Endbericht der Projektgruppe 539

NextWii
- Multimodale Interaktion mit Computerspielen -

Teilnehmer

Alexander Nimtz, Daniel Warzecha, Finn Siebert, Helena Kotthaus, Jan Hölscher, Julian Dittrich, Kyrill Risto, Michael Mülle, Oskar Opalinski, Philipp Gerloff, Tristan Skudlik

Betreuer

Prof. Dr.-Ing. Gernot A. Fink,
Dr.-Ing. Thomas Plötz
Dipl.-Ing. Jan Richarz

technische universität
dortmund

Inhaltsverzeichnis

Autoren: Alexander Nimtz, Daniel Warzecha, Finn Siebert, Helena Kotthaus, Jan Hölscher, Julian Dittrich, Kyrill Risto, Michael Mülle, Oskar Opalinski, Philipp Gerloff, Tristan Skudlik
Erstellt am: 31. März 2010

1 Einleitung

Traditionell erfolgt die Kontrolle virtueller Avatare in Computerspielen über Joysticks, Gamepads, oder einfach mit der Maus oder Tastatur. Diese Möglichkeiten der Eingabe haben sich vielfach bewährt und sind noch immer die Methode der Wahl. Neben diesen klassischen Eingabemethoden gab es in den letzten Jahren Bestrebungen, alternative Steuerungsmethoden zu entwickeln. Die bekanntesten sind dabei das Sony Eye-Toy (siehe Abschnitt 2.5.2), die Nintendo Wiimote (siehe Abschnitt 2.5.3), aber auch die aus Guitar-Hero bekannte Gitarre. Diese neu entwickelten Eingabemethoden haben das Ziel, die Steuerung intuitiver zu gestalten, sollen aber auch eine bessere Möglichkeit bieten, den Spieler in das Geschehen zu integrieren - der Spieler selbst wird durch eine stärkere physische Involvierung Teil der Handlung.

Bisherige alternative Eingabemethoden sind oftmals durch strikte finanzielle Vorgaben beschränkt, d.h. die volle Bandbreite technischer Möglichkeiten wird nicht ausgeschöpft, da das Produkt bezahlbar bleiben soll. Diese Beschränkungen limitieren effektiv die Einsatzmöglichkeiten neuartiger, komplexerer Eingabemethoden. So lassen sich über das Sony Eye-Toy nur stark vereinfachte Spiele steuern und die von einer Wiimote erwarteten Handbewegungen sind oft wenig intuitiv und verlangen auch nicht vollen Einsatz vom Spieler. Der Wiimote genügt es, wenn die richtigen Bewegungen aus dem Handgelenk kommen. Es stellt sich unweigerlich die Frage, welche Eingabemethoden unter konsequenter Ausnutzung moderner Techniken realisierbar wären und wie weit der Spieler heute ins Spielgeschehen eintauchen kann. Diesen Fragen wollen wir in dieser Projektgruppe nachgehen, indem wir bekannte und effiziente Verfahren nutzen, um eine „Videoeingabe“ so zu bearbeiten und zu erkennen, dass daraus eine „Spieleingabe“ wird.

Der spielerische Rahmen der Projektgruppe, eine Spielsteuerung über visuell erfasste und digital verarbeitete Aktionsdaten zu erstellen, nimmt natürlich auch in anderen Bereichen, des alltäglichen Lebens eine wichtige Rolle ein. Die Aktionserkennung wird in großen Teilen im Bereich der semiautomatischen Überwachung von Polizei, Militär, aber auch anderen Sicherheitsdienstleistern eingesetzt. Für die Erkennung von Angriffen oder sich dynamisch verändernden Umgebungen reichen einfache Bewegungsmelder nicht mehr aus, sodass die Erkennung von Aktionen einen Mehrwert liefert. Der Fokus dieser Projektgruppe liegt jedoch in der Möglichkeit, mittels Kamera eine alternative Eingabemöglichkeit für ein Computerprogramm zu liefern. Neben der hier verwendeten Spielsteuerung sind jedoch auch Steuerungen denkbar, die auch in anderen Szenarien zum Einsatz kommen können. Diese könnten in Umgebungen eingesetzt werden, die keine klassischen Eingabegeräte, wie Maus oder Tastatur, zulassen. Dies ist für einige Szenarien eine Alternative. Viele mobile Geräte werden immer leistungsfähiger. Da gerade

diese im Lauf der Zeit ebenfalls noch kleiner werden, bleiben großflächige und raumfordernde Eingabemöglichkeiten, wie z.B. die besagte Tastatur, immer häufiger auf der Strecke. Schon heute werden auf berührungsempfindlichen Displays Aktionen über Gesten ausgeführt [2] [47]. Eine komplette Steuerung über die meist eingebaute Kamera würde den Bewegungsfreiraum für den Nutzer vergrößern können. Ebenso würden auch behinderte Menschen von einer alternativen Eingabemöglichkeit profitieren. Menschen, mit körperlichen Einschränkungen, nutzen bereits häufig eine Spracheingabe. Für Personen, die weder die Tastatur noch ein Sprachmodul benutzen können, wäre der Ansatz zur videobasierten Erkennung überaus geeignet.

1.1 Ziel

Das Ziel dieser Projektgruppe ist die Entwicklungen neuartiger, multimodaler Eingabemethoden für Computerspiele. Hierbei liegt der Fokus vor allem auf einer intuitiven und natürlichen Eingabe, welche ihren Einsatz in der nächsten Generation von Spielekonsolen finden könnte (siehe Abschnitt 2.5). Die Grundlage für diese Aufgabenstellung stellt dabei die Sensorumgebung der FINCA[5] dar. Die FINCA-Experimentalumgebung am Institut für Roboterforschung der technischen Universität Dortmund ist ein Besprechungsraum, welcher mit mehreren Kameras und Mikrofonen ausgestattet ist. Diese erlauben eine nahezu vollständige Erfassung der Geschehnisse innerhalb des Raums. Die vorhandene Sensorik soll genutzt werden, um eine Steuerung von Computerspielen mit direktem Körpereinsatz zu erreichen. Duckt sich der Spieler zum Beispiel in der realen Welt, so sollen sich seine Aktionen auf den virtuellen Avatar übertragen. Als Ersatz für den bei Computerspiele üblichen Controller soll die Wiimote dienen. Über diese können verschiedene, nicht einfach erkennbare oder auf engen Raum nicht realisierbare Aktionen abgebildet werden. Die Wiimote verfügt über weitere Sensoren, wie einen Beschleunigungssensor, deren Daten ebenfalls genutzt werden können.

Als mögliche Testspiele kommen dabei klassische sog. Jump'n'Run-Spiele oder Sportspiele in Frage (siehe Abschnitt 2.1). Durch die Verfügbarkeit zahlreicher Open-Source Spiele sind darüber hinaus weitere Alternativen zur Wii als Konsole machbar. Das Computerspiel wird dabei mit Hilfe eines Beamers an eine Leinwand projiziert.

Die inhaltlichen Schwerpunkte der Projektgruppe konzentrieren sich auf die folgenden Arbeitsgebiete:

- Entwicklung/Implementierung von Verfahren der Musterklassifikation zur Aktionserkennung in Videodaten.
- Einbindung der Wiimote für die Steuerung eines Computerspiels unter Linux.

- Eine sinnvolle Integration multimodaler Daten, um ausgeführte Handlungen auf einen virtuellen Avatar in einem Computerspiel zu übertragen.

1.2 Problemstellung

Für die Realisierung dieser Ziele gibt es eine große Menge von Herangehensweisen. Diese müssen im Vorfeld untersucht werden, sodass die für das Projekt geeignetsten selektiert werden. Alle Verfahren unterscheiden sich stark in Bezug auf für das Projekt wichtige Aspekte, wie *Geschwindigkeit*, *Erkennungsrate*, den *Anlernaufwand* und die *Anforderungen an die Hardware*.

Es wird ein Verfahren gesucht, das in der Lage ist, eine große Datenmenge in Echtzeit zu verarbeiten. Die Eingaben der Wiimote sollen ebenfalls verarbeitet werden und zur Steuerung eines Spiels beitragen. Diese Daten sind in Echtzeit in einer vereinheitlichten und auf das Zielspiel zugeschnitten Form zu verarbeiten. Somit sind folgende Probleme zu lösen:

- Die Datenmengen sollen gering gehalten werden.
- Daten bzw. Merkmale müssen analysiert werden und damit
- Eingaben für das Spiel erzeugt werden.
- Die Verwendung des Systems soll in Echtzeit erfolgen.

2 Stand der Technik

Um die in Kapitel 1 erarbeiteten Ziele und Aufgaben zu erfüllen, ist die Identifikation der verschiedenen Teilprobleme und möglicher Lösungsansätze von entscheidender Bedeutung. Neben der sorgfältigen Auswahl eines geeigneten Spiels steht hierbei vor allem die Erkennung von aus mehreren Posen zusammengesetzten Aktionen im Vordergrund. Dazu ist zunächst die genaue Lokalisierung des Bildbereichs, der die Person enthält, notwendig, um im weiteren Verlauf die dort eingenommene Pose zu identifizieren. Aufbauend auf einer Sequenz von identifizierten Posen ist es dann möglich, die ausgeführte Aktion zu erkennen und das Spiel entsprechend anzusteuern. Zur Lösung jedes dieser drei Teilprobleme existiert in der Literatur ein breites Spektrum von verschiedenen Ansätzen unterschiedlichster Qualität, über die der folgende Abschnitt eine Übersicht bietet. Darüber hinaus bietet sich ein Blick auf vergleichbare Konzepte zur Spielsteuerung an, um weitere Ideen zu sammeln und einen Maßstab für die Qualität der eigenen Ergebnisse zu erhalten.

2.1 Computerspiele

Computerspiele sind seit den 1970er Jahren eine immer schneller wachsende Industrie. In den letzten Jahren haben die Firmen der Spielehersteller mehr Umsatz als die Filmindustrie in Hollywood erwirtschaftet [30][54][14]. Die Zeiten, in denen sich primär Computerfreaks damit beschäftigten, sind längst vorbei. Es ist sogar so, dass der rasante Fortschritt im Bereich der Hardware zu einem nicht unbedeutendem Teil der Spielindustrie zu verdanken ist, die immer realistischere Spielwelten erschaffen will und dafür immer schnellere Prozessoren und Grafikkarten benötigt. Aktuelle Spiele haben fast fotorealistische Grafiken.

Seit einigen Jahren ist ein neuer Trend in der Spielentwicklung zu beobachten, welcher nicht nur auf immer realistischere Grafiken und filmreife Geschichten setzt sondern auf eine neuartige Einbindung des Spielers. Es geht insbesondere den Herstellern der Spielekonsolen immer öfter darum, dass Spiele nicht mehr mit einem einfachen Eingabegerät bedient werden, sondern der ganze Körper zum Einsatz kommt.

Es gibt bereits einige Ansätze welche dieses Ziel verfolgen (siehe Kapitel 2.5), doch bisher wird das entsprechende Spiel entweder durch Auswertung von Bilddaten oder durch Informationen aus Bewegungssensoren gesteuert. Der nächste logische Schritt ist es, diese beiden Techniken zu verbinden. Um das Spielgefühl realistischer zu gestalten, muss es durch die Bewegungen des Spielers gesteuert werden können. Durch videobasierte Aktionserkennung und zusätzlich Informationen durch Bewegungssensoren lässt

sich dies für fast alle nötigen Steuerungskommandos gut realisieren. Alle anderen können nach wie vor durch das Drücken von Tasten und/oder die Verwendung eines Joysticks (bzw. eines Steuerkreuzes) ausgelöst werden.

2.1.1 Erkennung von Bewegungen

Bevor man sich Gedanken über die Realisierung von Eingaben für ein Spiel machen kann, muss zuerst untersucht werden, welche Arten von Bewegungen die Spielfigur durchführen kann bzw. soll. Leider ist die Vielfalt der Spielekonzepte und somit auch der Handlungen von Spielfiguren genau so groß wie im realen Leben. Daher müssen in der Regel Einschränkungen bei der Realitätsnähe der vom Spieler ausführbaren Aktionen zur Steuerung der Spielfigur hingenommen werden - egal welcher Art die Eingaben für das Spiel sein sollen. Das wohl einfachste Beispiel für die Notwendigkeit einer solchen Einschränkung ist die Verwendung und der Transport von Gegenständen innerhalb der Spielwelt. Es wäre unmöglich, dieses den Spieler wie in der echten Welt erleben zu lassen, vor allem im Bezug auf die haptische Wahrnehmung. Diese ist in Computerspielen sicherlich ein interessantes Forschungsgebiet, wird aber hier nicht weiter behandelt.

Aufgrund der getroffenen Überlegungen ist es sinnvoll, sich bei der Steuerung des Spiels mit dem Körper auf die Erkennung von Bewegungsprimitiven zu beschränken. Hierfür eignen sich vor allem Aktionen wie Springen, Ducken, Werfen und Boxen.

2.1.2 Ähnlichkeiten in der Spielesteuerung

Die Steuerung einer Spielfigur mittels Gamepad (welches in der Regel nicht mehr als 15 Tasten aufweist) führt zwangsläufig dazu, dass die Anzahl der ausführbaren Aktionen stark eingeschränkt ist. Dieser Tatsache sind sich auch Spielentwickler bei der Festlegung der im Spiel ausführbaren Aktionen bewusst. Durch den eingeschränkten Aktionsumfang sind sich die meisten Spiele recht ähnlich, was die Handlungsmöglichkeiten der Protagonisten anbelangt. Diese Ähnlichkeit wird noch stärker, wenn man sich Spiele eines speziellen Genres ansieht. „Jump'n'Run"-Spiele, welche lange Zeit einen Großteil der auf dem Markt befindlichen Spiele ausmachten, beschränken sich in erster Linie auf das Springen und Laufen.

Diese Homogenität der Aktionen lässt sich bei praktisch allen Spielegenres beobachten und gilt in vielen Fällen sogar genreübergreifend. Dies bedeutet noch lange nicht, dass ein Spiel dadurch an Realitätsnähe verliert. Auch im echten Leben führt der Mensch im Verhältnis zu den vorhanden

Möglichkeiten nur wenige Aktionen besonders häufig aus. Diese Fakten spielen auch dem Entwickler neuer Eingabemethoden in die Hände. Eine für ein Spiel entwickelte neue Eingabemethode lässt sich leicht auf andere Spiele anwenden, mit nur wenigen oder gar ohne Modifikationen. Dadurch ist es nicht zwingend notwendig, für die Konzeption einer neuartigen Spielsteuerung auch ein neues Spiel zu entwickeln.

2.1.3 Gewalt in Computerspielen

Ein in den letzten Jahren immer häufiger in den Fokus der Aufmerksamkeit gerücktes Thema ist Gewalt in Computerspielen. Diverse Amokläufe an Schulen[1] haben den sog. „Ego Shootern“ einen sehr zweifelhaften Ruf verschafft. Obgleich Studien belegen[57][33][17], dass Gewalt in Computerspielen zu einer zumindest kurzzeitig gesteigerten Aggressivität des Spielers führen kann, fehlt der Nachweis, dass dieser Effekt zwangsläufig in gewalttätigen Ausbrüchen mündet. Nichts desto trotz ist es notwendig, sich der Tatsache bewusst zu sein, dass es Nachteile mit sich bringt, ein neues und innovatives Steuerungssystem für ein Spiel (bzw. Genre) zu entwickeln, welches derartig vorbelastet ist. Obgleich die Ich-Perspektive sehr reizvoll für die Umsetzung einer Ganzkörpersteuerung ist, sollte darauf geachtet werden, dass das verwendete Spiel kein Ego-Shooter ist, und auch nicht versehentlich damit in Verbindung gebracht werden kann. Aufgrund dessen ist es empfehlenswert bei der Präsentation einer neuen Spielsteuerung auf Spiele mit gewalttätigen Inhalten, vor allem in Bezug auf Schusswaffen, zu verzichten. Durch die breite Palette an gewaltfreien Spielen, viele davon insbesondere für Spielekonsolen, gibt es jedoch ausreichend alternativen zur Umsetzung neuer Konzepte.

2.1.4 In-Game-Perspektive

Das zentrale Ziel bei der Entwicklung neuartiger Steuerung für Computerspiele liegt darin, ein authentisches Spielgefühl zu erzeugen. Im Zuge dessen wird es auch erforderlich, sich mit der Perspektive, aus der ein Spieler das Geschehen betrachtet, auseinander zu setzen. Zudem muss auch die Art des Spieles zu dem intendierten Steuerungsmodus passen. Führt man diese Überlegungen weiter, wird schnell klar: ein Puzzlespiel wie Tetris oder ein Kartenspiel wie Solitär sind nicht sehr ergiebig, was die Möglichkeiten zur multimodalen Steuerung anbelangt. Auch 2D Jump'n'Runs oder vergleichbare Spiele bieten zwar unter Umständen ausreichend viele ausführbare Aktionen, um die Möglichkeiten einer multimodalen Eingabe ausnutzen zu können; allerdings führt dies nicht zwingend dazu, dass sich der Spieler auch mit der Spielfigur identifizieren kann. Auf den ersten Blick eignet sich das Genre der

Ego-Shooter nahezu ideal für die Umsetzung einer intuitiven Spielsteuerung, aus den oben erläuterten Gründen wird jedoch von der Verwendung dieser Spielkategorie Abstand genommen.

Wie man sieht, ist es gar nicht so einfach ein Spiel zu finden, welches die Anforderungen für ein authentisches Spielgefühl und die Eignung zur multimodalen Steuerung erfüllt. Nach diesen Vorüberlegungen sind gewaltfreie Jump'n'Run (-artige) Spiele gut geeignet, wenn die Perspektive, aus welcher der Spieler das Spiel steuert, ein gutes Spielgefühl vermittelt. Hier bieten vor allem die diversen, für Spielekonsolen verfügbaren, Titel für uns interessante dreidimensionale Umsetzungen. Einige davon ermöglichen, das Geschehen aus einer Perspektive hinter der Spielfigur zu betrachten (Third-Person-Perspektive).

2.1.5 Geeignete Spiele

Zur Entwicklung einer neuen Eingabemethode, ohne dabei ein passendes Spiel entwickeln zu müssen, ist es notwendig die neue Steuerung einfach in ein bereits existierendes Spiel integrieren zu können. Open-Source-Spiele scheinen auf den ersten Blick die perfekten Kandidaten dafür zu sein, da der Quellcode bei Bedarf angepasst werden kann. Unter Berücksichtigung der getroffenen Vorüberlegungen und der Durchsicht der verfügbaren Spiele[60] haben sich einige für eine nähere Betrachtung qualifiziert. Als erstes wurden all diejenigen verworfen, welche das Spielgeschehen aus einer ungünstigen Perspektive zeigen. Dies hat die Auswahlmöglichkeiten erheblich reduziert. Folgende Spiele haben sich bezüglich der Perspektive und Quelloffenheit qualifiziert:

Soulride Ein anspruchsvolles und realistisches Snowboard-Rennen. Die Aktionsmöglichkeiten sind hier allerdings nicht ausreichend, da sie sich primär auf das lenken nach links und rechts, sowie das springen beschränkt.

TuxRacer Soulride sehr ähnlich, mit dem Unterschied das hier ein Pinguin anstelle eines Menschen auf einem Snowbaord einen Berg herunter rutscht. Aus den gleichen Gründen wie SoulRide nicht verwendbar.

Space Plumber Ein absolut gewaltfreies Spiel mit First-Person-Perspektive. Die Aktionsmöglichkeiten sind aber auf laufen und das betätigen von Pumpen beschränkt.

YoFrankie [4] Ein 3D-Jump'n'Run-Spiel welches viele Aktionsmöglichkeiten bietet.

Da aber nicht nur die Perspektive, sondern auch die Anzahl der Aktionsmöglichkeiten eines Spiels zu beachten sind, bleibt bei genauerer Betrachtung

nur YoFrankie übrig. Alle anderen verfügen über eine zu geringe Aktionsvielfalt. YoFrankie befindet sich jedoch in einer frühen Entwicklungsphase. Obwohl es sonst alle genannten Anforderungen erfüllt, läuft es bei weitem nicht stabil genug und benötigt zu viel Rechenleistung.
Der Mangel an geeigneten Spielen kann jedoch ausgeglichen werden, wenn man einmal über den Tellerrand blickt. Spielekonsolen bieten eine breite Auswahl an Titeln, welche sich ideal eignen, um sie mit einer multimodalen Steuerung zu versehen. Dabei handelt es sich allerdings um kommerzielle Produkte, deren Quellcode nicht verfügbar ist. Diese Problematik lässt sich durch die Vielzahl gut funktionierender Emulatoren lösen. Diese sind im Allgemeinen quelloffen und somit ein idealer Ansatzpunkt für die Integration von neuen Eingabemethoden.

Hier bietet sich Mupen64 [39] an, da dieser schon seit längerer Zeit entwickelt wird und praktisch voll ausgereift ist. Zudem ist er Plattformunabhängig und benötigt, da er die Konsole Nintendo 64 (N64) aus dem Jahr 1999 emuliert, nur verhältnismäßig wenig Rechenleistung. Das N64 bietet eine Vielzahl von ideal geeigneten Spielen, welche das Spielgeschehen aus der Third-Person-Perspektive zeigen, und zudem noch über ausreichend Aktionsmöglichkeiten verfügen.

Als Spiel wurde Banjo Kazooie ausgewählt, da dieses sehr gut emuliert wird und besonders viele Aktionsmöglichkeiten bietet. Wie bereits beschrieben, sind die ausführbaren Aktionen solch artverwandter Spiele sehr ähnlich, so dass prinzipiell auch andere Titel zum Einsatz kommen können. Banjo Kazooie ist ein Spiel, in dem die beiden Protagonisten Banjo und Kazooie sich auf die Suche nach Banjos kleiner Schwester begeben, die von einer bösen Hexe entführt wurde. Der Hauptcharakter Banjo ist ein Bär, der in einem Rucksack seine Freundin Kazooie, einen Vogel, mit sich herumträgt. Banjo kann laufen, seine Richtung ändern, boxen, sich ducken, etwas werfen, springen, besonders hoch springen und eine Art Ausfallschritt durchführen. Mit der Hilfe von Kazooie können die beiden außerdem eine kurze Strecke fliegen, einen Sprungangriff vollziehen, auf den Boden stampfen und etwas schneller rennen.

2.2 Personendetektion

Der zentrale Aspekt unserer Zielanwendung ist die Erkennung von aus mehreren Posen zusammengesetzten Aktionen. Viele der bekannten Verfahren zur Posenerkennung lassen sich deutlich beschleunigen, wenn der relevante Bildabschnitt, der die Person enthält, bereits eingegrenzt wurde. Zudem sind Verfahren, die auf ganzen Bildern arbeiten, wesentlich fehleranfälliger. Darüber hinaus stellt die Eingrenzung der Position einzelner Personen einheitliche Eingabedaten für die weitere Verarbeitung sicher und ermöglicht

die Trennung von verschiedenen Akteuren. Der folgende Abschnitt stellt verschiedene gängigen Verfahren zur Detektion von Personen vor.

2.2.1 Verfahren zur Personenerkennung

Ein Ansatz, der zwar ursprünglich zur Gesichtserkennung entwickelt wurde, aber auch zur Erkennung von Personen erweitert werden könnte, wird in [46] vorgestellte. Es beschreibt ein einfaches und sehr schnelles Verfahren auf Basis eines *Sliding Window*.

Hierbei wird ein, der zu erwartenden Größe des Gesichts entsprechendes, *Detection Window* über das gesamte Bild bewegt und dabei an jeder Position ausgewertet, bis eine zuverlässige Detektion erfolgt ist. Zur Auswertung der einzelnen *Detection Window* wird dabei eine Kaskade von Detektoren verwendet. Die Konstruktion der einzelnen Detektoren orientieren sich an dem Konzept des *Boosting*[65] welches die Konstruktion eines guten Klassifikator aus mehreren schwachen Klassifikatoren beschreibt. Bei der hier vorgestellten Umsetzung treffen die einzelnen, für sich genommen wenig aussagekräftigen, schwachen Klassifikatoren ihre Entscheidung (Gesicht / kein Gesicht) anhand von flächenweise bestimmten Pixel-Differenzen innerhalb des Fensters. Aus den Ergebnissen der einzelnen schwachen Klassifikatoren wird im weiteren Verlauf ein gewichtetes Gesamtergebnis bestimmt, anhand dessen der zusammengesetzte Klassifikator seine Entscheidung trifft. Die Wahl, der in den einzelnen schwachen Klassifikatoren verwendeten Flächen und ihrer Gewichtung, erfolgt dabei anhand der Ergebnisse auf Trainingsdaten.

Die hohe Geschwindigkeit wird durch steigende Komplexität der Teildetektoren innerhalb der Kaskade erreicht. Wird in einem Fenster vom ersten Teildetektor ein Gesicht erkannt, wird es an den nächst komplexeren Detektor weitergeleitet. Dieses Vorgehen wiederholt sich, bis das Ende der Kaskade erreicht wird. Lehnt ein Teildetektor ein Fenster ab, wird es von der gesamten Kaskade abgelehnt. Da in der Regel die große Mehrheit der Fenster kein Gesicht enthält, wird der komplexere (und langsamere) Teil der Kaskade sehr selten durchlaufen. Der Grad der Komplexität der einzelnen Teildetektoren wird dabei durch die Anzahl der zu ihrer Konstruktion verwendeten schwachen Klassifikatoren bestimmt.

Insgesamt ist das Verfahren schnell und erzielt hohe Erkennungsraten. Gegen die Verwendung spricht jedoch, die Empfindlichkeit des Systems bei einer Änderung der Blickrichtung. Dies verträgt sich nicht mit der Zielanwendung.
Der Ansatz nach [15] ist trotz hoher Erkennungsraten von bis zu 90% und Echtzeitfähigkeit für die Zielanwendung nicht optimal, da eine aufwendige manuelle Anlernphase vorausgesetzt wird. Das Verfahren beruht auf der Haar-Wavlet Transformierung, um Bilder in Merkmalsvektoren zu verwan-

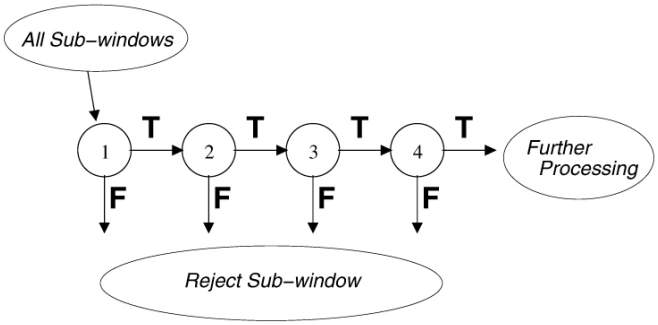

Abbildung 1: Eine Kaskade von Teildetektoren. Entnommen aus [46].

deln. Im Training werden die Bilder der zu erkennenden Objekte auf die gleiche Größe und Position gebracht und dann mit Haar-Wavlet transformiert. Eine *Support-Vektor-Machine* wird mit positiven Beispielen, also solchen die das Objekt enthalten sowie negativen Beispielen, solchen die das Objekt nicht enthalten, gefüttert. Im Erkennungsmodus wandert ein Slidingwindow fester Größe über das Bild. Jedes dieser Fenster wird mit der Haar-Wavlettransformierung in einen Merkmalsvektor transformiert und durch die SVM klassifiziert. Die Varianz bei der Größe wird durch mehrere Durchläufe mit veränderter Bildgröße bei gleich bleibendem Slidingwindow erreicht.

Ein weiterer, sehr interessanter Ansatz wurde in [40] vorgestellt und in [8] angewendet. Die Grundidee des Verfahrens basiert auf der Beobachtung, dass das Erscheinungsbild eines Objektes relativ gut durch die Verteilung der Kantenausrichtungen und Stärken charakterisiert wird. Dabei ist die konkrete Position der entsprechenden Kante von untergeordneter Bedeutung und muss nicht bekannt sein. Im Mittelpunkt des Verfahrens steht das *Histogram of Oriented Gradients* (HOG), welches die Häufigkeitsverteilung der verschiedenen Gradientenausrichtungen repräsentiert. Um dieses charakteristische Histogramm zur Personendetektion zu nutzen, greifen die Autoren auf das *Sliding Window* zurück. Hierbei wird für jedes der *Detection Windows* ein Vektor, der eine Reihe von, in diesem Fenster berechneten und lokal normalisierten, HOGs enthält, zusammengesetzt und an einen Klassifikator weitergeleitet. Der vorgestellte Ansatz zeichnet sich vor allem durch die äußerst aussagekräftigen Merkmalsvektoren aus, die es ermöglichen, die charakteristischen Umrisse von Menschen auch vor komplexen Hintergründen und unter schwierigen Lichtverhältnissen zu erkennen. Darüber hinaus liefert das Verfahren laut [40] in der Praxis herausragende Erkennungsraten. Leider ist die Berechnung des Merkmalsvektors relativ aufwendig, so dass Echtzeitfähigkeit bei Anwendung auf ein komplettes Bild nicht gegeben ist.

Die bisher vorgestellten Verfahren zur Personendetektion sind alle sehr rechenintensiv und führen dadurch zu einer schlechteren Reaktionszeit. Da in unserem Fall eine konstante Bewegung des Spielers angenommen werden kann, bietet sich ein alternativer Ansatz zur Eingrenzung des relevanten Bildausschnittes an, um die Vielzahl der *Detection Windows* auf wenige zu reduzieren. Da die Kameraposition fest ist, ist es möglich, ein Hintergrundmodell zu erstellen. Durch Einbeziehen des aktuellen Bildes in die Kalkulation des Modells können Regionen, in denen es hohe Abweichungen gibt, ermittelt werden. So ist es möglich, im Bild Abweichungen zum Hintergrund festzustellen und die Suche nach einer Person auf diesen Bereich zu beschränken. Im Folgenden wird dieser Bildausschnitt als *Region of Interest* (ROI) bezeichnet.

