P0	1	0	0	0
P1	1	1	0,4	1
P10	1	1	1	1
P18	1	0,8	0,4	1

zcs	P0	P1	P10	P18
P0	1,0000	0,6067	0,2067	0,9267
P1	0,9933	0,9000	0,5400	0,9867
P10	0,6867	0,7067	0,7800	0,8867
P18	0,9667	0,6067	0,4067	0,9333

zcrs	P0	P1	P10	P18
P0	0,9933	0,6000	0,3467	0,9333
P1	1,0000	0,9000	0,6667	0,9800
P10	0,8333	0,5400	0,8400	0,9400
P18	1,0000	0,5133	0,4133	0,9333

zcrps	P0	P1	P10	P18
P0	1,0000	0,6533	0,3333	0,9333
P1	0,9933	0,8800	0,6333	0,9800
P10	0,8000	0,5133	0,8333	0,9400
P18	0,9800	0,5267	0,4600	0,9333

Tabelle 8.16: Recognitionmatrizen der personenbezogenen Erkennung der Quadrat-Geste (Positionsdaten).

c	P0	P1	P10	P18
P0	0,5467	0,3267	0,0000	0,0000
P1	0,3267	0,6667	0,0333	0,0067
P10	0,0000	0,0000	0,9467	0,0000
P18	0,1267	0,0000	0,0000	0,9933
unknown	0,0000	0,0067	0,0200	0,0000

zc	P0	P1	P10	P18
P0	0,51333	0,14000	0,00000	0,02000
P1	0,34000	0,24000	0,07333	0,14000
P10	0,08000	0,10000	0,40000	0,02667
P18	0,06667	0,52000	0,52667	0,81333
unknown	0,00000	0,00000	0,00000	0,00000

zcs	P0	P1	P10	P18
P0	0,4667	0,2667	0,0467	0,0667
P1	0,3733	0,3733	0,1733	0,1600
P10	0,0467	0,1467	0,2733	0,2667
P18	0,1133	0,2133	0,5067	0,5067
unknown	0,0000	0,0000	0,0000	0,0000

zcrs	P0	P1	P10	P18
P0	0,2600	0,1933	0,0133	0,0267
P1	0,1533	0,2800	0,1267	0,1333
P10	0,2400	0,1733	0,3800	0,3067
P18	0,3467	0,3467	0,4800	0,5333
unknown	0,0000	0,0067	0,0000	0,0000

zcrps	P0	P1	P10	P18
P0	0,30667	0,29333	0,08667	0,15333
P1	0,20667	0,30000	0,18000	0,08000
P10	0,12000	0,09333	0,23333	0,20667
P18	0,36667	0,30000	0,50000	0,56000
unknown	0,00000	0,01333	0,00000	0,00000

Tabelle 8.17: Personenbezogene Auswertung der Konfusionsmatrizen (Variante 1) der Z-Geste.

c	P0	P1	P10	P18
P0	0,9267	0,7733	0,0067	0,1533
P1	0,5533	0,8867	0,1267	0,0200
P10	0,0000	0,0000	0,9133	0,0000
P18	0,4467	0,0333	0,0000	0,9667

zc	P0	P1	P10	P18
P0	1,0000	0,5867	0,1267	0,3200
P1	0,8133	1,0000	0,4667	0,7733
P10	0,2867	0,4667	0,9600	0,1467
P18	0,2667	0,8400	0,9733	1,0000

zcs	P0	P1	P10	P18
P0	0,9733	0,9533	0,2933	0,4600
P1	0,7800	0,9533	0,5667	0,6733
P10	0,2267	0,5933	0,9667	0,9400
P18	0,6267	0,7467	0,9533	0,9933

zcrs	P0	P1	P10	P18
P0	0,9600	0,6933	0,4200	0,6467
P1	0,9200	0,8400	0,8800	0,9333
P10	0,8733	0,7200	0,9867	1,0000
P18	0,7933	0,7667	0,9733	1,0000

zcrps	P0	P1	P10	P18
P0	0,9600	0,7933	0,5267	0,7200
P1	0,8867	0,7933	0,8533	0,8333
P10	0,8533	0,6600	0,9933	0,9400
P18	0,8067	0,7067	0,9867	1,0000

Tabelle 8.18: Recognitionmatrizen der personenbezogenen Erkennung der Z-Geste (Positionsdaten).

