CARL VON OSSIETZKY universität OLDENBURG

Department für Informatik
Abteilung für Medieninformatik und Multimedia-Systeme

Individuelles Projekt

Beschleunigungsbasierte 3D-Gestenerkennung mit dem Wii-Controller

Benjamin Poppinga

29. August 2007

1. Gutachterin: Prof. Dr. Susanne Boll
2. Gutachter: Dr. Dietrich Boles

Danksagung

Ich danke Prof. Dr. Susanne Boll für die Vergabe, Betreuung und Begutachtung des individuellen Projekts. Auch bei Dr. Dietrich Boles möchte ich mich herzlich für die Begutachtung meiner Arbeit bedanken. Weiteren besonderen Dank widme ich Dipl.-Inform. Thomas Schlömer und Dipl.-Inform. Niels Henze für die sehr hilfreichen Hinweise und stundenlangen Diskussionen.

Erklärung

Ich erkläre, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Benjamin Poppinga
Matrikelnummer 8897680
Oldenburg, den 01. September 2007

Zusammenfassung

Die wesentliche Thematik des Dokuments befasst sich mit der Frage, ob und wie der Wiimote für Zwecke der Gestenerkennung genutzt werden kann. Zunächst werden die Grundlagen des Wii-Controllers erklärt und mögliche Kommunikationswege abgegrenzt. Anschließend wird in die Mustererkennung, insbesondere in Hidden-Markov-Modelle, eingeführt und zugehörige Algorithmen dargestellt. Zusätzlich wird der k-mean-Algorithmus sowie die Bayes-Klassifikation beschrieben. Anschließend erfolgt anhand einer Anforderungsanalyse eine Komponentenbestimmung in der Systemarchitektur. Als wesentliche Komponenten können eine Diskretisierungskomponente, diverse Filter, ein mathematisches Modell und ein Klassifizierer erkannt werden. Im Rahmen des Entwurfs werden diese wesentlichen Komponenten, teilweise anhand von empirischen Versuchen, mit möglichen Initialisierungen belegt. In der vorgestellten Implementierung wird eine Umsetzung der Komponenten in der Programmiersprache Java beschrieben. Schließlich werden die ermittelten Erkennungsraten dargestellt und dessen Eignung für Anwendungen beschrieben. Im Ausblick finden sich mögliche Anwendungsgebiete und weitere Ideen zur Weiterarbeit in dieser Fragestellung.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Gestenerkennung wurde bereits auf vielfältigste Art und Weise erforscht und in Softwareprodukte umgesetzt. Mausbasierte Gestenerkennung auf Browserebene (Opera, usw.), optische Gestenerkennung via Kamera, Datenhandschuh basierte Gestenerkennung, usw. sind hier als wesentliche Aspekte der Mensch/Maschine-Interaktion zu nennen. Bisher wurde die Erkennung oft auf einen zweidimensionalen Raum beschränkt und es wurde mit absoluten, erkennbaren Positionen des Menschen/Geräts gearbeitet.
In diesem individuellen Projekt soll nun die Möglichkeit betrachtet und analysiert werden, ob und wie es möglich ist mit dem WiiMote, einem Spielekonsolencontroller von Nintendo, auf Ebene von einem integriertem 3D-Beschleunigungssensor eine Gestenerkennung zu verwirklichen. In gewissen Rahmen gezeigt wurde diese Möglichkeit bereits von AiLive, ein Unternehmen, welches für die Gestenerkennung im Nintendo-Wii verantwortlich ist. Besondere Schwierigkeit ist hierbei die unbekannte Position des WiiMote, die damit dem Gestikulierendem die völlig neue Freiheit ermöglicht seine Gesten an beliebiger Stelle, mit nahezu beliebiger Ausgangsposition, auszuführen. Einzige Vorraussetzung ist die korrekte Haltung vom Controller in der Hand.
In der Arbeit werden die technischen Grundlagen erörtert und Verfahren zur Mustererkennung gesichtet. Anhand einer geeigneten Softwarearchitektur wird ein Entwurf konzeptioniert und implementiert. Schließlich erfolgt eine Auswertung der vorgestellten Methode und es wird ein Bezug zu weiteren denkbaren oder existenten Projekte hergestellt. Ziel der Arbeit ist das individuelle Anlernen und echtzeitnahe Erkennen von Gesten, die im dreidimensionalen Raum mit dem WiiMote unter zu ermittelnden Randbedingungen ausgeführt werden.

2 Grundlagen

In diesem Kapitel werden die Nintendo Wii-Konsole und zugehöriger Controller vorgestellt und im Hinblick auf die Ziele dieser Arbeit genauer betrachtet. Der Wii-Controller (auch Wiimote genannt) wird auf Hardwareebene analysiert und die über die standardisierte Bluetooth-Schnittstelle bereitgestellten Funktionen erklärt. Darauf aufbauend wird beschrieben, wie Beschleunigungswerte aus dem Controller ausgelesen und normiert werden. Des Weiteren wird in diesem Kapitel das Thema Mustererkennung behandelt. Es werden unterschiedliche Verfahren vorgestellt und verglichen. Anschließend werden Hidden-Markov-Modelle definiert und zugehörige Probleme bei der Anwendung diskutiert. Zur Lösung dieser Probleme werden drei Algorithmen eingeführt. Abschließend wird in dem Kapitel auf Vektorquantisierung eingegangen sowie die Bayes-Klassifikation vorgestellt.

2.1 Wii-Konsole

Zu den Konsolen der siebten Generation zählt neben Microsoft x-Box 360 und Sony Playstation 3 die von Nintendo[1] entwickelte Konsole „Wii“. Die Konsole lockt nicht, wie sonst auf dem Konsolenmarkt üblich, mit Leistung und hochauflösender Grafik, sondern durch ein außergewöhnliches Bedienkonzept. Damit soll ein größeres Publikum als bisher angesprochen werden. Es kommt ein schnurloser Controller mit Tastenkreuz und mehreren Knöpfen zum Einsatz. Jeder der bis zu vier Spieler steuert mit dem Wiimote über Bluetooth Aktionen in den Spielen. Dabei ist die Konsole nicht nur auf die Nutzung des Steuerkreuzes angewiesen, sondern ermöglicht den Spielern eine Interaktion über im Spiel definierte Gesten. In dem im Lieferumfang befindlichen Spiel „Wii Sports“ kann der Spieler beispielsweise Baseball oder Bowling durch Gesten spielen. Ursprünglich wurde diese Art der Spiele als Demonstration der Technik angesehen. Eine Veröffentlichung führte später dann zu einem „Game Developers Choice Award[2]“ für Innovation und Spieldesign. Im Folgenden wird der Controller und dessen Konzept genauer analysiert.

2.2 Der Wii-Controller

Der Controller ähnelt von der Optik (siehe Abbildung 2.1) einer Fernbedienung und trägt deswegen den von Nintendo erdachten Namen „Wiimote“ (engl. Remote = Fernbedienung). Für eine Fernbedienung typisch greift sich der Controller intuitiv richtig

[1] `http://www.nintendo.com/` letzter Zugriff: 28.07.07
[2] `http://www.gamechoiceawards.com/` letzter Zugriff: 28.07.07

Abbildung 2.1: Foto eines Wiimote [6]

und liegt ergonomisch in der Hand. Neben herkömmlichen Knöpfen und einem Steuerkreuz verfügt der Wiimote über eine Infrarot-Kamera, einen Lautsprecher, einen Beschleunigungssensor und einen kleinen Vibrationsmotor. Das Bedienkonzept basiert hauptsächlich auf der Infrarot-Kamera und dem Beschleunigungssensor. Der Mauszeiger auf der Wii wird ausschließlich mittels der integrierten Infrarot-Kamera gesteuert. Durch den Beschleunigungssensor überträgt der Spieler typische Handbewegungen in die Spiele. So lässt sich beispielsweise ein virtueller Tennisschläger im Spiel auf und ab schwingen. Diese Funktion beruht auf dem Konzept der Gestenerkennung. Die Gestenerkennung in der Konsole wurde im Auftrag von Nintendo von der Firma AiLive[3] entwickelt. In dieser Arbeit wird der Beschleunigungssensor konsolenunabhängig für eine Gestenerkennnung genutzt. Um die vom Wiimote bereitgestellten Eigenschaften zu verstehen befasst sich der Grundlagenteil auch mit den verbauten Komponenten und deren Funktionsweise.

2.2.1 Hardwareanalyse

Anhand der Prägung an der Controller-Unterseite und der FCC[4]-Nummer kann bestimmt werden, dass der Controller von Mitsumi[5] und Foxconn[6] mit der Bezeich-

[3] `http://www.ailive.com/` letzter Zugriff: 28.07.07
[4] `http://www.fcc.gov/` letzter Zugriff: 28.07.07
[5] `http://www.mitsumi.co.jp/` letzter Zugriff: 28.07.07
[6] `http://www.foxconn.com/` letzter Zugriff: 28.07.07

Kanal-Nr. (HEX)	Funktion
11	LEDs, haptisches Feedback
13	IR-Kamera aktivieren (1)
14	Lautsprecher aktivieren
16	Rohdaten in EEPROM schreiben
17	Rohdaten aus EEPROM lesen
18	Lautsprecherdaten
19	Lautsprecher stummschalten
1a	IR-Kamera aktivieren (2)

Tabelle 2.1: Empfangskanäle des Wiimote

nung RVL-003 in China gefertigt wird. Kernstück des Wiimote ist ein Broadcom[7] BCM2042 Mikrocontroller [2] mit integrierter Bluetooth-Funktionalität. Ursprünglich ausgelegt für Computermäuse und Tastaturen unterstützt der Chip Bluetooth 2.0 sowie das Human Interface Device (HID) Protokoll. Im Wiimote gespeicherte Daten werden in einem 128kbit EEPROM vom Typ ST[8] [4] 4128 BWP hinterlegt. Als Audio-Prozessor kommt ein H7824HE von Rohm[9] zum Einsatz. Besonders interessant ist der Beschleunigungssensor ADXL330 [1] von Analog Devices[10], welcher Beschleunigungswerte auf allen drei Achsen X, Y und Z im Interval zwischen typischerweise $\pm 3{,}6g$ (Erdbeschleunigung $g = 9{,}81\frac{m}{s^2}$) erkennt [3].

2.2.2 HID-Schnittstelle

Die HID-Schnittstelle ist eine standardisierte Klasse des USB-Standards [25]. Sie dient dazu Geräte zu betreiben, die direkt in Kontakt mit dem Menschen stehen. Häufig findet diese Technik in Tastaturen oder Mäusen Verwendung. Nintendo verwendet diese Schnittstelle im Wiimote. Im Wesentlichen handelt es sich um eine standardisierte, erweiterbare, selbstbeschreibende Schnittstelle. Dies erleichtert die Kommunikation mit dem Wiimote, da über die Wiimote-Selbstbeschreibung eine Liste mit sämtlichen Kommunikationskanälen ausgelesen werden kann. Weitere Details zur Selbstbeschreibung von HID-Geräten kann der Spezifikation [25] entnommen werden. Nicht für Nintendo arbeitenden Personen ist es durch Reverse-Engineering gelungen zwei noch unvollständige Tabellen über die Funktionen der einzelnen Ein- und Ausgabekanäle (Abbildungen 2.1 und 2.2) zu erstellen. Besonders wichtig ist an dieser Stelle der Hinweis, das eventuell erwähnte Steuerkommandos oder ähnliches allein von den aus Reverse-Engineering gezogenen Erkenntnissen stammen und somit in späteren Revisionen des Wiimote nicht mehr korrekt funktionieren könnten.

[7] `http://www.broadcom.com/` letzter Zugriff: 28.07.07
[8] `http://www.st.com/` letzter Zugriff: 28.07.07
[9] `http://www.rohm.com/` letzter Zugriff: 28.07.07
[10] `http://www.analog.com/` letzter Zugriff: 28.07.07

Kanal-Nr. (HEX)	Funktion
20	Erweiterungsport
30	Statuszustand der Knöpfe
31	Statuszustand von Beschleunigung und Knöpfen

Tabelle 2.2: Sendekanäle des Wiimote

2.2.3 Sende- und Empfangskanäle

Der Wiimote wird über verschiedene Kanäle angesprochen. Jeder Kanal wird als einzelne Schnittstelle zum Gerät mit gesonderter Aufgabe betrachtet. Die Zahlenangaben in diesem Abschnitt sind grundsätzlich im hexadezimalen Format. Über Kanal 0x11 können die vier Leuchtdioden im unteren Teil des Wiimote angesteuert werden, hierbei gibt es keine Beschränkung auf die Ansteuerung einer Diode, es können auch mehrere und sogar alle Dioden ansteuern. Auf Kanal 0x14 empfängt der Wiimote Signale zur Aktivierung des integrierten Lautsprechers. Schreib- und Lesezugriff auf den virtuell verwalteten Speicher wird auf Kanal 0x16 bzw. 0x17 ermöglicht. An Kanal werden die wiederzugebenen Audio-Daten gesendet, nachdem der Lautsprecher aktiviert worden ist. Via 0x19 kann der Lautsprecher lautlos geschaltet werden. Wichtigster Empfangskanal für die Gestenerkennung ist 0x31, welcher Beschleunigungsdaten und Zustandsinformationen über die Knöpfe des Wiimote liefert [6].

2.2.4 Beschleunigungsdaten auslesen

Dieser Abschnitt beschreibt die Datenpakete vom Beschleunigungssensor, welche der Wiimote nach vorheriger Aktivierung überträgt. Es handelt sich hierbei um Daten auf Kanal 0x31, welcher unter anderem für das Übertragen von Beschleunigungsdaten zuständig ist (vgl. Tabelle 2.2). Ermittelt wurde eine Sequenz von Bytes durch das Programm `hcidump`[11], welches unter GNU/Linux frei zur Verfügung steht und alle ein- und ausgehenden Daten an der Bluetooth-Schnittstelle protokolliert. Eine empfangene Byte-Sequenz könnte beispielsweise so aussehen:

(A1) 31 40 20 86 8A A5

Das von Klammern umschlossene erste Byte enthält Metadaten und wird in der HID-Spezifikation genauer beschrieben [25]. Das zweite Byte zeigt den verwendeten Kanal und damit indirekt den Zweck der nachfolgenden Bytes an. Das dritte und vierte Byte dienen zur Übermittlung der Buttonzustände mit einem noch nicht näher bekannten Einfluss durch Bytes nach dem Schema 0x20, 0x40, 0x60, welche auf die Zustandsbytes von den Buttons hinzuaddiert werden. Byte fünf bis sieben repräsentieren eine X, Y, Z Beschleunigung in Form von jeweils positiven binären Werten. Im obigen Beispiel befindet sich der Wiimote in Ruhe und liegt mit den Knöpfen nach oben.

[11] `http://www.bluez.org` letzter Zugriff: 28.07.07

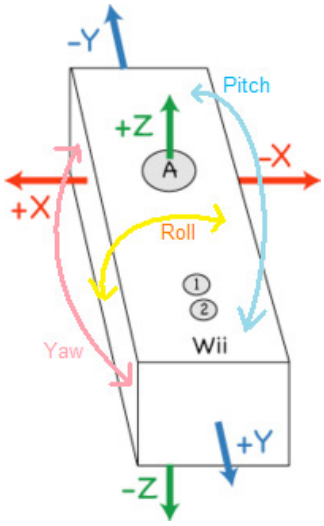


Abbildung 2.2: Wiimote Beschleunigungsachsen [6]

2.2.5 Normierung von Beschleunigungswerten

Die Rohdaten im binären Format sind für eine Beschleunigung nicht anschaulich und auch für spätere Rechenoperationen und Vergleiche nicht geeignet. Deswegen werden die Werte in diesem Abschnitt auf den sogenannten G-Faktor (auch: G-Kraft) normiert. Der G-Faktor beschreibt eine Kraft relativ zur immer und überall vorhandenen Erdanziehungskraft. Ein in Ruhe befindlicher Körper erfährt, wird der den Betrag über alle Dimensionen gebildet, eine Kraft a, welche bei jedem Körper in Ruhelage auf der Erde ca. $9{,}81\frac{m}{s^2}$ beträgt. Es handelt sich hierbei um einen G-Faktor von 1. Wird der Körper in eine beliebige Richtung beschleunigt nimmt der Betrag zu, damit steigt auch der G-Faktor. Folglich gibt der G-Faktor unabhängig von der Masse Auskunft über die erfahrene Beschleunigung. Zur Kalibrierung auf diesen Faktor wird der Wiimote in drei unterschiedliche Position bewegt und die Werte x, y und z als Elemente der ganzen Zahlen ermittelt [5].