2.3 Posenerkennung

Nach der Anwendung der in Abschnitt 2.2 vorgestellten Verfahren ist eine Weiterverarbeitung des eingegrenzten, relevanten Bildausschnitts notwendig. Um geeignete, aus einer Sequenz von Posen bestehende, Eingabedaten für den Aktionserkenner zu generieren ist die eindeutige Identifikation der zu jeweils sehenden Pose wünschenswert. Hierzu gibt es in der Literatur bereits eine Reihe von Ansätzen über die die folgenden Abschnitte eine Übersicht liefern.

2.3.1 Verschiedene Verfahren zur Posenerkennung

Ein Ansatz zur Posenerkennung sind *Active Shape Models* mit denen in [58] und in [28] gearbeitet wird. Bei diesem Ansatz wird ein System mit den Umrissen von verschiedenen Objekten des gleichen Typs trainiert. Die Umrisse werden mit nummerierten Punkten belegt, die entweder wichtige Stellen der Silhouette markieren oder mit gleichem Abstand zwischen diesen wichtigen Punkten auf einer Linie verteilt liegen. Dabei befinden sich jeweils die Punkte mit gleicher Nummer bei allen Objekten des selben Typs an der gleichen Stelle, dies kann zum Beispiel der Mittelpunkt eines Auges oder die Ecke eines Quadrats sein. Aus diesen Trainingsdaten lassen sich so die verschiedenen möglichen Abweichungen für ein Objekt des Typs berechnen. Im Erkennungsmodus wird ein Objekt als Objekt des bestimmten Typs klassifiziert wenn es innerhalb der möglichen Abweichungen bleibt. Da das Training dieser Verfahren sehr Fehler anfällig ist ("If a Point is not in the correct position on each shape, the model will be unable to correctly represent the position of that point - it will include terms describing the noise caused by errors in point location." [58]) und die Berechnung in Echtzeit keine Rolle spielt, kommen sie für die Zielanwendung nicht in Frage.

Weitere Verfahren, die ebenfalls auf Silhouetten beruhen werden in [6] und in [41] vorgestellt. In [6] wird hierzu auf die Silhouette der Person ein 3D-Körper Modell projiziert, anhand dessen die Pose klassifiziert wird. Dem gegenüber wird in [41] die Silhouette mit gerichteten Rechtecken angenähert. Die Ausrichtung und Verteilung der Rechtecke wird dabei in Histogrammen gespeichert und in verschiedenen Posen klassifiziert. In beiden Papern werden keine Angaben zur Laufzeit gemacht weswegen sie sich nicht für die Endanwendung eignen.

Ein effizienter, echtzeitfähiger Ansatz ist die Posenerkennung mit Blobs, wie sie in [28] und in [7] verwendet werden. Hierbei wird im Bild nach zusammenhängenden Pixelgruppen, genannt *Blobs*, mit ähnlichen Eigenschaften wie Farbe oder Helligkeit, gesucht. Damit werden in der Regel Gesichter und Hände identifiziert. Anhand der Lage der Blobs zueinander läßt sich die Pose bestimmen. Beide Verfahren sind zwar echtzeitfähig, jedoch sehr fehleranfällig gegenüber Änderung des Lichts, des Hintergrunds oder der Hautfarbe. Ein Ansatz, der sowohl mit Blobs als auch mit Silhouetten arbeitet, wurde in [49] vorgestellt. Dieser ist aufgrund der darin verwendeten Blobs ebenfalls anfällig für Änderungen der Umgebungsbedingungen.

Neben den oben erläuterten Ansätzen lohnt ein erneuter Blick auf das bereits in Abschnitt 2.2.1 vorgestellte Verfahren der *Histogram of Oriented Gradients*. Obwohl es ursprünglich zur Personendetektion konzipiert wurde, lässt es sich aufgrund der aussagekräftigen Merkmalsvektoren, mit einem geeigneten Klassifikator auch zur Unterscheidung von verschiedenen Posen verwenden. Eine ähnliche Anwendung wurde bereits 2008 in [16] vorgestellt. Da die eingenommene Pose ebenfalls durch die Verteilung der Kantenausrichtung in den Merkmalsvektoren repräsentiert wird, ist zur Posenerkennung keine Anpassung des eigentlichen Verfahrens notwendig. Durch die Reduktion der Anzahl der möglichen *Detection Window* mit den in Abschnitt 4.4 vorgestellten Ansätzen ist es möglich, die Verarbeitungsgeschwindigkeit soweit zu steigern, dass eine Anwendung unter Echtzeitbedingungen ermöglicht wird. Aufgrund der aussagekräftigen Merkmalsvektoren und sehr guter Ergebnisse bei der Posenerkennung im Rahmen anfänglicher Tests mit verschiedenen Klassifikatoren wurde die Entscheidung zur Verwendung dieses Verfahrens bereits relativ früh getroffen.

2.4 Aktionserkennung

In der Aktionserkennung wird hauptsächlich mittels zweier Verfahrensweisen gearbeitet. Das *Template Matching* erstellt während des Trainings beliebig komplexe Schablonen von Konturen oder Bewegungsabläufen des Ursprungmaterials, also den Bildsequenzen. Um eine ausgeführte Aktion zu rekonstruieren, werden die aus der aktuellen Sequenz extrahierten Daten mit zuvor den

gespeicherten Schablonen verglichen. Mit einem Klassifikator wird die ähnlichste Schablone, und damit Aktion, gefunden. Der *State-Space*-Ansatz setzt die vorhergehende Erkennung von Posen auf einzelnen Bildern von Bildsequenzen voraus, ist also auf die Posenerkennung als Vorverarbeitungsschritt angewiesen. Als Eingabe für den Aktionserkenner werden die einzelnen Posenlabels übergeben.

2.4.1 Template Matching

Der Bereich des Template Matching umfasst verschiedene Ansätze und Herangehensweisen.

In [62] werden die sogenannten *Canonical Poses* verwendet, welche darauf beruhen, dass während der Ausführung einer Aktion zu bestimmten Zeitpunkten fünf Knoten- bzw. Gelenkpunkte des Körpers auf einer Ebene liegen. Mit diesen Knotenpunkten lassen sich Invarianten berechnen, mit deren Hilfe ein Bewegungsverlauf einer Aktion abgebildet werden kann. Dieser wird *Invariance Space Trajectory* (IST) genannt. Die Berechnung des IST erfolgt durch die Verwendung von jeweils zwei Gelenkpunkten vom Anfang und Ende einer Videosequenz. Ausserdem wird ein fünfter, sich in Bewegung befindlicher Punkt, als variable Komponente genutzt.

Ebenfalls mit Gelenkpunkten arbeitet der Ansatz in [52], wobei hier nicht mit Invarianten, sondern direkt mit dem Bewegungsverlauf der dafür relevanten Punkte (in diesem Fall Hände, Füße und Kopf) während einer Aktion gearbeitet.

Um die von beiden Verfahren benötigten Punkte zu finden, werden jedoch Motion-Capturing-Daten benötigt. Da ein Motion-Capturing sehr aufwendig ist, wurde von der Verwendung dieses Verfahrens abgesehen.

Statt Motion-Capturing wird in [9] eine Kombination aus Tracking und Hintergrundsubstraktion zur Berechnung der Konturen der Personen genutzt. Die Punkte einer Kontur werden den entsprechenden Punkten jeder zeitlich benachbarten Kontur zugeordnet. So entsteht ein dreidimensionales *Spatiotemporal Volume* (STV), welches dann auf verschiedene Oberfächenarten untersucht werden kann, wie z.B. *Peaks* und *Ridges*. Die Menge und Verteilung dieser Merkmale wird *Action Sketch* genannt. Dieser Action Sketch ist invariant gegenüber Blickwinkelveränderung.

Da der Vergleich zweier Aktionen mit dieser Methode erst nach jeweils vollständig berechneten Spatiotemporal Volumes und Action Sketches möglich ist, ist diese Methode nicht echtzeitfähig und somit nicht für die Zielanwendung geeignet.

In [29] wird der Aufbau des menschlichen visuellen Kortex als Inspiration für ein System, in dem Bildsequenzen in mehreren Stufen analysiert werden, verwendet. Die Laufzeit dieses Verfahrens ist allerdings recht lang.

Als Beispiel wird für eine Bildsequenz aus 50 Frames eine Dauer von zwei Minuten angegeben, bis die Aktion erkannt wird. Es wird prognostiziert, dass verschiedene Optimierungen signifikante Verbesserungen bringen könnten. Jedoch sind hierfür keinerlei Beweise in der Arbeit vorhanden.

Die *Motion-History Volumes* (MHV) nach [50] werden erzeugt, indem Aktionen aus mehreren Blickwinkeln gleichzeitig aufgenommen werden. Mit den Silhouetten, die auf den einzelnen Frames, beispielsweise durch Hintergrundsubtraktion, berechnet werden, kann die visuelle Hülle der Person geschätzt werden. Auf diesen werden mit einer Verfallsfunktion MHI's erstellt, welche den Verlauf der Bewegung während einer Aktion repräsentieren. Mit zunehmende zur letzten Bewegung verringert sich der einem Voxel zugeordnete Wert. Aufgrund der Notwendigkeit von mehreren Kameras bei der Aufnahme von Aktionen wurde dieser Ansatz zunächst nicht weiter betrachtet.

Der Vorgänger dieser Methode, die *Motion-History Images* (MHI) nach [32], findet bei der Bestimmung der ROI in Abschnitt 4.4 Verwendung.

2.4.2 State-Space

Wie schon in Abschnitt 2.3 erwähnt, werden in [16] HOGs auf einzelnen Posen berechnet, um Aktionen zu erkennen. Anstatt jeden einzelne Pose aus jedem Bild zu identifizieren, wird versucht sog. Schlüsselposen zu erkennen. Dies sind Posen, die eine Aktion besonders gut repräsentieren. Andere Posen fließen nicht in die Berechnung ein. Dies erhöht in einigen Fällen sogar die Erkennungsleistung.

Zur einfacheren Trennung von Vorder- und Hintergrund in der späteren Erkennung einer Aktion werden die HOGs zunächst mittels *nicht-negativer Matrix Faktorisierung* (NMF) in Basisvektoren und Koeffizienten aufgeteilt. Um den Vordergrund abgrenzen zu können, werden beim Training diese Basisvektoren auf einem Bild berechnet, welches nur den Hintergrund enthält. Bei der Posenklassifikation werden diese als Basisvektoren für das neue Bild verwendet und darauf die Koeffizienten neu berechnet. Diese *Posenprimitive* werden dann mit den gespeicherten Daten abgeglichen, um die Pose zu bestimmen. Für die finale Bestimmung einer Aktion werden die als Sequenz gespeicherten Posenprimitive normalisiert in einem Histogramm angeordnet. Die Gewichtung der Posen geschieht anhand ihrer Aussagekraft bezüglich bestimmter Aktionen. Um den temporalen Kontext zu berücksichtigen, wird das Histogramm dementsprechend erweitert, dass es statt einzelner Posenprimitive bi-Gramme enthält. Unter Verwendung der Kullback-Leibler-Divergenz werden die berechneten Daten mit den gespeicherten Histogrammen verglichen und die Aktion damit bestimmt. Der Fokus auf Echtzeitberechnungen und die Aktualität der Arbeit stellen einen großen Vorteil dieses

Ansatzes dar. Die angegebene Erkennungsleistung von bis zu 94,4% ist ebenfalls mehr als ausreichend für unseren Zielanwendung, so dass der Ansatz aus [16] ins System mit einfloss.

Auch in [24] werden nur Schlüsselposen für die Berechnung von Aktionen verwendet. Diese werden im Gegensatz zum vorherigen Ansatz jedoch als 3D-Modell gespeichert. Da eine Kamera aber nur zwei Dimensionen liefert, werden mehrere Blickwinkel miteinander verknüpft. Die Klassifikation geschieht mit einem sog. *Action Net*, einer Kaskadierung mehrerer *Hidden Markov Models* (HMMs). Die versteckten Zustände repräsentieren die erkannten Posen aus jeweils einem einzelnen Blickwinkel. Die zu erkennenden Posen werden durch die Emissionen dargestellt. Die Verknüpfungen zwischen den HMMs sind dort vorhanden, wo eine Aktion in die nächste übergehen kann. Somit wird sichergestellt, dass unsinnige Aktionsabfolgen nicht berechnet werden. Die Verknüpfungen werden außerdem kategorisiert: Es ist möglich eine Aktion zu wiederholen (*back-link*) oder von einer Aktion in die nächste zu gelangen (*inter-link*). Insbesondere existieren viele inter-links zwischen Aktionen aus ähnlichen Blickwinkeln. Der Einsatz von HMMs bringt den Nachteil mit sich, dass eine Aktion erst erkannt werden kann, sobald das komplette Modell durchlaufen wurde. Aufgrund der effizienten inkrementellen Dekodierung verbunden mit der bekannt guten Erkennungsleistung von HMMs wird dieser Ansatz jedoch als viel versprechend eingestuft.

2.5 Andere Ansätze

Die Grundidee, Spielsteuerung mittels Erkennung von Bewegungen und Aktionen als ein „Produkt“ anzubieten, das einer breiten Masse zur Verfügung steht, ist neu. Diverse Spielekonsolenhersteller haben derartige Produkte für das Jahr 2010 angekündigt ([37] und [66]), auf dem End-Verbraucher-Markt sind sie jedoch noch nicht erhältlich.

Verfahren aus dem Bereich der *Computer Vision* werden in vielen Szenarien eingesetzt, wobei Echtzeitfähigkeit in der Regel nicht die wichtigste Anforderung darstellt. Bei der Überwachung von Fließbändern, Qualitätskontrollen und der Erkennung von Personen fällt eine gewisse Verzögerung bis zum Abschluss der Berechnungen kaum ins Gewicht. So bewegt sich Gut X zum Beispiel auf dem Förderband noch Y Zeiteinheiten, bevor es den Sortierer erreicht und die Berechnung abgeschlossen sein muss. Dasselbe gilt für Überwachungspersonal, deren eigenen Reakationszeit in den meisten Fällen deutlich über der des “Personenerkenners” liegen drüfte. Die in vielen Verfahren fehlende Echtzeitfähigkeit machte es jedoch bisher unmöglich, Systeme in einem bezahlbaren Rahmen so zu gestalten, dass der Benutzer über seine Bewegungen mit dem System interagieren kann.

In den letzten Jahren wurden, durch intensivierte Forschung, neue Sensorik, bessere Erkennungsmodelle und nicht zuletzt deutlich mehr verfügbarer Rechenleistung, neue und verbesserte Verfahren mit wesentlich kürzeren Reaktionszeiten entwickelt. Daraus sind einige kommerzielle Anwendungen, gerade im Bereich der reaktiven Systeme, entstanden. Insbesondere gibt es neue Syteme, die zur Posendetektion und daraus ableitbaren Aktionserkennung dienen. Kombiniert mit der dazu gewonnen hohen Geschwindigkeit und geringen Latenz bei der Erkennung bietet dieser Ansatz, gerade für die Spieleindustrie, ein innovatives Verfahren zur Steuerung von Konsolen und anderen technischen Geräten [38]. Erste Ansätze, ohne die Erkennung von Ganzköperposen, gibt es bereits in der Spielkonsole Nintendo DSi, die die Emotionen im Gesicht des Spielers erkennt und auch Bewegungen vor der Kamera fordert [42], welche in diversen kleinen Minispielen zum Einsatz kommen. Auch liegen heute vielen WebCams Anwendungen bei, die z.B. das Gesicht immer zentrieren [36]. Darüber hinaus gibt es Fotoaparate, die erst auslösen, wenn die Person vor der Kamera lächelt [55], Autos die die Spur halten und sich am Seitenstreifen orientieren [10], Strassenschilder erkennen [11] oder bald Personen auf der Strasse orten [34] und bremsen. Größere und leistungsfähigere Produkte, die aufbauend auf einer schnellen Videoverarbeitung Personen im ganzen Erfassen und deren Aktionen erkennen, folgen langsam.

2.5.1 Projekt Natal (Microsoft)

Microsoft [37] hat für den Sommer 2010 auf der *Electronic Entertainment Expo (E3)* eine Zusatzeinheit für die Xbox 360 angekündigt, die es dem Spieler ermöglichen soll, das Spiel durch Bewegung und Sprache zu steuern. Die Einheit enthält eine Kamera mit einer Tiefensensortechnik und ein Mehrbereichsmikrofon. Damit können sehr genau 3D-Modelle der sich bewegenden Person erstellt werden. Laut vagen Informationen [22] verarbeitet das System mit einem 3D-Ganzkörpermodell, bestehend aus 48 Verbindungspunkten, 30 Bilder pro Sekunde [21]. Über das Mehrbereichsmikrofon werden Kommandos entgegengenommen [61], Personen identifiziert und ihre Stimmungen erkannt.

2.5.2 Motion Controller (Sony)

Sony geht einen anderen Weg als Microsoft mit Projekt Natal. Es werden keine Ganzkörpermodelle genutzt, sondern Marker, die der Spieler in den Händen[66] hält. Diese Marker sind für die Kamera deutlich zu erkennen und erfassen außerdem deutlich präziser die Bewegung des Spielers.Durch die bereits erhältliche Kamera EyeToy [56] werden zusätzliche Raumkoordinaten

ermittelt, die es ermöglichen, sehr kleine Bewegungen genau zu erfassen. Bei diesem Konzept fließen in die Bewertung einer Aktion Sensor- und Videodaten ein. Ein fertiges Produkt ist für den Herbst 2010 angekündigt.

2.5.3 Wii Controller - MotionPlus (Nintendo)

Auch die Wii wird mit neuen Sensoren präziser [43] und damit noch besser nutzbar. Die Auflösung [63] des MotionPlus soll weitere Erhebungen von 3D Informationen, wie z.B. über eine Kamera unnötig machen [59] und dem Spieler eine bessere Interaktion ermöglichen. Trotz des Verzichts auf Videodaten ermöglicht die Wii die Integration von anderen Sensoren z.B. einem Trittbrett, einer Tanzmatte, diverse Instrumente usw., in die Steuerung. Hierbei ist jedoch zu beachten, dass es sich im Grunde lediglich um normale Controller in einer anderen Hülle handelt.

2.5.4 CamSpace (CamTrax Technologies)

Die Software [18] der Firma CamSpace ermöglicht eine Erkennung von Gegenständen, die in einem Spiel als Controller genutzt werden können. Nach einer Initialisierung, in der der Gegenstand in seiner Form und Farbe erfasst wird, werden laufend Positionsänderungen analysiert (*Motiontracking*) und so als Eingabe für ein beliebiges Spiel genutzt. Die Software unterstützt beliebige Video-Kameras als Eingangssignal und läuft unter Windows, Linux und Mac.

3 Grundlagen der verwendeten Verfahren

Da das spätere System aus mehreren Modulen besteht, die jeweils anfallende Teilprobleme lösen, sollen die Grundlagen verwendeter Verfahren für die Lösung hier erklärt werden. Die Aufteilung gestaltet sich entsprechend der Struktur des späteren Systems in die Teile Merkmalsextraktion und Hintergrundmodelle zur Eingrenzung zu verwendender Daten, sowie die eigentliche Klassifikation der Posen und schließlich der Aktionen.

3.1 Grundlagen der Merkmalsextraktion

Bevor im weiteren Verlauf der genaue Ablauf der Verfahren zur Bestimmung der Pose vorgestellt wird, sei etwas ausführlicher auf das Konzept der Merkmalsextraktion in Kombination mit einem *sliding window* verwiesen. Es ermöglicht die Datenmenge vor der weiteren Verarbeitung auf wesentliche Merkmale zu reduzieren.

3.1.1 Sliding Window

Grundlage dieses Ansatzes, der z. B. in [40] Anwendung findet, ist das *detection window*, welches der erwarteten Größe der Person entspricht und über das gesamte Bild bewegt wird. Dabei wird an jeder Position das entsprechende Detektionsverfahren angewendet und ausgewertet. Das *detection window* wird dabei solange über das Bild bewegt, bis eine hinreichend zuverlässige Detektion erfolgt ist oder alle möglichen Positionen abgearbeitet wurden. Die Geschwindigkeit vieler Verfahren wird durch die hohe Zahl der auszuwertenden *detection window* limitiert, so dass eine vorherige Einschränkung des relevanten Bildausschnitts und der damit einhergehenden Reduktion der zu betrachtenden Fenster eine weitere Geschwindigkeitssteigerung ermöglicht.

3.1.2 Merkmalsextraktion

Da die direkte Verarbeitung von Sensormessungen (in unserem Fall Bilder) aufgrund der hohen Datenmenge und komplexen Strukturen eine praktisch unlösbare Komplexität aufweist, stellt die Abstraktion von den Eingabedaten einen entscheidenden Verarbeitungsschritt da. In der Praxis der Bildverarbeitung findet das Konzept der Merkmalsextraktion häufig in Kombination mit dem oben erläuterten *sliding window* Verwendung. Der zentrale Ansatz basiert dabei auf der Reduktion der vorhandenen Datenmenge durch Extraktion von relevanten Informationen. Dazu wird in jedem *Detection Win-*

*dow*eine, durch das konkrete Verfahren definierte und in der Regel relativ kleine, Anzahl von Merkmalen berechnet und zu einem Merkmalsvektor zusammengefasst. Dieser repräsentiert die relevanten Informationen des aktuellen Fensters. Der Vektor wird danach mithilfe eines Klassifikators einer Klasse von bekannten Vektoren, in unserem Fall den verschiedenen Posen, zugeordnet.

3.2 Hintergrundmodelle

Um eine hohe Reaktionszeit des Systems gewährleisten zu können, ist es wichtig relevante Bildausschnitte zu ermitteln und weitere Berechnungen auf diese zu reduzieren. Hierfür bieten sich Verfahren an, Hintergrundmodelle des betrachteten Kamerabildes zu erstellen und mit ihrer Hilfe in jedem Bild Bereiche zu ermitteln, in denen eine hohe Abweichung zum Hintergrund festgestellt wurde.

Im Wesentlichen kann zwischen einem statischen und einem dynamischen Modell unterscheiden werden. Bei dem statischen Modell wird ein einzelnes Bild verwendet, um den Hintergrund zu modellieren. Anschließend werden neue Bilder durch unterschiedliche Verfahren mit dem Hintergrund verglichen. Die so ermittelten Unterschiede ermöglichen es, den Vordergrund vom Hintergrund zu trennen. Da dieser Ansatz ein festes Modell des Hintergrunds verwendet, ist er sehr empfindlich gegenüber dort auftretenden Variationen. Wird am Hintergrund etwas geändert, wird der Bereich der Änderungen solange dem Vordergrund zugeordnet, bis das Modell mit der entsprechenden Änderung neu erstellt wird. Ein dynamisches Hintergrundmodell hingegen ist anpassungsfähig und wird nicht nur durch ein einzelnes Bild erstellt. Stattdessen wird ein zeitliches Fenster über die Bildersequenz betrachtet und zu einem neuen Modell verrechnet. Wie beim statischen Modell, wird anschließend das aktuellste Bild mit dem Modell verglichen, um Unterschiede zu ermitteln.

3.2.1 Motion History Images

Das in [20] vorgestellte *Motion-History-Image* repäsentiert den zeitlichen Verlauf der Veränderungen in einer Sequenz von Bildern. Es berechnet sich über die folgende Gleichung:

$$MHI\delta(x,y) = \begin{cases} \tau & \text{if } \psi(I(x,y)) \neq 0 \\ 0 & \text{else if } MHI\delta(x,y) < \tau - \delta \end{cases}$$

Dabei erhält jeder Pixel (x, y) im MHI den Zeitstempel τ, falls die Funktion $\psi(I(x,y))$ eine Bewegung im Bild ermittelt hat. Diese Funktion ist beliebig wählbar und ermöglicht ein hohes Maß an flexibilität. Es bietet sich

beispielsweise eine Hintergrundsubtraktion an, bei der Abweichungen um einen gewissen Schwellwert tolleriert werden. Pixel, für die keine Veränderung errechnet wurde und deren letzte Änderung über die Verfallszeit $(\tau - \delta)$ hinausgeht, werden auf 0 gesetzt. Das Zeitfenster, innerhalb dessen das *MHI* erzeugt wird, kann durch den Parameter δ beliebig variiert werden. Ein Wert von 0 bedeutet, dass keinerlei Dynamik im Modell stattfinden kann, wodurch es statisch wird und nur Veränderungen des aktuellen Bildes zum vorher zur Modellierung des Hintergrunds genutzten Bild darstellt. Größere Werte machen die Anpassung weniger reaktiv, erlauben es aber ein größeres Zeitfenster zu analysieren. Betrachtet man ein auf diese Weise berechnetes *MHI*, so kann jeder Pixel, für den gilt $MHI\delta(x, y) \neq 0$ dem Vordergrund des Bildes zugeordnet werden, während jeder andere dem Hintergrund zugeordnet wird. Das *MHI* kann als Graustufenbild dargestellt werden, in dem die Helligkeit jedes Pixels die Dauer des Intervalls ohne Änderungen repräsentiert. Der Verfall der ermittelten Zugehörigkeit muss keinesfalls zeitlich linear sein, es ist möglich eine beliebige Funktion zur Aktualisierung des Wertes zu nutzen. So lässt sich der Verlauf, in dem ein Wert abnimmt, beispielsweise logistisch oder asymptotisch umsetzen. Je nach Einsatzgebiet haben unterschiedliche Parameter Vor- und Nachteile. So kann ein statisches Hintergrundmodell auf kleine Veränderungen, beispielsweise der Beleuchtung, nicht reagieren. Ein dynamisches Hintergrundmodell hingegen bedarf ständiger Bewegung, da ansonsten die Person im Vordergrund mit in das Modell einfließt und nicht mehr unterschieden wird.

3.2.2 Approximierter Median

Für dieses Verfahren wird eine feste Anzahl der vergangenen Bilder zwischengespeichert. Dabei wird für jedes neue Bild jeder Pixel mit dem Median der korrespondierenden Pixel aller Bilder in dem Zeitfenster vergleichen. Überschreitet die Abweichung einen Schwellwert, so folgt daraus, dass der Pixel nicht zum Hintergrund gehört. Um den hohen Platzbedarf zur Speicherung der vergangenen Bilder zu reduizieren, wurde in [45] ein Verfahren entwickelt, welches es ermöglicht, den Median zu approximieren. Hierfür wird jeder Pixel im aktuellen Bild mit dem entsprechenden vom Hintergrundmodell verglichen. Ist der aktuelle Wert größer, wird der Wert im Hintergrundmodell um 1 erhöht. Falls er kleiner ist, wird der Pixel im Hintergrundmodell um 1 verringert. Im Schnitt wird ungefähr die Hälfte aller Vergleiche kleiner, die andere größer ausfallen, womit der Median approximiert wäre. Als Ergebnis liefert das Verfahren ein Graustufenbild, welches dem approximierten Hintergrund entspricht. Durch Vergleich mit dem aktuellen Bild, kann ein binäres Bild errechnet werden, welches das Bild in Vorder- und Hintergrund einteilt.

3.2.3 Mixture of Gaussians

Das nachfolgende Verfahren ist wesentlich komplexer, als der approximierte Median, erlaubt allerdings auch eine größere Flexibilität bei Veränderungen und führt zu einem besseren Gesamtergebnis. Anstelle feste Pixelwerte für den Hintergrund anzunehmen, ist das gesamte Modell parametrisiert. Jedem Pixel wird eine feste Anzahl an Gaußfunktionen zugeordnet, die in ihrer Summe eine Wahrscheinlichkeitsverteilungsfunktion F um mögliche Farbwerte bilden.

$$F(i_t = \mu) = \sum_{i=1}^{k} \omega_{i,t} \cdot \eta(\mu, \sigma)$$

Der Erwartungswert μ jeder der Normalfunktionen kann als Schätzung des Pixelwertes im nächsten Bild interpretiert werden, während die Gewichtung und Standardabweichung von der Konfidenz abhängig sind. Je höher die Konfidenz, desto höher wird die Gewichtung ω und geringer die Standardabweichung σ. In einem neuen Bild wird nun für jeden Pixel sein Wert mit den Erwartungswerten des entsprechenden Hintergrundmodells verglichen. Befindet sich dieser nach einer Gesamtbewertung innerhalb eines Schwellwertes, wird er zum Hintergrund gezählt, andernfalls zum Vordergrund. Anschließend werden die Funktionen aktualisiert. Zählt der Pixel zum Hintergrund, wird die Gewichtung der entsprechenden Funktion erhöht und sie so angepasst, dass der Erwartungswert in Richtung des Pixelwertes verschoben wird und σ reduziert. Andernfalls wird lediglich die Gewichtung exponentiell reduziert.

Als Ergebnis liefert das Verfahren zu jedem Pixel eine Zuordnung, welche Vordergrund oder Hintergrund unterscheidet. Ein großer Vorteil des Verfahrens ist es, jedem Pixel eine Vielzahl von möglichen Werten zu gewähren, um dennoch zum Hintergrund gerechnet zu werden. Betrachtet man ein sich bewegendes Blatt vor dem Himmel, kann dieser Bildausschnitt grün oder blau sein. In beiden Fällen sollte er jedoch zum Hintergrund gezählt werden. Auf Grund der Komplexität des Verfahrens, jede Funktion zu berechnen und die Parameter anzupassen, ist es jedoch sehr rechenlastig.