Auswertungen der Phasenraums

c	Circle	Heart	Square	Z
Circle	0,626	0,192	0,504	0,16
Heart	0,202	0,476	0,376	0,384
Square	0,594	0,33	0,652	0,61
Z	0,102	0,21	0,126	0,488

zc	Circle	Heart	Square	Z
Circle	0,828	0	0,464	0,016
Heart	0	0,878	0	0,006
Square	0,882	0,078	0,878	0,132
Z	0	0	0	0,776

zcs	Circle	Heart	Square	Z
Circle	0,928	0	0,542	0
Heart	0	0,808	0	0
Square	0,664	0,016	0,858	0,492
Z	0	0	0	0,912

zcrs	Circle	Heart	Square	Z
Circle	0,944	0	0,47	0,002
Heart	0	0,77	0	0
Square	0,86	0	0,83	0,134
Z	0	0	0	0,846

zcrps	Circle	Heart	Square	Z
Circle	0,928	0	0,49	0,002
Heart	0	0,782	0	0
Square	0,85	0	0,816	0,244
Z	0	0	0	0,852

Tabelle 8.19: Auswertung der Recognition Matrizen (Phasenraum).

c	Circle	Heart	Square	Z
Circle	0,59	0,088	0,392	0,066
Heart	0,018	0,434	0,074	0,108
Square	0,266	0,084	0,32	0,268
Z	0,062	0,136	0,078	0,468
unknown	0,064	0,258	0,136	0,09

zc	Circle	Heart	Square	Z
Circle	0,744	0	0,146	0,012
Heart	0	0,9	0	0
Square	0,218	0,014	0,768	0,066
Z	0	0	0	0,848
unknown	0,038	0,086	0,086	0,074

zcs	Circle	Heart	Square	Z
Circle	0,936	0	0,252	0
Heart	0	0,87	0	0
Square	0,056	0,01	0,676	0,038
Z	0	0	0	0,938
unknown	0,008	0,12	0,072	0,024

zcrs	Circle	Heart	Square	Z
Circle	0,95	0	0,26	0
Heart	0	0,85	0	0
Square	0,046	0	0,634	0,03
Z	0	0	0	0,912
unknown	0,004	0,15	0,106	0,058

zcrps	Circle	Heart	Square	Z
Circle	0,904	0	0,252	0
Heart	0	0,928	0	0
Square	0,082	0	0,63	0,016
Z	0	0	0	0,904
unknown	0,014	0,072	0,118	0,08

Tabelle 8.20: Auswertung der Konfusionsmatrizen (Variante 1 im Phasenraum).

Personenbezogene Auswertung der Phasenraums

c	P0	P1	P10	P18	zc	P0	P1	P10	P18
P0	0,9933	0,24	0	0	P0	0,9733	0,6133	0	0,7133
P1	0,0467	0,9267	0,3733	0,82	P1	0,98	0,9133	0	0,8933
P10	0	0,1533	0,7933	0	P10	0,0333	0,0333	0,88	0,0333
P18	0,0333	0,6933	0,5267	0,8733	P18	0,98	0,6933	0	0,7867

zcs	P0	P1	P10	P18	zcrs	P0	P1	P10	P18
P0	0,9533	0,6933	0	0,6733	P0	0,9733	0,8533	0	0,7467
P1	0,9933	0,9267	0	0,7133	P1	0,9933	0,9533	0	0,7933
P10	0	0	0,7867	0	P10	0	0	0,8533	0
P18	1	0,86	0	0,7933	P18	1	0,9867	0	0,7933

zcrps	P0	P1	P10	P18
P0	0,9933	0,82	0	0,78
P1	1	0,9067	0	0,7133
P10	0	0	0,8533	0,0133
P18	1	0,96	0	0,7533

Tabelle 8.21: Personenbezogene Auswertung des Phasenraums: Recognition Matrizen der Kreis-Geste

c	P0	P1	P10	P18	zc	P0	P1	P10	P18
P0	1	0,1533	0	0	P0	0,6133	0,3333	0	0,3533
P1	0	0,4533	0,1	0,4	P1	0,2333	0,4067	0	0,3667
P10	0	0,0867	0,7067	0	P10	0	0,04	1	0,0067
P18	0	0,3	0,18	0,5867	P18	0,1533	0,1933	0	0,2667
unknown	0	0,0067	0,0133	0,0133	unknown	0	0,0267	0	0,0067