1. Der Controller liegt eben auf einem Tisch mit den Knöpfen auf der Oberseite. Diese Position liefert (x_1, y_1, z_1). *Position A in Abbildung 2.3*

2. Der Controller steht auf dem Infrarot-Sensor und der Erweiterungsport befindet sich auf der Oberseite. Werte: (x_2, y_2, z_2) ab. *Position B in Abbildung 2.3*

3. Der Controller liegt auf der rechten Seite, so dass sich das CE-Prüfzeichen auf der Oberseite befindet. Werte: (x_3, y_3, z_3). *Position C in Abbildung 2.3*

Anhand dieser neun Messwerte können die Nullpunkte für die Beschleunigung auf jeder Achse berechnet werden. Dies geschieht wie folgt:

Abbildung 2.3: Positionen zur Kalibrierung des Wiimote

- $x_0 = \frac{x_1+x_2}{2}$
- $y_0 = \frac{y_1+y_3}{2}$
- $z_0 = \frac{z_1+z_3}{2}$

Diese Nullstellen sind nötig um auf den G-Faktor normierte Daten zu berechnen. Für einen beliebigen Beschleunigungsvektor $(x_{raw}, y_{raw}, z_{raw})$ werden folgende Berechnungen durchgeführt.

- $x = \frac{x_{raw}-x_0}{x_1-x_0}$
- $y = \frac{y_{raw}-y_0}{y_2-y_0}$
- $z = \frac{z_{raw}-z_0}{z_3-z_0}$

Der ausgegebene Vektor (x, y, z) beschreibt die Beschleunigung damit G-Faktor konform. Das bedeutet, dass wenn der Wiimote ruhig auf einer Tischoberfläche liegt, der Wert für die Z-Achse $+1{,}0g$ beträgt. Der Controller erfährt genau die 1-fache Erdbeschleunigung in eine Richtung. Die so ermittelten Werte ermöglichen damit ein leichteres Verständnis für die betrachteten Vektoren.

In Abbildung 2.4 ist zur Veranschaulichung eine über die Zeit gemessene Beschleunigung graphisch dargestellt und farblich codiert (X rot, Y grün, Z blau).

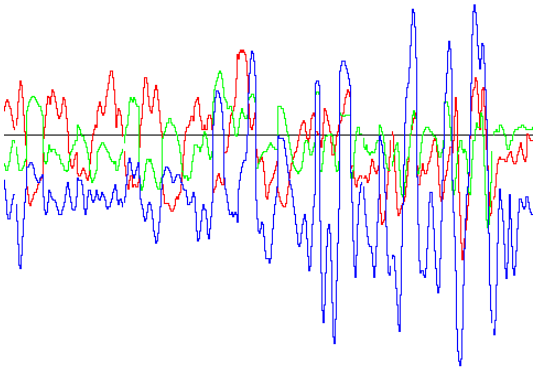


Abbildung 2.4: Beschleunigungsgraph vom Wiimote

2.3 Mustererkennung

Mustererkennung spielt insbesondere bei der Spracherkennung eine große Rolle. Für die Erkennung von biometrischen Daten oder Handschriften werden ebenfalls Technologien der Mustererkennung verwendet. Auch Gesten unterliegen einem Muster. In der Praxis bewährte Techniken stellen Dynamic Time Wrapping [13], künstliche neuronale Netze [11] oder Hidden-Markov-Modelle dar [13].

2.3.1 Techniken der Mustererkennung

Um im Vorfeld schon Methoden der Mustererkennung ausschließen zu können ist es hilfreich, die zu betrachtende Datenmenge genau zu beschreiben. Der Wiimote bewegt sich frei im Raum und die ausgeführten Gesten haben eine flexible zeitliche Länge. Es handelt sich somit um räumlich-zeitvariante Muster, die es zu erkennen gilt. Unbekannte Gesten sollen bei allen Ansätzen vernachlässigt werden.
Dynamic-Time-Wrapping ist eine Technik, die auf dynamischer Programmierung und einem Vergleich von Mustern mit Vorlagen basiert. Größter Nachteil der Methode ist eine hohe Anzahl von Vorlagen, die Gesten in unterschiedlichen Variationen enthalten [13]. Dies ist zwar grundsätzlich möglich, durch die hardwarebedingte Ungenauigkeit beim Wiimote jedoch nicht leicht zu realisieren. Deswegen und um die Flexibilität zum dynamischen Antrainieren von Gesten nicht zu verlieren ist diese Technik ungeeignet. Neuronale Netze eignen sich hauptsächlich für statische Mustererkennung. Sobald Dynamik, wie beispielsweise die Bewegung des Wiimote relevant wird, sind sie als ungeeignet einzustufen [13]. Die um eine Zeitbetrachtung erweiterten Time-Delay-Neural-Networks (deutsch etwa: zeitverzögertes neurales Netz) eignen sich prinzipiell für die Betrachtung der oben beschriebenen Daten. Ein großes Manko ist das feste Zeitfenster in denen sie mit Muster fester Länge auskommen.
Das selbe Problem tritt auch bei Hidden-Markov-Modellen auf, diese haben jedoch weniger Schwierigkeiten bzw. Einschränkungen bei der Verarbeitung von dynamischen Daten. Zudem erlauben sie es durch ihre Automatenstruktur, die Muster für

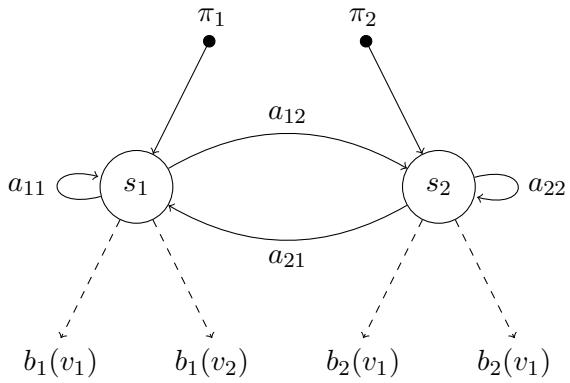


Abbildung 2.5: Schematische Darstellung eines Hidden-Markov-Modells

den Menschen verständlicher zu beschreiben als es neuronale Netze könnten. Besonders geeignet ist eine spezielle Art von Hidden-Markov-Modellen, Links-Rechts-Modell genannt. Dies stellt eine sehr gute Möglichkeit dar, räumlich und zeitlich flexible Informationen zu modellieren [19].

2.3.2 Hidden-Markov-Modelle

Ein Hidden-Markov-Modell [18] ist ein stochastisches Modell und setzt sich aus zwei separaten stochastischen Prozessen zusammen (vgl. Abbildung 2.5). Der versteckte (engl. hidden) Prozess ist eine Markov-Kette und besteht aus Zuständen und Übergangswahrscheinlichkeiten. Der Zustand, in dem sich dieser Prozess zum Zeitpunkt t befindet, wird mit q_t angegeben. Der äußere stochastische Prozess erzeugt zu jedem Zeitpunkt mit Wahrscheinlichkeiten belegte Ausgabesymbole. Diese Ausgabesymbole sind alles, was der Betrachter des Modells wahrnimmt. Das Erzeugen der Ausgangssymbole wird auch als „Emittieren“ bezeichnet. Formal wird ein Hidden-Markov-Modell als Fünftupel $\lambda = (S, \pi, V, A, B)$ definiert [18]. Da es sich bei S und V um interne Modellparameter handelt, findet sich in der Literatur auch gelegentlich die Kurzschreibweise $\lambda = (A, B, \pi)$.

Im Folgenden werden die einzelnen Elemente definiert:

1. Eine Zustandsmenge S mit N Zuständen

$$S = \{s_1, s_2, \ldots, s_N\}$$

2. Eine Menge von Wahrscheinlichkeiten $\pi = \{\pi_i\}$, dass Zustand i Startzustand ist.

$$\pi_i = P\left(q_1 = s_i\right), 1 \leq i \leq N$$

3. Eine Menge V von M Beobachtungssymbolen alias diskretes Beobachtungs-Alphabet

$$V = \{v_1, v_2, \ldots, v_M\}$$

4. Eine Menge von Wahrscheinlichkeiten $A = \{a_{ij}\}$, wobei a_{ij} die Wahrscheinlichkeit angibt, dass ein Übergang von Zustand s_i nach s_j geschieht.

$$a_{ij} = P\left(q_{t+1} = s_j | q_t = s_i\right), 1 \leq i, j \leq N$$

5. Eine Menge von Emissionswahrscheinlichkeiten $B = \{b_j\left(k\right)\}$, die angibt wie wahrscheinlich das Beobachten von Symbol v_k in Zustand s_j ist.

$$b_j = P\left(O_t = v_k | q_t = s_j\right), 1 \leq j \leq N \text{ und } 1 \leq k \leq M$$

O_t ist hierbei die Beobachtung zum Zeitpunkt t. $O_t \in V$.

Desweiteren gelten folgende stochastische Beschränkungen:

1. Die Summe der Wahrscheinlichkeiten um von Zustand i zu allen anderen Zuständen j zu kommen ist 1.

$$\forall i : \sum_{j=1}^{N} a_{ij} = 1, \quad a_{ij} \geq 0$$

2. Die Summe der Emissionswahrscheinlichkeiten von allen Symbolen k in Zustand j ist 1.

$$\forall j : \sum_{k=1}^{M} b_j\left(k\right) = 1, \quad b_j\left(k\right) \geq 0$$

3. Die Summe der Startwahrscheinlichkeiten ist 1.

$$\sum_{i=1}^{N} \pi_i = 1, \quad \pi_i \geq 0$$

2.3.3 Arten von Hidden-Markov-Modellen

Durch Einschränkung der Modellparameter lassen sich bestimmte Arten von Hidden-Markov-Modellen erzeugen. Ein Modell ohne Einschränkungen heißt „Ergodisches Hidden-Markov-Modell“. In dem ergodischen Modell ist jeder Zustand in einem Schritt erreichbar. Formal wird diese Aussage durch folgende Gleichung definiert:

$$a_{ij} > 0, \quad 0 \leq i, j \leq N.$$

In der Spracherkennung populär ist das „Links-Rechts Hidden-Markov-Modell“ [21]. Für das Links-Rechts-Modell ist nicht jeder Zustand in endlich vielen Schritten erreichbar. Vielmehr nimmt der Zustandsindex zu oder bleibt gleich. Dabei wird die maximale Zunahme vom Zustandsindex nach einer Transition als Sprungweite bezeichnet. Ein Modell mit einer Sprungweite von eins kann folglich mit jeder Transition in dem Zustand i bleiben oder in Zustand $i+1$ wechseln. In Links-Rechts-Modellen hat der Zustand s_0 grundsätzlich die maximale Wahrscheinlichkeit, sonst würde das Modell Zustände besitzen, die nie besucht werden könnten. Der Endzustand s_N darf nach einmaligem Besuchen nicht mehr verlassen werden.

2.3.4 Probleme von Markov-Modellen in der Mustererkennung

Das mathematisch definierte Modell sagt noch nichts über die Verwendungsmöglichkeiten und den damit zusammenhängenden Problemen aus. Wird das Modell zur Mustererkennung verwendet stellen sich drei Probleme:

1. Zu einer Beobachtungssequenz $O = O_1, O_2, \ldots, O_n$ muss die Wahrscheinlichkeit $P(O|\lambda)$ der Zustandsfolge im Hidden-Markov-Modell $\lambda = (A, B, \pi)$ ermittelt werden können.

2. Bei einem gegebenen Modell $\lambda = (A, B, \pi)$ und einer gegebenen Folge von Beobachtungen $O = O_1, O_2, \ldots, O_n$ muss diejenige Zustandsfolge ermittelt werden können, die die Emissions-Wahrscheinlichkeit maximiert.

3. Ein Modell λ muss so verändert werden können, dass $P(O|\lambda)$ maximal wird.

Hinsichtlich einer Mustererkennung beschreibt das letzte Problem das Trainieren des Modells, während sich das erste Problem mit der eigentlichen Mustererkennung befasst. Problem Nummer 2 ist für die Gestenerkennung mit den Markov-Modellen nicht relevant, weil auf diesem Gebiet nur das Training und die Erkennung von Gesten relevant ist. Die Zustandsfolge für eine Folge von Beobachtungen ist für folgende Algorithmen nicht notwendig. In den nächsten Abschnitten werden Vorschläge und Algorithmen zur Problemlösung beschrieben.

2.3.5 Vorwärts-Algorithmus

Der Vorwärts-Algorithmus (auch Forward-Algorithmus, Vorwärts-Prozedur) wie in [21] beschrieben ermöglicht eine effiziente Lösung des ersten Problems. Für eine beliebige Beobachtung $O = O_1, O_2 \ldots, O_n$ soll die Wahrscheinlichkeit ermittelt werden, mit der diese Beobachtung von einem Hidden-Markov-Modell λ emittiert worden ist. Hierzu kann die Wahrscheinlichkeit aller möglichen Zustandsfolgen betrachtet werden. Dies würde exponentiellen Aufwand fordern: $\mathcal{O}\left(N^T\right)$ mit N = Anzahl der Zustände und T = Anzahl der diskreten Zeitpunkte. Der Vorwärts-Algorithmus ermöglicht die Berechnung unter Zuhilfenahme einer sogenannten Vorwärtsvariablen $v_t(i)$ in $\mathcal{O}(N^2 \cdot T)$.

$$v_t(i) = P(O_1, O_2, \ldots, O_t, q_t = s_i|\lambda)$$

Initialisiert wird die Vorwärts-Prozedur dabei mit

$$v_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N.$$

Für weitere Zeitpunkte t und Zustände j wird die Hilfsvariable induktiv weiterberechnet.

$$v_{t+1}(j) = \left(\sum_{i=1}^{N} v_t(i)\, a_{ij}\right) b_j(O_{t+1}), \quad 1 \leq t \leq T-1 \text{ und } 1 \leq j \leq N$$

Wird die Induktion über die ganze Länge der Beobachtung geführt kann letztlich die Summe über die Wahrscheinlichkeiten gebildet und damit die Gesamtwahrscheinlichkeit $P(O|\lambda)$ errechnet werden.

$$P\left(O|\lambda\right) = \sum_{i=1}^{N} v_T\left(i\right)$$

Der in Abschnitt 2.3.7 vorgestellte Baum-Welch-Algorithmus zum Training des Modells verwendet ebenfalls den Vorwärts-Algorithmus, benötigt zusätzlich aber auch den Rückwärts-Algorithmus.