3.2.4 ROI Tree

Nachdem durch eines der bisher vorgestellten Verfahren ermittelt wurde, welche Pixel zum Vordergrund und welche zum Hintergrund des Bildes zählen, ist es notwendig, die dort lokalisierten Veränderungen weiter zu analysieren und zusammenzufügen, um daraus den relevanten Bildausschnitt zu berechnen. Dies kann effizient und schnell mit Hilfe des in [31] entwickelten *ROI Tree* umgesetzt werden, welcher eine Spezialisierung des *Quadtree* ist. Es werden jedoch keine Daten des Bildes, sondern lediglich die Koordinaten des zu

einem Knoten gehörenden Bildausschnittes gespeichert. Der *ROI Tree* unterteilt alle Dimensionen eines Bildes rekursiv gleichmäßig, bis ein Schwellwert für die Mindestgröße oder die Größe eines einzelnen Pixels erreicht wurde. Zur Auswertung muss zunächst ein *Integral Image* des aktuellen Bildes berechnet werden. Darin wird gemäß der Formel

$$sat(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y')$$

zu jedem Punkt (x, y) die Summe aller Pixel links und oberhalb, inklusive des betrachteten Pixels berechnet. Wird das *Integral Image* an den *ROI Tree* übergeben, kann dieser von der Wurzel aus in alle Verzweigungen hinab traversiert werden. Bei jedem Kindknoten muss die Summe der in ihm enthaltenen Pixel neu ermittelt werden und das *Integralimage* ermöglicht dies in linearer Zeit. Es müssen lediglich die Werte an den Eckpunkten betrachtet werden. So kann die Menge enthaltener Pixel bestimmt werden, welche vom entsprechenden Hintergrundmodell zum Vordergrund gehörend eingestuft wurden. Übersteigt diese Menge einen festgelegten Schwellwert, ist es nicht mehr nötig, die Kinder dieses Knotens weiter zu betrachten. Der Knoten, beziehungsweise die Region, die er repräsentiert wird vermerkt und im nächsten Schritt alle benachbarten Regionen verschmolzen. Das Resultat ist ein Baum, in dessen Knoten alle Koordinaten von Bildbereichen, die dem Vordergrund zugeordnet wurden, gespeichert sind. Werden die Extremwerte aller Koordinaten betrachtet, kann eine große Region berechnet werden, die den relevanten Bildausschnitt beschreibt.

3.3 Posenerkennung

Aufgrund der bereits in Abschnitt 2.3 besprochenen Vorteile des Verfahrens nach [40] stellt der Ansatz der *Histograms of Oriented Gradients* eine gute Grundlage für den Posenerkenner dar. Zur Klassifikation der Merkmalsvektoren bietet sich die Verwendung eines künstlichen neuronalen Netzes an, da dieses sich gut für die Zuordnung der Vektoren in viele verschiedene Klassen eignet. Der folgende Abschnitt beschreibt die beiden Konzepte im Detail.

3.3.1 Histograms of Oriented Gradients

Neben der bereits erläuterten Grundidee, Objekte anhand der Verteilung von Kantenausrichtungen und -stärken zu charakterisieren basiert der entscheidende neue Ansatz des vorgestellten Verfahrens auf der Aufteilung des aktuellen *Detection Window* in mehrere Zellen und ihrer Normalisierung. Dazu wird für jede Zelle jeweils ein lokaler Merkmalsvektor bestehend aus den

einzelnen Kanälen des HOGs der Zelle berechnet und lokal normalisiert. Im folgenden werden die wichtigsten Schritte des Verfahrens im Detail erläutert.

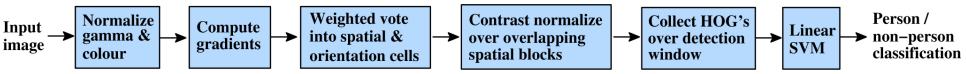

Abbildung 2: Übersicht über den Ablauf der Merkmalsextraktion. Entnommen aus [40].

Schritt 1 - Gamma-Kompression

Im ersten Schritt der Merkmalsextraktion wird eine Gamma-Kompression des Bildes gemäß der Formel $s = cr^{\gamma}$, wobei r dem original Pixel-Wert und s dem Ergebnis Pixel-Wert entspricht, durchgeführt. Aufgrund der später vorgenommenen lokalen Normalisierung hat die Gamma-Kompression jedoch nur geringe Auswirkungen auf die Ergebnisqualität.

Schritt 2 - Gradienten-Berechnung

Zur Berechnung der Gradienten stehen eine Reihe von Verfahren zur Verfügung. Üblicherweise erfolgt die Berechnung näherungsweise durch die Anwendung von Filter-Masken. Hierbei wird die Maske Pixel für Pixel durch das Bild bewegt, der Ergebniswert für jeden Pixel ergibt sich dabei durch die von der Maske gewichteten Summe der Nachbar Pixel. Abbildung 3 veranschaulicht die Vorgehensweise. Dabei gilt als Ergebnis der Anwendung für jeden Pixel

$$g(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

$$\text{mit } a = (m-1)/2, b = (m-1)/2$$

wobei f(x,y) das Original Bild, g(x,y) das Ergebnis der Anwendung und w(x,y) die Filter-Maske der Dimension $m \times m$ darstellt. Bei den gesuchten Gradienten handelt es sich um zwei dimensionale Vektoren, wobei der Wert jeder Komponente separat aproximiert wird. Hierzu wird auf jeden Pixel je eine Maske für die X- und eine für die Y-Komponente der Gradienten-Vektoren angewendet. Das Ergebnis der Gradienten Berechnung ist in Abbildung 4 beispielhaft zu sehen.

Schritt 3 - Zellen und Histogramm Berechnung

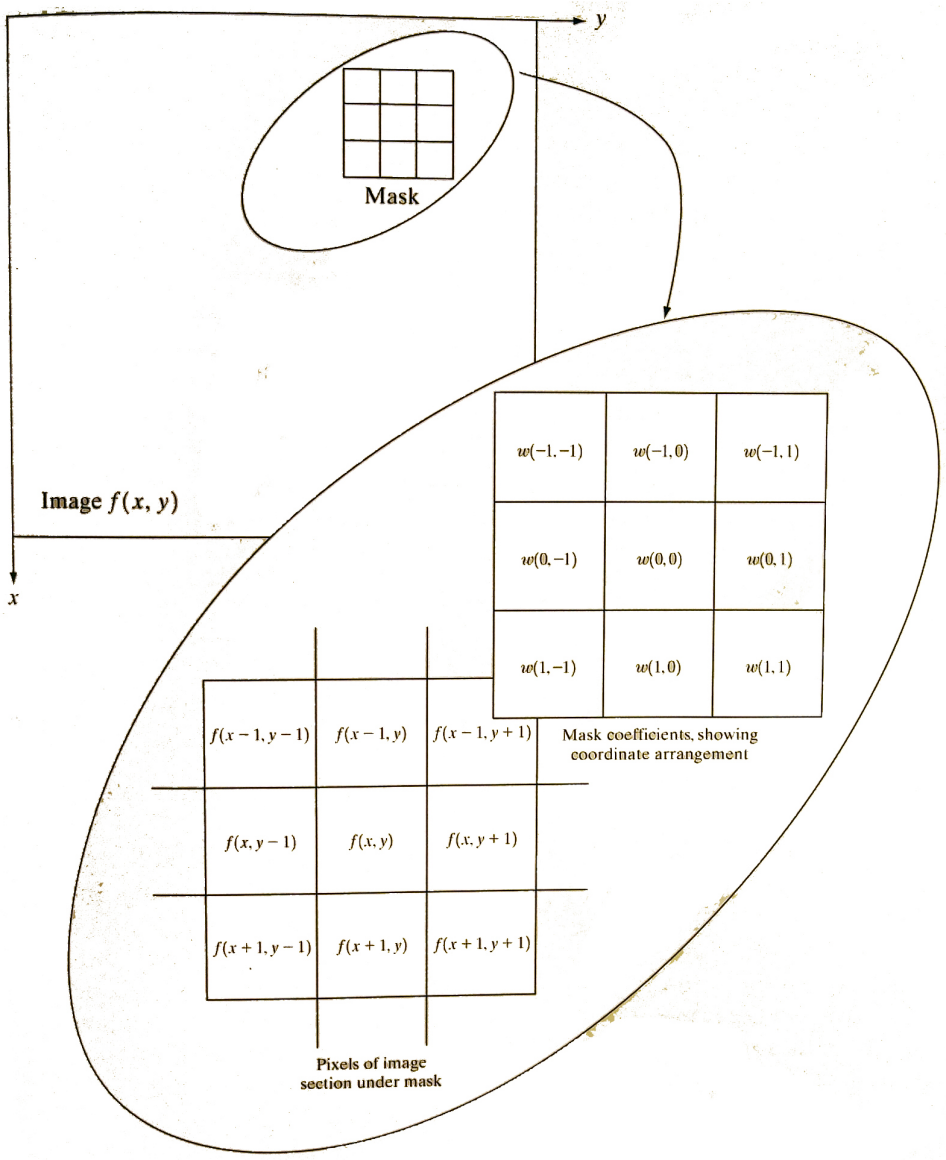

Abbildung 3: Beispiel der Masken-Anwendung. Entnommen aus [48].

Im dritten Schritt der Merkmalsextraktion wird das Detection Window in mehrere gleich große Zellen eingeteilt. Danach wird für jede Zelle ein *Histogram of Oriented Gradients* berechnet. Das Histogramm repräsentiert dabei die Verteilung der Gradientenausrichtungen in der jeweiligen Zelle. Hierbei entsprechen die einzelnen Kanäle des Histogramms den möglichen Ausrichtungsbereichen der Gradienten. Die Anzahl der Kanäle bestimmt damit die Größe der einzelnen Ausrichtungsbereiche (z. B. $0° - 20°, 20° - 40°$, ... oder $0° - 40°, 40° - 80°, ...$) und ist prinzipiell frei wählbar. Zur Berechnung des

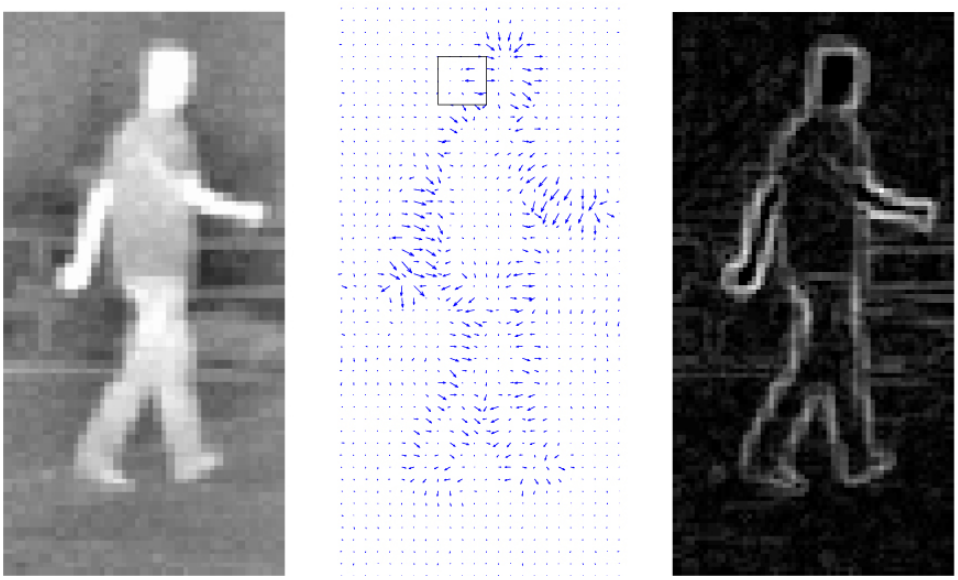

Abbildung 4: Ergebnis der Gradientenberechnung. Entnommen aus [23].

Histograms für jede Zelle werden die einzelnen Gradienten in die vorher definierten Ausrichtungsbereiche eingeteilt und ihre Beträge in den jeweiligen Kanälen aufsummiert.

Schritt 4 - Blockweise Normalisierung

Da die Beträge der einzelnen Gradienten aufgrund lokaler Unterschiede in Beleuchtung, Schatten und Vorder-Hintergrund-Kontrast relativ stark varieren, führt eine lokale Normalisierung der HOGs zu einer deutlichen Verbesserung der Ergebnisse. Um diese Normalisierung durchzuführen werden mehrere Zellen zu überlappenden Blöcken zusammengefasst und blockweise mittels L1- oder L2-Norm normalisiert. Der Merkmalsvektor eines Blocks ergibt sich aus der Konkatenation der normalisierten Vektoren der Zellen des Blocks.

Schritt 5 - Berechnung des finalen Merkmalsvektors

Der globale Ergebnisvektor des Fensters ergibt sich aus der Konkatenation der Merkmalsvektoren der einzelnen Blöcke. Da die Blöcke überlappend gewählt werden, geht ein und die selbe Zelle mehrfach mit unterschiedlichen Normalisierungsfaktoren in den Ergebnisvektor ein. Diese Redundanz trägt mit zu den guten Ergebnissen des Verfahrens bei. Nach diesem Verar-

beitungsschritt steht der Merkmalsvektor zur weiteren Verarbeitung durch einen Klassifikator bereit.

3.3.2 Künstliches Neuronales Netz

Künstliche Neuronale Netze (KNN) kommen aus dem Bereich der künstlichen Intelligenz und bieten sich zur Klassifizierung der beschriebenen Merkmalsvektoren an. Das Prinzip ist hierbei vergleichbar mit dem neuronalen Netzwerk eines menschlichen Gehirnes. Ein einzelnes *Neuron* kann dabei mehrere Eingänge, als auch mehrere Ausgänge besitzen. Die Werte der Eingänge werden gewichtet und an die *Übertragsfunktion* übergeben. Die von der Übertragsfunktion berechnete Summe der gewichteten Eingangswerte wird als *Netzeingabe* bezeichnet. Die Aktivierungsfunktion entscheidet ob ein Neuron aktiv ist. Es gibt verschiedene Typen von Aktivierungsfunktionen. Die einfachste ist die sogenannte Schwellenwertfunktion, bei der das Neuron nur den binären Wert 1 oder 0 annehmen kann. Das Neuron wird als 1 oder „on“ betrachtet, wenn die Netzeingabe einen Schwellwert übersteigt. Solch ein Neuron wird auch McCulloch-Pitts-Zelle [64] genannt. Aktivierungsfunktionen, die eine Lineare oder Sigmoidfunktionen [53] nutzen, bilden die Netzeingabe auf einen Wert zwischen 0 und 1 ab. Dieser Wert ist die Ausgabeaktivierung des Neurons.

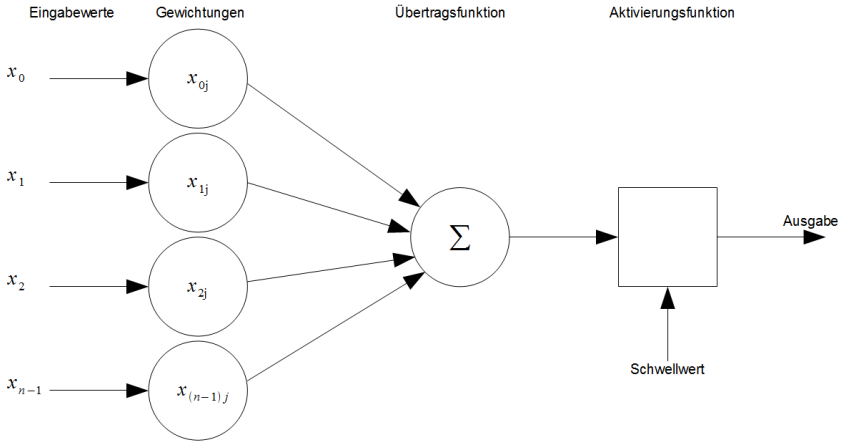

Abbildung 5: Aufbau eines Neurons nach Rosenblatt [51]

Ein KNN ist eine lineare Verknüpfung solcher Neuronen. Solch ein Netz besteht dabei aus mindestens zwei Schichten. In der ersten Schicht befinden sich die Eingangsneuronen, welche in ihrer Anzahl der Menge der Elemente des zu klassifizierenden Merkmalsvektor entsprechen. Die Neuronen der letzten Schicht werden als Ausgabeneuronen bezeichnet, die das Ergebnis des KNN repräsentieren. Zwischen der Ein- und Ausgabeschicht liegen ver-

steckten Schichten. Die optimale Anzahl der versteckten Schichten kann je nach Problem variieren, denn diese sind dafür verantwortlich, dass die zu klassifizierenden Merkmale auf die gewünschte Ausgabe abgebildet werden. Ein Netz wird mit Hilfe einer *Backpropagation* Variante trainiert[12]. Es werden Trainingsdaten in das Netz eingegeben, von denen bekannt ist, welches Ergebnis sie repräsentieren. Der Unterschied zwischen realem und erwartetem Ergebnis wird als Fehler des Netzes interpretiert. Dieser wird rückwärts durch das Netz propagiert und es werden die Gewichtungen der relevanten Neuronen angepasst. Somit wird bei erneuter Eingabe der Daten der Fehler geringer. Für die Klassifikation durchlaufen die zu klassifizierenden Daten das trainierte Netz. Das am stärksten aktivierte Neuron der Ausgabeschicht repräsentiert die Zuordnung.

Obwohl das vorgestellte Verfahren für die Klassifikation verschiedener Daten genutzt werden kann, wird es hier nur für die Posenerkennung eingesetzt. Die zu den Eingabedaten zugeordneten Posen können von einem Aktionserkenner weiterverabeitet werden.

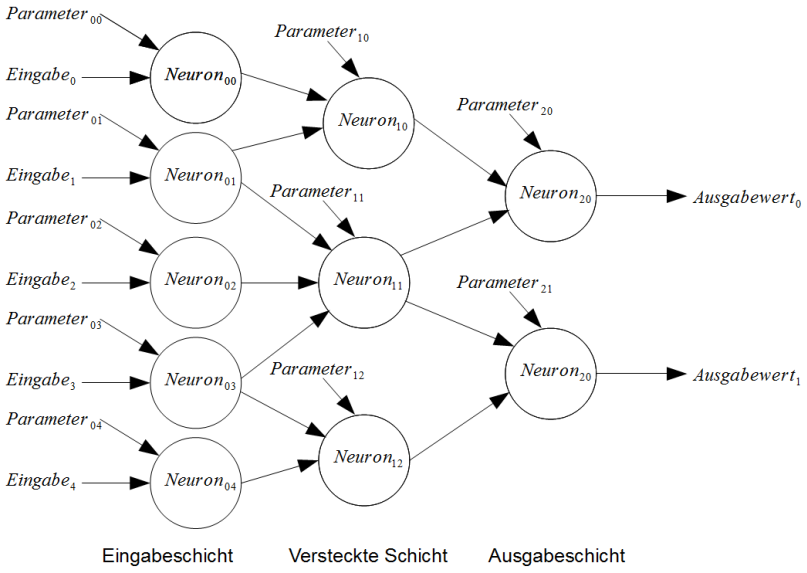

Abbildung 6: Aufbau eines KNN

3.4 Aktionserkennung

Nach dem Entschluss (Abschnitt 2.4.2) sich sowohl auf den Ansatz in [16] als auch auf Sprachmodelle zu fixieren, sollen einige Grundlagen der Aktionserkennung erläutert werden. Es folgen die Erläuterung zweier Klassifikatoren und die Beschreibung eines Verfahrens, mit dem es möglich wird, den zeitlichen Kontext eines Merkmalsvektors in die Klassifikation mit einzubeziehen.

Obwohl noch viele weitere Verfahren existieren, sind die genannten für den Anwendungsfall besonders nützlich.

3.4.1 k-Nearest-Neighbor

Bei dem *k-Nearest-Neighbor*-Klassifikator wird eine Klassenzuordnung eines Merkmalvektors unter Berücksichtigung seiner k nächsten Nachbarn vorgenommen. Der Teil des Lernens besteht aus simplem Abspeichern der Trainingsbeispiele. Die Klassifikation besteht in dem Vergleich des aktuellen Datums mit allen gespeicherten Daten, wobei die Zugehörigkeit der meisten der k nächsten Nachbarn über die Zuordnung entscheidet.

Dieser Ansatz zeichnet sich durch seine Simplizität aus, ist für viele Anwendungsfälle jedoch ausreichend. Da der Implementierungs- und Berechnungsaufwand gering ist, ist er besonders für Echtzeitanwendungen geeignet.

3.4.2 Hidden Markov Model

Hidden Markov Models (HMMs) werden schon lange im Bereich der Spracherkennung verwendet, sind jedoch auch für die Klassifizierung anderer Daten nützlich und können wie in [24] beschrieben auch zur Aktionserkennung eingesetzt werden. Ein HMM setzt sich aus zwei Stufen zusammen. Die erste Stufe besteht aus einem stochastischen Prozess S, der als Folge von Zufallsvariablen

$$S = S_1, S_2, ..., S_T$$

beschrieben wird und nicht beobachtbar ist. Die Werte für diese entstammen einer diskreten, endlichen Zustandsmenge:

$$S_t \in \{1, 2, ...N\}$$

Der Prozess S besitzt dabei folgende Eigenschaften: Er ist *stationär*, und demnach von der absoluten Zeit unabhängig. Die Wahrscheinlichkeitsverteilung einer Zufallsvariablen hängt des Weiteren jeweils nur von vergangenen Zuständen ab; der Prozess ist also *kausal*. Für die hier betrachteten HMMs erster Ordnung ist der Prozess zusätzlich *einfach*: Die Verteilung einer Zustandsvariablen hängt dabei nur vom unmittelbaren Vorgänger ab.

Übergänge zwischen den Zuständen erfolgen entsprechend der *Zustandsübergangswahrscheinlichkeiten*

$$P(S_t = i | S_{t-1} = j)$$

Ferner ist bekannt, mit welcher Wahrscheinlichkeit das Modell im Zustand i startet. Die Formel

$$\pi = \{\pi_i | \pi_i = P(S_1 = i)\}$$

modelliert dabei diese Startwahrscheinlichkeiten.

Die zweite Stufe besteht aus einem weiteren stochastischen Prozess, der zu jedem Zeitpunkt beobachtbare Emissionen O_t in Abhängigkeit vom aktuellen Zustand mit den entsprechenden Emissionswahrscheinlichkeiten generiert:

$$P(O_t | O_1 ... O_{t-1}, S_1 ... S_t) = P(O_t | S_t)$$

Dies gilt jedoch nur für diskrete HMMs, die für vektorwertige Daten nicht geeignet sind. Statt dessen verwendet man für diesen Zweck gewöhnlich *kontinuierliche* HMMs, bei denen die Emissionen $X = x_1, x_2, ..., x_T$ Folgen von Vektoren $x_t \in \mathbb{R}^n$ aus einem n-dimensionalen Vektorraum darstellen. Der Vektor von Emissionsdichten

$$b_j(x) = p(x | S_t = j)$$

beschreibt dann die Ausgaben des HMMs und kann aus Normal- und Mischverteilungen bestehen.

Für das Training des HMMs besteht die Aufgabe nun darin, Emissions- und Zustandsübergangswahrscheinlichkeiten so zu wählen, dass sich mittels der Emissionen möglichst gut auf die versteckte Zustandsfolge schließen lässt. Zu diesem Zweck wird der Baum-Welch-Algorithmus verwendet.

Für die Klassifikation ist die wahrscheinlichste Folge von versteckten Zuständen ausschlaggebend. Diese lässt sich mit dem Viterbi-Algorithmus berechnen.[25]

Bei der Aktionserkennung entspricht die Folge einzelner Posen, die z.B. als Bild oder als Merkmalsvektor vorliegen können, den beobachtbaren Emissionen. Die versteckte Zustandsfolge ist die gesuchte Aktion, die näherungsweise berechnet wird. Ein Problem bei der Sequenzklassifikation ist die schon erwähnte Abhängigkeit von der gesamten Sequenz. Somit kann eine Aktion mittels HMM erst bestimmt werden, wenn sie vorüber ist. Die daraus folgende Verzögerung muss für ein reaktives System möglichst gering gehalten werden.

HMMs sind ungleich komplexer als auf kNN basierende Klassifikatoren, liefern jedoch gerade für sequentielle Daten gute Ergebnisse. Für die Klassifikation von aus Sequenzen von Posenlabels bestehende Aktionen bieten sie sich demnach an. Die Reaktivität ist jedoch strukturbedingt gefährdet. Ihr muss bei der Implementierung besondere Aufmerksamkeit zuteil werden.

3.4.3 N-Gramme

N-Gramme können für die Einbeziehung des zeitlichen Kontext eines Datensatzes verwendet werden und sind gerade daher für die Aktionserkennung besonders interessant. Sie finden ansonsten Anwendung in der Kryptologie und Linguistik, speziell auch in der Computerlinguistik, Computerforensik und quantitativen Linguistik. Einzelne Wörter, ganze Sätze oder komplette Texte werden hierbei zur Analyse oder statistischen Auswertung in N-Gramme zerlegt. Ein N-Gramm ist eine Folge aus n Zeichen, beispielsweise ein Wortfragment. Bei der Aktionserkennung stellen die N-Gramme eine Folge von n Symbolen dar, welche z.B. Posenprimitive abbilden. Das Wort, welches aus beliebig vielen N-Grammen bestehen kann, stellt die Aktion selbst dar.

Die Aktionserkennung wird durch die Verwendung von N-Grammen robuster, da unsinnige Abfolgen von Posen verworfen werden.

4 Realisierung

In diesem Kapitel werden zunächst die Räumlichkeit der Spielumgebung sowie der Hardwareaufbau der Zielanwendung beschrieben. Anschließend erfolgt eine Erläuterung der Architektur der Zielanwendung als Ganzes und der Kommunikation zwischen den einzelnen Modulen. In den darauf folgenden Unterabschnitten werden die einzelnen Module in ihrer Funktionalität detailliert beschrieben. Dabei werden einzelne Parameter genauer erläutert. Die zuvor erwähnten grundlegenden Verfahrensweisen kommen dabei jeweils zur Anwendung.

4.1 Räumlichkeit der Spielumgebung

Zur Nutzung der Zielanwendung wird ein Raum benötigt, der groß genug ist, um den Benutzer des Spieles als Ganzes aufnehmen zu können. Der Abstand zwischen Spieler und Kamera ist dabei so zu wählen, dass der gesamte Körper des Spielers in der Aufnahme zu erkennen ist. Nur so ist es möglich die benötigten Posen und Aktionen, welche zur Bedienung des Spieles verwendet werden, zu erkennen. Wie in Abbildung 7 zu sehen ist, steht der Benutzer hierbei vor einem Hintergrund. Um eine kontrollierte Umgebung zu schaffen, wurde zur Vereinfachung ein statischer Hintergrund gewählt. Zur Aufnahme der Spielaktionen ist eine Kamera frontal zur Person gerichtet. Das Spiel wird über einen Projektor angezeigt und auf eine ebenfalls frontal zum Benutzer gerichtete Leinwand übertragen, so dass der Benutzer die Aktionen der Spielfigur verfolgen kann.

Die Änderungen der Lichtverhältnisse im Raum sind relativ gering und beeinflussen somit nicht die Erkennung der Aktionen und Posen. Jedoch ist darauf zu achten, dass keine weitere Person sich während des Spieles im Hintergrund des Benutzers aufhält. Dies kann zur Störungen bei der Erkennung der Spielaktionen führen. Die Störungen sind dabei zeitlich beschränkt, das heißt sie sind nur so lange vorhanden, wie eine andere Person sich im Kamerabereich aufhält.

Nach der Beschreibung des Aufbaus der Spielumgebung, wird folgend der Aufbau der in der Zielanwendung benötigten Hardware dargestellt.

4.2 Aufbau der Hardware

Für die Realisierung der Zielanwendung werden unterschiedliche Hardwarekomponenten benötigt. Diese lassen sich in zwei Aspekte gliedern. Zunächst werden die technischen Daten der zur Aufnahme der Posen und Aktionen

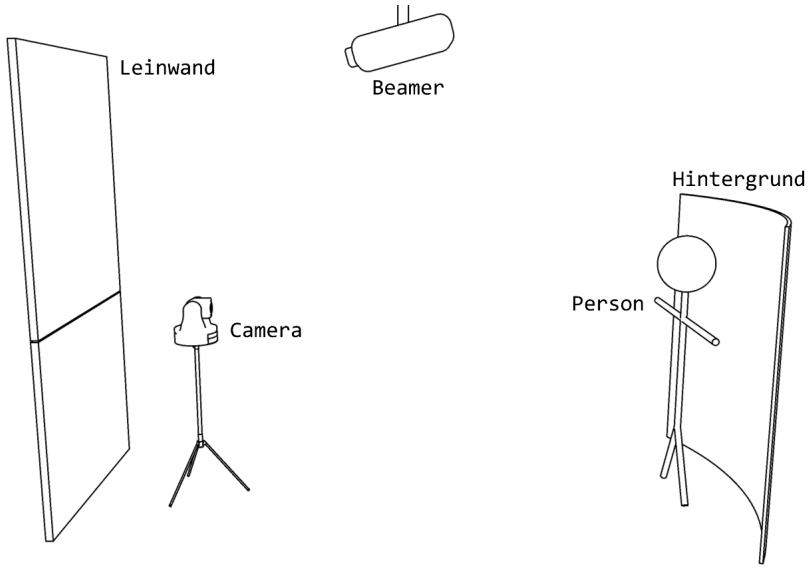

Abbildung 7: Aufbau der Spielumgebung

verwendeten Kamera erläutert. Danach erfolgt eine Beschreibung der eingesetzten Rechnerhardware und ihrer Umgebungsvariablen.

4.2.1 Kamera und Kameraserver

Zur Aufnahme der Bilddaten wird eine Sony *EVI-D70-Kamera* verwendet, welche Bilder in einer PAL Auflösung liefert. Um die Datenmenge der Bilder zu reduzieren und damit die Verarbeitung der Bilddaten zu beschleunigen, werden Halbbilder verwendet, deren Auflösung über einen Kameraserver auf 284x288 Pixel reduziert wird. Über diesen Server gelangen die Bilder in die Zielanwendung, welche als Pipelinearchitektur realisiert ist. Zur eindeutigen Identifizierung der Bilder werden hierbei Zeitstempel erstellt. Da die Zielanwendung stets das aktuelle Bild vom Server anfordert, richtet sich die Rate mit der die Bilder in die Anwendung gelangen, nach der Verarbeitungsgeschwindigkeit des Pipelinemoduls für die Bildbeschaffung. Wobei alte Bilddaten überschrieben werden. Dabei ist die Kamera, wie in Abbildung 8 zu sehen, über Firewire mit dem Kameraserver verbunden. Der Kameraserver wiederum kommuniziert über ein Gigabit Netzwerk mit der Zielanwendung.