zcs	P0	P1	P10	P18	zcrs	P0	P1	P10	P18
P0	0,5	0,28	0	0,3133	P0	0,3533	0,2533	0	0,24
P1	0,2933	0,3133	0	0,2867	P1	0,4	0,4733	0	0,4467
P10	0	0	0,9267	0	P10	0	0	0,98	0
P18	0,2067	0,3467	0	0,3467	P18	0,2467	0,2733	0	0,1933
unknown	0	0,06	0,0733	0,053333333	unknown	0	0	0,02	0,12

zcrps	Circle	Heart	Square	Z
P0	0,4667	0,28	0	0,433333333
P1	0,2933	0,3667	0	0,3267
P10	0	0	0,9933	0
P18	0,24	0,3533	0	0,2067
unknown	0	0	0,0067	0,0333

Tabelle 8.22: Personenbezogene Auswertung des Phasenraums: Konfusionsmatrizen der Kreis-Geste, Variante 1

Personenbezogene Auswertung der Geschwindigkeitsfeatures

c	P0	P1	P10	P18	zc	P0	P1	P10	P18
P0	0,8600	0,2200	0,0000	0,0267	P0	0,8733	0,2267	0,0000	0,0000
P1	0,6667	0,9800	0,0400	0,0000	P1	0,6467	0,8667	0,0400	0,0000
P10	0,0000	0,0000	0,7933	0,0000	P10	0,0000	0,0000	0,8267	0,0000
P18	0,9600	0,8800	0,0467	0,8667	P18	0,9600	0,8867	0,0467	0,8000

zcs	P0	P1	P10	P18	zcrs	P0	P1	P10	P18
P0	0,8267	0,2467	0,0000	0,0733	P0	0,7600	0,3400	0,0000	0,1133
P1	0,6733	0,9733	0,0533	0,4333	P1	0,7400	0,9067	0,0000	0,4933
P10	0,0000	0,0000	0,8467	0,0000	P10	0,0000	0,0000	0,8467	0,0000
P18	0,8667	0,9533	0,0200	0,8533	P18	0,8933	0,9733	0,0067	0,8800

zcrps	P0	P1	P10	P18
P0	0,8400	0,3200	0,0000	0,0733
P1	0,7067	0,9467	0,0000	0,5267
P10	0,0000	0,0000	0,8267	0,0000
P18	0,8867	0,9600	0,0200	0,8867

Tabelle 8.23: Personenbezogene Auswertung der Geschwindigkeitsfeatures: Recognition Matrizen der Kreis-Geste

c	P0	P1	P10	P18	zc	P0	P1	P10	P18
P0	0,5933	0,1400	0,0000	0,0000	P0	0,6000	0,1067	0,0000	0,0000
P1	0,2267	0,5800	0,0133	0,0000	P1	0,2067	0,5333	0,0000	0,0000
P10	0,0000	0,0000	0,9667	0,0000	P10	0,0000	0,0000	0,9800	0,0000
P18	0,1800	0,2667	0,0000	0,9333	P18	0,1867	0,3200	0,0133	0,9200
unknown	0,0000	0,0133	0,0200	0,0667	unknown	0,0067	0,0400	0,0067	0,0800

zcs	P0	P1	P10	P18	zcrs	P0	P1	P10	P18
P0	0,6200	0,1067	0,0000	0,0600	P0	0,4467	0,0667	0,0000	0,0200
P1	0,1400	0,5533	0,0067	0,1200	P1	0,2400	0,6267	0,0000	0,2933
P10	0,0000	0,0000	0,9800	0,0000	P10	0,0000	0,0000	0,9800	0,0000
P18	0,2333	0,3267	0,0067	0,7800	P18	0,3133	0,3067	0,0067	0,6667
unknown	0,0067	0,0133	0,0067	0,0400	unknown	0,0000	0,0000	0,0133	0,0200

zcrps	Circle	Heart	Square	Z
P0	0,7200	0,1667	0,0000	0,0467
P1	0,1000	0,5467	0,0000	0,3133
P10	0,0000	0,0000	0,9733	0,0000
P18	0,1733	0,2800	0,0200	0,6267
unknown	0,0067	0,0067	0,0067	0,0133

Tabelle 8.24: Personenbezogener Tests mit Geschwindigkeitsfeatures, Kreis-Geste, Variante 1

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Gummersbach, 15. August 2013

Renée Schulz