2.3.6 Rückwärts-Algorithmus

Der Algorithmus an sich ist nicht hilfreich für die Bewältigung eines der in Abschnitt 2.3.4 beschriebenen Probleme. Trotzdem ist der Algorithmus für die Lösung der Probleme relevant, da er im Baum-Welch-Algorithmus verwendet wird. Der Vorwärts-Algorithmus berechnet die Wahrscheinlichkeit eine Beobachtungssequenz $O_1, O_2, \ldots, O_t$ zu betrachten und dabei in Zustand s_i zu enden. Dagegen berechnet der Rückwärts-Algorithmus die Wahrscheinlichkeit in Zustand s_i zu starten und dann die Sequenz $O_{t+1}, O_{t+2}, \ldots, O_t$ zu beobachten:

$$r_t\left(i\right) = P\left(O_{t+1}, O_{t+2}, \ldots, O_T|q_t = s_i, \lambda\right).$$

Der Rückwärts-Algorithmus wird definiert durch die Initialwerte

$$r_i\left(T\right) = 1, \quad 1 \leq i \leq N$$

und weitergeführt durch

$$r_i\left(t\right) = \sum_{j=1}^{N} a_{ij} b_j\left(O_{t+1}\right) r_j\left(t+1\right), \quad 1 \leq t \leq T-1 \text{ und } 1 \leq i \leq N.$$

Somit ergibt sich die Wahrscheinlichkeit aus folgender Summe

$$P\left(O|\lambda\right) = \sum_{i=1}^{N} r_i\left(1\right) \pi_i b_i\left(O_1\right).$$

2.3.7 Baum-Welch-Algorithmus

Der Baum-Welch-Algorithmus [7], [8] ist in der Lage die Wahrscheinlichkeit $P\left(O|\lambda\right)$ einer Beobachtungsfolge O auf dem Modell $\lambda = (A, B, \pi)$ zu maximieren. Dabei werden iterativ die Modellparameter verfeinert (trainiert), bis ein lokales Maximum der Wahrscheinlichkeit erreicht wird. Der Algorithmus zählt dabei die Häufigkeiten der aufgetretenen Beobachtungen und errechnet daraus eine Wahrscheinlichkeitsverteilung. Mit diesen Werten werden in jeder Iteration die Werte für π, A und B neu

berechnet. Um den Algorithmus zu beschreiben werden zwei Hilfsvariablen eingeführt. Die Funktion $\xi_t(i,j)$ beschreibt die Wahrscheinlichkeit sich innerhalb eines Modells λ bei einer Beobachtungsfolge O zum Zeitpunkt t im Zustand s_i und zum Zeitpunkt $t+1$ im Zustand s_j zu befinden.

$$\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j | \lambda, O) = \frac{v_t(i)\, a_{ij} b_j(O_{t+1})\, r_{t+1}(j)}{\sum_{i=1}^{N}\sum_{j=1}^{N} v_t(i)\, a_{ij} b_j(O_{t+1})\, r_{t+1}(j)}$$

Mit dieser Funktion lässt sich eine weitere Hilfsfunktion $\gamma_t(i)$ definieren, die die Wahrscheinlichkeit angibt sich zum Zeitpunkt t im Zustand s_i zu befinden.

$$\gamma_t(i) = \sum_{j=1}^{N} \xi_t(i,j)$$

Nach J. Bilmes [10] beschreibt sich das neue Modell $\tilde{\lambda} = \left(\tilde{A}, \tilde{B}, \tilde{\pi}\right)$ wie folgt:

$$\tilde{\pi} = \gamma_1(i)$$

$$\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)}$$

$$\tilde{b}_j(k) = \frac{\sum_{t=1, O_t=k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \quad \text{mit } k \in V.$$

Diese Neuzuweisung von Werten geschieht beliebig oft oder so lange, bis ein lokales Maximum gefunden worden ist. Da es keine Anhaltspunkte für die zu erwartenden Modellparameter gibt, wird zur Maxima-Erkennung eine Änderungsrate benutzt (vgl. Abbildung 2.6). Liegt der neue Modellparameter unterhalb dieser Änderungsrate terminiert der Algorithmus und es wurde ein neues lokales Maximum gefunden. Alternativ zu dieser Methode kann der Algorithmus auch nach einer vorher bestimmten Anzahl von Schritten terminiert werden. Dies führt jedoch nicht zwangsläufig zu einer Optimierung der Wahrscheinlichkeit.

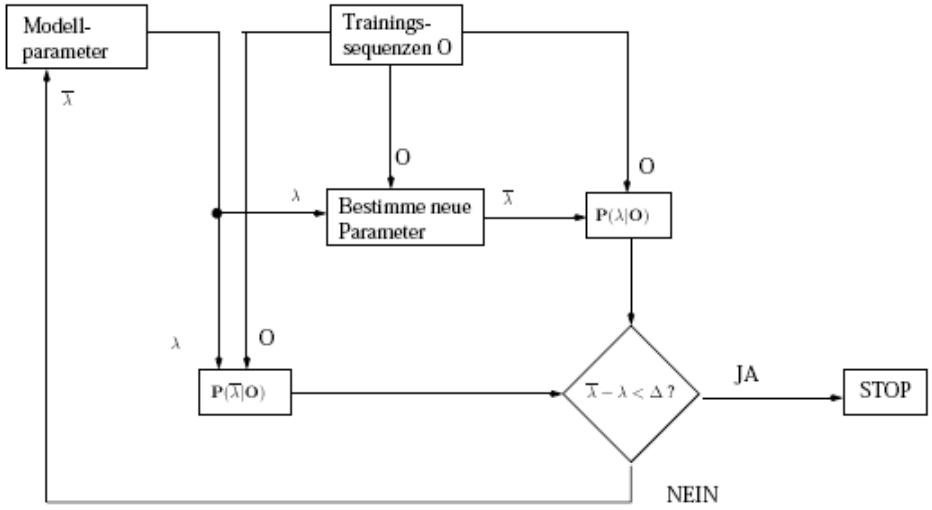


Abbildung 2.6: Training mit dem Baum-Welch-Algorithmus [16]

2.4 Vektorquantisierung

Die vorgestellten Algorithmen verwenden die beschriebenen Beobachtungsfolgen $O = O_1, O_2, \ldots, O_n$ um Wahrscheinlichkeiten zu ermitteln oder Modellparameter zu optimieren. Der Wiimote liefert Beschleunigungswerte in drei Dimensionen. Diese Werte weichen bedingt durch Sensorungenauigkeit bei völlig identisch ausgeführten Bewegungen voneinander ab. Ungenaue Daten sind für obige Algorithmen nicht gut geeignet, da die Modelloptimierung und Gestenerkennung unter dynamischen Werten spürbare Qualitätseinbußen zeigt. Der Abschnitt beschreibt Techniken zur Gruppierung von Daten, in der Literatur auch als Quantisierung oder Clusteranalyse bekannt. Ähnliche Vektoren, die vom Wiimote gesendet werden, können unter Berücksichtigung von definierten Ähnlichkeiten zusammengefasst und abstrahiert werden. Diese Art von Verfahren beseitigt die angesprochenden Probleme der variierenden Werte und zieht letztlich eine Optimierung der Modellergebnisse nach sich.

2.4.1 k-mean-Algorithmus

Bei den Clustering-Technologien ist der k-mean-Algorithmus [17] eine sehr bekannte Methode. Das begründet sich nicht nur dadurch, dass er einfach zu implementieren ist. Vielmehr erlaubt er bei geringer Berechnungskomplexität ein performantes Ergebnis. In Hinblick auf spätere Echtzeitanwendung ist dies ein entscheidener Vorteil. Zudem kommt die Tatsache, dass es wohl keinen anderen unüberwachten Clusteralgorithmus gibt, der so intensiv studiert und erweitert worden ist [24]. Vorerst wird der Algorithmus allgemein beschrieben, anschließend folgt ein Beispiel für Werte in zwei Dimensionen.

Angenommen es existiert eine Menge von Vektoren und für jedes Element soll eine Gruppenzugehörigkeit bestimmt werden. Das k in k-mean steht für die Anzahl der Gruppen (auch: Zentren, Centeroids, Cluster) in die diese Vektoren zugewiesen werden sollen. Die Gruppenzentren werden bei Initialisierung vom Algorithmus über die Menge der Vektoren verteilt. Eine völlig zufallsbasierte Erstellung der k Zentren ist auch möglich, erhöht aber die Komplexität. Eine optimale Anordnung wird bei größtmöglicher Distanz zwischen den initialen Gruppenzentren erreicht. Nach der Erstellung wird jeder Vektor aus dem Datensatz einem Zentrum zugewiesen. Die Zuweisung erfolgt über die euklid'sche Distanz. Dem Zentrum, welches die geringste Distanz zu einem Element aus der Menge der Vektoren hat, wird das Element zugeordnet. Nachdem alle Elemente ihrer optimalen Gruppe zugeordnet worden sind, werden die Zentren neu berechnet. Neue Werte berechnen sich aus dem Schwerpunkt der den Gruppen zugeordneten Elementen. Mit den neuen Zentren wird anschließend das Verfahren wiederholt bis sich keine Veränderung der Gruppenzentren mehr einstellt.

2.4.2 k-mean-Algorithmus Beispiel

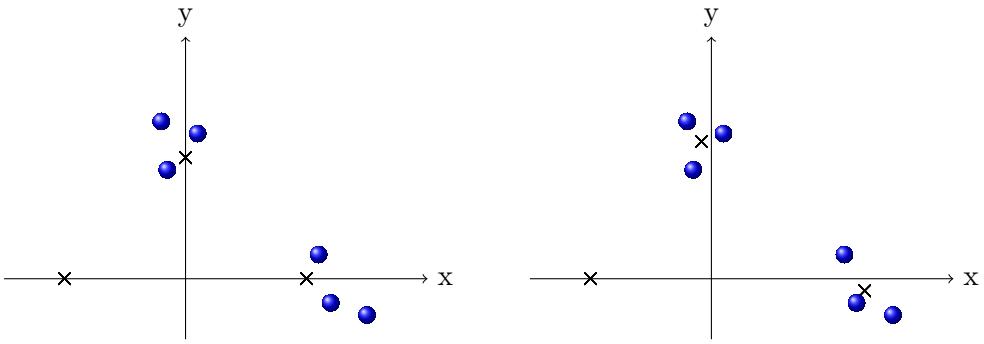
In Abbildung 2.7 ist $k = 3$ gewählt worden und die Gruppenzentren jeweils durch ein Kreuz gekennzeichnet. Die Anordnung der Zentren ist so gewählt, dass die euklid'sche Distanz zwischen den Koordinaten möglichst groß ist. Die initialen Positionen der Gruppenzentren sind im Optimalfall durch vorher ausgeführte Tests bekannt und zusätzlich durch einen möglichen maximalen Wertebereich beschränkt. Beim Wiimote beispielsweise ist diese Grenze $\pm 3{,}6g$. Die zuzuordnenden Vektoren sind in der Abbildung als Kreise dargestellt. Der Abstand zu den initialen Gruppenzentren ist ziemlich gering. Ein Vorteil, der durch gute Vorarbeit bei der Wahl der Gruppenzentren erreicht werden kann. Der entstandene Rechenaufwand ist minimiert worden. Nach der Berechnung haben sich zwei Gruppenzentren etwas verändert. Wird über die zwei Gruppen mit je drei Vektoren ein Dreieck aufgespannt, ist das neue Zentrum gerade der Schwerpunkt.

2.5 Bayes Klassifikation

Gegeben eine beliebige Sequenz $v = v_1, v_2, \ldots, v_j$ mit $v_i \in V, 1 \leq i \leq j$. Für M Hidden-Markov-Modelle $\lambda_1, \lambda_2, \ldots, \lambda_M$ und einer zu klassifizierenden Sequenz v gibt es M Wahrscheinlichkeiten: $P(\lambda_i|v), 1 \leq i \leq M$. Jede dieser Wahrscheinlichkeiten berechnet sich nach dem Satz von Bayes [9], wie beispielsweise in [20] beschrieben, wie folgt:

$$P(\lambda_i|v) = \frac{P(\lambda_i)\, P(v|\lambda_i)}{P(v)} = \frac{P(\lambda_i)\, P(v|\lambda_i)}{\sum_{k=1}^{M} P(\lambda_k)\, P(v|\lambda_k)}$$

Hierbei ist $P(\lambda_i)$ die Modellwahrscheinlichkeit, mit der ein Modell λ eine der Trainingssequenzen erkennt. $P(v|\lambda)$ ist die Modellwahrscheinlichkeit für die Sequenz v.



(a) Vor der Ausführung (b) Nach der Ausführung

Abbildung 2.7: Anwendungsbeispiel k-mean-Algorithmus

Nachdem für alle Modelle λ die Wahrscheinlichkeiten bestimmt worden sind, wird die größte von allen ausgewählt. Zur maximalen Wahrscheinlichkeit gehörendes Modell repräsentiert die klassifizierte Geste. Ein Vorteil der Klassifikation nach Bayes ist, dass die Werte von $P(\lambda_i|v)$ im Intervall $[0, 1]$ liegen. Damit lassen sich anschaulich Wahrscheinlichkeiten in Prozent angeben.

3 Anforderungsanalyse und Grobentwurf

Dieses Kapitel stellt zunächst funktionale und nicht-funktionale Anforderungen heraus um daraus für das System notwendige Komponenten zu ermitteln. Die ermittelten Komponenten werden kurz vorgestellt und anschließend deren Initialwerte erfasst. Es wird ein abstrakter Automat entwickelt, der das Nutzervorgehen beschreibt und beim Verständnis der Arbeitsweise der Komponenten unterstützt. Der Automat wird in der Implementierung erneut aufgegriffen und ermöglicht so ein leichtes Übertragen der Anforderungen auf die praktische Anwendung.

3.1 Funktionale Anforderungen

Das Programm soll in der Lage sein mit einem Eingabegerät, z.B. Wiimote, beschleunigungsbasierte Gesten zu erkennen. Die zu erkennenden Gesten müssen dem System bereits bekannt sein oder können diesem individuell beigebracht werden. Daraus resultiert die zweite funktionale Anforderung des Trainings von Gesten. Im Folgenden werden beide Funktionen genauer beschrieben und die benötigten Komponenten ermittelt [23].

Aufnehmen der Beschleunigungsvektoren

Der Wiimote, oder ein vergleichbares Gerät, stellt Beschleunigungsvektoren zur Verfügung. Diese müssen über eine Schnittstelle zum Gerät abgegriffen werden. Auch eine Normierung der Daten, wie in Abschnitt 2.2.5 vorgestellt, ist je nach Format der Vektoren notwendig. Die vorverarbeiteten Vektoren werden anschließend über eine zu definierende Schnittstelle weiteren Komponenten zur Verfügung gestellt.

Erkennen von Gesten

Die vorverarbeiteten Vektoren werden analysiert und einer Geste zugeordnet. Neben bekannten Gesten müssen auch unbekannte Gesten berücksichtigt werden. Ergebnis von dieser Prozedur könnte z.B. eine Wahrscheinlichkeit sein.

Klassifizieren von Gesten

Anhand der Werte, die durch die Gestenerkennung zur Verfügung gestellt werden, wird eine Entscheidung abgegeben ob und um welche Geste es sich handelt. Die Geste wird klassifiziert, beispielsweise in „erkannt“ oder „nicht erkannt“. Das Ergebniss der Klassifikation wird dem Nutzer zugänglich gemacht oder für weitere Funktionen verwendet.

Training von Gesten

Die Komponente der Gestenerkennung soll zur Ausführungszeit anpassungsfähig sein. Damit kann ein Nutzer dem System eine neue Geste antrainieren. Die zur Geste gehörenden Vektoren werden entsprechend integriert und vollständig im Ablauf der Erkennung und Klassifizierung berücksichtigt.

3.2 Nicht-Funktionale Anforderungen

Neben den funktionalen Anforderungen sollen auch nicht-funktionale Anforderungen erfüllt werden. Das Ergebnis der Gestenerkennung sollte bereits nach möglichst kurzer Zeit verfügbar sein. Auch eine Erweiterung auf mehrere Wiimotes sollte dem System keine Schwierigkeiten bereiten. Zusätzlich werden gehobene Anforderungen an die Zuverlässigkeit gestellt.

Echtzeitfähigkeit

Der Zeitraum zwischen Ende der Ausführung einer Geste und Ergebnis ist für die Anwendungstauglichkeit in Multimediaumgebungen entscheidend. Ein Nutzer möchte nach Ausführung einer Geste nicht erst mehrere Minuten warten bis das System reagiert und die damit assoziierte Funktion ausführt. Das Ergebnis einer Geste sollte daher schon innerhalb der Latenzzeit verfügbar sein. Dazu ist es notwendig das Datenaufkommen zu verringern und vereinfachte Gesten für das Training und die Erkennung zu verwenden. Für eine sehr gute Reaktionszeit wird ein mehrstufiges Filterkonzept unumgänglich sein.

Latenzzeit

Im Rahmen der Echtzeit wird eine maximale Latenzzeit definiert. Eine durch Laufzeiten der Algorithmen und Übertragung der Signale entstehende Zeit sollte maximal wenige Millisekunden betragen. Als nominaler Richtwert ist die Ausführungsdauer der Geste zu verwenden. Der Nutzer erwartet also nach der Ausführung innerhalb der Ausführungszeit der Geste ein Ergebnis.

Erweiterbarkeit

Denkbar ist, dass ein Nutzer mehrere Wiimotes in seiner Anwendungsumgebung verwenden möchte oder aber mehrere Nutzer jeweils einen Wiimote nutzen wollen. Weitere Wiimotes sollten sich daher problemlos in das System integrieren lassen und die selbe Funktionalität bereitstellen. Die ausgeführten Gesten sollten eindeutig einem Wiimote zugewiesen werden können. Auch ein visuelles Feedback an die Nutzer sollte mehrere Wiimotes berücksichtigen können.

Portabilität

Das System soll auf den großen Plattformen (Windows, MacOS X, Linux) lauffähig sein. Außerdem soll die Hardware für das Liefern der Beschleunigungswerte ausgetauscht werden können. Dabei müssen evtl. auch andere Normierungsschritte berücksichtigt werden.