4.2.2 Rechnerhardware

Für die Zielanwendung werden minimal zwei Rechner benötigt. Auf einem von diesen Rechnern werden ein Emulator zur Spieleansteuerung, sowie die Software zur Aktionserkennung und Ansteuerung des Spieles, ausgeführt. Auf dem zweiten Rechner finden Berechnungen auf den Bilddaten statt. Hier erfolgt die Ermittlung der ROI, sowie die Erkennung der Posen. Die Kommunikation zwischen diesen Rechnern erfolgt über UDP Sockets. Hierbei werden die erkannten Posen in Form von Labeln an ein Aktionserkennungsmodul gereicht. Die Verteilung der Rechenlast auf zwei Rechner optimiert die Antwortzeit des Systems. Es besteht jedoch auch die Möglichkeit die Zielanwendung auf drei Rechner zu verteilen, da der Emulator ebenfalls über UDP Sockets mit der restlichen Software kommuniziert.

Die Rechner verfügen über jeweils zwei Kerne mit 1.86 bis 3.17 GHz. Hierbei ist eine gute Ausstattung bzgl. der Leistungsfähigkeit des Rechners, der die Kalkulationen auf den Bilddaten ausführt, wichtig. Dieser Teil der Anwendung benötigt im Gegensatz zur Aktionserkennung und Emulatorsteuerung eine wesentlich höhere Rechenkapazität. Auf beiden Systemen ist das Betriebssystem Linux installiert. Die Rechner sind Arbeitsstationen des Instituts für Roboterforschung und sind daher an das dort vorhandene interne Netzwerk angeschlossen. In der nachstehenden Abbildung 8 ist ein beispielhafter Aufbau der Zielhardware dargestellt.

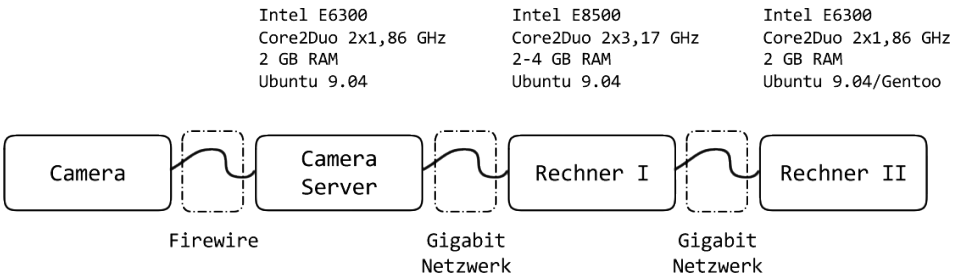

Abbildung 8: Aufbau der Zielhardware

Nach dem nun die Hardware der Zielanwendung beschrieben wurde, folgt die Darstellung der Softwarearchitektur.

4.3 Softwarearchitektur

Ausgehend von der Aufnahme eines Bildes einer Person, wird eine Pose oder eine Aktion bestimmt, über die wiederum die Ansteuerung des Spieles erfolgt. Hierfür sind mehrere Berechnungsschritte notwendig. Darunter fällt zunächst die Ermittlung der Position der Person innerhalb des Bildes (Personendetektion). Davon ausgehend können Merkmale extrahiert werden, die

der Klassifikation von Posen dienen. Die Reihenfolge der Berechnungsschritte ist hierbei fest. So kann zum Beispiel die Klassifizierung erst durchgeführt werden, wenn die Merkmale extrahiert worden sind. Doch auch wenn die einzelnen Berechnungsschritte voneinander abhängen, können die Berechnungen unabhängig von den Daten parallel ausgeführt werden. Das heißt jedes Modul, welches einen der Berechnungsschritte darstellt, arbeitet auf anderen Bilddaten als das Vorgänger- oder Nachfolgermodul.

Der Begriff des Moduls wird anhand einer Tätigkeit innerhalb der Anwendung definiert. Ein Modul kapselt daher eine der benötigten Berechnungsschritte, wie zum Beispiel die Bildbeschaffung, die Positionsbestimmung, die Merkmalsextraktion, die Ermittlung der Pose aus den Merkmalen, die Aktionserkennung oder die Ansteuerung des Spieles. Ausgehend hiervon ist die Anwendung als Pipeline in C/C++ implementiert. Die Anforderung, die eintreffenden Bilder in schneller Geschwindigkeit verarbeiten zu können, hat dabei die Wahl der Sprache auf C/C++ reduziert.

Das Modul ViscaConnector bildet den Anfang der Pipeline. Dieses Modul stellt einen Client innerhalb eines Client-/Server Prinzips dar. Der Server selbst steht mittels eines VISCA-Protokolls [19] mit einer hierfür geeigneten Kamera (Sony EVI D70) in Verbindung. Auf Anfrage des Clients wird das Bild in einer unkomprimierten Form über das Netzwerk an den Client versendet. Wie in Abbildung 9 zu sehen besteht die Tätigkeit des Moduls ViscaConnector aus einer endlosen Anfrage nach neuen Bilddaten. Eintreffende Bilder werden mit einer numerischen Identifikation versehen und an eine vorher vereinbarte Stelle im Speicher abgelegt. Diese Vereinbarung geschieht im Vorfeld durch ein geeignetes Exchange-Objekt. Der Zugriff auf dieses Exchange-Objekt wird durch Mutex-Operationen gegenüber mehrfachem Zugriff geschützt.

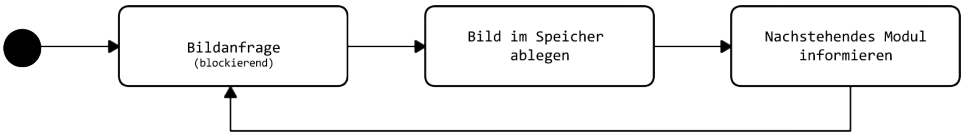

Abbildung 9: Aktivität des ViscaConnectors.

Zwei Module verhalten sich entsprechend dem Erzeuger-Verbraucher-Problem. Für die Synchronisation dieses Speichers stellt die POSIX-Bibliothek die notwendigen Mittel zur Verfügung. Das Lesen und Schreiben in den Speicherbereich wird dahingehend, wie in Abbildung 10 dargestellt sequentialisiert. Zusätzlich zum gegenseitigen Ausschluss bei Schreib- beziehungsweise Leseoperationen, wird eine zusätzliche Mutex-Struktur dazu verwen-

det, um eine abgeschlossene Schreiboperation zu signalisieren.

Schreibthread:

```
lock (memory);

    /* Schreiboperationen */

unlock (memory);
unlock (signal);
```

Lesethread:

```
lock (signal);
lock (memory);

    /* Leseoperationen */

unlock (memory);
```

Abbildung 10: Verfahrensweise zwischen schreibendem (Erzeuger) und lesendem (Verbraucher) Thread.

Das lesende Modul wartet auf ausgeführte Schreiboperationen des schreibenden Moduls und blockiert während dieser Zeit. Bewusst wurde hierbei das aktive Warten vermieden, und die Freigabe des Prozessors erzwungen, falls keine aktuellen Daten vorliegen. Rechenwillige Threads werden in dieser Form der Implementierung begünstigt. Da neue Daten die alten stets überschreiben und das nachfolgende Modul nur bei Eingang neuer Daten informiert wird, rechnet jeder Thread stets auf aktuellen Datensätzen.

Ausgehend von der Bildbeschaffung – dem Modul ViscaConnector - wird im nachfolgenden Modul Region-of-Interest eine Personendetektion durchgeführt. Dies kann auf Basis der Bewegungen innerhalb mehrerer Bilder geschehen, oder mit Hilfe einer Hintergrundsubtraktion. Das Ergebnis hiervon ist ein Bereich, der innerhalb der Implementierung durch $P1(x, y)$ und $P2(x, y)$ definiert wird. Innerhalb dieses Bereichs werden im nachfolgenden Modul HoGCalculator die Merkmale aus dem Bild extrahiert. Diese Merkmale dienen im nächsten Modul Posenerkenner als Eingabe für die Klassifizierung der Pose. Ein vorher trainiertes, künstliches neuronales Netz fungiert hierbei als Klassifikator. Das Resultat wird im nachstehenden Modul ActionDetector verarbeitet um hieraus eine Aktion ermitteln zu können. Diese Aktion wird dem letzten Modul, dem Controller, als Eingabe weitergeleitet, um eine Reaktion innerhalb des Spieles auszulösen.

Von Anfang an stand neben der Funktion, vor allem die Geschwindigkeit dieser Anwendung im Vordergrund. Die Reaktionszeit, vom aufgenommenen Bild, bis hin zur letztendlichen Reaktion im Spiel musste weitestgehend gering gehalten werden. Hierfür wurde die Pipeline, wie in Abbildung 11 zu sehen, an geeigneten Stellen um eine Socketkommunikation erweitert. Die Pipeline lässt sich daher in drei Komponenten aufteilen. Innerhalb der ersten Komponente wird sowohl die Bildbeschaffung, als auch die Personendetek-

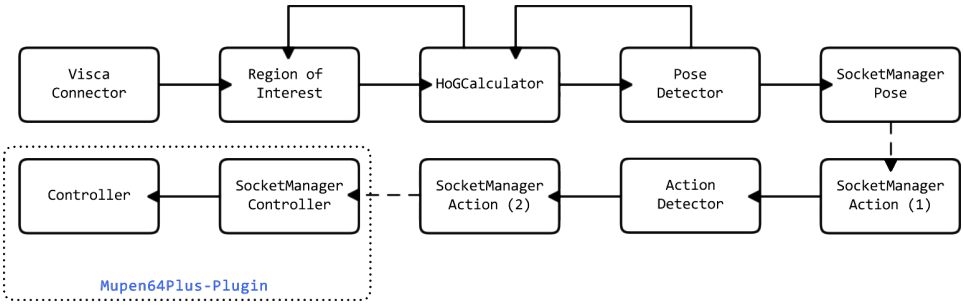

Abbildung 11: Datenfluss innerhalb der NextWii-Pipeline.

tion, die Merkmalsextraktion und die Posenerkennung durchgeführt. Erst das Ergebnis des Posenerkenners ist für eine effiziente Netzwerkkommunikation zu gebrauchen, da im Vorfeld lediglich unkomprimierte Bilddaten, bzw. hochdimensionale Gleitkommavektoren ausgetauscht werden. An dieser Stelle setzen die jeweiligen SocketManager-Module an. Diese verhalten sich zu ihren Vorgängern wie Berechnungsschritte, so dass die Übergabe der Ergebnisse aus dem vorhergehenden Schritt problemlos erfolgen kann. Die Übermittlung der Daten erfolgt dabei vollständig transparent zum nachstehenden SocketManager-Modul. Die zweite Komponente der Pipeline besteht lediglich aus dem ActionDetector-Modul. Der Empfang der Daten erfolgt hierbei ähnlich wie auch das Senden vollständig transparent. Ergebnisse des ActionDetector-Moduls werden ebenfalls über eine Socketverbindung dem Controller-Modul mitgeteilt. Dieses Modul kann als dritte Komponente der Pipeline separat auf einem eigenen Rechner platziert werden. Die Netzwerkübertragungen erfolgen, zu Gunsten der Geschwindigkeit, über UDP-Sockets. In Abbildung 12 sind die erläuterten Komponenten dargestellt.

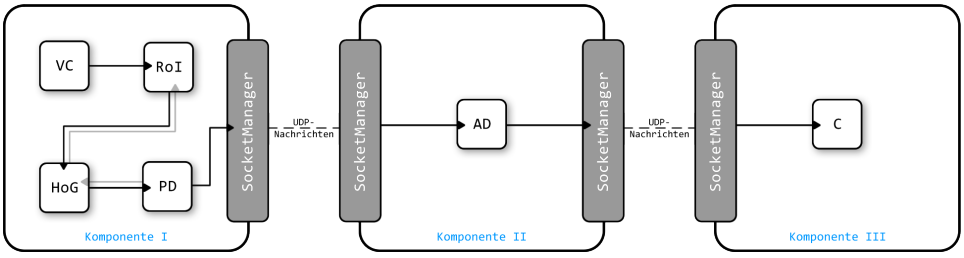

Abbildung 12: Komponenten der Pipeline. VC (Bildbeschaffung [Visca-Connector]), ROI (Positionsbestimmung [Region-of-Interest]), HoG (Merkmalsextraktion [HoGCalculator]), PD (Posenerkennung [PoseDetector]), AD (Aktionserkennung [ActionDetector]), C (Spieleansteuerung [Controller])

Zusätzlich zum Datenfluss durch die Pipeline, sind an den Modulen für die Posenerkennung und der Merkmalsextraktion rückwärtige Kommunika-

tionsmöglichkeiten gegeben. Eine durch den Posenerkenner gefundene Pose führt zu einem Berechnungsabbruch beim Modul für die Merkmalsextraktion ab. Der Posenerkenner bestätigt dabei eine erfolgreiche Klassifikation anhand der Bild-ID. Das vorhergehende Modul berechnet dahingehend keine weiteren Merkmalsvektoren aus diesem Bild. Die Bestätigung einer solchen Klassifikation wird auch dem Modul für die Personendetektion mitgeteilt. Hierbei wird - neben der Bild-ID - zusätzlich die Bildregion bestätigt. Der bestätigte Bereich wird seitens der Personendetektion in nachfolgenden Bildern nur wenig verändert und erlaubt es in kleinem Umfang auf sporadisch auftretende Objekte innerhalb des Bildes zu reagieren.

4.4 ROI

Das *ROI-Modul* dient dazu, den Bereich des Bildes ausfindig zu machen, in dem sich eine Person befindet. Weitere Berechnungen müssen anschließend nicht mehr auf dem gesamten Bild, sondern lediglich auf dem eingeschränkten Bereich durchgeführt werden. Im Folgenden wird genauer auf die Arbeitsweise des Moduls eingegangen. Von den im Abschnitt 3.2 vorgestellten Verfahren wurde das *MHI* ausgewählt. Es liefert bessere Ergebnisse, als der approximierte Median, ist jedoch nicht so rechenintensiv wie die *Mixture of Gaussians*. Weiterhin kann es sowohl für statische, als auch dynamische Modelle genutzt werden und bietet durch die vielen Parametrisierungsoptionen Möglichkeiten zur Optimierung der Ergebnisse.

4.4.1 Funktionsweise

In direktem Anschluss an den *Visca-Connector* erhält das *ROI-Modul* alle verfügbaren Bilder. Je nach Konfiguration des Moduls wird ein dynamisches oder statisches Hintergrundmodell erzeugt. Das Ergebnis wird an den *ROI Tree* weitergereicht, welcher als Ergebnis einzelne Bereiche des Bildes liefert, die nicht zum Hintergrund gehören. Tritt eine Person in das Bild ein, unterscheidet sie sich deutlich vom Hintergrund. Im *ROI Tree* lassen sich nun zusammenhängende Regionen ermitteln, von denen es häufig eine Vielzahl gibt. Anschließend werden diese einzeln betrachtet und alle Extremwerte der Koordinaten zusammengefügt, welche letztendlich eine einzige, große *ROI* ergeben. Da diese Region häufig noch sehr eng um eine Person herum ist, wird sie in alle Richtungen ein wenig erweitert.

4.4.2 Statisches Hintergrundmodell

Beim statischen Modell wird lediglich eine Hintergrundsubtraktion durchgeführt. Hierzu wird als erstes Eingabebild der leere Hintergrund ohne Person

erwartet und als Modell gespeichert. Zur Berechnung der ROI wird für jeden Pixel des aktuellen Bildes die Differenz mit den zugehörigen Pixel des Modells bestimmt und anhand eines Schwellenwerts entschieden, ob er dem Vorder- oder Hintergrund zugeordnet wird.

4.4.3 Dynamisches Hintergrundmodell

Das dynamische Hintergrundmodell erfordert weitaus mehr Anpassungen, um die *ROI* optimal zu berechnen. Insbesondere wirkt es sich sehr negativ auf das Ergebnis aus, wenn einige Gliedmaßen weniger Bewegung zeigen, als andere. Betrachtet man beispielsweise die Aktion „Fliegen“, wird deutlich, dass nur die Arme in Bewegung sind, wohingegen der Rest des Körpers kaum Bewegung zeigt und daher nach kurzer Zeit dem Hintergrund zugeordnet wird.

Wird nun der *ROI Tree* zu solch einer Bewegung ausgewertet, ent-

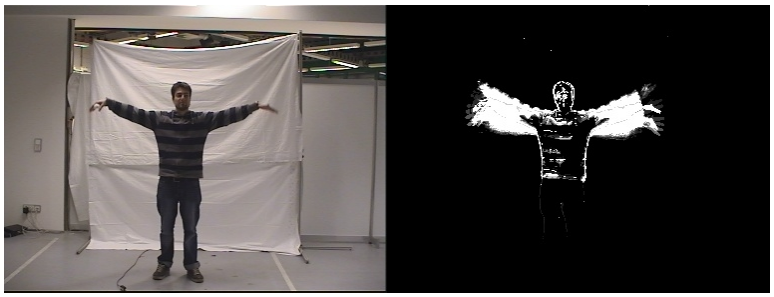

Abbildung 13: Die Aktion „Fliegen“ mit entsprechendem MHI. Keine Bewegung der Beine

steht nur eine recht schmale *ROI*, welche die beiden Arme überspannt. Um dennoch den ganzen Körper zu erfassen, wird zunächst eine Anpassung der Seitenverhältnisse vorgenommen. Die *ROI* ist somit immer quadratisch und erfasst sowohl Kopf, als auch Beine, falls nur Bewegung in den äußeren Bereichen stattfindet. Genau wie beim statischen Modell, wird auch hier die *ROI* noch um 3 Pixel in jede Richtung vergrößert, um einen kleinen Rand um die Person herum zu erhalten. Ein häufig auftretendes Problem stellt dennoch Bewegungsmangel in Kopf und Beinen dar. Diese Regionen werden nach kurzer Zeit nicht mehr von der *ROI* erfasst, wodurch sie sich nur noch auf den Oberkörper beschränkt und beim nachfolgenden Modul zu falschen berechnungen führt.

Um dies zu verhindern, wurde ein Qualitätsfeedback vom HOG-Modul eingeführt. *ROIs*, welche zu einem eindeutigen Ergebnis führen, werden dem *ROI*-Modul bestätigt. Da recht viel Zeit von einer Berechnung bis zur Bestätigung dieser vergeht und es dazu kommen kann, das gänzlich unterschiedliche *ROIs* zu guten Ergebnissen führen, wird ein Puffer verwendet. Dieser

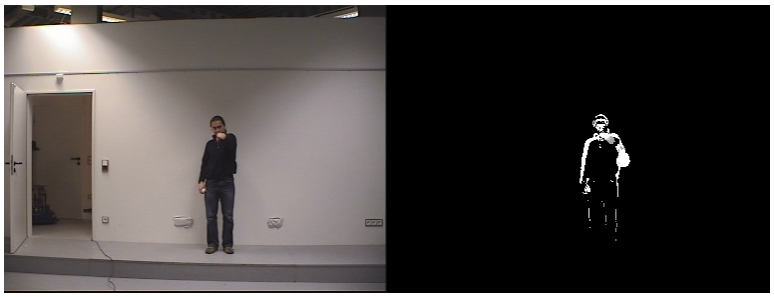

Abbildung 14: Die Aktion „Boxen“ mit entsprechendem MHI. Die Beine sind kaum betont.

beinhaltet die letzten *ROIs*, welche zu einer hohen Erkennungsrate beim HOG-Modul führten. Aus diesem Puffer wird der Durchschnitt ermittelt, wobei aktuellere *ROIs* eine höhere Gewichtung erhalten, als ältere. Die aktuell berechnete *ROI* wird anschließend an diesem Durchschnitt ausgerichtet. Hierbei werden nur geringe Abweichungen zugelassen, da es unwahrscheinlich ist, dass eine Person sich in kurzer Zeit durch das gesamte Bild bewegt.

4.4.4 Binär-Maske zur Verbesserung der HOG-Berechnung

Um die Aussagekräftigkeit der Merkmalsvektoren zu steigern, bietet es sich an, lediglich Gradienten, die tatsächlich zur Person gehören in die *HOG*-Berechnung einfließen zu lassen. Dies ermöglicht die Reduktion des Einflusses von Störungen im Hintergrund bei gleichbleibendem Informationsgehalt in den relevanten Regionen. Um dem HOG-Modul die Entscheidung, ob der Gradient einer Pixel-Position in den Merkmalsverktor einfließen soll oder nicht zu ermöglichen, wird eine Binär-Maske, die jedem Pixel die Information Person / keine Person zuordnet, benötigt. Die Berechnung einer entsprechenden Maske gestaltet sich, aufbauend auf den durch die oben beschriebenen Verfahren bereits berechneten, Vorder- bzw. Hintergrundmodellen relativ leicht.

Da die durch die Modelle berechnete Einteilung in Vorder- und Hintergrund in der Regel nicht ganz exakt die Umrisse der Person abbildet, entstehen gerade am Rand der Person Lücken, in denen einzelne Pixel fälschlicherweise dem Hintergrund zugeordnet werden. Darüber hinaus können aufgrund von Übereinstimmungen zwischen der Farbe des Hintergrunds und der Kleidung unter Umständen Teile der Person vom Modell fälschlicherweise dem Hintergrund zugeordnet werden, so dass eine weitere Verarbeitung der Maske notwendig ist. Um die beschriebenen Lücken zu füllen bietet sich die Verwendung eines Maximumfilters an. Hierbei handelt es sich um einen Rangordnungsfilter bei dessen Anwendung, vergleichbar mit der in Abschnitt

3.3 beschriebenen Anwendung von Filter-Masken, alle Nachbarn des aktuellen Pixels aufgesammelt und nach ihrer Größe sortiert werden. Als Ergebnis der Anwendung wird der Wert des größten Pixels zurückgeliefert. Durch die Anwendung des Filters werden die meisten Lücken in der Maske geschlossen, so dass sie zur Weiterverarbeitung durch das HOG-Modul bereitsteht.

4.5 Posenerkenner

Folgend wird die Funktionsweise des Moduls zur Posenerkennung dargestellt. Hierbei werden zunächst die für die Zielanwendung ausgewählten Posen beschrieben. Anschließend erfolgt eine Erläuterung zum Aufbau des Posenerkenners, sowie zur Merkmalsextraktion und zur Klassifikation der Posen.

4.5.1 Auswahl der Posen

Die Auswahl der Posen zur Ansteuerung des Spieles ist in erster Linie abhängig vom ausgewählten Spielgenre. Da das Spiel der Zielanwendung dem klassischen *Jump'n'Run* Genre zuzuordnen ist, werden Posen wie z. B. Boxen, Springen und Werfen verwendet jedoch auch speziell auf das Spiel zugeschnittene Posen, wie z. B. Hacken, Fliegen oder Ducken kommen zum Einsatz. Zusätzlich werden verschiedene Variationen von Posen genutzt, die bei der Spielansteuerung die gleiche Auswirkung haben. Dies dient einer intuitiveren Steuerung des Spieles. Die Auswahl der zu klassifizierenden Posen ist in Abbildung 15 aufgeführt.

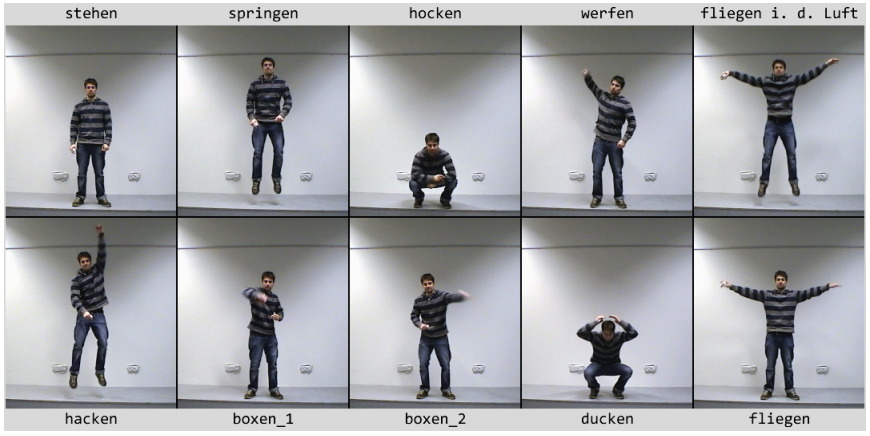

Abbildung 15: Verwendete Schlüsselposen

Um einer Posenverwechslung bei der Klassifikation vorzubeugen, sind die Posen so ausgewählt, dass sie sich stark unterscheiden. Bei Posen die sich

sehr ähnlich sind, wie z.B. die in Abbildung 16 dargestellten Posen, werden auch für die Klassifikation ähnliche Merkmalsvektoren generiert.

Abbildung 16: negatives Beispiel für eine Posenauswahl

Diese Ähnlichkeit führt dabei zu schlechten Erkennungsraten, daher ist die Verwendung solcher Posen in der Zielanwendung zu vermeiden.

4.5.2 Aufbau

Das ROI Modul, welches der Posenerkennung vorgeschaltet ist, liefert die Koordinaten, die zur Einschränkung des Bildbereiches dienen, in dem sich die Person befindet. Um die von der Person ausgeführten Posen erkennen zu können, wird zunächst eine Merkmalsextraktion auf den Bilddaten durchgeführt. Die Einschränkung des Bildbereiches reduziert hierbei die Rechenzeit. Die Extraktion basiert auf dem in Abschnitt 3.3 vorgestellten HOG-Verfahren. Der so erzeugte Ergebnisvektor eines Bildes dient als Eingabe des Posenerkenners. Mit Hilfe eines *KNN* kann hier die Klassifikation der Posen erfolgen. Die erkannten Posen werden in Form von Labeln an das dem Posenerkenner nachgeschaltete Modul zur Aktionserkennung gereicht. Der Aufbau dieses Teilbereiches der Pipeline ist in Abbildung 17 zu sehen.

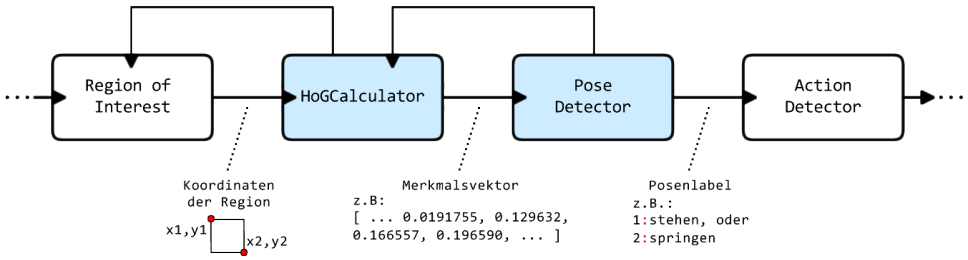

Abbildung 17: Aufbau des Posenerkenners innerhalb der Pipeline

Hierbei existieren zusätzliche Rückkanten die zur Qualitätsverbesserung der Berechungen des ROI Moduls und damit auch den nachfolgenden Modulen dienen. Meldet der Posenerkenner z. B. eine erkannte Pose, so kann diese Information über das Modul der HOG Berechnung an das ROI Mo-

dul weiter gereicht werden. Eine größere Anpassung der aktuell berechneten Koordinaten wäre dann nicht mehr nötig.

4.5.3 Merkmalsextraktion

Die Merkmalsextraktion erfolgt über das in 3.3 vorgestellten HOG-Verfahren. Die Konfiguration der Parameter entspricht hierbei den in [40] empfohlenen Werten. Die Anzahl der *Bins* beträgt hier 9, die Aufteilung der *Blöcke* und *Zellen* ist in Abbildung 18 dargestellt. Es wird eine Überlappung der Blöcke von 1 gewählt und als Normalisierungsschema wird eine *L2-Norm* verwendet. Durch diese Konfiguration der Parameter entsteht ein Merkmalsvektor mit 729 (3 x 3 x 3 x 3 x 9) Werten.

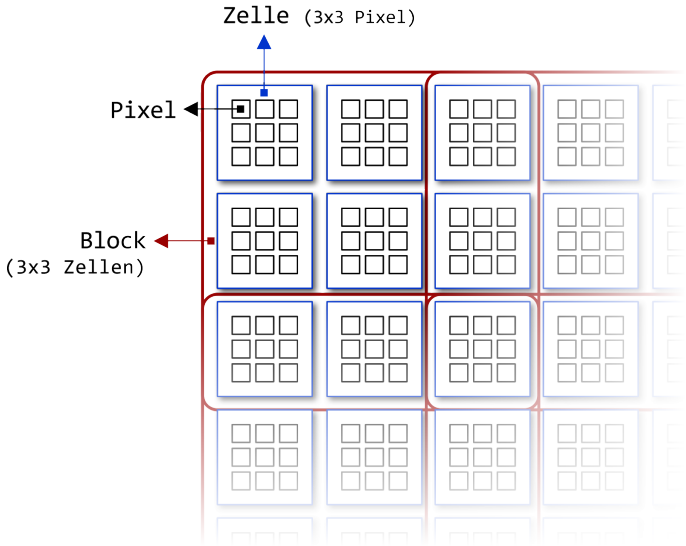

Abbildung 18: Aufteilung der Blöcke und Zellen zur HOG-Berechnung

Wie weiterhin in Abbildung 19 zu sehen, bestehen die Eingabedaten für die Berechnung der Merkmalsvektoren aus dem aktuellen Bild, der *Bild-ID*, der Koordinaten der Region und einer *Subtraktionsmaske*.

Die Bild-ID dient hier der eindeutigen Zuordnung von später berechneten Posen zum Bild, sie wird über die ganze Pipeline hinweg genutzt. Die Subtraktionsmaske stellt ein weiteres Ergebnis des ROI Moduls dar. Sie wird dazu genutzt den Hintergrund von der Person zu trennen. Anhand dieser Maske wird der berechnete Merkmalsvektor angepasst, so dass die Werte im Vektor, welche stellvertretend für den Hintergrund der Person stehen, auf 0

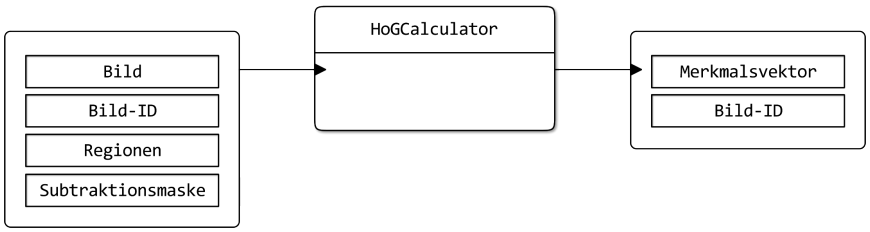

Abbildung 19: Eingabe- und Ausgabedaten des HOG Moduls

gesetzt werden. So kann die Posenerkennung unabhängig vom Hintergrundbild erfolgen.