Erkennungsrate

Die Erkennungsrate beeinflusst maßgeblich die Zuverlässigkeit des Gesamtsystems und damit auch die Akzeptanz beim Nutzer. Bei definiertem Trainingsaufwand sollte die Erkennungsrate im Mittel für jede Geste 80% nicht unterschreiten um dem Nutzer erkennbare Funktionalität zu bieten. Unter Verletzung von Randbedingungen für die Gesten darf sich die Erkennungsrate beliebig verschlechtern.

Wiederholgenauigkeit

Wird exakt die selbe Geste mehrmals hintereinander ausgeführt darf vom System wiederholend mit der selben Klassifikation gerechnet werden. Dem Nutzer wird dadurch der Umgang mit dem System erleichtert. Zusätzlich wird eine bessere Prüfbarkeit der Ergebnisse ermöglicht.

3.3 Nutzerintention erkennen

Die Handlungen des Nutzers lassen sich in einem Automaten veranschaulichen. Der in Abbildung 3.1 dargestellte Automat besteht aus drei Zuständen: `Ruhelage`, `Erkennung` und `Training`. Manche Zustandsübergänge sind mit dem Label [Aktion] versehen. Eine Aktion ist ein Nutzerkommando, welches über den Wiimote eingegeben wird. Zu Aktionen zählt beispielsweise das Drücken eines Knopfes oder das Verlassen oder Betreten der Ruhelage. Das Label [Bewegung] beschreibt, dass der Wiimote sich in Bewegung befindet und damit über der Dauer des Aufenthalts im jeweiligem Zustand eine Geste ausgeführt wird. Erst durch eine erneute Aktion können diese Zustände wieder verlassen werden.

3.3.1 Ruhelage

Der Betrag der Beschleunigung ist in der Ruhelage 1,0g. Die Bedeutung des g-Faktors unterstützt diese Aussage, der Wiimote erfährt das 1,0-fache seines Gewichts. Ist der Wiimote dagegen in Bewegung erfährt er grundsätzlich eine höhere Beschleunigung. Diese wichtige Eigenschaft dient als Grundlage zum Setzen von Ausgangs- und Endpunkt der Gesten. In dem Automaten stellt der Zustand `Pause` den Initialzustand dar. Für einen Beschleunigungsvektor $\vec{a}$ berechnet sich der Betrag nach euklid'scher Norm wie folgt:

$$|\vec{a}| = \sqrt{a_x^2 + a_y^2 + a_z^2}.$$

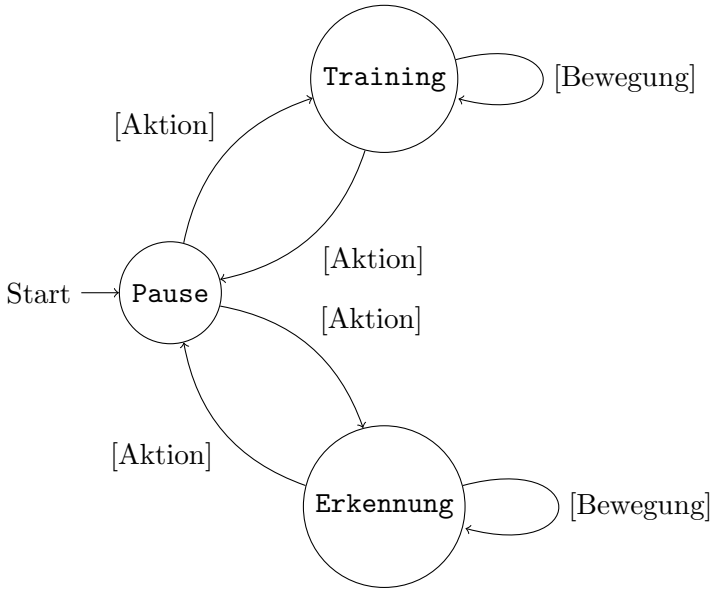


Abbildung 3.1: Veranschaulichung Erkennungsablauf

3.3.2 Training

Möchte der Nutzer eine neue Geste hinzufügen wechselt der Automat auf Anweisung des Nutzers in den Zustand `Training`. Die Anweisung des Nutzers kann dabei implizit, durch schlichtes Bewegen des Wiimote, oder durch Drücken eines Knopfes auf dem Wiimote initiiert werden. Es wird je nach Nutzerwunsch eine neue Gesten-Repräsentation erzeugt oder eine bestehende mit einer weiteren Instanz von Beobachtungen erweitert. Nach abgeschlossener Geste und fertigem Training wechselt der Automat durch eine [Aktion] wieder in den Zustand `Pause`.

3.3.3 Gestenerkennung

Der häufigste Anwendungsfall ist die Erkennung einer Geste. Der Automat wird wieder entweder auf implizite oder explizite Anweisung in den Zustand `Erkennung` geschaltet. Nach Erkennung einer [Aktion] zum Beenden der Erkennung wechselt der Automat auch hier wieder in `Pause`. Durch implizites Aktivieren der Gestenerkennung können unbeabsichtigt Gesten durchgeführt werden. Um dies zu verhindern ist es notwendig, das Wechseln in den Zustand `Erkennung` durch einen Knopfdruck zu aktivieren. Dadurch wird sichergestellt, dass nur Gesten erkannt werden, wenn der Nutzer auch wirklich eine Geste ausgeführt hat.

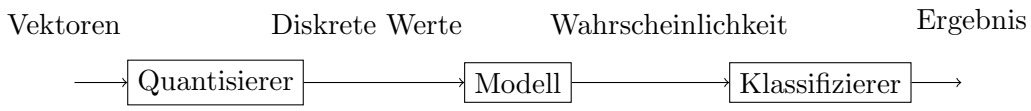


Abbildung 3.2: Identifizierte Komponenten zur Gestenerkennung

3.4 Identifizierte Komponenten

Aus den funktionalen und nicht-funktionalen Anforderungen lassen sich die wichtigsten Komponenten des Systems herleiten. Eine grobe Übersicht über die identifizierten Komponenten findet sich in Abbildung 3.2. In der Abbildung werden ausschließlich die erkannten Hauptkomponenten aufgeführt, die vereinfachenden Filter werden später separat vorgestellt. Jede dieser Komponenten wird in diesem Abschnitt genauer betrachtet und es wird eine für das Gesamtsystem sinnvolle Initialisierung bestimmt.

3.4.1 Quantisierungskomponente

Die Quantisierungskomponente dient der Vektorquantisierung. Eingaben sind Beschleunigungsvektoren, die in der Ausgabe durch diskrete Werte repräsentiert werden. Für diese Komponente wird der vorgestellte k-mean-Algorithmus verwendet. Während des Trainings werden die initialen Parameter des Algorithmus entsprechend an die Geste angepasst um in der späteren Erkennung optimale Ergebnisse liefern zu können. Wichtige Aspekte zur Initialisierung der Komponente sind die Anzahl der Gruppen, und die initialen Gruppenzentren.

3.4.2 Modell

In dieser Arbeit kommt als Modell ein Hidden-Markov-Modell zum Einsatz. Das Hidden-Markov-Modell bekommt als Eingabe eine Sequenz von diskreten Symbolen und liefert eine Modellwahrscheinlichkeit. Bei den unterschiedlichen Arten von Hidden-Markov-Modellen muss ermittelt werden, welches die beste Lösung für unser Problem darstellt. Außerdem besitzt ein Modell Parameter, die initialisiert werden müssen: Startwahrscheinlichkeit, Übergangswahrscheinlichkeit, Emissionswahrscheinlichkeit, Anzahl der Zustände und Anzahl der diskreten Symbole im Beobachtungsalphabet. Üblicherweise ist eine konkret mathematische Initialisierung nicht möglich, so dass auf empirische Ergebnisse zurückgegriffen werden muss.

3.4.3 Klassifizierer

Das Endstück der Komponentenkette ist der Klassifizierer. Dieser bedient sich an den Modellwahrscheinlichkeiten einzelner Hidden-Markov-Modelle und ermittelt daraus als Ergebnis die Geste, welche erkannt worden ist. Wird keine Geste erkannt liefert er eine Wahrscheinlichkeit gegen Null für alle Gesten. Diese Komponente wird

durch einen Bayes-Klassifizierer initialisiert. Der Klassifizierer muss nicht initialisiert werden.

3.4.4 Filter

Durch die nicht-funktionalen Anforderungen wird die Echtzeitfähigkeit eingeführt. Die Komplexität der Algorithmen hängt maßgeblich von der Länge der Geste ab. Die Aufgabe eines Filters ist es, aus einer Menge von n Eingabesignalen eine Menge von m Ausgabesignalen zu erzeugen, wobei $n \geq m$ gilt und die Differenz zwischen n und m möglichst groß sein soll. Dabei wird eine Geste durch weniger Elemente repräsentiert als vorher. Ein entstehender Informationsverlust soll minimal gehalten werden. Auch das mehrfache Filtern an verschiedenen Stellen im Ablauf ist möglich. Filter werden nicht initialisiert.

4 Entwurf

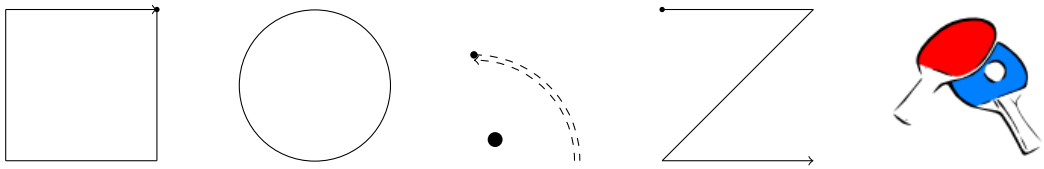
Nachdem die Anforderungen erarbeitet und zugehörige Komponenten identifiziert wurden ist es Inhalt dieses Kapitels, die problembezogene Initialisierung der Komponenten zu beschreiben. Um optimale Werte für die Komponenten zu ermitteln ist es teilweise nötig empirische Versuche zu betreiben und Werte aus bekannten Problemen auf dieses Projekt zu übertragen. Zu diesem Zweck werden in diesem Kapitel Referenzgesten eingeführt. Nach dem Initialisieren der Diskretisierungskomponente werden die verwendeten Filter detailliert beschrieben und definiert. Anschließend werden die Modellparameter entwickelt und unter Berücksichtigung der stochastischen Randbedingungen festgesetzt. Abschließend wird der Klassifizierer nach Bayes in seiner Funktionsweise beschrieben.

4.1 Rahmenbedingungen für Gesten

Für die Parameterbestimmung der Komponenten sind empirische Versuche notwendig, hierfür werden Rahmenbedingungen für Gesten und beispielhafte Referenzgesten eingeführt. Gesten können grundsätzlich völlig frei gewählt werden. Trotzdem gibt es einige Hinweise, die eine bessere Erkennung garantieren, und daher bei der Auswahl der Gesten zu beachten sind. Eine Geste sollte den Sinn einer Geste nicht verfehlen – Einfachheit und Unmissverständlichkeit. Daher sollte eine gewählte Geste nicht übermäßig komplex ausfallen. Außerdem wird die Erkennung bei voneinander verschiedenen Gesten positiv unterstützt. Dies schließt ein, dass Gesten nicht ineinander enthalten sein dürfen, weil damit nicht mehr eindeutig ist um welche Geste es sich handelt. In [19] wird beschrieben, dass Gesten bevorzugt selbst definiert werden. Durch die Verwendung von diskreten Hidden-Markov-Modellen kann der Trainingsaufwand zusätzlich reduziert werden [19].

4.1.1 Referenzgesten

Als Referenzgesten, hauptsächlich zum Vergleich und Einschätzung von Messwerten, werden fünf einfache Gesten (Abbildung 4.1) definiert. Alle Referenzgesten werden auf einer zweidimensionalen Fläche ausgeführt. Dabei wird der Wiimote in der späteren Implementierung nicht erkennen, dass die Geste auf einer 2D-Ebene ausgeführt worden ist. Bei der Kreisgeste *Kreis* ist der Startpunkt gleich dem Endpunkt, jedoch beliebig wählbar. Dabei wird der Kreis genau einmal durchlaufen. Geste *Kippen* stellt das dreimalige Kippen des Wiimote um die eigene Achse nach rechts dar. Nach jedem Kippen wird der Wiimote in die Ursprungsstellung zurückgeführt. Zusätzlich zu den grafisch darstellbaren Gesten wird eine kompliziertere Referenzgeste *Tennis*



(a) *Viereck* (b) *Kreis* (c) *Kippen* (d) *Zett* (e) *Tennis*

Abbildung 4.1: Referenzgesten

eingeführt. Bei dieser Geste wird mit dem Wiimote Tischtennis gespielt. Die Geste lässt sich wie folgt beschreiben: Der Nutzer wirft einen Ball nach oben und schlägt diesen, mit dem Wiimote als Schläger, weg. Der Wiimote wird also in einer flüssigen Bewegung nach oben beschleunigt und anschließend ruckartig nach vorne bewegt.

4.1.2 Unbekannte Gesten

Wird eine unbekannte Geste ausgeführt ist es hilfreich auf die fehlerhafte Ausführung oder die noch nicht definierte Geste aufmerksam zu machen. Hidden-Markov-Modelle bieten zur Erkennung von unbekannten Gesten keine Möglichkeit. Jedes Modell reagiert auf die ihm antrainierte Geste mit einer Wahrscheinlichkeit, unbekannte Gesten rufen eine Wahrscheinlichkeit gegen Null hervor. Rubine stellt in seiner Ausarbeitung [22] einen linearen Klassifizierer vor, der die Wahrscheinlichkeit mit der Gesten richtig erkannt worden sind, betrachtet. Unter Zuhilfenahme der Zustände, wie sie im Vorfeld vorgestellt wurden, lässt sich die Nutzerintention erkennen. Sollte der Versuch eine Geste zu machen erkannt werden, liefert jedoch kein Modell eine über einem Schwellwert liegende Wahrscheinlichkeit, wird angenommen, dass es sich um eine unbekannte Geste handelt. Das in dieser Arbeit verwendete Modell ist komplexer als diese einfache Variante. Die genaue Funktionsweise wird in Abschnitt 2.5 beschrieben.

4.2 Quantisierungskomponente

Der Wiimote liefert für die eingeführten Hidden-Markov-Modelle zu viele variierende Datenvektoren. Dies erschwert eine Zuweisung zu eindeutigen Gruppen. Um dieses Problem zu lösen werden die Beschleunigungsvektoren des Wiimote mit Hilfe des k-mean-Algorithmus diskretisiert. Ein sehr wichtiger Punkt bei Anwendung des k-mean-Algorithmus ist die Bestimmung von k, welches die Anzahl der Gruppen angibt.

4.2.1 Anzahl der Gruppen

Prinzipiell liefert der Wiimote im Betrag unterschiedlich lange Vektoren im dreidimensionalen Raum. Einzige Grenze für die Dimension ist die maximale Beschleu-

k / Geste	*Viereck*	*Kreis*	*Kippen*	*Zett*	*Tennis*
6	0,029	0,16	0,47	$5{,}9 \cdot 10^{-4}$	$6{,}1 \cdot 10^{-6}$
14	$6{,}88 \cdot 10^{-7}$	$6{,}23 \cdot 10^{-7}$	0,13	$9{,}35 \cdot 10^{-12}$	$1{,}68 \cdot 10^{-17}$
18	$6{,}88 \cdot 10^{-7}$	$6{,}23 \cdot 10^{-7}$	0,027	$1{,}35 \cdot 10^{-15}$	$1{,}68 \cdot 10^{-17}$

Tabelle 4.1: Modellwahrscheinlichkeiten bei variierender Anzahl von Gruppen

nigung, die der Sensor in jeder Dimension messen kann. Für eine maximale Gruppierungsgenauigkeit müsste jeder Ort in dem Raum mit sich selbst als Zentrum gewählt werden. Das sogenannte Codebuch, die Menge aller Zentren, würde mit dieser Menge an Elementen jedoch schon keine hohe Abstraktion mehr darstellen. Dagegen ist eine zu geringe Anzahl an Elementen im Codebuch wieder eine zu grobe Abstraktion, was sich als nicht hilfreich für eine eindeutige Erkennung der Gesten auswirkt. Der Informationsverlust ist in diesem Fall zu hoch. Die Anzahl der Zentren ist im Idealfall gestenabhängig zu wählen und anhand der Hidden-Markov-Modelle und dessen Wahrscheinlichkeit zu optimieren. In [15] wird $k = 8$ als Anzahl an Codebuch-Elementen für Gestenerkennung im zweidimensionalen Raum verwendet. Wird eine gleichmäßige Verteilung der Werte auf einem Kreis angenommen und auf eine Kugel im dreidimensionalen Raum übertragen kann $k = 14$ als Pendent zu $k = 8$ in 2D bestimmt werden.
Um diese Theorie zu unterstützen wird eine Versuchsreihe bei unterschiedlichem k antraininert und die Modellwahrscheinlichkeit ermittelt (vgl. Tabelle 4.1). Für $k = 6$ hat sich gezeigt, dass die ermittelten Modellwahrscheinlichkeiten sehr gut ausfallen. Dafür besteht das Problem, dass die Auflösung zu gering gewählt ist und sehr leicht ähnliche Gesten als eine falsche Geste erkannt werden. Insbesondere die Referenzgesten *Viereck* und *Kippen* zeigten hier hohe Ähnlichkeiten. Eine Übergenauigkeit kann für den Wert $k = 18$ festgestellt werden. Bei den Gesten *Viereck*, *Kreis* und *Tennis* sind die Modellwahrscheinlichkeiten gleich, für die Gesten *Kippen* und *Zett* wird dafür ein geringerer Wert als für $k = 14$ geliefert. Für diese Arbeit wird $k = 14$ als optimaler Wert für die Anzahl der Gruppen im dreidimensionalen Raum bestätigt.