4.5.4 Klassifikation

Um die HOG Merkmalsvektoren zu klassifizieren, wird ein Künstliches Neuronales Netz genutzt. Als *Aktivierungsfunktion* der einzelnen Neuronen wird die, von der FANN Bibliothek [44] bereitgestellte, *FANN_SIGMOID* Funktion genutzt. Spätere Ergebnisse der Evaluierung aus Abschnitt 5.2 zeigten, dass ein Netz mit drei Schichten am besten geeignet ist. Die Anzahl der Neuronen der ersten Schicht wird vom Merkmalsvektor bestimmt und beträgt in diesem Fall 729. Die Zwischenschicht besteht aus 470, die Ausgabeschicht aus 14 Neuronen. Somit repräsentiert das vom HOG-Merkmalsvektor am stärksten aktivierte Neuron, der letzten Schicht, die erkannte Pose. Um Fehlinterpretationen des Eingabevektors zu verhindern, wird überprüft, ob der Wert des aktivsten Ausgabeneurons einen Schwellwert übersteigt. Nur wenn dies der Fall ist, wird der HOG-Merkmalsvektor als klassifiziert eingestuft und das Label der entsprechenden Pose an den Aktionserkenner weitergegeben (siehe Abbildung 20). Bei einer Klassifizierung der Eingabe wird zusätzlich noch das ROI Modul informiert, welcher Vektor klassifiziert werden konnte.

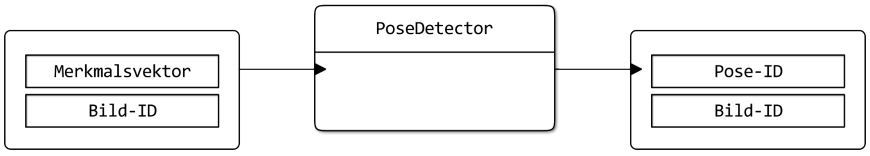

Abbildung 20: Eingabe- und Ausgabedaten des Posenerkenner Moduls

Als Alternative zum KNN [12] wurden die in Abschnitt 3.4.2 beschriebenen HMMs betrachtet. Im direkten Vergleich mit einem entsprechenden KNN aus der FANN-Bibliothek [44] waren die entsprechenden Erkennungsraten des HMM jedoch unzureichend. Daher wird ein KNN als Klassifikator eingesetzt.

4.6 Aktionserkenner

Die vom Posenerkenner ausgegebenen Posenlabels liegen nun als Sequenzen vor und dienen als Eingabe für den Aktionserkenner. Um Aktionen zu erkennen, ist es zuerst nötig, sich über die Elemente, aus denen die Sequenzen bestehen, Gedanken zu machen. Zudem ist es erforderlich, sich für eine Ebene zu entscheiden, auf der die Erkennung stattfinden wird. Eine Möglichkeit liegt darin, direkt auf der Ebene der aus den Bildern (einer Videosequenz) extrahierten Merkmalen zu arbeiten. Eine andere höhere Ebene, auf der man agieren kann, ist eine symbolische Ebene. Hier bilden die Hypothesen, welche aus den Merkmalen resultieren die Elemente der Sequenzen. Letzterem Ansatz wird der Vorzug gegeben, da ungewollte Varianzen in den zu beurteilenden Videosequenzen durch eine zuvor erfolgte Bewertung und Kategorisierung der extrahierten Merkmale stark gemindert werden. Daher werden Schlüsselposen verwendet. Diese werden vom Posenerkenner an den Aktionserkenner weitergeleitet und sind auf entsprechende Symbole (Posenlabels) abgebildet.

4.6.1 DiBaNGARS

„DiBaNGARS“ steht für Distance Based NGram Action Recognition System und verwendet den Abstand von N-Gramm Historien zueinander, um Aktion zu klassifizieren.

Um die Erkennung von Aktionen (im Sinne von [16]) zu realisieren, sind N-Gramme, Histogramme, ein Abstandsmaß sowie ein Verfahren zur Bestimmung des nächsten Nachbarn erforderlich. Um Aktionen („Worte“) miteinander zu vergleichen ist es, unter der Annahme, dass eine Aktion durch mehr als ein N-Gramm beschrieben wird, notwendig, eine einheitliche Abbildung solcher Wörter zu finden. In diesem Fall bieten sich Vektoren an, welche eine Häufigkeitsverteilung der N-Gramme innerhalb eines Wortes darstellen - die Histogramme. Als Abstandsmaß wird der euklidische Abstand verwendet. Obgleich die Manhattan-Metrik nach [13] ebenfalls ein guter Kandidat dafür ist, hat sich durch Testläufe gezeigt, dass der euklidische Abstand besser geeignet ist. Zur Klassifizierung wird der k-Nearest-Neighbor-Algorithmus verwendet.

Erkennen von Aktionen

Unter Verwendung geeigneter Trainingsdaten können unbekannte Sequenzen von Schlüsselposen klassifiziert werden. Zu Beginn wird das Sliding Window mit den erhaltenen Posenlabels gefüllt, dann wird das zugehörige Histogramm berechnet und normalisiert. Dieses wird mit jedem Histogramm aus

den Trainingsdaten anhand des Abstands verglichen und jenes ausgewählt, welches den geringsten Abstand hat. Dann wird die entsprechende Aktion zum Abruf bereit gestellt. Dabei ist zu beachten, dass auch ein Schwellwert für den Abstand angegeben werden kann. Somit werden nur solche Aktionen ausgegeben, deren Vektoren nicht nur den geringsten Abstand aufweisen, sondern zudem noch unter dem Schwellwert liegen. Dies ist dann von Vorteil, wenn die betrachtete Teilsequenz keiner bereits bekannten ähnlich ist. In diesem Fall wird sie als unbekannt zurückgewiesen. Nun wird bei jedem neuen Label, welches beim Aktionserkenner ankommt, dieser Vorgang wiederholt. Wenn eine neue Aktion erkannt worden ist, wird diese zum Abruf bereit gestellt. Die zuvor erkannte Aktion wird dabei überschrieben. Somit kann nach jedem Label die erkannte Aktion abgefragt werden, dies geschieht aber nicht zwangsläufig. Da diese Abfrage zu jedem Zeitpunkt geschehen kann, insbesondere auch dann, wenn noch gar kein neues Label angekommen ist bzw. zu dem Zeitpunkt keine Aktion erkannt wurde, wird die erkannte Aktion höchstens einmal ausgegeben. Sollte die selbe Aktion ein zweites mal abgefragt werden, so wird „Nichts erkannt“ ausgegeben. Auf diesem Weg wird sichergestellt, dass dieselbe Aktion höchstens einmal an das Spiel weitergegeben werden kann.

Training

Damit eine betrachtete Sequenz von Labels, welche die verschiedenen Schlüsselposen repräsentieren, klassifiziert werden kann, ist es notwendig sie mit Referenzdaten (den Trainingsdaten) abzugleichen.

Diese Referenzdaten werden erzeugt indem ein Sliding Window über eine Sequenz läuft, für die bekannt ist welche Aktion sie darstellt. Dafür wird zuerst das Sliding Window von links nach rechts gefüllt. Danach werden die N-Gramme für diese Teilsequenz berechnet, ihre Häufigkeitsverteilung in einem entsprechenden Vektor gespeichert und dieser dann der vorgegebenen Aktion zugeordnet. Da aber nie sichergestellt werden kann, dass die Größe des Sliding Window genau der Länge der Trainingssequenz entspricht, wird das Sliding Window nun ein Label weiter geschoben. Es werden erneut die N-Gramme sowie ihre Häufigkeitsverteilungen berechnet und der gegebenen Aktion zugeordnet. Dieser Vorgang wiederholt sich solange bis die Trainingssequenz komplett erfasst wurde.

Diese Schritte werden für eine ausreichende Anzahl von Sequenzen aller zu erkennenden Aktionen durchgeführt. Obgleich es sinnvoll ist, für das Sliding Window sowohl beim Training als auch später bei der Klassifizierung, die selbe Länge zu verwenden, so würde es auch bedeuten, dass ein Satz von Trainingsdaten nur für eine spezifische Größe des Sliding Window verwendet werden kann. Um robust gegen Längenvarianzen der betrachteten

Teilsequenzen zu sein, werden die Histogramme normalisiert. Dies geschieht, indem der größte Wert im Vektor auf 1 gesetzt wird und alle anderen proportional dazu (zwischen 0 und 1). Auf diesem Wege entsteht leider eine sehr große Menge an Vektoren. Dies kann bei der späteren Erkennung aus Gründen der Performanz von Nachteil sein.

Abhängig davon, wie mächtig die Menge der verwendeten Symbole ist und wie stark sich einzelne Sequenzen unterscheiden, kommt es recht häufig vor, dass viele gleiche Vektoren entstehen. Ferner kann es leicht passieren, dass verschiedene Vektoren zwar die gleiche N-Gramm Historie abbilden, aber verschiedenen Aktionslabels zugeordnet sind. Daher ist es zwingend erforderlich, die gewonnenen Referenzdaten von solchen Mehrdeutigkeiten und Duplikaten zu befreien.

Der Abstand zweier Vektoren beschreibt, wie ähnlich sie sich sind. Daraus folgt, dass, wenn ihr Abstand gleich Null ist, die Vektoren ebenfalls gleich sind. Dies kann ausgenutzt werden, um solche Vektoren zu identifizieren, die mehrdeutig bzw. Duplikate sind. Um Vektoren zu finden, die nicht exakt gleich sind, sich aber nur geringfügig unterscheiden, kann ein Schwellwert verwendet werden.

Bei der Bereinigung der Trainingsdaten werden auf diesem Weg alle Vektoren identifiziert, welche dieselbe N-Gramm Historie abbilden und einer entsprechenden Äquivalenzklasse zugeordnet. Wird kein Schwellwert verwendet, so sind diese Äquivalenzklassen wohldefiniert. Bei wohldefinierten Äquivalenzkassen spielt die Wahl des Repräsentanten keine Rolle. Besitzen alle Vektoren einer solchen Äquivalenzklasse das gleiche Aktionslabel, so werden alle Vektoren bis auf den ersten gelöscht und nur dieser übernommen. Wird ein Schwellwert verwendet, so sind die Äquivalenzklassen nicht mehr wohldefiniert und es kann nicht einfach der erste Vektor als Repräsentant gewählt werden. In diesem Fall werden alle Vektoren dieser Äquivalenzklasse erneut bezüglich ihres Abstandes in Subklassen unterteilt (ohne Schwellwert) und dann aus jeder der nun wohldefinierten Äquivalenzklassen ein Repräsentant ausgewählt. Befinden sich in einer Äquivalenzklasse mindestens zwei Vektoren mit unterschiedlichen Aktionslabels, so wird diese Klasse vollständig verworfen, da nicht automatisiert festgestellt werden kann welches Aktionslabel das richtige ist.

Parameter

Die Erkennung von Aktionen erfordert die Einstellung diverser Parameter. Der Aktionserkenner ermöglicht es, diese Parameter entweder über Standardwerte einzustellen oder sie zur Laufzeit zu modifizieren. Die wohl wichtigsten Werte, die bei der Erkennung und auch dem Training variiert werden kön-

nen, sind die Größe der N-Gramme, sowie die Länge des Sliding Window. Hierbei handelt es sich um Konstanten, die sich zum einen während einer Ausführung des Programms nicht ändern dürfen, und zum anderen an vielen verschiedenen Stellen benötigt werden.

Die beiden anderen Parameter sind die Schwellwerte. Wie bereits erwähnt, kann ein solcher Schwellwert sowohl bei der Bereinigung der Trainingsdaten als auch bei der Erkennung von Aktionen zum Einsatz kommen. Der Schwellwert-Paramteter ist beim Erstellen der Trainingsdaten optional, kann jedoch bei der Optimierung hilfreich sein.

4.6.2 Markov-Modelle

Der Aktionserkenner *Markov-Modelle* beruht auf dem *Language Modell*-Modul der Entwicklungstoolbox Esmeralda[27], das mit statistischen Sprachmodellen zur Spracherkennung eingesetzt wird. Eine Sequenz von Posenlabels, die der Posenerkenner generiert, wird mit vorher aufgenommenen Trainigsdaten verglichen. Die Posensequenzen werden als Wortfolgen betrachtet und mit Markov-Modellen verglichen, die aus den Traingsdaten für jede Aktion erstellt werden. Zum Vergleich wird über die Perplexität die Wahrscheinlichkeit berechnet, dass eine Wortfolge aus einer der Sprachen stammt. Eine niedrige Perplexität bedeutet eine hohe Wahrscheinlichkeit, dass ein Wort Teil der untersuchten Sprache und damit eine bestimmten Aktion ist. Der Aktionserkenner gibt die Aktion aus, zu welcher die niedrigste Perplexität berechnet wird.

Erkennen von Aktionen

Ziel ist die Berechnung von $P(w)$, also der Wahrscheinlichkeit, dass die gegebene Wortfolge $w = w_1, w_2, w_3....w_k$ Teil der Sprache ist. Hierfür wird die Wahrscheinlichkeit $P(w)$ faktorisiert:

$P(w) = P(w_1)P(w_2|w_1)...P(w_k|w_1, w_2, ..., w_{k-1}) =$
$\prod_{i=1}^{k} P(w_i|w_1, w_2, ..., w_{i-1})$

Um die Berechnung effizient zuhalten, wird die *Geschichte*, also die Anzahl der betrachteten Wörter, begrenzt:

$\prod_{i=1}^{k} P(w_i|w_{i-n+1}, w_2, ..., w_{i-1})$

In der Realisierung werden Bi- ($n = 2$) oder Trigramme ($n = 3$) verwendet. Die Perplexität ist die mittlere Anzahl möglicher Fortsetzungen einer Teilwortfolge aus der Sprache. Eine geringe Perplexität lässt also darauf schließen, dass das gegebene Wort zur Sprache gehört. Sie ist definiert als:

$PP(L) = 2^{H(L)}$ und kann für N-Gramme mit
$PP(L) \approx P(w_1, w_2, ..., w_n)^{\frac{1}{N}}$. abgeschätzt werden.

Zur Berechnung der bedingten Wahrscheinlichkeiten werden zunächst die Vorkommensanzahlen der N-Gramme und der N-1-Gramme ermittelt.
$P(w_n|w_1, w_2, ..., w_{n-1}) = \frac{c(w_1,w_2,...,w_n)}{c(w_1,w_2,...,w_{n-1}}$
Hierbei ensteht jedoch das Problem, dass viele N-Gramme nicht beobachtet, also *unseen events* sind. Dies führt dazu, dass die gesamte Wahrscheinlichkeit gleich null ist. Deshalb wird im Verfahren ein sinnvolles $P(z|xy)$ gewählt für $c(xyz) = 0$.

Training

Für die Adaption des Erkenners wird das Standard-Trainingsverfahren für HMMs (siehe Abschnitt 3.4.2) verwendet. Dabei werden zunächst die einzelnen Posensequenzen jeder Aktion konkateniert und jeweils das N-Gramm-Vorkommen ermittelt. Dies wird in der Zielanwendung mit der Funktion lm_count aus esmeralda realisiert. Um aus diesen Daten ein brauchbares Markov Modell zu erhalten werde neben den statistischen Vorkommen der N-Gramme noch Wahrscheinlichkeiten für nicht beobachtete N-Gramme benötigt. In der Zielanwendung wird dies mit der esmeralda-Funktion lm_param und dem Parameter *S1*, welcher für *absolute discounting* steht, realisiert. Beim *absolute discounting* wird jede empirisch ermittelte Häufigkeit um einen Konstanten Betrag verringert, anhand dessen wiederum die Nullwahrscheinlichkeit berrechnet wird. Mit der Funktion lm_perp, wird zur Laufzeit die Perplexität von einer Texttestsequenz zu einem vorher erstellen Markov-Modell berechnet.

4.7 Anbindung an das Spiel

Wie bereits im Abschnitt 2.1 erwähnt, kann die Spielfigur eine Reihe von Aktionen ausführen. In weiteren Verlauf soll betrachtet werden, wie der Spieler diese Aktionen auslösen kann. Der verwendete Emulator, in diesem Fall Mupen64plus [39], bietet die Möglichkeit, mit Hilfe von Plugins die Spielsteuerung selbst zu definieren.

4.7.1 Verarbeitung der Eingabedaten

Das verwendete Plugin empfängt die Signale der Wiimote und die berechneten Aktionen des Aktionserkenners, fasst sie zusammen und sendet sie codiert als Tasten des N64-Controller an den Emulator. Es verwendet dabei die Mupen64Plus Plugin API[3], um mit dem Emulator zu kommunizieren.

Als Basis für die Verknüpfung von Aktionserkenner und Emulator wurde das Mupen64plus-Plugin *Wiinput64* [26] verwendet. Dieses bietet grundlegende Funktionen für die Steuerung des Emulators per Wiimote und nutzt seinerseits die Bibliothek *libwiimote* [35], um die Steuersignale der Wiimote auszulesen.

Zur Kommunikation mit der Wiimote wird ein Bluetooth Dongle benötigt. Für die Steuerung der Laufrichtung der Spielfigur wurde anfänglich die Wiimote-Erweiterung *Nunchuk* verwendet, da diese im Gegensatz zur Wiimote einen Analogstick bietet. Mit diesem sind Bewegungen im Spiel präziser ausführbar, als mit dem digitalen Steuerkreuz der Wiimote. Die Steuersignale des Nunchuk werden dabei von der Wiimote versendet und analog zu anderen Eingaben von Wiinput64 verarbeitet. Die Verbindung des Nunchuk mit der Wiimote erfolgt jedoch über ein relativ kurzes Kabel. Bei einigen Aktionen, wie z.B. *Fliegen*, reicht dieses nicht aus, um die Aktion sauber auszuführen, während man beide Geräte in der Hand hält. Die Unterbringung der Wiimote in der Hosentasche o.ä. erwies sich als unkomfortabel und verfälschte die Ergebnisse der Bewegungssensoren. Ein Probelauf mit einem kabellosen Nunchuk eines Fremdherstellers zeigte außerdem, dass die verwendete Treiberarchitektur *libwiimote* auf dessen Eingabesignale nicht reagiert. Es wird daher das digitale Steuerkreuz der Wiimote zur Richtungsangabe genutzt und auf die Verwendung des Nunchuks verzichtet.

4.7.2 Einbezug der Bewegungssensorik

Für eine robustere Erkennung der vor der Kamera durchgeführten Aktionen, werden die Bewegungssensoren der Wiimote in die Steuerung des Spiels mit einbezogen. Diese gehen jedoch nicht in die Berechnungen des Aktionserkenners ein, sondern werden getrennt betrachtet. Im Controller-Plugin des Emulators werden ununterbrochen die aktuellen Beschleunigungswerte der Wiimote ausgelesen. Dabei wird für alle drei geometrischen Achsen des Controllers die in libcwiimote enthaltene Funktion *force* verwendet, die mittels

$$\sqrt{(wiimote.force.x)^2 + (wiimote.force.y)^2 + (wiimote.force.z)^2}$$

den aktuellen mittleren Ausschlag der Bewegungssensoren berechnet. Dieser wird anschließend mit einem vorgegebenen Schwellwert verglichen. Erst wenn dieser Wert überschritten ist, kann von einer Aktion ausgegangen werden und die letzte vom Aktionserkenner gelieferte Aktion wird an den Emulator weitergeleitet. Dies hilft vor allem in Situationen, in denen der Spieler keine Aktion durchführen möchte, aber der Aktionserkenner dennoch fälschlicherweise eine Aktion ausgegeben hat.

Für eine möglichst komfortable Bedienung des Spiels ist außerdem die

Synchronisierung der Signale vom Aktionserkenner mit denen der Wiimote wichtig. Da das Signal von der Kamera bis zum Spiel eine Reihe von Modulen durchläuft, benötigt es etwa 100ms bis 150ms bis die berechnete Aktion den Emulator erreicht. Auch wenn eine weitere Verkürzung dieser Zeitspanne wünschenswert wäre, ist diese für den Anwendungsfall durchaus akzeptabel. Die Steuersignale der Wiimote, welche direkt mittels Bluetooth mit dem Spielerechner kommuniziert, treffen jedoch ohne erwähnenswerte Verzögerung direkt beim Controller-Plugin des Emulators ein. Da das Zusammenwirken beider Eingabegeräte das Spielgefühl maßgeblich beeinflusst, wurde daher eine Verzögerung der Wiimotesignale in das Plugin integriert. Nach einigen Probeläufen erwies sich eine Verzögerung um 100ms als praktikabel. Aufgrund späterer, genauerer Messungen (siehe Tabelle 18) wurde die künstliche Verzögerung des Wiimote-Signals entsprechend angepasst.

4.7.3 Spielbare Aktionen

In der Tabelle 1 werden die Aktionen des Spielers, die dafür benötigten Tastenkombinationen (dargestellt als N64-Controller Tasten) und die Reaktionen der Spielfigur aufgelistet.

Der Spieler...	Tasten des N64	Aktion der Spielfigur
...boxt	A-Taste	Banjo boxt.
...duckt sich	Z-Taste	Banjo duckt sich.
...macht eine Wurfbewegung	Z-Taste kombiniert mit Richtungsangabe vom Steuerkreuz	Banjo wirft etwas nach vorne oder nach hinten.
...springt	B-Taste	Banjo springt.
...duckt sich und springt anschließend	Z-Taste, dann B-Taste	Banjo springt besonders hoch.
...breitet die Arme aus	B-Taste	Banjo fliegt ein Stück mit Kazooie.
...springt, während er die Arme ausbreitet	B-Taste, dann eine kurze Wartezeit und dann wieder B-Taste	Banjo springt und fliegt anschließend mit Kazooie.
...springt und führt eine Hackbewegung mit der Hand über seinem Kopf aus	B-Taste, dann A-Taste	Banjo springt und Kazooie hackt mit seinem Schnabel.
...springt und duckt sich danach	B-Taste, dann Z-Taste	Banjo springt und stampft auf den Boden.

Tabelle 1: Zuordnungstabelle von Aktion zu N64-Controller Tasten

Der Ausfallschritt wurde aus Gründen der Vereinfachung nicht mit einbezogen, da der Spieler auch ohne diese auskommt.

Abbildung 21: N64 Controller

Laufen, *Richtungsänderung* und *schnell Rennen* führt der Spieler mit dem Steuerkreuz und der Z-Taste aus. Beim Werfen wird eine Richtungsangabe, in die Banjo werfen soll, benötigt. Diese erfolgt ebenfalls mit dem Steuerkreuz.

5 Evaluierung

Das folgende Kapitel beschäftigt sich mit der qualitativen Betrachtung des Systems. Zunächst werden die verwendeten Datensätze beschrieben und die angelegten Qualitätskriterien erläutert. Da die einzelnen Module des Systems unterschiedliche Datentypen verwenden, kann für die Evaluation kein einheitlicher Datensatz auf alle Teile der Pipeline angewendet werden. Ebenso sind angelegte Kriterien unterschiedlich und werden daher in den Unterkapiteln behandelt. Zur Verbesserung dienende Änderungen an der Software werden gegebenenfalls erläutert. Die Unterteilung des Kapitels findet anhand der Pipeline-Struktur statt. Nach der Zusammenfassung der Teilergebnisse werden abschließend die Ergebnisse der Benutzertests beschrieben, um außer Berechnungsergebnissen auch das „Spielgefühl“ zu bewerten. Da für die Erstellung der Videodaten ein gegebenes System verwendet wird, beginnt die Evaluierung der Software mit der qualitativen Betrachtung des ROI-Moduls.

5.1 ROI

Zur Beurteilung der Qualität der eingesetzten Verfahren zur Berechnung der ROI und des Einflusses der verschiedenen Parameter ist die Entwicklung eines Auswertungsverfahren notwendig. Um die Nachvollziehbarkeit und damit auch die Vergleichbarkeit der Ergebnisse für Außenstehende zu gewährleisten, ist die eindeutige und klare Definition der Qualitätsmaße von zentraler Bedeutung. Um weiterhin sicherzustellen, dass die ermittelten Qualitätswerte auch die wirkliche Leistungsfähigkeit der eingesetzten Verfahren widerspiegeln, ist die sorgfältige Auswahl der Eingabedaten besonders wichtig. Hier sollte eine für den typischen Einsatz des Systems repräsentative Menge von Testdaten ausgewählt werden, die das üblicherweise vorkommendes Spektrum von Bildfolgen abdecken.

5.1.1 Evaluierungssystem

Den Qualitätsmaßstab für die optimale ROI liefern manuell vorgenommene Annotationen der Eingabebilder. Diese enthalten pro Bild genau eine Region, die die Person genau umschließt. Die berechneten und zu bewertenden Daten enthalten pro Bild ebenfalls eine Region. Diese stellt das, vom ROI-Modul mit dem jeweiligen Verfahren, berechnete Endergebnis nach Anwendung aller Optimierungen der ROI da. Aufgrund der manuellen Annotationen, die eine optimale ROI nur annähern, ist eine absolut exakte Übereinstimmung zwischen Annotation und Berechnungsergebnis nicht zu erwarten.

Um, unter Berücksichtigung der oben beschriebenen Anforderungen, möglichst aussagekräftige Qualitätsmaßstäbe zu definieren, bieten sich die im Rahmen des *Information Retrieval* verwendeten Konzepte von *Precision* und *Recall* an. Diese liefern etablierte und robuste Maße, um die Güte von Treffermengen bei der Suche nach Dokumenten zu beurteilen. Die Menge der vorhandenen Dokumente läßt sich in relevante und irrelevante Dokumente trennen. Dabei ist es Ziel jeder durchgeführten Suche, möglichst alle relevanten Dokumente zu finden, während dabei keine irrelevanten Dokumente in den Ergebnissen erscheinen. Um die etablierten Qualitätsmaße zur Evaluierung der ROI zu verwenden, ist es notwendig, die Begriffe von relevanten, irrelevanten und gefundenen Dokumenten in diesem Kontext entsprechend zu definieren. Dabei stellt die Pixel-Weise Sicht auf die Eingabedaten einen sinnvollen Ansatz da, wobei jeder Pixel einem Dokument entspricht. Hierbei repräsentiert die berechnete ROI die Ergebnismenge der Suche, wogegen die annotierte ROI die Einteilung in relevante und irrelevante Dokumenten darstellt. Jeder betrachtete Pixel in jedem annotierten und verarbeiteten Bild liegt also in genau zwei der folgenden Mengen:

1. Innerhalb berechneter ROI ($\widehat{=}$ Menge der gefundenen Dokumente, P)
2. Außerhalb berechneter ROI ($\widehat{=}$ Menge der nicht gefundenen Dokumente)
3. Innerhalb annotierter ROI ($\widehat{=}$ Menge der relevanten Dokumente, R)
4. Außerhalb annotierter ROI ($\widehat{=}$ Menge der irrelevanten Dokumente, I)

Gemäß ihrer Definition gibt die *Precision* den Anteil der annotierten gefundenen Pixel an allen gefunden Pixeln an. Damit kann sie als Maß für die Genauigkeit der Ergebnisse angesehen werden. Dem gegenüber stellt der *Recall* der Anteil der annotierten gefunden Pixel an allen annotierten Pixeln dar. Er kann als Maß für die Trefferquote bzw. die Vollständigkeit der Ergebnisse gesehen werden. Zur Verdeutlichung der beschriebenen Zusammenhänge folgt eine kurze formale Definition.

$$Recall = \frac{|R \cap P|}{|R|}$$
$$Precision = \frac{|R \cap P|}{|P|}$$

Damit kann in unserem Kontext der *Recall* als die Wahrscheinlichkeit, mit der ein Pixel der annotierten ROI auch tatsächlich gefunden[1] wird, interpretiert werden, wogegen die *Precision* die Wahrscheinlichkeit, mit der ein Pixel innerhalb der berechneten ROI tatsächlich gesucht[2] ist, angibt.

Zur Berechnung der Maße ist es notwendig, die Kardinalität der oben aufgelisteten Mengen (und der ihrer Schnittmengen) zu bestimmen. Dabei

[1] Gefunden bezieht sich hierbei auf die Zugehörigkeit zur berechneten ROI.
[2] Gesucht bezeichnet hier die Zugehörigkeit zur annotierten ROI.

bietet es sich an, über alle Pixel zu iterieren und dabei, je nach Zugehörigkeit des aktuellen Pixels, die Kardinalität der entsprechenden Mengen zu erhöhen. Um für jedes Bild eine eindeutige Qualitätsmaßzahl zu ermitteln, wird abschließend der gewichtete Mittelwert f von *Recall* und *Precision* gebildet. Er ergibt sich als:

$$f = \frac{1}{\alpha/precision+(1-\alpha)/recall}$$

Durch die Wahl des Faktors α kann die Gewichtung zwischen *Recall* und *Precision* angepasst werden, wobei für α typischerweise $0,5$ gewählt wird.

5.1.2 Eingabedaten

Als Eingabemenge für die Evaluierung der ROI wurden 5 typische Spielsequenzen ausgewählt, die insgesamt ca. 2000 Bilder enthalten. Die Auswertung der, mit den verschiedenen Parametern berechneten, Regionen wurde durch ein selbstentwickeltes Tool und diverse Skripte vorgenommen. Dabei wird für jede interessante Parameterkombination für jedes Bild der F-Wert gemäß der oben beschriebenen Verfahren bestimmt. Der Ergebnis F-Wert einer Parameterkombination ergibt sich aus dem Durchschnitt über allen Bildern.

5.1.3 Evaluierung der Parameterwahl

Bevor die einzelnen Modelle miteinander verglichen werden können, ist es zunächst notwendig die für jedes Modell optimale Parameterkombination zu ermitteln. Für die Berechnung der dynamischen ROI wird das in Abschnitt 3.2.1 beschriebene Verfahren MHI genutzt, welches viele Möglichkeiten der Parametrisierung bietet.