4.2.2 Wahl initialer Gruppenzentren

Neben der Anzahl der Zentren ist auch die Verteilung über alle Datensätze relevant für eine gute Gruppenzuordnung über alle Vektoren, die vom Wiimote gesendet werden. Für die Funktionsweise des Algorithmus ist eine zufällige Verteilung der Vektoren im Raum nicht optimal. Die Wahl des initialen Codebuchs ist daher besonders relevant. Für einen dreidimensionalen Raum wird eine Kugel mit Radius r um den Ursprung (vgl. Abbildung 4.2) erzeugt. Darauf werden in maximalem Abstand zueinander die k initialen Zentren für den k-mean-Algorithmus platziert. Alle initialen Gruppenzentren sind dadurch paarweise verschieden und das Zusammenfallen von zwei Gruppen daher unwahrscheinlich. Nach der Ausführung des k-mean-Algorithmus auf einer Datenmenge bilden sich Regionen, in denen neue Vektoren eindeutig einem Gruppenzentrum zugeordnet werden können. Jede initiale Bele-

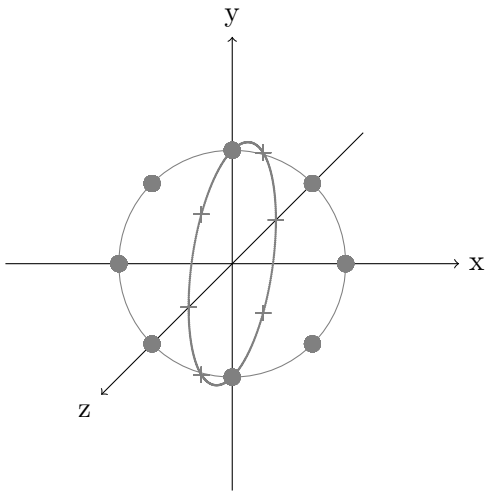


Abbildung 4.2: Die 17 initialen Gruppenzentren

gung endet in unterschiedlichen Regionenaufteilungen [16]. Eine Garantie für eine optimale Lösung gibt es nicht. Um eine annährend optimale Lösung zu erhalten werden unterschiedliche initiale Gruppenzentren verwendet und das Ergebnis nach der minimalen Abweichung ausgewählt [14].

4.2.3 Quantisierung der Vektoren

Der Wiimote sendet, wie bereits in Abschnitt 2.2.4 genauer erläutert, permanent Beschleunigungsvektoren. Hidden-Markov-Modelle in ihrer diskreten Form versprechen hingegen den größten Erfolg bei dem Anwendungsgebiet der Gestenerkennung [19]. Daher werden die Vektoren nach dem vorgestelltem k-mean Algorithmus diskretisiert. Aus dreidimensionalen Vektoren entstehen, wie in Abbildung 4.3 dargestellt, eindimensionale Vektoren. Diese Vektoren können durch eine beliebige Koordinierung repräsentiert werden. Um die Anschaulichkeit zu wahren werden im Folgenden ganzzahlige, positive Werte in aufsteigender Reihenfolge verwendet. Eine aus 10 Vektoren bestehende Geste mit dem Wiimote besteht nach der Diskretisierung weiterhin aus 10 Werten, die einen Wertebereich aus der Menge der k Gruppen haben. Diese Folge von Werten wird im Folgenden mit Hidden-Markov-Modellen als Trainings- oder Erkennungssequenz von Beobachtungen verwendet.

4.3 Hidden-Markov-Modelle

Für jede Geste wird ein eigenes Hidden-Markov-Modell initialisiert, welches mit Hilfe des Baum-Welch-Algorithmus auf eine Geste optimiert wird. Nach [21] ist das Links-Rechts-Modell besonders für die Modellierung von Signalen geeignet, deren Eigenschaften sich über die Zeit verändern.

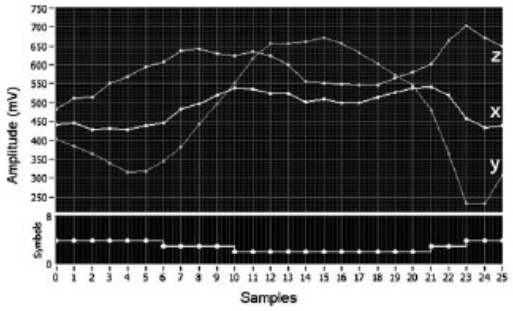


Abbildung 4.3: Diskretisierung der Beschleunigungswerte [15]

4.3.1 Ergodisches vs. Links-Rechts-Modell

Um festzustellen, welches Modell für unsere Anwendung geeigneter ist, wird ein Versuch unternommen und die resultierenden Modellwahrscheinlichkeiten miteinander verglichen. Der Versuch besteht aus drei Messreihen pro Modell über einer variierenden Anzahl von Zuständen. Die vier Referenzgesten werden einmalig für diesen Versuch aufgezeichnet und die Modelle pro Messreihe damit trainiert. Anschließend wird überprüft, wie hoch die Wahrscheinlichkeit ist, dass genau die selbe Beobachtung zu dem Modell gehört. Je höher die Zahl ist, desto optimaler ist das Modell an die Geste angepasst und desto geeigneter ist das Modell im Allgemeinen für Zwecke der Gestenerkennung mit dem Wiimote. Als mögliche Anzahl von Zuständen wird 5, 8 und 10 gewählt. Im ergodischen Modell werden die Startwahrscheinlichkeiten als gleichverteilt angenommen. Aus den Tabellen 4.2 ist ablesbar, dass die Modellwahrscheinlichkeiten sich in beiden Modellen etwa gleich verhalten. Die Aussage nach [12], dass die Wahl der Modellart nicht entscheidend für gute Modellwahrscheinlichkeiten ist, wird für die hier vorliegenden Daten bestätigt. Für den Zweck der Übersichtlichkeit ist das Links-Rechts-Modell dann aber zu bevorzugen, da hier die Zustandsübergänge und Emissionswahrscheinlichkeiten an den Werten von a_{ij}, $b_j\,(k)$ und π_i sehr gut nachvollzogen werden können.

4.3.2 Initiale Modellparameter

Das Hidden-Markov-Modell besteht aus vielen Parametern, die bereits vor dem ersten Training gut konfiguriert werden müssen, um einen Trainingserfolg zu gewährleisten. Die Menge an Beobachtungssymbolen wird durch die Anzahl der Gruppen k im k-mean-Algorithmus vorgegeben. Es wird das Links-Rechts-Modell verwendet, da sich dies als geeigneter herausgestellt hat. Die Menge der Zustände N wird mit fünf Zuständen $s_1, \ldots, s_5$ initiiert, da sich dies bereits bewährt hat [12]. Die Anzahl der Zustände hat nach [20] keinen besonderen Einfluss auf die Ergebnisse.
Für die Initialisierung der Parameter A und B gibt es keine konkrete Vorschrift, sondern heuristische Bestimmungsmethoden [21]. Eine zufällige Belegung ist daher möglich, aber nicht sinnvoll. Zufällige Belegungen können das Modell in der Trai-

5 Zustände	*Viereck*	*Kreis*	*Kippen*	*Zett*	*Tennis*
Ergodisch	$2{,}37 \cdot 10^{-7}$	$1{,}42 \cdot 10^{-5}$	0,0078	$2{,}916 \cdot 10^{-8}$	$2{,}5 \cdot 10^{-15}$
Links-Rechts	$6{,}88 \cdot 10^{-7}$	$6{,}2 \cdot 10^{-7}$	0,132	$9{,}35 \cdot 10^{-12}$	$2{,}8 \cdot 10^{-19}$

8 Zustände	*Viereck*	*Kreis*	*Kippen*	*Zett*	*Tennis*
Ergodisch	$2{,}32 \cdot 10^{-7}$	$5{,}55 \cdot 10^{-6}$	0,0027	$2{,}9 \cdot 10^{-8}$	$2{,}58 \cdot 10^{-15}$
Links-Rechts	$1{,}16 \cdot 10^{-8}$	$1{,}5 \cdot 10^{-5}$	0,124	$1{,}55 \cdot 10^{-11}$	$1{,}68 \cdot 10^{-17}$

10 Zustände	*Viereck*	*Kreis*	*Kippen*	*Zett*	*Tennis*
Ergodisch	$2{,}37 \cdot 10^{-7}$	$1{,}17 \cdot 10^{-6}$	0,0016	$2{,}916 \cdot 10^{-8}$	$2{,}58 \cdot 10^{-15}$
Links-Rechts	$1{,}4 \cdot 10^{-8}$	$1{,}3 \cdot 10^{-4}$	0,125	$1{,}73 \cdot 10^{-11}$	$1{,}68 \cdot 10^{-17}$

Tabelle 4.2: Vergleich der Modellwahrscheinlichkeit bei variabler Zustandsanzahl

ningsphase schon stark beeinträchtigen und somit Trainingsergebnisse verfälschen. Aus diesem Grund ist eine möglichst neutrale Initialisierung zu bevorzugen, da diese das Modell nicht negativ beeinflussen kann. Zu beachten ist hierbei, dass die definierten stochastischen Randbedingungen aus Abschnitt 2.3.2 nicht verletzt werden.

$$a_{ij} = \frac{1}{N - i}, \quad j \geq i$$

Dies schafft für Transitionen zwischen allen Zuständen die selbe Wahrscheinlichkeit und beeinflusst das Modell damit nicht.
Für den Parameter B darf keine zufällige Wahrscheinlichkeit gewählt werden, weil dadurch die späteren Operationen beeinflusst werden. Auch hier gilt, dass eine Gleichverteilung das Modell neutral initialisiert.

$$b_j(k) = \frac{1}{M}$$

Im Links-Rechts-Modell ist die Initialwahrscheinlichkeit π fest definiert: $\pi_0 = 1$. Für ein ergodisches Modell ist die Startwahrscheinlichkeit der Zustände unbekannt. Da zur Initialisierung jedoch bei A und B Gleichverteilungen vorliegen ist es uninteressant, wie wir die Startwahrscheinlichkeiten definieren. Für das Modell ändern sich die Parameter im Training in gleicher Weise, unabhängig von dem Startzustand. Sinnvoll ist daher, die Startzustandswahrscheinlichkeit vom ersten Zustand π_0 mit 1 zu definieren.

4.3.3 Training mit mehreren Trainingssequenzen

In der Theorie wird das Hidden-Markov-Modell mit einer einzigen Beobachtungssequenz trainiert und die Wahrscheinlichkeit auch ausschließlich auf diese eine Sequenz optimiert. Bei der Ausführung von Gesten variieren die Werte durch Messfehler und die Ausführung durch mehrere mögliche Personen teilweise erheblich. Wird ein Hidden-Markov-Modell mit mehreren Gesten hintereinander trainiert wird jeweils die Wahrscheinlichkeit an die letzte Trainingssequenz angepasst. In der Praxis

werden daher, wie in [21] und [20] beschrieben, alle vorliegenden k Beobachtungssequenzen

$$O = \left(O^{(1)}, O^{(2)}, \ldots, O^{(k)}\right)$$

zur Berechnung der Modellwahrscheinlichkeiten herangezogen. Die Sequenz an Stelle k wird wie folgt formuliert:

$$O^{(k)} = \left(O_1^{(k)}, O_2^{(k)}, \ldots, O_n^{(k)}\right).$$

Ziel ist, die Parameter des Referenzmodells so zu verändern, dass die Gesamtwahrscheinlichkeit über alle Teiltrainingssequenzen maximal wird. Für eine Menge von Trainingssequenzen und ein Modell λ bedeutet dies formal:

$$P\left(O|\lambda\right) = \prod_{k=1}^{K} P\left(O^k|\lambda\right) = \prod_{k=1}^{K} P_k.$$

Die Startzustandsverteilung wird von dem Training nicht beeinflusst, da diese beim Links-Rechts-Modell fest initialisiert wurde und keine weiteren Änderungen zugelassen werden. Die Übergangsmatrix und die Emissionswahrscheinlichkeiten werden unter Zuhilfenahme einer Hilfsvariablen und der Vor- und Rückwärtsvariablen berechnet. Die Übergangswahrscheinlichkeiten berechnen sich mittels

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} v_t^k\left(i\right) a_{ij} b_j\left(O_{t+1}^{(k)}\right) r_{t+1}^k\left(j\right)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} v_t^k\left(i\right) r_t^k\left(i\right)}.$$

Für die Emissionswahrscheinlichkeiten gilt:

$$\bar{b}_i\left(k\right) = \frac{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} v_t^k\left(i\right) r_t^k\left(i\right)}{\sum_{k=1}^{K} \frac{1}{P_k} \sum_{t=1}^{T_k-1} v_t^k\left(i\right) r_t^k\left(i\right)}$$

4.4 Klassifizierer

Zur Klassifizikation wird das Verfahren von Bayes gewählt, wie in 2.5 vorgestellt. Für den Wert $P\left(\lambda_i\right)$ wird die höchste Wahrscheinlichkeit gewählt, die von einer der Trainingssequenzen für Modell λ_i erzeugt wird. Für jede zu klassifizierende Geste wird für alle vorhandenen Modelle die Modellwahrscheinlichkeit gebildet. Anschließend wird anhand der aus den Grundlagen bekannten Formel ein Wert zwischen Null und

Eins berechnet. Dieser Wert stellt die Grundlage für die Entscheidung dar. Dem Modell, welches den größten Wert liefert, wird die zu klassifizierende Geste zugeordnet. Vorteil der Bayes-Klassifizierung ist, dass nicht nur die Modellwahrscheinlichkeit der aktuellen Geste berücksichtigt wird, sondern auch die Wahrscheinlichkeit der Trainingsdaten verwendet wird. Dadurch werden selbst fehlerhafterweise festgestellte sehr hohe Modellwahrscheinlichkeiten kompensiert, die z.B. durch eine sehr kurze Sequenz für eine Geste entstehen können.

4.5 Filter

Die vom Wiimote gesendeten Beschleunigungsdaten durchlaufen drei Filter. Dabei werden die Daten aufbereitet, vereinfacht oder zusammengefasst. Ohne Aufbereitung sendet der Wiimote selbst in der Ruhelage permanent bis zu 100 Beschleunigungsdaten pro Sekunde. Wird die Komplexität der Algorithmen betrachtet, stellt solch ein hohes Datenaufkommen die Echtzeitfähigkeit stark in Frage. Zuerst werden die Daten im Ruhelagefilter gefiltert, da für die Gestenerkennung nur Vektoren wichtig sind in denen der Wiimote sich nicht in Ruhe befindet. Anschließend werden Vektoren, die einen ähnlichen Wert haben zu einem Vektor zusammengefasst. Schließlich werden, nach der Diskretisierung, aufeinander folgende, gleiche Elemente zu einem zusammengefasst.

4.5.1 Bedeutung der Filter

Die Hauptaufgabe der Filter ist die Reduktion des Datenaufkommens um die komplexen Algorithmen im Rahmen der Echtzeit zu betreiben. Eine weitere Aufgabe ist die Reduktion von Fehlern, die z.B. durch Zittern des Nutzers entstehen können. Nach durchlaufen der Vektoren durch alle drei Filter entsteht eine minimale Repräsentation einer Geste, mit der das Modell trainiert oder verglichen wird. Auf diese Art gekürzte Gesten ermöglichen, durch die mathematischen Zusammenhänge im Modell bedingt, bessere Modellwahrscheinlichkeiten. In der Praxis können sehr kurze Gesten aber auch zu Problemen bei der Erkennung führen. Ein beliebiges Modell kann, bedingt durch die hohe Wahrscheinlichkeit des Emittierens von einem einzigen Symbol, schnell die global höchste Modellwahrscheinlichkeit erlangen. Diese Situation tritt, je höher die Anzahl Gruppenzentren für den k-mean-Algorithmus gewählt wird, weniger häufig ein. Unter sinnvollen Anwendungsbedingungen ist dieses Problem nicht relevant. Als zusätzlicher Schutz gegen diese Art von Trivialgesten-Erkennung dient auch der Bayes-Klassifizierer, der eine Geste auch unter Zuhilfenahme der Modellwahrscheinlichkeit für Trainingsgesten als erkannt einstuft oder nicht. Unbekannte Gesten werden durch die Bayes-Klassifiktation ausgeschlossen.

4.5.2 Ruhelagefilter

Wie in Abschnitt 3.3.1 vorgestellt wird anhand des Betrags ermittelt, ob sich der Wiimote in Ruhelage befindet. Wenn der Betrag einen bestimmten Schwellwert über-

Geste	Messung 1	2	3	4	5	Mittel
Viereck	109	111	131	102	110	113
Kreis	98	105	112	92	95	100
Kippen	29	29	38	31	28	31
Zett	118	130	131	142	139	132
Tennis	110	109	109	110	110	110

Tabelle 4.3: Anzahl der Vektoren vor dem Richtungsäquivalenzfilter

schreitet, werden die Beschleunigungsdaten weiterverarbeitet. Für die aus Vektoren bestehende geordnete Eingabemenge N gilt nach Anwendung des Filters für die geordnete Ausgabemenge N'

$$N' = \{\vec{a} \in N \mid |\vec{a}| \geq \Delta\}$$

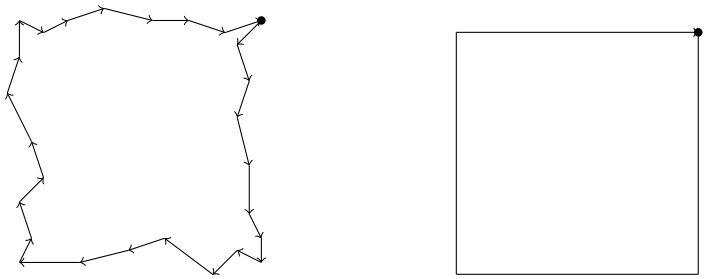
wobei Δ der zu wählende Schwellwert ist. Der Schwellwert repräsentiert die gewünschte Empfindlichkeit des Filters. Ein Wert von $\Delta = 1{,}2g$ hat sich beim Wiimote als geeignet erwiesen. Für Werte über diesem Wert war die Empfindlichkeit des Wiimote zu gering, während sie bei geringeren Werten zu hoch war und sich somit das Datenvolumen nicht nennenswert reduziert hat.

4.5.3 Richtungsäquivalenzfilter

Durch die hohe Rate an Werten, die der Wiimote liefert, sind die Richtungen der Beschleunigungsvektoren bei aufeinanderfolgenden Werten annähernd gleich. Der Wiimote ändert bei Gesten üblicherweise nicht mehrfach in kürzester Zeit die Richtung, da die Gesten sonst keiner natürlichen Handbewegung mehr unterliegen würden. Ähneln sich mehrere aufeinanderfolgende Vektoren können diese zu einem Vektor zusammengefasst werden. Für die geordnete Ausgabemenge des Filters N' und einer beliebigen Koordinate $c \in \{x, y, z\}$ heißt das

$$N' = \left\{\vec{a} \in N \mid \exists c : \left|\vec{a}_c^{(n)} - \vec{a}_c^{(n-1)}\right| \geq \varepsilon\right\}.$$

$\vec{a}_c^{(n)}$ bezeichnet hierbei den Vektor $\vec{a}$ an der Position n mit Koordinate $c \in \{x, y, z\}$. Diese Notation bleibt auch für den folgenden Abschnitt bestehen. Der Schwellwert ε wird nach gewünschter Genauigkeit frei gewählt. Für den Wiimote hat sich ein Wert von $\varepsilon = 0{,}2$ als ausreichend genau herausgestellt.
Anhand der quadratischen Referenzgeste *Viereck* wird diese Art der Filterung in Abbildung 4.4 vereinfacht dargestellt. Um die Wirksamkeit vom Filter für dieses Projekt zu veranschaulichen sind in Tabelle 4.3 und 4.4 entsprechende Werte aufgelistet. Die in der Anzahl verringerten Vektoren werden anschließend mittels des k-mean-Algorithmus diskretisiert.



(a) Vor der Filterung (b) Nach der Filterung

Abbildung 4.4: Richtungsäquivalenzfilter

Geste	Messung 1	2	3	4	5	Mittel
Viereck	56	45	54	51	44	50
Kreis	23	31	25	22	27	26
Kippen	19	22	22	24	20	21
Zett	78	78	69	66	66	71
Tennis	67	61	66	65	67	65

Tabelle 4.4: Anzahl der Vektoren nach dem Richtungsäquivalenzfilter

4.5.4 Gruppenäquivalenzfilter

Beim Diskretisieren werden aufeinander folgende Vektoren häufig selben Klassen zugeordnet. Ein weiterer Filter entfernt gleiche aufeinanderfolgende Elemente. Nach dem Filtern sind alle Werte in der Sequenz paarweise voneinander verschieden.

$$N' = \left\{ a \in N \mid a^{(n)} \neq a^{(n-1)} \right\}$$

Im Optimalfall würde Geste *Viereck* somit noch aus vier diskreten Elementen bestehen, da hier der Wiimote in vier unterschiedliche Richtungen beschleunigt wird und die Diskretisierung daher im Optimalfall vier Werte liefern wird. Wie stark sich die Glättung tatsächlich auswirkt zeigt Tabelle 4.5. Die Anzahl der Werte vor der Glättung kann aus Tabelle 4.4 entnommen werden. Ein Plot der Filterwerte findet sich in Abbildung 4.5.

Geste	Messung 1	2	3	4	5	Mittel
Viereck	15	17	18	18	15	17
Kreis	8	8	8	8	8	8
Kippen	14	14	16	15	14	15
Zett	16	15	14	17	15	15
Tennis	13	14	14	13	16	14

Tabelle 4.5: Anzahl der diskreten Werte nach der Glättung

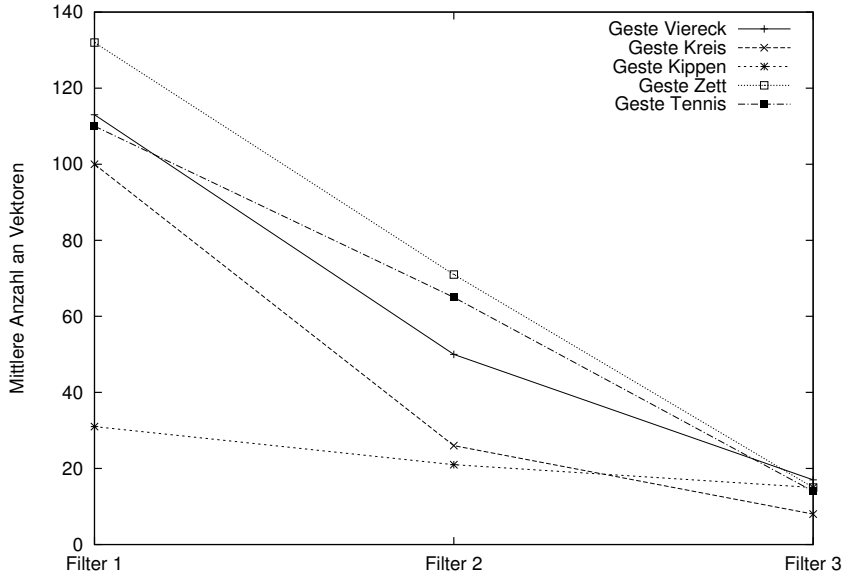


Abbildung 4.5: Plot der Filterwerte

5 Implementierung

Das Kapitel der Implementierung begründet die Wahl der Programmiersprache Java und geht auf das verwendete Entwurfsmuster mit Vor- und Nachteilen ein. Ein Klassendiagramm gibt eine Übersicht über sämtliche im System befindliche Klassen. Zum tieferen Verständnis werden wichtige Schnittstellen beschrieben und dessen Zusammenhang mit dem Entwurfsmuster geklärt. Außerdem werden die identifizierten Komponenten explizit in ihrer Implementierung beschrieben und die Funktionsweise anhand von Grafiken und Automaten erklärt. Ein vollständiger Ablauf eines Gestentrainings sowie einer Gestenerkennung ermöglicht praxisorientierten Lesern einen unmittelbaren Einstieg in die Arbeit. Am Ende der Arbeit stellen mehrere Tabellen und eine Grafik die Leistungsfähigkeit der Implementierung unter Beweis und zeigen die Grenzen des Systems auf. Während des ganzen Kapitels wird teilweise die englische Sprache für Fachbegriffe verwendet.

5.1 Wahl der Programmiersprache

Als Programmiersprache wird Java[1] verwendet, da die Sprache die drei großen Betriebssysteme (Windows, MacOS X, Linux) problemlos unterstützt. Zudem gestaltet sich das Zusammenspiel von Java und Bluetooth als sehr einfach und exakt spezifiziert. Dies war für die Ansteuerung vom Wiimote und erste Datenauswertungen sehr hilfreich.

5.1.1 Java und Bluetooth

Die Kommunikation mit dem Wiimote erfordert ohne Java eine betriebssystemabhängige Ansteuerung der Bluetooth-Schnittstelle. Leider enthält Java keine Implementierung der JSR-82-Spezifikation[2], es stehen aber mehrere Implementierungen zur Verfügung. Als sehr ausführlich dokumentiert und zumindest für Linux-Nutzer unter der GPL-Lizenz verfügbar hat sich Avetana Bluetooth[3] herausgestellt. Für Windows- oder MacOSX-Nutzer ist kostenlos eine zeitlich beschränkte Demoversion verfügbar. Für eine Vollversion sind 25 Euro zu zahlen. Selbstverständlich kann die Implementierung gegen jede andere JSR-82 konforme Implementierung ausgetauscht werden, da Methoden und Klassen von der Arbeitsgruppe um JSR-82 genau spezifiziert werden.

[1] `http://java.sun.com/` letzter Zugriff: 28.07.07

[2] JSR-82 Spezifikation `http://jcp.org/en/jsr/detail?id=82`, letzter Zugriff: 28.07.07

[3] Avetana-Bluetooth Bibliothek `http://www.avetana-gmbh.de/avetana-gmbh/produkte/jsr82.eng.xml` letzter Zugriff: 28.07.07

5.2 Klassendiagramm

In Abbildung 5.1 ist das Klassendiagramm der Logik abgebildet. Die Klasse `Quantizer` ist unsere Quantisierungskomponente in Form des k-mean-Algorithmus. `Gesture` modelliert eine Geste mit einer Sequenz von Beschleunigungswerten in dem Attribut `data`. Das Hidden-Markov-Modell befindet sich in der Klasse `HMM`. Es stellt Methoden, wie `getProbability()` und `train()` zur Verfügung. Die Klasse `GestureModel` verknüpft alle Komponenten miteinander und enthält den Ablauf über das Training oder die Erkennung.

5.3 Entwurfsmuster

Grafische Oberflächen in Java werden über sogenannte Events (deutsch: Ereignisse) gesteuert. Ein Event kann z.B. ein Tastendruck oder eine Mausbewegung sein. Die Architektur dieser Arbeit sieht vor, dass der Wiimote bei Gesten ebenfalls solche Events erzeugt. Erzeuger der Events sind die Klassen `Wiimote` und `AccelerationStreamAnalyzer`. Möchte beispielsweise die fiktive Klasse `MP3Player` die Events empfangen, implementiert Sie die abstrakten Methoden aus den Interfaces `WiimoteListener` und `GestureListener`. Die hardwarenahe Ebene, der `WiimoteListener`, empfängt z.B. die Events, wenn ein Knopf am Wiimote gedrückt wird oder der Wiimote beschleunigt wird. Dagegen wird der `GestureListner` mit Gestenevents oder Zustandsänderungen der in Abschnitt 3.3 vorgestellten Statemachine bedient.

5.3.1 Vorteile

Das Entwurfsmuster erleichtert dem Programmierer individuell auf sehr hardwarenahe Funktionen des Wiimote zu reagieren. Zudem erleichtern abstrakte Events, wie ein `GestureEvent`, ein Umgehen der komplexen Logik der Gestenerkennung. Für die Integration der Gestenerkennung muss ein Entwickler seine Anwendung nur den Events beliebige Handlungen zuweisen. Events werden in grafischen Anwendungen sowieso verwendet, eine Einarbeitung in andere Techniken ist daher nicht nötig. Zudem muss eine Anwendung nicht regelmäßig Informationen einholen (*pull*), ob eine neue Geste ausgeführt worden ist, sondern wird direkt über neue Ereignisse informiert (*push*). Das spart in der Anwendung viel Logik und Rechenzeit für andere Aufgaben. Alternativ sind Schnittstellen denkbar, die entsprechende Gestenevents in z.B. MIDI-Steuersignale umwandeln und so auch Programme außerhalb der Programmiersprache Java bedienen können.

5.3.2 Nachteile

Ein Nachteil der Methode ist der hohe Speicherbedarf im Vergleich zu einfacheren Anwendungsmustern. Für jede Art von Event wird ein neues Objekt erzeugt, welches anschließend an die Klassen, die auf die Events reagieren sollen, geschickt wird. Wird berücksichtigt, dass jede Beschleunigung ein Event auslöst kommen so sehr

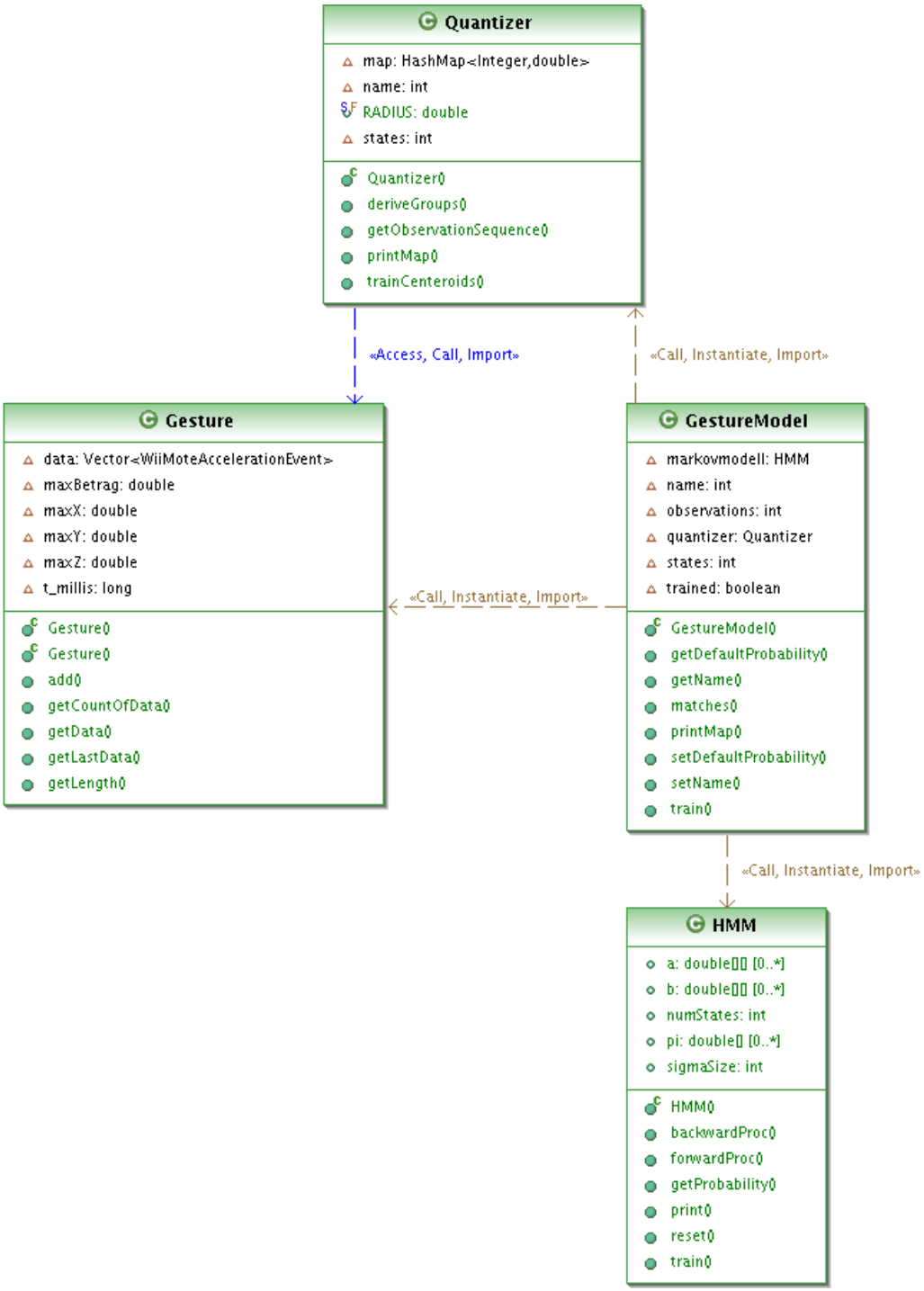


Abbildung 5.1: Klassendiagramm der Logik

schnell viele Objekte zusammen, die alle einen Teil vom Speicher für sich beanspruchen. Durch eine geschickte Filterung der Daten lässt sich das Datenaufkommen aber so reduzieren, dass der Nachteil des Speicherverbrauchs vernachlässigt werden kann. Zudem gilt zu berücksichtigen, dass die Gestenerkennung in dieser Arbeit auf Heimcomputer und nicht portable, ressourcenbegrenzte Geräte ausgelegt ist. Für die Zielarchitektur ist die Performanz im Echtzeitbereich anzusiedeln.