Eine entscheidende Rolle spielt in diesem Verfahren die Wahl der Funktion, welche den zeitlichen Verfall der Pixelwerte beschreibt. Im Folgenden werden Auswirkungen eines asymptotischen, linearen und logistischen Verfalls untersucht. Jede dieser Funktionen ist so implementiert worden, dass sie über einen Parameter k entsprechend angepasst werden kann. n bezeichnet die Anzahl der Bilder, die ein Wert höchstens anhalten kann.

Bei der asymptotischen Funktion errechnet sich diese wie folgt:

$$n = \lceil log(0.1)/log(k) \rceil$$

Es gilt also für den Parameter $0 < k < 1$ und die Funktion bricht ab, sobald ein Schwellwert von 0.1 erreicht wurde.

Im Fall der linearen Funktion bezeichnet der Parameter direkt die Anzahl der Bilder, über die ein Wert bestehen kann.

Für die logistische Funktion gilt die Gleichung:

$$n = \lceil 16/k \rceil$$

Der Wert 16 stellt sicher, dass eine ausreichende Granularität der Länge eingestellt werden kann.

Da der Updatefaktor der MHI-Funktion einen großen Einfluss auf das Ergebnis besitzt, wird zunächst nur dieser evaluiert. Anschließend folgt eine Evaluation der Parameter der unterschiedlichen Verfallsfunktionen, da keine große Abhängigkeiten zum Updatefaktor zu erwarten sind. Zunächst werden die Parameter der einzelnen Funktionen so gesetzt, dass die länge der Bilder 3 beträgt. Geht man davon aus, dass die Kamera 25 Bilder pro Sekunde liefert, entspricht dies einer zeitlichen Auflösung von 120ms. Höhere Werte könnten die Reaktivität des Systems gefährden. Somit ergeben sich nach den oben genannten Formeln die Werte 0,4 für die asymptotische Funktion, 3 für die lineare und 6 für die logistische.

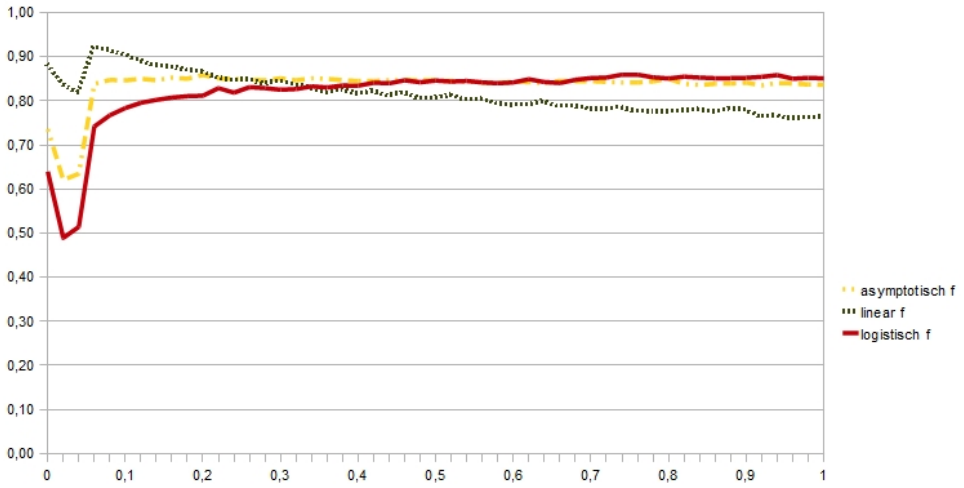

Abbildung 22: Ermittelter F-Wert bei entsprechendem Updatefaktor für die einzelnen Funktionen

Aus Abbildung 22 lässt sich ablesen, dass der Updatefaktor unter 0.1 nur schlechte Resultate liefert. Insgesamt hat der Faktor einen besonders großen Einfluss auf die lineare Funktion, während die asymptotische und logistische Funktion nur gering beeinflusst werden. Die Tabelle 2 beschreibt die jeweils besten Werte für die Funktionen.

Nachdem der optimale Updatefaktor für jede Funktion ermittelt wurde, können unterschiedliche Werte der Parameter evaluiert werden. Um sinnvolle Werte betrachten zu können, ist es zunächst notwendig, sich die Funktionen

Funktion	Updatefaktor	F-Wert
Asymptotisch	0.2	0.85
Linear	0.1	0.85
Logistisch	0.94	0.84

Tabelle 2: Beste ermittelte Werte für die Funktionen

anzuschauen.
Die asymptotische Funktion ist wie folgt definiert:

$$f(x) = k^x, 0 < k < 1$$

Die lineare Funktion:

$$f(x) = 255 * (k - x)/x$$

Die Gleichung für die logistische Funktion:

$$f(x) = 1 - 1/(1 + exp(-k * (x + 8)))$$

Aufgrund der Beschräkung für die asymptotische Funktion $0 < k < 1$ werden für k Werte in 0.02 Schritten von 0.02 an betrachtet.

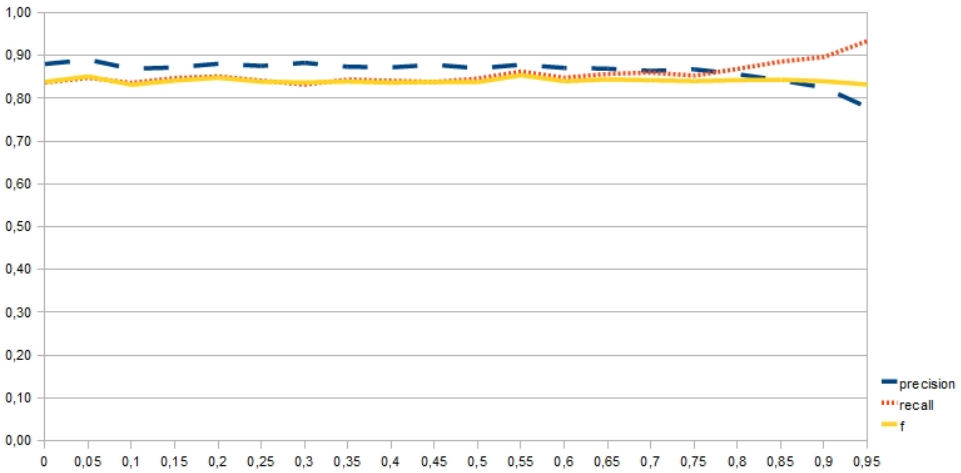

Abbildung 23: Ermittelter F-Wert bei entsprechendem Parameter für die asymptotische Funktion

Aus dem Graphen in Abbildung 23 ist ersichtlich, dass ein Wert von 0.55 als optimaler Parameter für die Funktion dient. Werte über 0.8 liefern nur noch unbrauchbare Ergebnisse.

Die lineare Funktion wird mit Werten von 0 bis 20 in Einerschritten evaluiert.

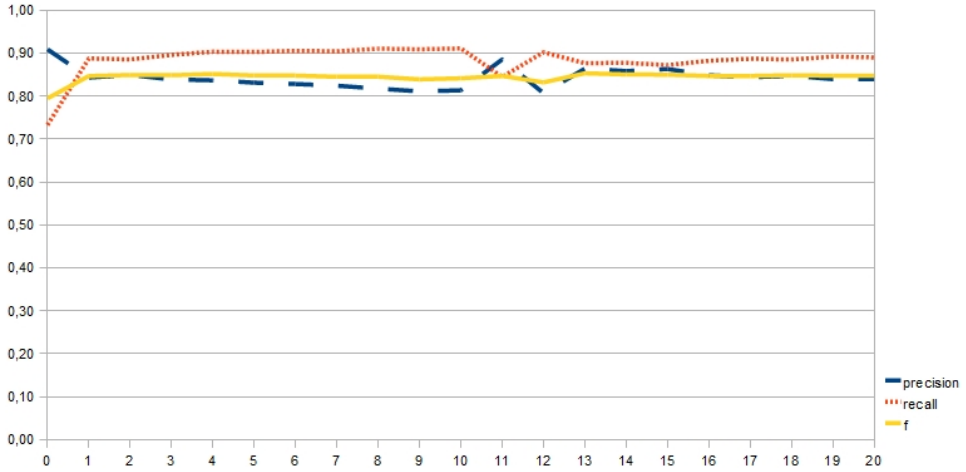

Abbildung 24: Ermittelter F-Wert bei entsprechendem Parameter für die linea re Funktion

Ergebnisse hierzu sind in Abbildung 24 dargestellt. Interessanterweise gibt es zwischen den Werten 11 und 13 höhere Fluktuationen. Es ist denkbar, dass diese durch Sequenzen wie Springen verursacht werden. Solch eine Aktion verändert schlagartig die Größe der ROI nach oben hin und passt auch zeitlich in das Fenster.

Aufgrund der Beschaffenheit der logistische Funktion werden zunächst Werte von 0 bis 1 in 0.05er Schritten betrachtet, anschließend die Werte 2, 4, 8, 16.

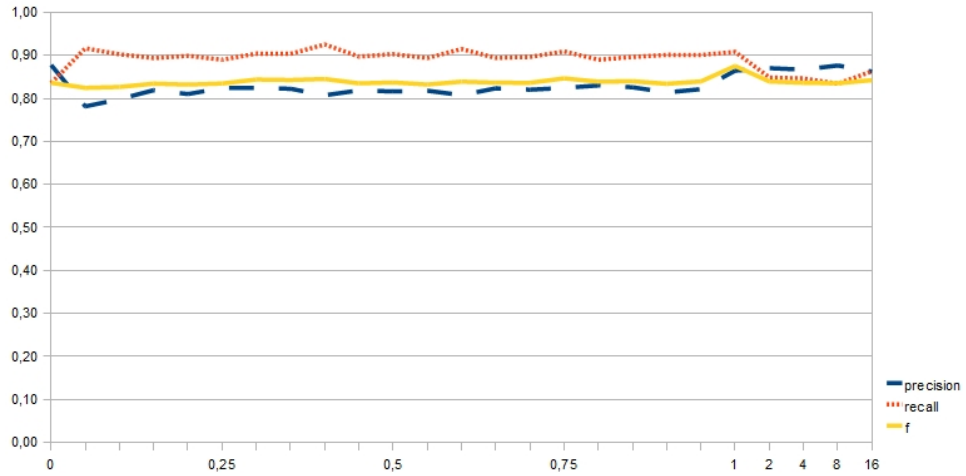

Abbildung 25: Ermittelter F-Wert bei entsprechendem Parameter für die logis tische Funktion

Funktion	Parameter	F-Wert
Asymptotisch	0.55	0.85
Linear	4	0.85
Logistisch	1.0	0.88

Tabelle 3: Beste ermittelte Werte für die Funktionen

Wie in Abbildung 25 ersichtlich, liefert ein Wert von 1.0 hierbei das beste Ergebnis für die Funktion.

Bei Betrachtung der Evaluationsergebnise des dynamischen Hintergrundmodells, fällt der relativ geringe Einfluss der einzelnen Parameter auf die Qualität der ROIs auf. Unter Verwendung der logistischen Verfallsfunktion können um 4% größere F-Werte als mit den beiden anderen Funktionen erreicht werden. Die Ergebnisse sind in Tabelle 3 veranschaulicht. Dabei sei darauf verwiesen, dass alle bisher ausgewerteten Funktionen ohne Qualitätsfeedback des Posenerkenners durchgeführt wurden.

Da beim statischen Hintergrundmodell lediglich der Schwellenwert variiert werden kann, wird hier auf eine separate Behandlung verzichtet und auf den folgenden Abschnitt verwiesen.

5.1.4 Evaluierung der Modelle

Neben den weiter oben für die einzelnen Modelle evaluierten optimalen Parametern stellt der Schwellenwert, ab welchem Modellergebnis ein Pixel dem Vordergrund zugeordnet wird, eine wichtige, allen Modellen gemeinsame, Einflussgröße dar. Durch ihn wird die Empfindlichkeit des Modells festgelegt, womit er sich direkt auf die Ergebnisqualität auswirkt. Darüber hinaus stellt sich die Frage, ob die Rückmeldung über gute ROIs vom Posenerkenner wirklich zu einer Verbesserung der berechneten ROIs beiträgt. Dazu wurden die Messserien des Schwellenwerts einmal mit und einmal ohne Qualitätsfeedback durchgeführt und die Ergebnisse gemeinsam in einem Diagramm dargestellt. In Abbildung 26 ist für jede Einstellung (Statisch mit, Statisch ohne, Dynamisch mit, Dynamisch ohne) die Auswirkung der verschiedenen Schwellenwerte auf den F-Wert dargestellt.

Neben der Ergebissqualität der ROIs stellt in einem reaktiven System auch die Laufzeit ein wichtiges Entscheidungskriterium dar. Die ersten Implementierungen des statischen Hintergrundmodells verwendeten die selbe Bibliothek die auch zur Berechnung der dynamischen Modelle zum Einsatz kommt, wobei die Umsetzung der Hintergrundsubtraktion lediglich durch entsprechende Parameterwahl (D-Funktionsparameter = 0, Updatefaktor = 0) erfolgte. Dadurch benötigten alle Modelle nahezu die gleiche Berechnungszeit. Im Rahmen der Evaluierung wurde eine eigenen Implementierung der

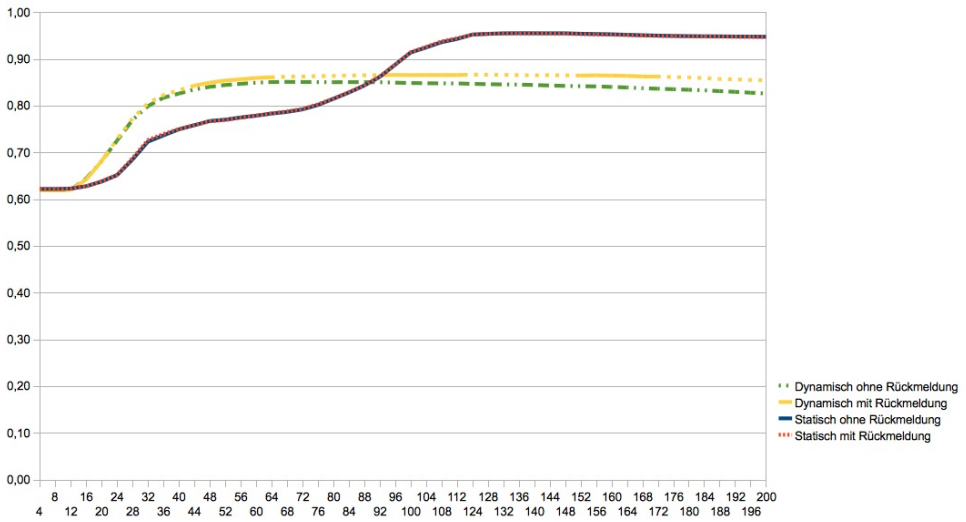

Abbildung 26: Gemeinsame Darstellung der F-Werte für die beiden besten Modelle

Hintergrundsubtraktion umgesetzt, durch die eine Reduzierung der durchschnittlichen Berechnungszeit des statischen Modells um 70 % gegenüber dem dynamischen Modell erreicht werden konnte.

5.1.5 Bewertung der Ergebnisse

Wie Abbildung 26 entnommen werden kann, werden die besten Ergebnisse mit dem statischen Hintergrundmodell und einem Schwellenwert von 136 erreicht. Für das dynamische Hintergrundmodell liefert ein Schwellenwert von 124 die besten Ergebnisse.[3] Weiterhin ist festzustellen, dass die Rückmeldung des Posenerkenners keinen Einfluss auf die Ergebnissqualität des statischen Hintergrundmodells hat, und lediglich zu einer minimalen Verbesserung, mit um 1,5 % größerem F-Wert, des dynamischen Modells führt.

Das statische Hintergrundmodell erreicht bei optimaler Wahl des Schwellenwerts (136) mit einem F-Wert von 0,9551 eine sehr gute und für die Zielanwendung absolut ausreichende Ergebnissqualität, welche eine deutliche Verbesserung gegenüber den dynamischen Modellen darstellt. Mit diesem hohen F-Wert ist sichergestellt, dass die berechnete ROI nur minimal von der gesuchten ROI abweicht und somit optimale Voraussetzungen für die weitere Verarbeitung des Bildausschnitts liefert. Allein aufgrund der Tatsache, dass manuelle Annotationen eine optimale ROI nur annähern, ist eine

[3] Hierbei sei angemerkt, dass es sich um den Schwellenwert für die Berechnung auf Farbbildern handelt. Wird mit Graustufen gearbeitet, ist der Wert durch drei zu dividieren.

absolut exakte Übereinstimmung zwischen Annotation und Berechnungsergebnis nicht zu erwarten. Neben der höheren Ergebnisqualität spricht auch die deutlich kürzere Berechnungszeit für das statische Hintergrundmodell. Wenn, wie bei dem hier betrachteten Anwendungsfall, ein konstanter Hintergrund angenommen werden kann, stellt zur Bestimmung der ROI das statischen Hintergrundmodell mit einem Schwellenwert von 136 die unter allen Gesichtspunkten beste Option dar.

5.2 Posenerkenner

Der Posenerkenner führt auf der Ausgabe der ROI die Merkmalsextraktion für den relevanten Bildbereich durch. Diese erfolgt mittels des bereits vorgestellten HOG-Verfahren. Die Konfiguration der Parameter zur Berechnung der Merkmalsvektoren wird nach den in [40] empfohlenen Werten durchgeführt.

5.2.1 Evaluierungssystem

Das Qualitätsmaß des Posenerkenners ist die Erkennungsrate der Posen. Für die Ermittlung dieser sei M die Anzahl der zu klassifizierenden Posen, K die Anzahl der korrekt klassifizierten Beispiele der Pose n und A die Anzahl aller Beispiele der Pose n.

$$\text{Erkennungsrate} = \sum_{n=1}^{M} \left(\frac{K_n}{A_n}\right) \setminus M$$

Die Normierung über die einzelnen Posen stellt sicher, dass die Anzahl der Merkmalsvektoren der Posen keinen Einfluss auf das Ergebnis nimmt.

Bei der Erstellung eines KNN sind neben den Trainings- und Validierungsdaten verschiedene Parameter von Bedeutung.

- Anzahl der Neuronenschichten
- Anzahl der Neuronen innerhalb der Schichten
- Lernrate
- Anzahl der Trainingsdurchläufe

Die Lernrate ist der Faktor, um den sich die Gewichte eines Neurons ändern, wenn ein Beispiel falsch klassifiziert wurde. Die Anzahl der Trainingsdurchläufe gibt an wie oft, durch eine Änderung der Gewichtung, versucht werden soll ein besseres Ergebnis zu erzielen. Die Ergebnisse eines Durchlaufs werden nur gespeichert, wenn ihre Fehlerrate geringer ist, als die bisherige minimale

Fehlerrate. Daher kann ein größerer Durchlaufwert keine schlechteren Ergebnisse liefern, als ein kleinerer. Ein zu großer Wert hat eine längere Laufzeit und damit eine höhere Nutzung von Rechenkapazitäten zur Folge. Bei Tests zeigt sich, dass sich nach ca. 150 Durchläufen keine Verbesserungen mehr einstellen. Um sicher zu stellen, dass eventuelle Ausreißer berücksichtig werden, wird die Anzahl der Trainingsdurchläufe auf bei allen Netzen auf 200 festgelegt.

5.2.2 Eingabedaten

Für die Klassifizierung der Beispieldaten werden Merkmalsvektoren benötigt, von denen bekannt ist, welche Pose sie repräsentieren. Diese Merkmalsvektoren entsprechenen denen in Abschnitt 3.3 vorgestellten HOG-Vektoren. Solche Merkmalsvektoren werden für die Trainingsphase des KNN erstellt.

Für die Erstellung der als Eingabe verwendeten Merkmalsvektoren sind mehrere Schritte notwendig. Zunächst werden für jede der zu erkennenden Posen Einzelbilder benötigt, aus denen der entsprechende Merkmalsvektor extrahiert werden kann. Um eine zusätzliche Varianz in die Bilder zu bringen, sollten hierbei Bildermengen von vielen verschiedenen Personen vor unterschiedlichem Hintergrund genutzt werden. Diese Bilder sind den vorhandenen Posen zuzuordnen. Um die Merkmalsvektoren der einzelnen Bilder zu erhalten, wurde die Software so modifiziert, dass das Ergebnis der Merkmalsextraktion, anstatt klassifiziert zu werden, in eine Datei geschrieben wird. Beim Erstellen dieser Trainingsdaten wird noch nicht die in Abschnitt 4.4.4 beschriebene Hintergrundsubtraktion angewendet. Durch die Berechnung ist sichergestellt, dass der Merkmalsvektor dem von der ROI ermitteln Bereich entspricht. Durch die Zuordnung der Bilder, zu den zu erkennenden Posen, ist auch die benötigte Zuordnung der Merkmalsvektoren gegeben.

In diesem Fall bestehen die Trainingsdaten aus Aufnahmen von neun Personen vor drei verschiedenen Hintergründen. Die Bilder jeder Person sind dabei zu einer Bildmenge zusammengefasst. Insgesamt ergeben sich 15000 Bilder, diese sind in 9 disjunkte Bildmengen unterteilt. Wie in Abschnitt 4.5.4 erwähnt, sind diese Daten noch in Trainings und Validierungsdaten aufzuteilen. Dazu wurden sechs Bildmengen als Trainingsdaten und drei Bildmengen als Validierungsdaten genutzt. Die schon aus Abschnitt 4.5.1 bekannte Abbildung 27 zeigt 10 der 14 zu klassifizierenden Posen. Die Posen „werfen“ , „hacken“ , „boxen_1“ und „boxen_2“ existieren jeweils auch für die Ausführung mit dem anderem Arm.

Zur Bewertung der Netze wurde analog ein weiterer Datensatz, von einer zehnten Person vor einem vierten Hintergrund, erstellt. Dieser besteht

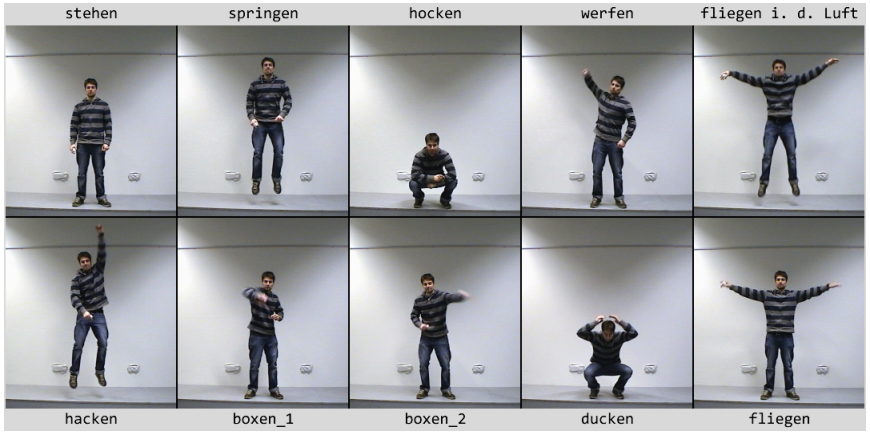

Abbildung 27: Beispiel der Posen aus dem Trainingsdatensatz

aus 1400 Merkmalsvektoren. Die Abbildung 28 zeigt die 10 Posen dieses Datensatzes.

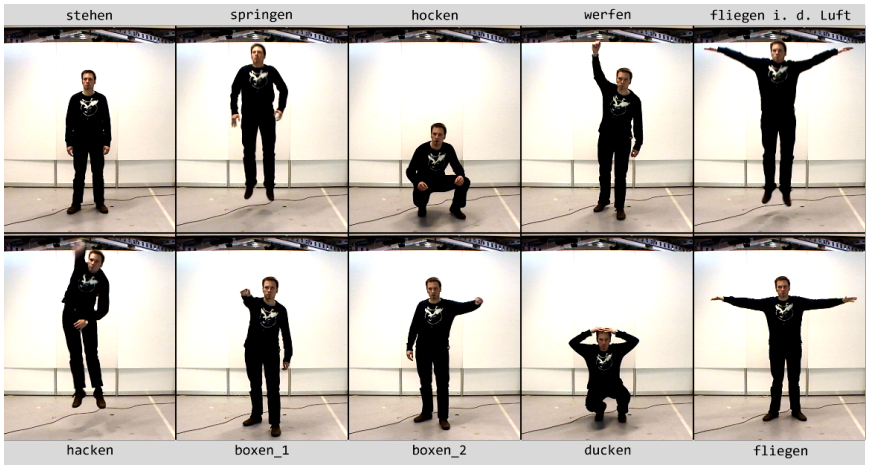

Abbildung 28: Beispiel der Posen aus dem Testdatensatz

5.2.3 Evaluierung der Parameter

Da die zum Trainieren des KNNs eingesetzte Software die Eingabereihenfolge der Merkmalsvektoren zufällig wählt, ist es nötig, für jede Parametrisierung mehrere Netze zu erstellen. Von diesen Netzen wird nur das jeweils Beste betrachtet. So wird sichergestellt, dass die Qualität der Parameter und nicht die Reihenfolge der Eingabe der Merkmalsvektoren verglichen wird.

Ausgewertet wird nach der durchschnittlichen Erkennungsrate der einzelnen Schlüsselposen, wobei hier Posen, die zur selben Aktion führen sollen,

Pose	Erkennungsrate
STEHEN	58.8 %
IN DER LUFT	36.8 %
HOCKE	95.3 %
WERFEN	84.1 %
FLIEGEN	100.0 %
FLUG LUFT	80.0 %
LUFT HACK	35.0 %
BOXEN	66.3 %

Tabelle 4: Erkennungsrate nach Posen unterteilt

zusammengefasst werden. Eine Verwechslung der Pose „boxen_1“ mit „boxen_2“ oder der Posen „hocken“ und „ducken“ aus Abbildung 28 wird nicht als Fehler gewertet, da für beide das Label „*BOXEN*“ bzw. "*DUCKEN*“ an den Aktionserkenner weitergegeben wird. Das zweischichtige Netz mit 729 Eingangs- und 14 Ausgabeneuronen erreicht, nach einem Training mit einer Lernrate von 0.9, eine Erkennungsrate von 69.6 %. Die Tabelle 4 zeigt die Erkennungsrate der einzelenen Posen.

Um eine Steigerung der Erkennungsrate zu erreichen, werden verschiedene Netze mit einer und zwei Zwischenschichten trainiert. Insgesamt werden für 46 verschiedene Parametrisierungen Netze mit drei Schichten verglichen. Diese unterscheiden sich in der Anzahl der Neuronen der Zwischenschicht und der gewählte Lernrate. Die Anzahl der Neuronen der Zwischenschicht wird von initial 40 in 30er Schritten auf 700 angehoben. Weiterhin werden 364 vierschichtige Neze getestet. Bei diesen wird darauf geachtet, dass alle Wertebereiche zwischen den 729 Eingabeneuronen und 14 Ausgabeneuronen von Kombinationen der Neuronenanzahl der Zwischenschichten abgedeckt werden. Für jede Neuronenkombination wird ein Netz mit der Lernrate 0.5 und 0.9 trainiert. Abbildung 29 zeigt die durchschnittliche Erkennungsrate, unterteilt nach der Neuronenanzahl der ersten bzw. einzigen Zwischenschicht. Es ist deutlich zu erkennen, dass die Netze mit drei Schichten, denen mit vier überlegen sind.

Während sich die Erkennungsrate der dreischichtigen Netze in jedem Bereich auf einem ähnlichen Niveau befindet, steigt die Erkennungsrate bei den vierschichtigen Netzen im Bereich mit wenigen Neuronen in der ersten Zwischenschicht an. Die durchschnittliche Erkennungsrate der dreischichtigen Netze liegt noch unterhalb der Erkennungsrate des zweischichtigen Netzes von $\sim$70%.

Die Erkennungsraten der besten Netze werden in Abbildung 30 dargestellt. Die dreischichtigen Netze, mit wenigen Neuronen in der Zwischenschicht, sind für das gegebene Problem der Posenklassifizierung am besten geeignet.

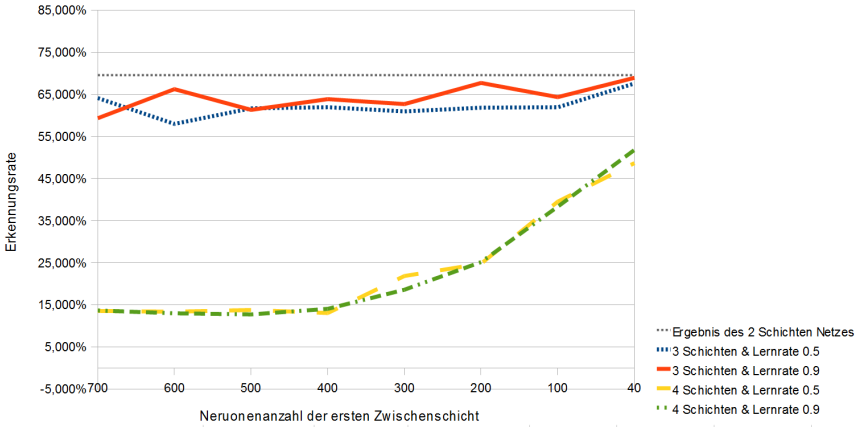

Abbildung 29: Durchschnittliche Erkennungsrate auf die Testmenge ohne Nutzung der Hintergrundsubtraktion

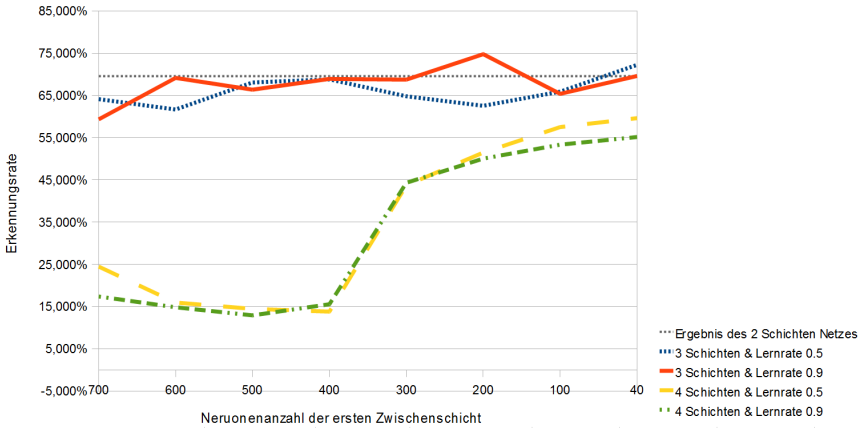

Abbildung 30: Beste Erkennungsrate auf die Testmenge ohne Nutzung der Hintergrundsubtraktion

Bei dem besten ermittelten Netz, das mit 250 Neuronen in der Zwischenschicht mit einer Lernrate von 0.9 trainiert wird, liegt die Erkennungsrate bei 74.8% und somit um 5.2% höher als bei dem Ausgangsnetz mit zwei Schichten. Die Erkennungsrate für die einzelnen Schlüsselposen wird in Tabelle 5 dargestellt. Kein Netz mit vier Schichten, konnte die Erkennungsrate des zweischichtigen Netzes erreichen. Das beste vierschichtige Netz, das bei einer Lernrate von 0.5 mit 60 Neuronen in der ersten Zwischenschicht und 20 Neuronen in der zweiten Zwischenschicht trainiert wird, hat eine um 10% schlechtere Erkennungsrate als das zweischichtige Netz. Dieses unerwartete Ergebnis weist auf Überadaption oder eine mangelnde Anzahl von Eingabe-

Pose	Erkennungsrate
STEHEN	98.0 %
IN DER LUFT	47.3 %
HOCKE	96.2 %
WERFEN	62.4 %
FLIEGEN	96.8 %
FLUG LUFT	100.0 %
LUFT HACK	25.0 %
BOXEN	74.5 %

Tabelle 5: Erkennungsraten des besten Netzes mit drei Schichten ohne Hintergrundsubtraktion nach Posen unterteilt

daten hin.