5.4 Interfaces und Events

In diesem Abschnitt finden sich Beschreibungen zu erstellten Interfaces und erzeugten Events. Dabei werden ausschließlich die wichtigsten Events vorgestellt, da nur wenige Beispiele zum Verständnis notwendig sind.

5.4.1 Interface: WiimoteListener

Klassen, die das Interface `WiimoteListener` implementieren und entsprechend von einem Wiimote mit Daten versorgt werden, können mit drei unterschiedlichen Events versorgt werden. Zum einen wird ein `ButtonPressedEvent` gesendet, in welchem kodiert ist welcher Knopf auf dem Wiimote soeben gedrückt worden ist. Das entsprechende Pendant für das Loslassen eines Knopfes heißt `ButtonReleasedEvent`. Zum anderen wird ein Beschleunigungsevent `AccelerationEvent` erzeugt für jede Beschleunigung, die einen gewissen Schwellwert überschreitet. In diesem Event enthalten sind alle relevanten Daten, wie z.B. der Betrag oder die Beschleunigung in x-Richtung. Insgesamt lassen sich so die direkten Interaktionen am und mit dem Wiimote festhalten und für Anwendungszwecke nutzen.

5.4.2 Interface: GestureListener

Wird das Interface `GestureListener` implementiert, empfängt die Klasse Events, die Auskunft über den aktuellen Zustand des Automaten geben (vgl. Abschnitt 5.5) und ein `GestureEvent`, wenn eine Geste erkannt worden ist. Bei einem `GestureEvent` handelt es sich um ein sehr abstraktes Event, das die gesamte Gestenerkennungslogik maskiert. Intern trägt jedes Gestenmodell eine aufsteigende Nummer aus den natürlichen Zahlen. Die Zuweisung eines Namens und einer Funktion erfolgt außerhalb der Anwendung durch die Semantik mit denen die Gesten behaftet sein könnten.

5.4.3 Event: GestureEvent

Das `GestureEvent` wird erzeugt und an die Listener übergeben, wenn von einem Klassifizierer eine Geste erkannt worden ist. Ohne eine ausgeführte Bewegung mit Erkennungswunsch wird dieses Event nie im System erscheinen. Ein `GestureEvent` ist eine Klasse bestehend aus drei Attributen: Eine Nummer, einer Wahrscheinlichkeit und einer Quelle. Die Nummer entspricht der programminternen Repräsentation aus den natürlichen Zahlen. In einer möglichen Umgebung wird der Geste anhand

der eindeutigen Nummer sicherlich eine aussagekräftigere Bezeichnung zugewiesen. Als zusätzliche Informationen zur Nummer wird eine Wahrscheinlichkeit geliefert, die dem Nutzer die Wahrscheinlichkeit ausgibt, zu der die Geste erkannt worden ist. Die Quelle dient dazu, dass mehrere Eingabegeräte benutzt werden können und dass beliebig viele Gestenerkennungseinheiten erzeugt werden können. Damit bleibt die Möglichkeit offen eine ausgeführte Geste mit unterschiedlichen Modellen zu erkennen und z.B. über die Wahrscheinlichkeit den Mittelwert zu bilden.

5.4.4 Event: WiimoteAccelerationEvent

Dieses Event wird erzeugt, wenn der Wiimote beschleunigt wird und die Beschleunigung nicht innerhalb der beschriebenen Filter entfernt wird. Anhand des `WiimoteAccelerationEvents` lässt sich beispielsweise ein Graph darstellen, wie in Abbildung 2.4 dargestellt. Das Event besteht aus den Parametern x, y und z für die Beschleunigungswerte in den drei Dimensionen sowie aus dem daraus berechneten Betrag und einer Quelle. Bis auf die Quelle sind alle Werte selbsterklärend. Als Quelle wird ein Wiimote angegeben, damit können mehrere Wiimotes im System unabhängig voneinander betrieben werden und mögliche Gesten mit mehreren Geräten ausgeführt werden.

5.5 Implementierter Automat

Der in Abschnitt 3.3 vorgestellte Automat in der allgemeinen Fassung wird in diesem Abschnitt für die konkrete Implementierung beschrieben. Um unbeabsichtigt ausgeführte Gesten zu vermeiden wechselt der Zustand ausschließlich mit explizitem Kommando in einen anderen Zustand. In Zustand `Pause` kann der Wiimote bewegt werden, ohne das es zu Handlungen im Programm kommt. Befindet sich der Automat in diesem Zustand und wird [Knopf A] gedrückt wechselt der Automat in den Zustand `Training`. Bis der Knopf losgelassen wird, werden Beschleunigungswerte gespeichert und mit dem Loslassen als eine Trainingseinheit abgelegt. Durch erneutes Drücken von [Knopf A] wird eine weitere Trainingsgeste aufgenommen, bis der Knopf wieder losgelassen wird. Mit dem [Knopf Home] wird ein neues Hidden-Markov-Modell erzeugt und mit den gesammelten Trainingsdaten, wie in Abschnitt 4.3.3 vorgestellt, trainiert. Wird in Zustand `Pause` [Knopf B] gedrückt und bis zum Ende der Geste gehalten, wird eine Geste aufgezeichnet. Mit dem Loslassen des Knopfes wird die aufgezeichnete Geste mit allen Modellen verglichen und Anhand der Bayes-Klassifikation die zugehörige Geste ermittelt.

5.6 Quantisierungskomponente

Für jedes Modell und damit für jede Geste wird eine eigene Quantisierungskomponente genutzt. Explizit bedeutet dies, dass für jedes Hidden-Markov-Modell jeweils

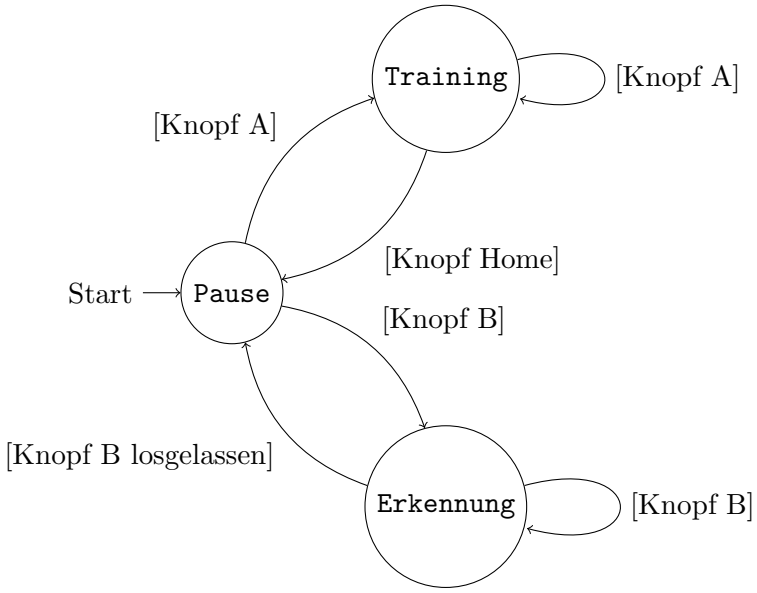


Abbildung 5.2: Implementierter Automat

ein k-mean-Algorithmus existiert. Das ist insofern sinnvoll, dass jede Geste die initialen Gruppenzentren anders verändert, weil die Beschleunigungen der Gesten in unterschiedlichste Richtungen verteilt sein können. Der erhöhte Rechenaufwand fällt bei der Gestenerkennung aber gering aus, weil dabei die Gruppenzentren nicht neu gesetzt werden sondern die Vektoren lediglich einem diskreten Symbol zugeordnet werden. Jeder k-mean-Algorithmus wird mit vierzehn initialen Gruppenzentren initialisiert, die auf einer Kugel mit Radius r angeordnet verteilt sind (vgl. Abbildung 4.2). Der Radius r sollte möglichst nahe an $r = \Delta = 1{,}2g$ gewählt werden, um den Rechenaufwand zu reduzieren. Prinzipiell ist ein beliebiger, positiver Radius zulässig.

5.7 Hidden-Markov-Modell

Die Hidden-Markov-Modelle werden für jede Geste gleich initialisiert und bestehen aus fünf Zuständen sowie einem diskreten Beobachtungsalphabet der Länge vierzehn. Es wird ein Links-Rechts-Modell verwendet. Eine detaillierte Beschreibung des Trainings- und Erkennungsablaufs beschreibt die Integration der Modelle in die Umgebung.

5.7.1 Trainingsablauf

Der Trainingsablauf ist in Abbildung 5.3 dargestellt. Die vom Wiimote empfangenen Daten sammeln sich in einer Menge von Trainingssequenzen, welche für jedes Modell separat dem k-mean-Algorithmus zugeführt werden. Die diskreten Werte werden anschließend dem Hidden-Markov-Modell übermittelt und dort mit vorgestellten Ver-

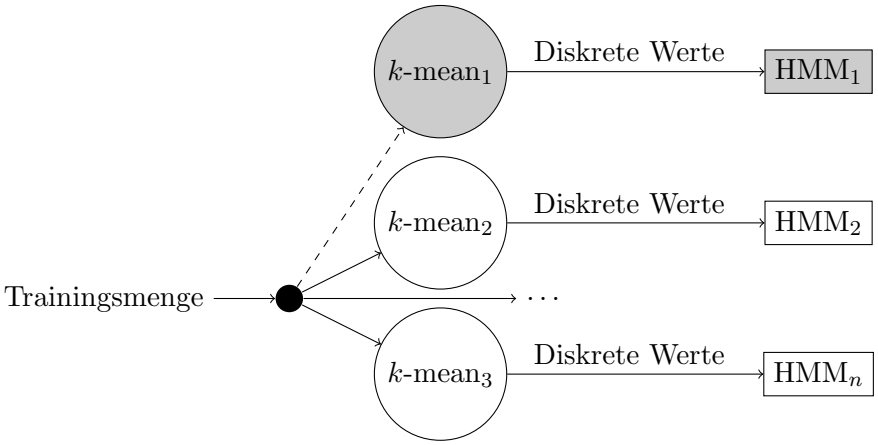


Abbildung 5.3: Schematische Darstellung des Trainingsablaufs

fahren trainiert. Das Modell kann nicht erneut trainiert werden, trainierte Modelle werden daher in der Abbildung grau markiert. Die nächste Menge von Trainingssequenzen wird daher dem nächsten, freien Modell zugeführt. Durch die unbegrenzte Anzahl an Modellen lässt sich die Prozedur unendlich lange fortführen. Um eine Geste zu entfernen genügt es, das Modell samt gespeicherten k-mean-Gruppenzentren zu löschen.

5.7.2 Erkennungsablauf

Die Gestenerkennung erfordert weniger Eingriffe und Vorverarbeitung als das Antrainieren von Gesten. Die Beschleunigungsvektoren werden direkt in den k-mean-Algorithmus gegeben und resultierende diskrete Sequenzen mit der Vorwärts-Prozedur ausgewertet. Eine nachgestellter Bayes-Klassifizierer ermittelt schließlich die entsprechende Geste. Bei der Gestenerkennung existiert eine einzige Sequenz von Beschleunigungswerten. Diese Sequenz wird von allen k-mean-Algorithmen in eine Sequenz von diskreten Werten umgewandelt und diese werden jeweils dem zugehörigen Hidden-Markov-Modell übergeben (vgl. Abbildung 5.4). Im k-mean-Algorithmus werden dabei die Gruppenzentren nicht neu ausgerichtet, sondern bleiben so bestehen, wie im Training ermittelt. Im Modell wird die Wahrscheinlichkeit zu der Sequenz vom Vorwärts-Algorithmus berechnet. Die ganzen Wahrscheinlichkeiten der verschiedenen Modelle werden einem Entscheider zugeführt, der die höchste Wahrscheinlichkeit auswählt und die Sequenz der Beschleunigungswerte eindeutig einem Modell und damit einer Geste zuordnen kann.

5.8 Klassifizierer

Der Klassifizierer liefert nach der Wahrscheinlichkeitsbestimmung durch die diskreten Hidden-Markov-Modelle eine Einschätzung, um welche Geste es sich tatsäch-

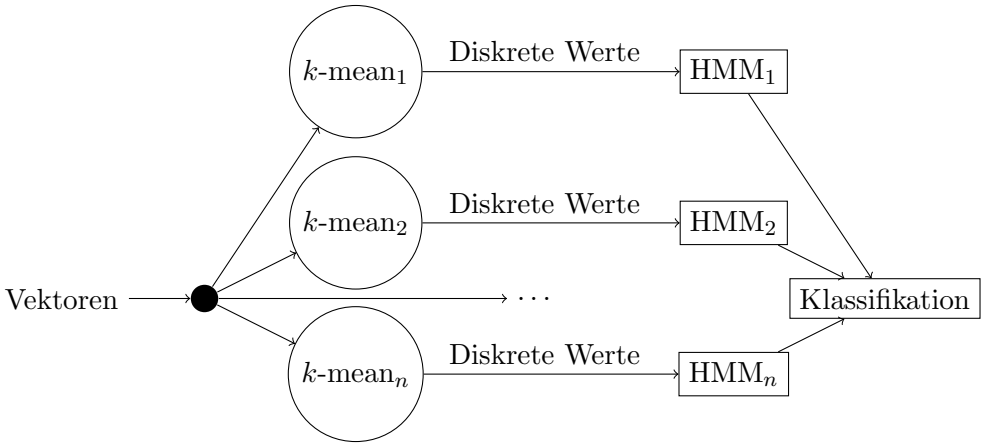


Abbildung 5.4: Schematische Darstellung des Erkennungsablaufs

lich handelt. Anfängliche Versuche sich auf die höchste Modellwahrscheinlichkeit zu stützen verliefen schnell in einer häufigen Fehlklassifizierung der Gesten. Der Klassifizierer nach Bayes berücksichtigt neben der Modellwahrscheinlichkeit für die zu klassifizierende Geste auch die höchste Modellwahrscheinlichkeit von den Trainingsdaten. Dadurch lässt sich das Ergebnis genauer feststellen und ein anschließender Schwellwert von 0,6 für die Erkennungswahrscheinlichkeit eliminiert fehlerhaft erkannte Gesten.