Steigerung der Erkennungsrate durch die Anwendung eines Filters auf die Merkmalsvektoren

Beim Erstellen der Trainings-, Validierungs- und Testmengen werden die Merkmalsvektoren, mittels des in Abschnitt 4.4.4 beschriebenen Verfahren der Hintergrundsubtraktion, manipuliert. Ein zweischichtiges Netz erreichte, auf den Testdaten eine Erkennungsrate von 66.2%. Zweischichtige Netze erreichen durch die Anwendung der Hintergrundsubtraktion keine Verbesserung. Die durchschnittlichen Erkennungsraten für drei- und vierschichtigen Netze wird in Abbildung 31 dargestellt.

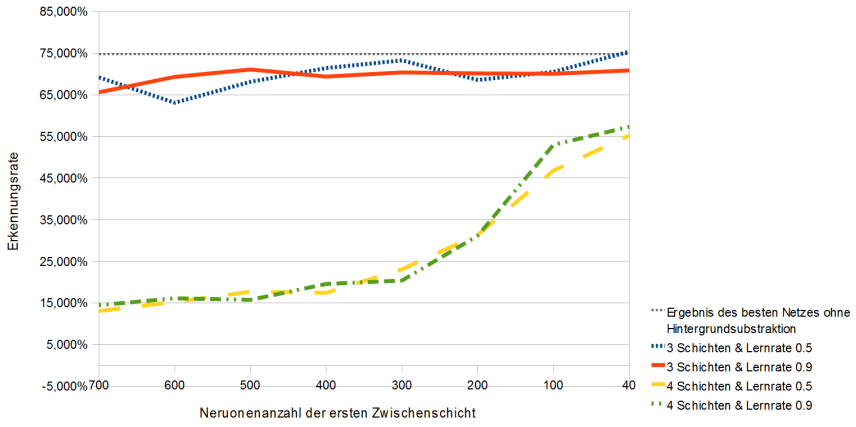

Abbildung 31: Durchschnittliche Erkennungsrate auf die Testmenge mit Nutzung der Hintergrundsubtraktion

Die drei- und vierschichtigen Netze können ihre durchschnittlichen Erkennungsraten mit der Hintergrundsubtraktion steigern. Die dreischichtigen

Pose	Erkennungsrate
STEHEN	97.7 %
IN DER LUFT	88.2 %
HOCKE	96.7 %
WERFEN	79.1 %
FLIEGEN	99.4 %
FLUG LUFT	90.9 %
LUFT HACK	33.3 %
BOXEN	55.1 %

Tabelle 6: Erkennungsraten der einzelnen Schlüsselposen des besten drei schichtigen Netzes mit Hintergrundsubtraktion

Netze erzielen bessere Erkennungsraten als die vierschichtigen Netze. Abbildung 32 zeigt die besten Erkennungsraten, die auf den Testdaten mit Hintergrundsubtraktion erreicht werden. Das beste vierschichte Netz erreichte

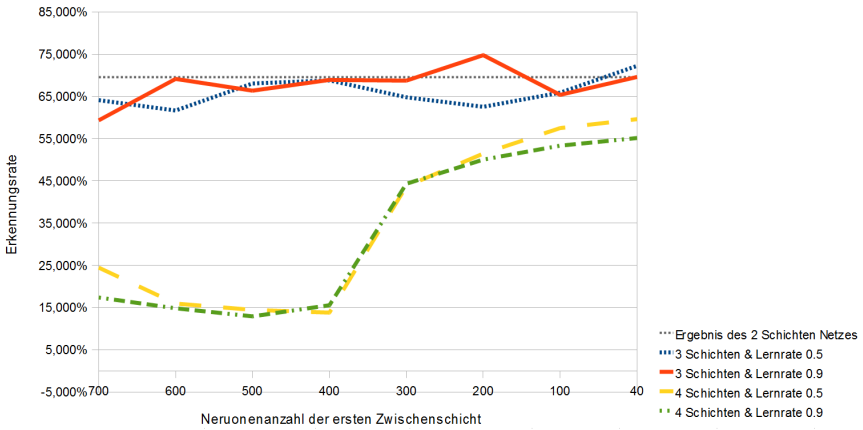

Abbildung 32: Beste Erkennungsrate auf die Testmenge mit Nutzung der Hintergrundsubtraktion

ein Erkennungsrate von 71.8%, was einer Steigerung, im Vergleich mit dem besten vierschichtigen Netz ohne Hintergrundsubstraktion, um 12.2% entspricht. Dieses wird mit 80 Neuronen in der ersten und mit 20 Neuronen in der zweiten Zwischenschicht bei einer Lernrate von 0.9 erreicht. Die Steigerung des besten dreischichtigen Netzes beträgt 5.3%. Dieses wird mit 470 Neuronen in der Zwischenschicht und mit einer Lernrate von 0.5 erreicht. Die Erkennungsrate dieses Netzes liegt bei 80.6%. Diese setzt sich aus den Erkennungsraten der einzelnen Posen, wie Tabelle 6 zeigt, zusammen. Einen Vergleich der Erkennungsraten, der Verfahren ohne und mit Hintergrundsubtraktion, zeigt Abbildung 33.

Die durchschnittliche Ausgabeaktvierung der Neuronen wird in Tabelle 7 dargestellt. Ein höherer Wert bedeutet, dass das entsprechende Ausga-

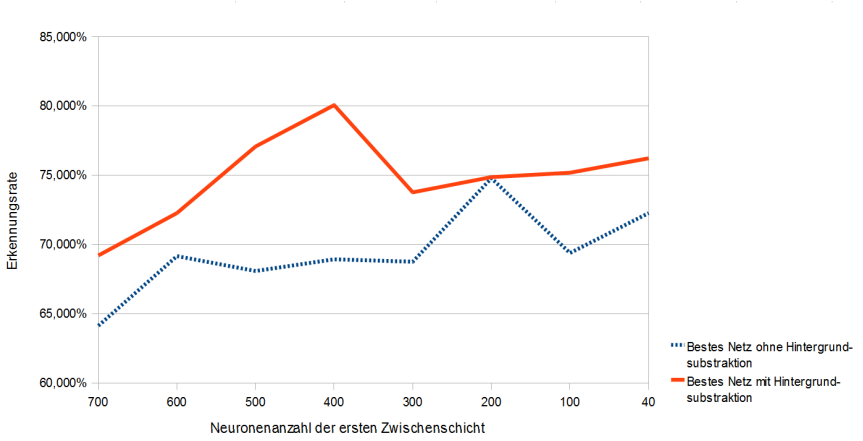

Abbildung 33: Vergleich der Erkennungsraten ohne und mit Hintergrundsubstraktion

Posen	STEHEN	IN DER LUFT	HOCKE	WERFEN
STEHEN	0.9267235	0.0002387	0.0000000	0.0356059
IN DER LUFT	0.1471678	0.7412906	0.0000000	0.0035691
HOCKE	0.0038784	0.0000000	0.9945570	0.0402916
WERFEN	0.0074290	0.0035279	0.0000001	0.5919557
FLIEGEN	0.0000000	0.0000000	0.0000000	0.0000000
FLUG LUFT	0.0000000	0.0000000	0.0000000	0.0000000
LUFT HACK	0.0000835	0.0743831	0.0000000	0.1680907
BOXEN	0.1167402	0.0192356	0.0000000	0.0956066
Posen	FLIEGEN	FLUG LUFT	LUFT HACK	BOXEN
STEHEN	0.0000000	0.0000000	0.0000000	0.0060818
IN DER LUFT	0.0000320	0.0000000	0.0000014	0.0605645
HOCKE	0.0000000	0.0000399	0.0000000	0.0747964
WERFEN	0.0000052	0.0000000	0.0407214	0.1216247
FLIEGEN	0.9893453	0.0000074	0.0000000	0.0177961
FLUG LUFT	0.1709645	0.9837126	0.0000000	0.0000008
LUFT HACK	0.0000000	0.0000000	0.2447260	0.1754472
BOXEN	0.0056333	0.0000000	0.0082253	0.3531328

Tabelle 7: Durchschnittliche Werte der Ausgabeneuronen des besten KNN

beneuron stärker aktiviert ist. Das KNN klassifiziert den Merkmalsvektor anhand der stärksten Aktivierung eines Ausgabeneurons. Bei einem Merkmalsvektor, der die zu 97% erkannte Pose „Stehen“ repräsentiert, ist die durchschnittliche Aktivierung des Ausgabeneurons mit einem Wert von 0.93 um 0.87 höher als der zweithöchste Wert, der der Pose „Boxen“ zugeordnet ist. Wenn ein Merkmalsvektor „Boxen“ repräsentiert, ist das zugehörige Ausgabeneuron nur um einen Wert von 0.24 stärker aktiviert, als das Ausgabeneron der Pose „Stehen“. Die durchschnittliche Aktivierung der schlecht erkannten Posen ist deutlich geringer, als die der gut erkannten Posen.

Absicherung der Klassifizierung durch Schwellwerte

Durch die Einführung von Schwellwerten für die Aktivierung der Ausgabeneuronen jeder Pose, kann eine Steigerung der Erkennungsrate und eine Verringerung der Fehlklassifizierungen erreicht werden. Für eine Klassifizierung ist es notwendig, dass die Aktivierung eines Ausgabeneurons seinen Schwellwert überschreitet. Überschreitet das aktivste Neuron diesen nicht, wird nach dem nächst aktivsten Neuron klassifiziert, welches seinen Schwellwert überschreitet. Ist kein Neuron stärker aktiviert als es sein Schwellwert verlangt, so wird der Merkmalsvektor als unbekannt klassifiziert. Durch die Anwendung von Schwellwerten als Klassifizierungsabsicherung ist es möglich das Qualitätskriterium zu erweitern. Die als unbekannt klassifizierten Merkmalsvektoren sind noch zu unterteilen. Ein Merkmalsvektor, der ohne Schwellwerte korrekt klassifiziert wird, aber mit diesen nicht mehr, wird als „zu unrecht abgelehnt“ bezeichnet. Die Bezeichnung „zu recht abgelehnt“ wird für die Merkmalsvektoren verwendet, die ohne Schwellwerte falsch klassifiziert worden wären.

Die Schwellwerte können jeden Wert zwischen 0 und 1 annehmen. Wenn ein Schwellwert den Wert 0 einnimmt, so wird kein Merkmalsvektor, der das entsprechende Ausgabeneuron aktiviert, abgelehnt. Bei einem Schwellwert von 1 kann kein Merkmalsvektor das entsprechende Ausgabeneuron ausreichend stark aktivieren, um danach klassifiziert zu werden. Für ein Netz mit 14 Ausgabeneuronen werden 14 Schwellwerte benötigt. Bei 14 Schwellwerten und einer Abstufung von 0.05 gibt es 20^{14} mögliche Kombinationen von Schwellwerten. Damit nicht alle getestet werden müssen, wird der Wertebereich eingegrenzt. Als Obergrenze sollte die durchschnittliche Aktivierung des zugehörigen Ausgabeneurons bei korrekter Klassifizierung nicht überschritten werden, da ansonsten zu viele korrekt klassifizerte Merkmalsvektoren abgelehnt werden. Die Untergrenze sollte einen Wert größer 0 einnehmen. In diesem Fall wird die Untergrenze auf das Maximum aus 0.001 und der um 0.5 verringerten durchschnittlichen Aktivierung gewählt. Die Abstufung der Grenzwerte wird auf 0.1 festgelegt. Für das beste gefundene Netz, mit drei Schichten und 470 Neuronen in der Zwischenschicht, existieren sieben Ausgabeneuronen, bei denen fünf verschiedene Schwellwerte möglich sind. Bei zwei Ausgabeneuronen sind vier unterschiedliche möglich. Für jeweils ein Ausgabeneuron sind zwei bzw. drei Schwellwerte möglich. Bei insgesamt drei Ausgabeneuronen ist durch die Einschränkung nur ein einziger Schwellwerte möglich. Daraus ergeben sich

$$5^7 * 4^2 * 3^1 * 2^1 * 1^3 = 7500000$$

mögliche Kombinationen. Die Abbildung 34 zeigt die Erkennungsraten und Fehlklassifizierungen der besten Netze. Die Verteilung der korrekt und falsch abgelehnten Maerkmalsvektoren zeigt die Abbildung 35.

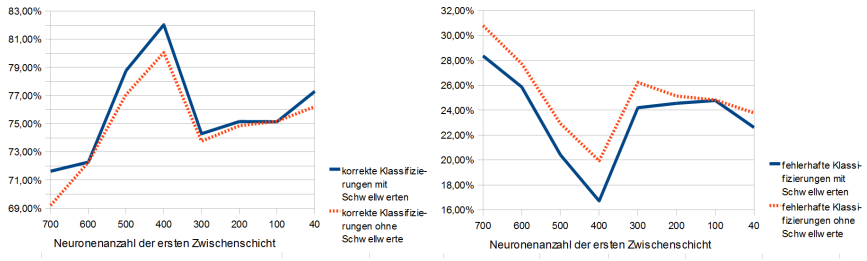

Abbildung 34: Vergleich der Erkennungsraten ohne und mit Hintergrundsubstraktion

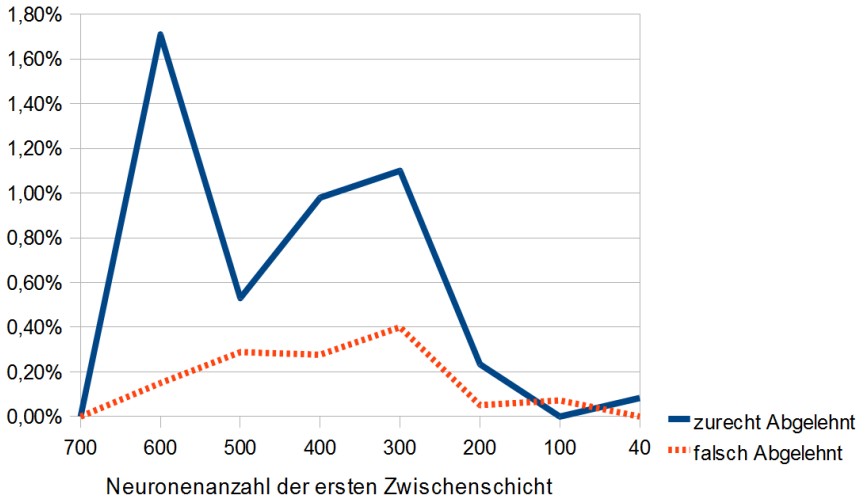

Abbildung 35: Vergleich der Erkennungsraten ohne und mit Hintergrundsubstraktion

5.2.4 Bewertung der Ergebnisse

Durch Anwendung der Hintergrundsubtraktion, in Kombination mit den Schwellwerten für die Ausgabeneuronen, konnte eine Steigerung der Erkennungsrate und eine Senkung der Fehlklassifizierung erreicht werden. Das dreischichtige Netz, mit 450 Neuronen in der Zwischenschicht, erzielt eine Erkennungsrate von 82.0%. Die Rate der Fehlklassifizierung liegt bei 16.7%. Das entspricht einer Verbesserung der Erkennungsrate um 12.5% im Vergleich zu dem initinalen zweischichtigen Netz. Die Fehlklassifizierungen konnte um 13.7% verringert werden. Wie die Tests der beiden Aktionserkenner und die End-to-End Evaluierung gezeigt haben, stellt die erzielte Erkennungsrate eine zufriedenstellende Grundlage für unterscheidbare Posensequenzen zur Erkennung der Aktion dar.

Ausgabeneuron	Schwellwert
AN 1	0.626724
AN 2	0.541290
AN 3	0.496879
AN 4	0.101000
AN 5	0.617637
AN 6	0.489345
AN 7	0.483713
AN 8	0.001000
AN 9	0.001000
AN 10	0.001000
AN 11	0.001000
AN 12	0.001000
AN 13	0.001000
AN 14	0.492801

Tabelle 8: Die Schwellwerte jedes Ausgabeneurons

5.3 Aktionserkenner

Die qualitative Betrachtung des Aktionserkenners ist besonders wichtig, da dieser sich maßgeblich für die Spielbarkeit des angesteuerten Spiels verantwortlich zeichnet. Eine falsch erkannte Aktion kann den Spielfluss durcheinander bringen und das Spiel im schlechtesten Fall unspielbar machen. Aus diesem Grund ist die Detektion der Aktionen nicht nur zügig, sondern auch möglichst korrekt durchzuführen.

5.3.1 Evaluierungssystem

Folgend werden die korrekt und falsch erkannten Aktionen in Relation zu der Gesamtzahl der erkannten Aktionen angegeben. Der Vollständigkeit halber wird auch die Anzahl der als „nicht erkannt“ zurückgewiesenen Aktionen gezeigt, allerdings ist die Anzahl der zurückgewiesenen Aktionen keinesfalls ein gutes Qualitätsmaß für die Genauigkeit der Aktionserkenner. Wie schon erwähnt ist es wichtiger, dass eine Aktion nicht falsch erkannt wird. Dabei ist allerdings zu beachten, dass das Spiel auch bei zu wenig erkannten Aktionen nicht mehr spielbar ist.

Eine Aktion wird ausreichend gut erkannt, wenn ihre Erkennungsrate mindestens bei 90% liegt. Für die Zielanwendung sollten mindestens drei Aktionen ausreichend gut erkannt werden. Der Aktionserkenner sollte insgesamt mindestens 80% der Aktionen richtig erkennen und nicht mehr als 60% zurückweisen. Diese Zahlen basieren auf Erfahrungen, welche bei ersten Tests am Gesamtsystem gemacht wurden. Schlechtere Erkennungsraten bzw. ein höherer Anteil an Zurückweisungen führen dazu, dass die Spielbarkeit nicht mehr gegeben ist.

Um zu entscheiden, ob eine Aktion richtig oder falsch erkannt, bzw. zurückgewiesen wurde, ist es nötig, genau festzulegen, welche Granularität bei der Datenerhebung verwendet wird. Es bieten sich hier zwei Arten von Granularität an, zum einen nach jedem vom Posenerkenner weitergereichten Posenlabel (*Granularität-Pose*), und zum anderen jedes mal, wenn das Sliding Window vollständig ist (*Granularität-SlidingWindow*). Bei der ersten Variante ist lediglich zu beachten, dass das Sliding Window vollständig sein muss, bevor eine Aktion erkannt werden kann. Daher wird die Länge einer Labelsequenz, welche der Größe des Sliding Window entspricht, von der Anzahl der als „nicht erkannt“ zurückgewiesenen Aktionen abgezogen. Bei der zweiten Art der Granularität wird das Sliding Window nach jeder erkannten Aktion gelöscht, und muss bis zur Erkennung der nächsten Aktion erst wieder vollständig sein. Auch hier ist es nötig die entsprechende Anzahl von den zurückgewiesenen Aktionen zu subtrahieren.

Beide Varianten der Granularität haben ihre Vor- und Nachteile. Wird nach jedem neuen Posenlabel versucht, eine Aktion zu erkennen, so kann es leicht passieren, dass insbesondere bei solchen Aktionen, welche längere Sequenzen benötigen (z.B. „fliegen“), oft falsche Aktionen erkannt werden. Der Vorteil hier ist die hohe Reaktivität bei Aktionen, die durch verhältnismäßig kurze Sequenzen repräsentiert werden (z.B. „hocken“). Dies ist insbesondere bei einer niedrigen Rate erhaltener Labels praktisch. Wird hingegen das Sliding Window nach jeder erkannten Aktion gelöscht, so leidet zwar die Reaktivität darunter, bei einer hohen Frequenz von neuen Posenlables fällt dies bei einem kleinen Sliding Window jedoch nicht sehr ins Gewicht. Der Vorteil dieser Granularität liegt darin, dass während langen Sequenzen potenziell weniger Aktionen falsch erkannt werden können.

5.3.2 Eingabedaten

Zur Evaluierung der Aktionserkenner wird eine Videosequenz von einer Person erstellt, welche die verschiedenen Aktionen ausführt. Der Person sind die Aktionen vorher nicht bekannt, sie vollzieht sie rein intuitiv anhand der Aktionsnamen. Die zu erkennenden Aktionen sind: „springen“, „boxen“, „fliegen“, und „hocken“.
Die Bildsequenzen werden dann durch den vorderen Teil der Pipeline gegeben, welcher als Ausgabe Labelsequenzen generiert. Diese dienen dann als Eingabe für die Aktionserkenner.
Auf Grundlage von vorher per Hand eingegeben Aktionen für jede Sequenz von Bildern, lässt sich die Korrektheit des Aktionserkenners überprüfen.

5.3.3 Evaluierung DiBaNGARS

Um zu ermitteln, wie gut *DiBaNGARS* dazu geeignet ist Aktionen für die Steuerung des Spiels zu erkennen, wurde die Verwendung von Bi- und Tri-Grammen, jeweils mit einem Sliding Window der Länge 8 und 16 ausgewertet.
Wie in Abschnitt 4.6.1 beschrieben kann ein Schwellwert für die Erkennung der Aktionen verwendet werden. Ein zu hoher Schwellwert würde dazu führen, dass viele Aktionen erkannt werden, diese aber auch leicht falsch sein können. Wenn der Schwellwert hingen zu niedrig angesetzt ist, so führt dies zwar zu einer deutlich besseren Rate von richtig erkannten Aktionen, jedoch wird oft überhaupt nichts erkannt. Damit eine ausreichende Reaktivität gegeben ist, und nicht zu viele Aktionen falsch erkannt werden, ist es nötig, einen passenden Schwellwert zu finden. Es wurden für beide Varianten von Datenerhebungspunkten (wie in Abschnitt 5.3.1 beschrieben) verschiedene Schwellwerte getestet. Die Ergebnisse dieser Auswertung sind in den Abbildungen 36, 37, 38, und 39 zu sehen. Dabei ist zu beachten, dass hier lediglich die Gesamterkennungsraten, summiert über alle Aktionen aufgeführt sind, das gleiche gilt für die Zurückweisungen. Aus Gründen der Übersichtlichkeit sind die Parameterkombinationen in den Grafiken abgekürzt. „Bi-G_SW8“ steht für die Verwendung von Bi-Grammen mit einem Sliding Window der Länge 8. „Tri-G_SW16“ bedeutet, dass Tri-Gramme mit einem Sliding Window der Länge 16 verwendet wurden. Die Bedeutung der anderen beiden Abkürzungen ergibt sich analog zu den eben genannten Beispielen.

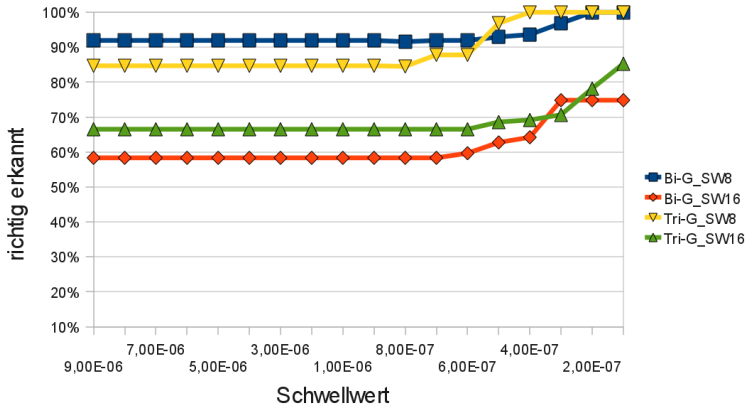

Abbildung 36: DiBaNGARS - Richtig erkannte Aktionen (Granulatität-Pose)

Die Ergebnisse der Schwellwertexperimente weisen darauf hin, dass es von Vorteil ist, nicht nach jedem Posenlabel zu versuchen eine Aktion zu erkennen. Bei der Verwendung von *DiBaNGARS* zur Steuerung des Spiels, muss dementsprechend nach jeder erkannten Aktion das Sliding Window neu erstellt werden. Der starke Anstieg der richtig erkannten Aktionen, sowie der

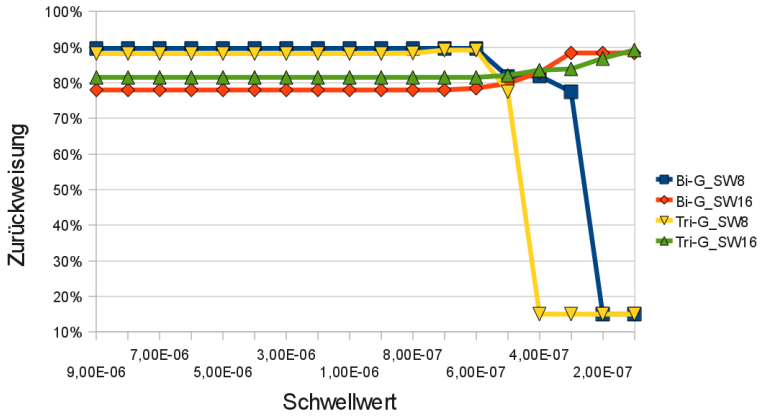

Abbildung 37: DiBaNGARS - Zurückweisungen (Granulatität-Pose)

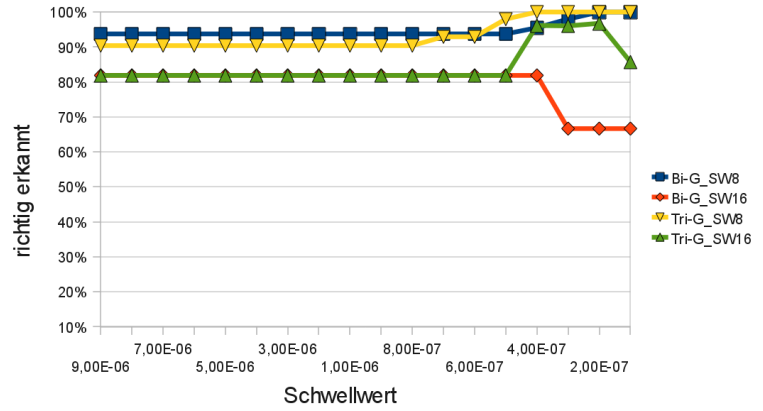

Abbildung 38: DiBaNGARS - Richtig erkannte Aktionen (Granularität-SlidingWindow)

abrupte Abfall der Rate der Zurückweisungen bei sehr niedrigen Schwellwerten resultiert aus der Tatsache, dass einige Aktionen gar nicht mehr erkannt werden. Es wird in diesem Fall nur noch „hocken“ erkannt. Für eine multimodale Spielsteuerung sind diese Einstellungen nicht verwendbar, da mehr als eine Aktion dafür benötigt wird.
Ein Sliding Window der Länge 16 erzielte die besten Resultate, dabei spielt es keine Rolle ob Bi- oder Tri-Gramme verwendet wurden. Da Tri-Gramme deutlich längere Histogramm-Vektoren erzeugen, ist die Verwendung von Bi-Grammen, aus Gründen der Performanz zu bevorzugen.
Als guter Schwellwert hat sich 1,0E-06 (0.000001) erwiesen. Die für diese Kombination von N-Gramm-Größe, Sliding-Window-Länge und Erkennungs-Schwellwert ermittelten Daten, sind in den Tabellen 9 und 10 aufgeführt. Dabei zeigt Tabelle 9 die Erkennungsraten für die einzelnen Aktionen, Ta-

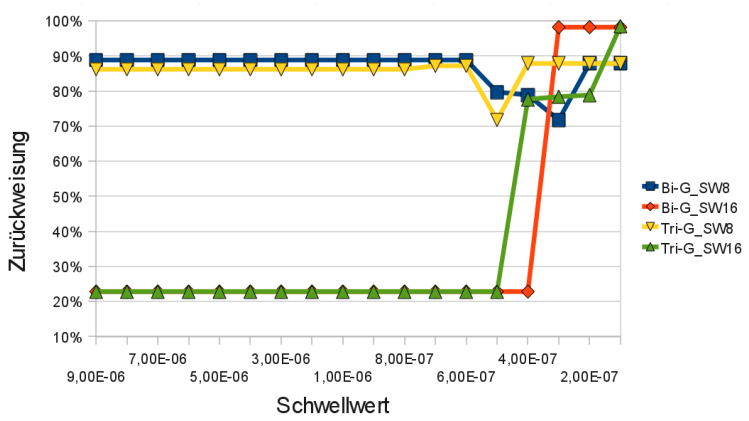

Abbildung 39: DiBaNGARS - Zurückweisungen (Granularität-SlidingWindow)

Aktion	richtig erkannt	Datenerhebungen	erkannt	zurückgewiesen
stehen	82,6%	658	609	49
springen	33,3%	40	3	37
hocken	97,0%	140	33	107
fliegen	95,9%	388	386	2
boxen	2,7%	134	75	59

Tabelle 9: DiBaNGARS - Erkennungsraten der einzelnen Aktionen mit Slidingwindow-Größe von 16 unter Verwendung von Bi-Grammen

belle 10 zeigt auf, welche Aktionen erkannt, und welche erwartet wurden.
Die Ergebnisse der Auswertung zeigen, dass sich mit *DiBaNGARS*, unter den in Abschnitt 5.3.1 beschriebenen Versuchsbedingungen, eine Gesamterkennungsrate von 81,9% bei 22,8% Zurückweisung insgesamt erreichen lässt. Dies erfüllt die ebenfalls in Abschnitt 5.3.1 genannten Mindestanforderungen.