5.9 Filter

Bevor ein Event erzeugt wird, greifen bereits der Ruhelagefilter und der Richtungsäquivalenzfilter. Damit wird ein unnötiges Erzeugen von Events verhindert und damit Speicher und Rechenzeit gespart. Nach dem Durchlaufen der Diskretisierung werden die Events verworfen und der dritte Filter glättet die Folge von diskreten Symbolen. Diese verkürzte Folge wird anschließend im Modell entsprechend algorithmisch behandelt. Für den Wiimote haben sich beim Ruhelagefilter ein Schwellwert für den Betrag von $\Delta = 1{,}2g$ ergeben. Für den Richtungsäquivalenzfilter wird $\varepsilon = 0{,}2$ gewählt.

5.10 Vollständiger Ablauf

Dieser Abschnitt beschreibt abschließend einen kompletten Trainings- und Erkennungsablauf aus Nutzersicht. Damit werden dem Leser die Zusammenhänge der erklärten Theorie verdeutlicht und praktisch veranlagten Lesern ein guter Einstieg in die Arbeit gegeben. Für das Szenario wird angenommen, dass der Nutzer die Referenzgeste *Viereck* trainieren und erkennen möchte.

5.10.1 Geste antrainieren

Der Nutzer nimmt seinen Wiimote in die Hand und verbindet ihn über eine Bluetoothverbindung mit seinem Rechner und der Java-Anwendung zur Gestenerkennung. Nach erfolgreichem Verbindungsaufbau wird Knopf A gedrückt und ein Rechteck in die Luft gezeichnet. Die Applikation zeichnet hierbei durchschnittlich 113 Beschleunigungswerte auf, die durch einen Ähnlichkeitsfilter auf 50 reduziert werden. Nach dem Loslassen von Knopf A kann der Nutzer die Geste noch mehrfach trainieren oder die Lernprozedur durch einen Druck auf den Knopf Home den Lernvorgang abschließen. Mit Abschluss des Lernvorgangs wird ein k-mean-Algorithmus und ein zugehöriges Hidden-Markov-Modell initialisiert. Für jede Trainingssequenz werden die 50 Beschleunigungswerte durch den k-mean-Algorithmus diskretisiert. Dabei passen sich die Gruppenzentren an die Trainingsmuster an. Ergebnis der Diskretisierung ist eine Menge von diskreten Werten aus dem Hidden-Markov-Alphabet mit pro Element einer durchschnittlichen Länge von 17. Mit den diskreten Werten wird im Anschluss das Hidden-Markov-Modell trainiert und damit die Wahrscheinlichkeit für diese Art von Geste optimiert.

5.10.2 Geste erkennen

Die Erkennung wird vom Nutzer mit dem Knopf B gestartet. Die Geste wird auch aufgezeichnet und, genau wie beim Training, vereinfacht. Die vereinfachten Werte werden von allen vorhandenen k-mean-Algorithmen diskretisiert und anschließend dem zugehörigem Modell zugeführt. Das Modell gibt die Wahrscheinlichkeit, dass die Geste zu dem Modell gehört, an einen Klassifizierer weiter, welcher dem Nutzer angibt, dass die ausgeführte Sequenz zu der Geste *Viereck* gehört. Wird keine akzeptable Wahrscheinlichkeit vom Modell geliefert teilt dies der Bayes-Klassifizierer dem Nutzer ebenfalls mit.

5.11 Erkennungsraten

In diesem Abschnitt wird eine Auswertung der Erkennungsraten durchgeführt. Dafür wird der Wert nach der Bayes-Klassifikation über 20 Versuchsreihen gemittelt. Im Vorfeld wird jede Geste n Mal trainiert. Es wurden alle fünf Referenzgesten antrainiert und anschließend zwanzig Mal jede Geste ausgeführt. Durch dieses Verfahren sind Verwechslungen mit anderen Gesten und Nichterkennung der Geste eingeschlossen. Die Modellparameter sind so gewählt, wie bereits beschrieben.

5.11.1 Fünf Trainingsdurchläufe

Bei fünf Trainingsläufen pro Geste sind die Erkennungsraten noch ziemlich gering, da es oft zu falschen Klassifikationen kommt. Bei fünf unterschiedlichen Sequenzen kann sich das Modell noch nicht so gut an die Gegebenheiten der Geste anpassen. Die so erzielten Raten sind für praktisch anspruchsvolle Aufgaben noch nicht ausreichend.

Geste	Anzahl Treffer (bei 20 Versuchen)	Prozent
Viereck	11	55%
Kreis	14	70%
Kippen	17	85%
Zett	14	70%
Tennis	6	30%

Tabelle 5.1: Erkennungswahrscheinlichkeit bei fünf Trainingssequenzen

Geste	Anzahl Treffer (bei 20 Versuchen)	Prozent
Viereck	18	90%
Kreis	19	95%
Kippen	19	95%
Zett	17	85%
Tennis	18	90%

Tabelle 5.2: Erkennungswahrscheinlichkeit bei zehn Trainingssequenzen

Die einzelnen Werte sind in Tabelle 5.1 aufgeführt. Die Geste *Tennis* kann, bedingt durch ihre Komplexität, bei fünf Trainingsdurchläufen nicht akzeptabel trainiert werden und so ist die maximal erreichbare Erkennungsrate sehr gering.

5.11.2 Zehn Trainingsdurchläufe

Bei zehn Trainingsdurchläufen werden dem Modell ausreichend viele Variationen einer Geste antrainiert, was sich schließlich in höheren Erkennungsraten auswirkt. Nach 10 Trainingsdurchläufen kann die Gestenerkennung mit den Referenzgesten als ausreichend genau bezeichnet werden. Deutlich zu sehen an Tabelle 5.1 und Tabelle 5.2 ist, dass die Komplexität der Geste maßgeblich für das Ergebnis ist. Anhand der Werte kann festgestellt werden, dass die Gesten *Viereck* und *Zett* sehr komplex aufgebaut sind, während die Kreisgeste *Kreis* bei den Raten im Mittelfeld agiert. Von der Komplexität ist Geste *Kippen* als sehr einfach einzustufen. Geste *Tennis* wird bei dieser Anzahl von Trainingsdurchläufen schon erstaunlich oft erkannt. Vermutlich, weil wir bei dieser Geste sehr markante Maximalbeschleunigungen erreichen und sich diese im Modell deutlich zeigen. Zudem besteht fast keine Verwechselungsgefahr mit anderen Gesten bei der Erkennung. Die Einschätzung der Komplexität deckt sich mit dem persönlichen Empfinden beim Ausführen der Gesten.

5.11.3 Fünfzehn Trainingsdurchläufe

Bei fünfzehn Trainingsdurchläufen werden alle Gesten noch zuverlässiger Erkannt, als bei zehn Trainingsvorgängen. Das System lernt noch mehr unterschiedliche Ausprägungen von den Gesten und kann die zugehörigen Modelle so wesentlich besser optimieren. In Tabelle 5.3 werden die Erkennungsraten veranschaulicht. Die Kom-

Geste	Anzahl Treffer (bei 20 Versuchen)	Prozent
Viereck	19	95%
Kreis	20	100%
Kippen	20	100%
Zett	19	95%
Tennis	20	100%

Tabelle 5.3: Erkennungswahrscheinlichkeit bei fünfzehn Trainingssequenzen

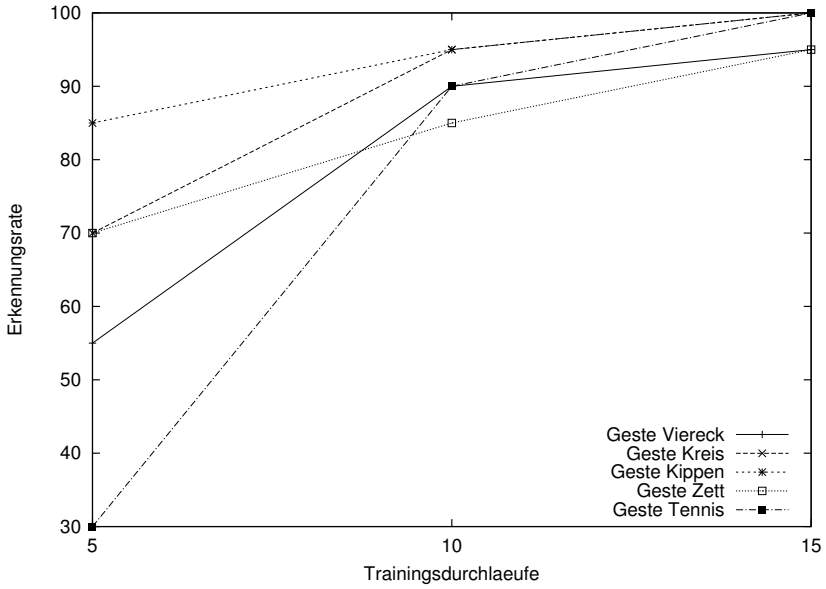


Abbildung 5.5: Plot der Erkennungsraten

plexitätseinschätzung für die Gesten bei zehn Trainingsdurchläufen bleibt auch bei fünfzehn Trainingsdurchläufen erhalten.

5.11.4 Zusammenfassung

Zusammenfassend lässt sich sagen, dass ab zehn Trainingsdurchläufen pro Geste brauchbare Ergebnisse festgestellt werden können. Fünf Trainingsdurchläufe bescheren insbesondere bei den komplexen Gesten keine akzeptabelen Erkennungsraten. Bei fünfzehn Trainingsdurchläufen kommt die Erkennungsrate, insbesondere bei den einfachen Gesten, den 100% ziemlich nahe. Die grafische Veranschaulichung in Abbildung 5.5 ermöglicht einen visuellen Eindruck über die Erkennungsraten.

6 Resümee und Ausblick

In der Arbeit wird gezeigt, dass der Wiimote nicht nur im Zusammenspiel mit der Konsole, sondern auch autonom für die beschleunigungsbasierte Gestenerkennung verwendet werden kann. Die dazu notwendige Bluetooth-Kommunikation wird beschrieben und für die Gestenerkennung aufbereitet. Für den Prozess der Gestenerkennung und zugehörigem Training werden diskrete Hidden-Markov-Modelle eingesetzt. Diese Modelle in Kombination mit einem Filterkonzept und einer dreidimensionalen Diskretisierungskomponente eignen sich sehr gut. Die Eignung der gewählten Komponenten wird durch die hohen Erkennungsraten hervorgehoben, die ab zehn Traininigsdurchläufen pro Geste erreicht werden können.
Auch das Ziel der Arbeit, die beschleunigungsbasierte 3D-Gestenerkennung, kann damit als erreicht angesehen werden. Durch das gewählte Entwurfsmuster kann die Gestenerkennung mit dem Wiimote nun für andere Applikationen ohne weitere Hintergrundkenntnisse bzgl. der Mustererkennung verwendet werden. Denkbar wären konsolenähnliche Spiele auf dem Heimcomputer ohne die Kosten für eine zusätzliche Konsole aufbringen zu müssen. Sollte in der Zukunft ein Wii-Emulator für den Heimgebrauch existieren ist denkbar auch dort die Gestenerkennung zu verwenden um die Spiele so originalgetreu wie möglich nachzubilden. Ein weiterer Aspekt, der in der Zukunft berücksichtigt werden sollte ist unter anderem die Portierung der Gestenerkennung auf mobile Plattformen. Dort besteht der Vorteil, dass die Hersteller meist eine JSR-82-Implementierung mitliefern und diese nicht separat erworben werden muss.
Weitergehende Forschungen könnten sich mit dem Nutzen der Gestenerkennung in diverser Software beschäftigen. Zusätzlich können auch alternative Komponenten für das Gesamtsystem gewählt werden und dessen Resultate mit den Ergebnissen dieser Arbeit verglichen werden.

Literaturverzeichnis

[1] *Analog Devices Inc.* URL: `http://www.analog.com/UploadedFiles/Data_Sheets/ADXL330.pdf`, letzter Zugriff: 29.08.07.

[2] *Broadcom Corporation.* URL: `http://www.broadcom.com/collateral/pb/2042-PB03-R.pdf`, letzter Zugriff: 29.08.07.

[3] *Spark Fun Electronics.* URL: `http://www.sparkfun.com/commerce/present.php?p=Wii-Internals`, letzter Zugriff: 29.08.07.

[4] *ST Microelectronics.* URL: `http://www.st.com/stonline/products/literature/ds/4578/m24128-bw.pdf`, letzter Zugriff: 29.08.07.

[5] *Wii Homebrew.* URL: `http://www.wiibrew.org/`, letzter Zugriff: 29.08.07.

[6] *Wii Linux.* URL: `http://www.wiili.org/`, letzter Zugriff: 29.08.07.

[7] Baum, L.E. und T. Petrie: *Statistical inference for probabilistic functions of finte state Markov chains.* In: *Annls of Mathematical Statistics*, Band 37, Seiten 1554–1563, 1966.

[8] Baum, L.E., T. Petrie, G. Soules und N. Weiss: *A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains.* In: *Annls of Mathematical Statistics*, Band 41, Seiten 164–171, 1970.

[9] Bayes, Thomas: *Thomas Bayes' Essay Towards Solving a Problem in the Doctrine of Chances.* Band 45, Seiten 296–315, 1958. Bayes' Essay aus dem Jahr 1763 in moderner Notation.

[10] Bilmes, J.: *A Gentle Tutorial on the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models*, 1998.

[11] Finlay, Janet und Russell Beale: *Neural networks and pattern recognition in human-computer interaction.* SIGCHI Bull., 25(2):25–35, 1993.

[12] Hofmann, Frank G., Peter Heyer und Günter Hommel: *Velocity Profile Based Recognition of Dynamic Gestures with Discrete Hidden Markov Models.* In: *Proceedings of the International Gesture Workshop on Gesture and Sign Language in Human-Computer Interaction*, Seiten 81–95, London, UK, 1998. Springer-Verlag.

[13] Hyeon-Kyu Lee, Jin Hyung Kim: *An HMM-Based Threshold Model Approach for Gesture Recognition.* IEEE Transactions on Pattern Analysis And Machine Intelligence, 21(10):961–973, 1999.

[14] Jiang, Xiaoyi: *Vorlesung Mustererkennung Folien*, 2006.

[15] Kela, Juha, Panu Korpipää, Jani Mäntyjärvi, Sanna Kallio, Giuseppe Savino, Luca Jozzo und Di Marca: *Accelerometer-based gesture control for a design environment.* Personal Ubiquitous Comput., 10(5):285–299, 2006.

[16] Lütticke, Tobias: *Gestenerkennung zur Anweisung eines mobilen Roboters.* Technische Universität Karlsruhe, Institut für Prozessrechentechnik, Automation und Robotik, 2000. Diplomarbeit.

[17] MacQueen, J. B.: *Some Methods for classification and Analysis of Multivariate Observations.* In: *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Band 1, Seiten 281–297, 1967.

[18] Markov, Andrei: *An Example of Statistical Investigation of the Text Eugene Onegin Concerning the Connection of Samples in Chains.* In: *Science in Context*, Seiten 591–600. Cambridge University Press, 2006. Übersetzt ins Englische von Gloria Custance und David Link.

[19] Mäntyjärvi, Jani, Juha Kela, Panu Korpipää und Sanna Kallio: *Enabling fast and effortless customisation in accelerometer based gesture interaction.* In: *MUM '04: Proceedings of the 3rd international conference on Mobile and ubiquitous multimedia*, Seiten 25–31, New York, NY, USA, 2004. ACM Press.

[20] Mäntylä, Vesa-Matti: *Discrete hidden Markov models with application to isolated user-dependent hand gesture recognition.* VTT Publications, 2001.

[21] Rabiner, L.R.: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* In: *Proceedings of the IEEE*, Band 77, Seiten 257–286, 1989.

[22] Rubine, Dean: *Specifying gestures by example.* In: *SIGGRAPH '91: Proceedings of the 18th annual conference on Computer graphics and interactive techniques*, Seiten 329–337, New York, NY, USA, 1991. ACM Press.

[23] Rubine, Dean Harris: *The automatic recognition of gestures.* Doktorarbeit, 1992.

[24] Torre, Fernando De la und Takeo Kanade: *Discriminative cluster analysis.* In: *ICML '06: Proceedings of the 23rd international conference on Machine learning*, Seiten 241–248, New York, NY, USA, 2006. ACM Press.

[25] USB Implementers Forum, Inc.: *Device Class Definition for Human Interface Devices (HID)*, 1.11 Auflage, June 2001. URL: `http://www.usb.org/developers/devclass_docs/HID1_11.pdf`, letzter Zugriff: 29.08.07.

Index