5.3.4 Auswertung Markov-Modelle

Der *Markov-Modell-Erkenner* basiert auf der Entwicklungstoolbox Esmeralda [27] (siehe Abschnitt 4.6.2). Bei der Erstellung der Trainingsdaten werden die Programme „lm_count“ sowie „lm_param“ aus dem *Language Models* Modul verwendet. Zu diesem Zeitpunkt wird außerdem bereits die Größe der N-Gramme festgelegt. Wie schon in Abschnitt 4.6.2 beschrieben, ist es sinnvoll, sich auf Bi- und Tri-Gramme zu beschränken. Zur Laufzeit wird mit jedem übertragenen Posenlabel das Sliding Window ein Label weiter geschoben. Die Erkennung der Aktionen beruht auf der von dem Programm

	stehen	springen	hocken	fliegen	boxen
zurückgewiesen	49	37	107	2	59
stehen	503	2	0	0	9
springen	106	1	1	16	64
hocken	0	0	32	0	0
fliegen	0	0	0	370	0
boxen	0	0	0	0	2

Tabelle 10: DiBaNGARS - Erwartete Aktionen (Spalten) und erkannte Aktionen (Zeilen)

Schwellwert	richtig erkannt	zurückgewiesen
ohne	64.3 %	0.0 %
5	64.3 %	4.2 %
3	66.4 %	10.4 %
2	73.3 %	20.8 %
1.9	73.7 %	21.3 %
1.7	81.6 %	33.6 %
1.6	91.2 %	43.7 %
1.5	98.7 %	49.9 %
1.4	99.9 %	52.6 %
1.3	100.0 %	52.8 %

Tabelle 11: Erkennungsrate mit Sliding Window-Größe von 8 und Verwendung von Tri-Grammen

„lm_perp“ berechneten Perplexität. Eine niedrige Perplexität bedeutet, dass die betrachtete Sequenz mit einer hohen Wahrscheinlichkeit der zugeordneten Aktion entspricht. Die Berechnung der Perplexität geschieht aufgrund des Inhaltes des Sliding Window, sowie einem Markov-Modell, welches für die jeweilige Aktion übergeben werden. „ lm_perp“ gibt dann die Perplexität für die jeweilige Aktion an das System zurück. Die Erhebung der Daten erfolgt mit der *Granularität-Pose*. Es wird die Aktion ausgegeben, für die die geringste Perplexität berechnet wurde. Wird ein Schwellwert für die Erkennung verwendet, so wird die Sequenz vom Aktionserkenner als „Nicht erkannt“ zurückgewiesen, wenn für alle getesteten Aktionen eine Perplexität ermittelt wurde, welche über dem Schwellwert liegt. Dies ist sinnvoll, um den Aktionserkenner zu stabilisieren und eine geringere Rate von Fehlerkennungen zu erreichen. Beim Markov-Modell-Erkenner ist es möglich, den Schwellwert sowie die Länge des Sliding Window zu variieren. Zudem ist es möglich, festzulegen, ob Bi- oder Tri-Gramme verwendet werden. Für die Evaluation werden verschiedene Schwellwerte getestet. Die Auswertung erfolgt für Bi- und Tri-Gramme, jeweils mit einem Sliding Window der Längen 8 und 16.

Schwellwert	richtig erkannt	zurückgewiesen
ohne	73.0%	0.0%
5	74.6%	5.9%
3	77.9%	12.8%
2	86.0%	24.0%
1.9	87.9%	26.0%
1.7	90.4%	29.3%
1.6	90.4%	29.3%
1.5	90.4%	29.3%
1.4	90.4%	29.3%
1.3	90.4%	29.3%
1.2	84.3%	56.7%
1.15	84.3%	56.7%
1.1	84.3%	56.7%
1.05	98.6%	76.6%

Tabelle 12: Erkennungsrate mit Sliding Window-Größe 16 und Verwendung von Tri-Grammen

Schwellwert	richtig erkannt	zurückgewiesen
ohne	73.6%	0.0%
5	75.0%	5.9%
3	77.9%	12.5%
2	86.0%	24.0%
1.9	89.2%	27.1%
1.7	90.4%	29.3%
1.6	90.4%	29.3%
1.5	90.4%	29.3%
1.4	90.4%	29.3%
1.3	90.4%	29.3%
1.2	90.4%	29.4%
1.15	84.3%	56.7%
1.1	84.3%	56.7%
1.05	98.6%	76.6%

Tabelle 13: Erkennungsrate mit Sliding Window-Größe 8 und Verwendung von Bi-Grammen

Schwellwert	richtig erkannt	zurückgewiesen
ohne	65.7%	0.7%
5	65.3%	3.6%
3	67.2%	10.1%
2	72.8%	19.8%
1.9	73.9%	21.1%
1.7	83.8%	31.1%
1.6	91.4%	43.8%
1.5	98.8%	50.0%
1.4	99.9%	52.6%
1.3	100.0%	52.8%

Tabelle 14: Erkennungsrate mit Slidingwindow-Größe 16 und Verwendung von Bi-Grammen

Aktion	richtig erkannt	zurückgewiesen
Stehen	100.0%	55.6%
Springen	—	100.0%
Hocken	100.0%	30.7%
Fliegen	100.0%	16.2%
Boxen	—	100.0%

Tabelle 15: Erkennungsrate der einzelnen Aktionen bei einer Sliding Window-Größe von 16, einem Schwellwert von 1.3 und Verwendung von Tri-Grammen.

Aktion	richtig erkannt	zurückgewiesen
Stehen	65.2%	0.0%
Springen	52.5%	0.0%
Hocken	99.3%	0.0%
Fliegen	92.5%	0.0%
Boxen	89.9%	0.0%

Tabelle 16: Erkennungsrate der einzelnen Aktionen ohne einen Schwellwert bei einer Sliding Window-Größe von 8 und Verwendung von Tri-Grammen.

Aktion	richtig erkannt	zurückgewiesen
Stehen	89.0%	29.5%
Springen	0.0%	95.0%
Hocken	100.0%	14.3%
Fliegen	96.9%	7.2%
Boxen	100.0%	37.3%

Tabelle 17: Erkennungsrate der einzelnen Aktionen mit einen Schwellwert von 1.5 bei einer Sliding Window-Größe von 8 und Verwendung von Tri-Grammen.

Wie die Ergebnisse zeigen, führt ein geringer Schwellwert dazu, dass die Erkennungsrate bis zu 100% beträgt. Ferner ist zu beobachten, dass die Rate der Zurückweisungen dabei steigt. Bei genauer Betrachtung der Erkennungsraten für die einzelnen Aktionen (siehe Tabelle 17) fällt auf, dass die Rate der Zurückweisungen nicht gleich verteilt ist. So werden die Aktionen Springen und Boxen zu 100% als „Nicht erkannt“ zurückgewiesen. Wenn das System ohne einen Schwellwert gestartet wird, so wird eine Erkennungsrate von höchstens knapp 74% erreicht. Dies ist für die Zielanwendung deutlich zu gering (siehe Abschnitt 5.3.1). Die Einstellung, welche die besten Ergebnisse erzielt, ergibt sich mittels folgender Parameter: Schwellwert 1.5, Sliding Windowgröße 8 und die Verwendung von Tri-Grammen. Bei der Verwendung dieser Parameter wird nur Springen nicht ausreichend gut erkannt. Alle anderen Aktion werden allerdings mit einer ausreichenden Genauigkeit detektiert. Insgesamt lässt sich aus den Ergebnissen ableiten, dass ein großes Sliding Window insgesamt die Stabilität des Systems erhöht. Gleichzeitig wirkt es sich aber negativ auf die Erkennungsraten von schnellen Aktionen wie Boxen und Springen aus. Die Verwendung von Tri-Grammen macht das System insgesamt stabiler. Durch die deutlich besseren Erkennungsraten kommen sie daher in Zielanwendung zum Einsatz, obwohl sie rechenintensiver als Bi-Gramme sind. Auf die Auswertung mit *Granulatität Sliding Window* wurde aufgrund der bereits hohen Erkennungsraten verzichtet, da *Granularität Pose* reaktiver ist.

5.3.5 Bewertung der Ergebnisse

Nachdem beide Aktionserkenner getestet und die Daten ausgewertet sind, werden die Ergebnisse daraufhin überprüft ob die Erkenner für die Zielanwendung, also die Steuerung des Spiels „Banjo Kazooie“, geeignet sind.

In Tabelle 11 ist zu erkennen, dass eine Erkennungsrate von 99.9% bei einer Rate zurückgewiesener Aktionen von 52.6% möglich ist, wenn ein Sliding Window der Länge 8 und Tri-Gramme, für den *Sprachmodell-Erkenner* verwendet werden. *DiBaNGARS* erzielt mit keiner Konfiguration eine solch gute Erkennungsrate, obgleich die Rate zurückgewiesener Aktionen in der besten Konfiguration nur bei 22.8% liegt (siehe Abbildung 39), ist die Erkennungsrate hingegen nur bei 81.9% (siehe Abbildung 38). Wie in Abschnitt 5.3.1 bereits erwähnt ist die Anzahl der Zurückweisungen zu vernachlässigen, wenn die Erkennungsrate hoch genug ist. Bei der Steuerung des Spiels bewirkt eine zu niedrige Erkennungsrate, unkontrollierte und unbeabsichtigte Reaktionen der Spielfigur. Dies kann soweit gehen, dass das Spiel nicht mehr spielbar ist.

Der Aktionserkenner *DiBaNGARS* erzielte die beste Erkennungsrate, wenn nach jeder erkannten Aktion das Sliding Window gelöscht wird. Ein

Sliding Window der Länge 16 hat deutlich bessere Resultate erzielt als eines der Länge 8. Wird aber nun ein Sliding Window der Länge 16 verwendet und nach jeder erkannten Aktion das Sliding Window zurückgesetzt, so kann nur nach jedem 16. Posenlabel überhaupt eine Aktion erkannt werden. Ausgehend von 25 Posenlabels pro Sekunde würde dies bedeuten das zwischen zwei Aktionen über eine halbe Sekunde vergehen muss. Dies macht die Steuerung träge und eine ausreichende Reaktivität ist, zumindest bei kurzen Aktionen wie „boxen“, nicht mehr gegeben.

Der *Markov-Modell-Erkenner* hingegen kann nach jeder empfangenen Pose eine Aktion ausgeben und hat dennoch eine bessere Erkennungsrate als *DiBaNGARS*. Zudem kommt der *Markov-Modell-Erkenner* mit einem Sliding Window der Länge 8 aus, und ist somit überlegen was die Reaktivität anbelangt.

Für eine sinnvolle Steuerung des Spiels sollten mindestens drei Aktionen videobasiert erkannt werden können. Dabei ist „stehen“ nicht als Aktion im eigentlichen Sinne zu bewerten, da es im Spiel keinen Unterschied macht, ob „stehen“ oder keine Aktion erkannt wird. Diese Anforderung wird nur vom *Markov-Modell-Erkenner* erfüllt (siehe Tabelle 17), *DiBaNGARS* hingegen erkennt nur „fliegen“ und „hocken“ in ausreichender Genauigkeit (siehe Tabelle 9).

Aufgrund der vorgenommenen Auswertungen, sowie dem Vergleich der beiden Aktionserkenner, hat sich gezeigt, dass der *Markov-Modell-Erkenner* für die Zielanwendung besser geeignet ist.

5.4 Zeitmessung

Die vorgestellte Software ist als Teil einer interaktiven Spieleumgebung an zeitliche Beschränkungen gebunden. Der zeitliche Aspekt ist dahingehend relevant, als dass die Eingabedaten mehrere Berechnungsschritte durchlaufen, bis diese eine Aktion im Spiel auslösen. Jeder Berechnungsschritt verzögert die eingehenden Daten und trägt daher zur Gesamtverzögerung bei. Für den Nutzer einer solchen Anwendung ist dies durch eine verzögerte Ausführung seiner Aktionen bemerkbar.

5.4.1 Evaluierungssystem

Für die Betrachtung der Laufzeiten wird der in Abbildung 8 dargestellte Aufbau aus Abschnitt 4.2.2 verwendet. Die Verwendung von zwei Rechnern stellt eine schnelle Kommunikation zwischen den einzelnen Modulen sicher. Den Ausgangspunkt für die Messung stellt das Modul für die Bildbeschaffung dar.

Vor der eigentlichen Anfrage an den Camera-Server markiert ein Messpunkt den Beginn dieses Verarbeitungsschrittes. Abgeschlossen wird die Zeitmessung mit einem Messpunkt nach der vollständigen Übertragung und damit beim Speichern der Bilddaten in der entsprechenden Austauschstruktur zum nächsten Modul. Der letzte Messpunkt markiert dabei gleichzeitig den Zeitpunkt der Bereitstellung der Daten für das nachfolgende Modul. Beginnt das nachfolgende Modul, also z.B. das Modul für die Personendetektion – mit der Berechnung so dient in diesem Fall ein Messpunkt am Anfang zusätzlich als Endzeit für die Daten in der Austauschstruktur. Abbildung 40 verdeutlicht die einzelnen Messpunkte zwischen den Modulen für die Bildbeschaffung und der Personendetektion. Der erste Messpunkt innerhalb des Moduls für die Bildbeschaffung dient weiterhin als Startzeitpunkt für den Gesamtdurchlauf durch die gesamte Anwendung. So können nicht nur die jeweiligen Bearbeitungszeiten der Daten, sondern ebenfalls die einzelnen Wartezeiten zwischen Modulgrenzen und die Gesamtlaufzeit ermittelt werden.

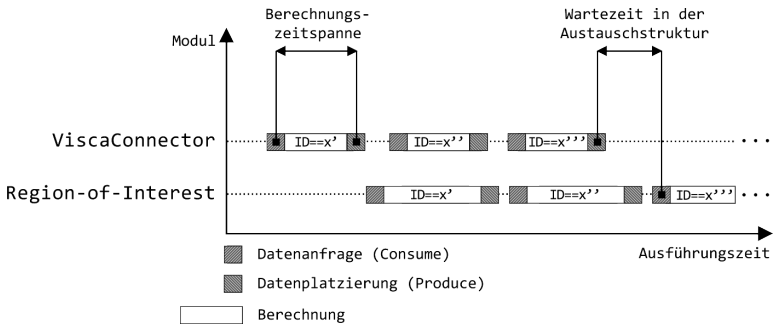

Abbildung 40: Ablauf der Zeitmessung über Modulgrenzen hinweg.

5.4.2 Eingabedaten

Zur Ermittlung des zeitlichen Verhaltens standen für beide AD-Konfigurationen (DiBaNGARS, Markov-Modell) Bildsequenzen aus 7000 Bildern als Eingabe bereit.

5.4.3 Evaluierung der Laufzeit

Rechner I bildet mit den darauf laufenden Modulen in dieser Kette den rechenintensivsten Teil der Anwendung. Dieser kapselt die Module für die Bildbeschaffung, Personendetektion, Merkmalsextraktion und die Posenerkennung. Rechner II hingegen ist sowohl für die Aktionserkennung, als auch für die Steuerung des Spieles durch den Emulator zuständig. Die Übertragungszeit zwischen den einzelnen Rechnern wird dabei zur Gesamtlaufzeit hinzugerechnet. Im Vorfeld erfolgt dabei eine zeitliche Synchronisation der

Modul (M)/ Austauschstruktur (S)	Verzögerung in Millisekunden
M-Bildbeschaffung	18.0
S-Bildbeschaffung->Personendetektion M-Personendetektion	7.3 22.6
S-Personendetektion->Merkmalsextraktion M-Merkmalsextraktion	0.7 11.1
S-Merkmalsextraktion->Posenerkennung M-Posenerkennung	0.3 0.8
(Markov-Modell) S-Posenerkennung->Aktionserkennung (Markov-Modell) M-Aktionserkennung	23.8 45.4
(DiBaNGARS) S-Posenerkennung->Aktionserkennung (DiBaNGARS) M-Aktionserkennung	22.0 0.8
GESAMT (Markov-Modell) GESAMT (DiBaNGARS)	130.0 83.6

Tabelle 18: Gemittelte Ergebnisse der zeitlichen Messung

beiden Rechner auch eine Ermittlung des Zeitunterschiedes. Der zeitliche Unterschied betrug in Tests wenige Millisekunden und wird unterhalb dieser Zeitauflösung vernachlässigt, da die Gesamtlaufzeit dadurch gering beeinträchtigt wird. Rechner II wird in diesem Aufbau abwechselnd mit zwei Konfigurationen betrieben. In der ersten erfolgt die Aktionserkennung mit Hilfe der DiBaNGARS-Methode, die zweite Konfiguration hingegen mittels der beschriebenen Markov-Modell-Funktionalität. Tabelle 18 stellt die Ergebnisse beider Konfigurationen gemittelt über 7000 Bilder dar.

5.4.4 Bewertung der Ergebnisse

Die beiden evaluierten Konfigurationen der Aktionserkenner unterscheiden sich dabei ganz erheblich in der Ausführungszeit. Während die Verweildauer in der jeweiligen Austauschstruktur bedingt durch die Netzwerkkommunikation entsprechend gesteigert wird, fügt der Markov-Modell-AD zusätzlich einen erheblichen zeitlichen Versatz zu den Daten hinzu. Während bei der DiBaNGARS-Konfiguration etwa 12 Aktionen pro Sekunde generiert werden können, sind es bei der Markov-Modell-Konfiguration des Aktionserkenners lediglich 8 Aktionen pro Sekunde.

5.5 End-to-End Evaluation

Nachdem alle einzelnen Module evaluiert und optimal eingestellt wurden, ist es nötig, das System als Ganzes auf seine Tauglichkeit zu prüfen. Zusammenfassend sind in Tabelle 19 alle verwendeten Parameter und ihre Werte dargestellt. Da das System dafür konzipiert wurde, als Eingabegerät für ein

Modul	Parameter	Wert
ROI	Modell	Statisch
	Schwellwert	136
HOG	Bins	9
	Zelle	3x3 Pixel
	Block	3x3 Zellen
	Überlappung	1
KNN	Aktivierungsfunktion	FANN_SIGMOID
	Neuronenschichten	3
	Neuronen in Eingabeschicht	729
	Neuronen in Zwischenschicht	470
	Neuronen in Ausgabeschicht	14
	Lernrate	0,5
AD	Modell	Spracherkennung
	Slidingwindow Länge	8
	Gramme	Tri-Gramme
	Schwellwert	1,5
Controller	Verzögerung	130ms

Tabelle 19: Für die Endevaluierung verwendete Konfiguration.

Spiel zu wirken, werden Versuchspersonen zurate gezogen, um es anzuwenden und unterschiedliche Kriterien zu bewerten.

Nach kurzer Einweisung, wie das System funktioniert, welche Aktionen durchgeführt werden können und wie die Wiimote verwendet wird, um die Steuerung zu vervollständigen, soll jede Person vorher definierte Aufgaben im Spiel erledigen. Hierfür wurde im Vorhinein ein Speicherstand erstellt, damit jeder Teilnehmer unter denselben Voraussetzungen spielt. Die Aufgaben bestehen darin, einen Gegner mit Eiern abzuwerfen, einen weiteren mit der Aktion Boxen zu erledigen, über vorheriges Ducken und anschließendes Springen einen besonders hohen Sprung auf ein erhöhtes Podium zu schaffen und zuletzt von diesem hinabzufliegen. Die Testpersonen sollen mittels eines Fragebogen ihre Bewertungen abgeben.

Insgesamt nahmen 18 Personen an der Auswertung teil, von denen 5 weiblich und 13 männlich waren. Das Alter schwankte zwischen 23 und 37 Jahren, wobei der Durchschnitt 27,11 betrug.

Im Folgenden wird genauer auf die Ergebnisse der einzelnen Abschnitte eingegangen.

Zunächst ist interessant, wie gut die einzelnen Aufgaben erfüllt werden konnten. Die Antwortmöglichkeiten hierzu sind *sehr gut, gut, machbar, schwer, unmöglich*. Die Ergebnisse hierzu sind der Tabelle 20 zu entnehmen, dabei steht 1 für *sehr gut*, 2 für *gut*, etc. Daraus kann geschlossen werden, dass die Aktionen Eierwerfen sowie Boxen besonders gut umgesetzt werden können, die anderen jedoch einigen Teilnehmern Schwierigkeiten bereiten.

Aktion	Min	Max	Durchschnitt
Eierwerfen	1	3	2,06
Boxen	1	2	1,61
Hacken	1	5	2,36
Springen	1	5	2,22
Fliegen	1	4	2,44
Gesamt	1	5	2,14

Tabelle 20: Erfüllbarkeit einzelner Aktionen.

Eigenschaft	Min	Max	Durchschnitt
Reaktionszeit	2	4	2,5
Erkennungsrate	1	4	2,44
Intuition	1	3	1,89
Wiimote	2	4	2,33
Gesamt	1	4	2,18

Tabelle 21: Bewertung unterschiedlicher Eigenschaften des Systems.

Ein wichtiges Kriterium zur Evaluierung des Systems ist die Steuerung, beziehungsweise die Umsetzung der Aktionen im Spiel.

Hierfür sollen die Probanden die Reaktionszeit des Systems, die empfundene Erkennungsrate, die Steuerung mit der Wiimote, sowie die Intuition der ausführbaren Aktionen bewerten. Zur Bewertung bestehen dabei folgende Optionen: *sehr gut, gut, befriedigend, schlecht, inakzeptabel* Ergebnisse hierzu sind in Tabelle 21 dargestellt, wobei die Optionen aufsteigend nummeriert wurden.

Es kann also gesagt werden, dass das System sehr intuitiv zu bewerten ist, jedoch einige Schwächen aufweist. So wird die Reaktionszeit von einem Teilnehmer auf 1 bis 1.5 Sekunden geschätzt, was er als schlecht bewertet. Bei einem weiteren Teilnehmer wurde häufig Boxen erkannt, obwohl er dies nicht beabsichtigte. Weiterhin wird die Richtungssteuerung mit der Wiimote bemängelt, welche feine Korrekturen der Blickrichtung sehr schwierig macht.

Da das System sowohl für Einsteiger, als auch für Personen mit viel Spielerfahrung geeignet sein soll, wird jeder Teilnehmer nach Erfahrungen mit anderen Eingabegeräten befragt. So kann ermittelt werden, ob das System mit einem existierenden vergleichbar ist. Jeder Teilnehmer kann seine Erfahrung über die Möglichkeiten *sehr hoch, hoch, normal, gering, keine* einstufen. Tabelle 22 stellt die Ergebnisse dazu dar, auch hier wurden den Möglichkeiten aufsteigende Zahlen zugeordnet.

Im Schnitt haben die meisten Teilnehmer viel Erfahrung mit Spielen am PC, nur normale bis geringe Erfahrung mit der Wii und sehr wenig mit dem Eyetoy.

Plattform	Min	Max	Durchschnitt
Wii	2	5	3,44
Eyetoy	2	5	4,28
PC	1	4	2,28
Andere	1	5	3,28
Gesamt	1	4	2,94

Tabelle 22: Spielerfahrung der Teilnehmer.

Auf die Frage, wie das System im Vergleich zur Wii bewertet wird, empfanden es 8 Teilnehmer als besser, 4 gleich gut und 2 schlechter. Die übrigen Teilnehmer hatten keine Erfahrung mit der Wii und konnten somit keinen Vergleich führen.

Das gesamte Spielerlebnis empfand ein Teilnehmer als sehr gut, 11 als gut, 4 als befriedigend, während keiner es als schlecht oder inakzeptabel beurteilte. 2 Teilnehmer gaben nichts an.

Zuletzt ist es interessant zu wissen, ob die Teilnehmer das System auch privat nutzen würden und welchen Preis sie dafür zu zahlen bereit wären. So gaben 10 Personen an, das System auch privat nutzen zu wollen, 6 taten dies nicht. 2 Personen gaben nichts an. Der Preis, welcher dem System zugestanden wird, schwankt sehr stark zwischen einem Minimalwert von 15 Euro bis zu einem Höchstwert von 350 Euro, wobei er durchschnittlich 168,38 Euro beträgt. Die hohen Schwankungen können auf unterschiedliche Einschätzungen, was unter dem System zu verstehen ist, zurückgeführt werden. Einige Teilnehmer bewerten nur die Software, während andere das System als vollständige Hardware in Form einer Spielekonsole ansehen.

Über ein Textfeld steht es jedem Teilnehmer frei, seinen Gesamteindruck zu kommentieren. Hierbei lobte eine Person das System als eine Verbindung von Eyetoy und Wii, welche sehr intuitiv und witzig wirkt. Eine andere empfindet das Konzept der Spielesteuerung als spannender, als jenes der Wii und des Eyetoy, da der ganze Körper bewegt werden muss, nicht nur das Handgelenk und die Erkennung sich nicht nur auf grobe Formen beschränkt.

Insgesamt betrachtet erprobten alle Teilnehmer das System erfolgreich und konnten die gewünschten Aufgaben erfüllen. Der Gesamteindruck wird als gut empfunden und schneidet im Vergleich zur Wii besser ab. Eine Verbesserung könnte jedoch noch erzielt werden, indem die Reaktionszeit verringert und die Erkennungsrate gesteigert wird. Das digitale Steuerkreuz der Wiimote macht eine präzise Steuerung schwierig. Hier wäre eine analoge Alternative besser geeignet.

6 Fazit

Es ist schon heute möglich, mit einem moderaten Hardwareaufwand und aktuellen Forschungsergebnissen ein leistungsfähiges System zu entwickeln, dass vor einer Kamera ausgeführte Aktionen verarbeiten und diese auf einen Avatar in einem Spiel abbilden kann. In der frontalen Kameraperspektive wurden Aktionen, wie „stehen“, „ducken“, „boxen“, „fliegen“ und, mit Einschränkungen, „springen“, „werfen“ und „hacken“ erkannt. Die Einschränkungen betreffen jedoch lediglich schlecht abzubildende Abfolgen im Spiel, wie z.B. „hoch springen“ (ducken und dann springen). Da, bedingt durch die vom Spieleentwickler eigentlich vorgesehen Controller, nur wenige „Tasten“ für Aktionen zur Verfügung stehen und somit bestimmte Folgen von Aktionen eine andere Bedeutung haben. Da einige Aktionsfolgen im Spiel zusätzlich kontextabhängig sein können, sind die damit verknüpften Aktionen bisher nur eingeschränkt möglich.

Der Avatar des Spieler in dem Testspiel „Banjo Kazooie“ ist ein Bär, so dass wir für eine intuitive Erkennung die Bewegungen *ducken* und *boxen* auch noch in einer sog. *Bärenart* erkennen. Diese Bewegungen wurden der Spielfigur nachempfunden (siehe Abbildung 27 „boxen_2“).

Um gute Erkennungsergebnisse zu erhalten, mussten wir vorab eine intensive Sichtung bisheriger Forschungsarbeiten und genaue Überlegungen durchführen, welche für uns in die engere Wahl kommen. Nach einigen Tests konnten wir eine einfache, aber durch die vielen, strikt aufgetrennten Threads der einzelnen Verarbeitungsstufen, leistungsfähige Software planen, die es uns erlaubte, auf den uns zur Verfügung gestellten Computern, diese Erkennung durchzuführen. Während der gesamten Entwicklung der Software wurden die Aspekte der Geschwindigkeit und Skalierbarkeit berücksichtigt.

Die Ansteuerung des Spiels wurde einige Male geändert, da ein klassischer Controller ein Steuerkreuz und zusätzlich begrenzt viele Knöpfe besitzt und alle, durchaus vielfältige Aktionen des Avatars, bei der Spielentwicklung bereits fest bestimmten Tasten zugeordnet sind. Auch hier konnten Kompromisse zwischen der Spielbarkeit und der gewünschten videobasierten Aktionserkennung gefunden werden. Bei der Entwicklung einer eigenen Spielkonsole ist dies irrelevant, da die Spiele an den Controller genau angepasst werden und damit derartige Einschränkungen vorab ausgeräumt werden.

Die Projektgruppe konnte im Verlauf ermitteln, welche Techniken heute zum Einsatz gebracht werden können, um ein System zu entwickeln, das einem eine intuitive und einfache Steuerung eines Avatars in einem Spiel erlaubt. Der Spieler kann in die Mitte der Geschehnisse gestellt werden und muss seinen gesamten Körper zur Steuerung der Spielfigur einsetzen. Damit

kann das Spiel nicht mehr von der Couch aus gespielt werden. Diese neue Art könnte man als „Bewegungsspiel“ bezeichnen. Es wäre damit auch möglich, Fitnessprogramme zu gestalten und z.B. Wettbewerbe, die auch bei Computerspielen von der körperlichen Fitness des Spielers abhängen und so in einem anderen Rahmen, als dem bisher üblichen, stattfinden könnten.

6.1 Technische Verbesserungsmöglichkeiten

Folgend findet eine kritische Betrachtung, der gegebenen technischen Voraussetzungen (siehe Abschnitt 4.1 und 4.2), statt. Es wird aufgezeigt inwiefern Veränderungen der technischen Vorraussetzungen Auswirkungen auf die Qualität der Zielanwendung haben könnten. Diese Betrachtung erfolgt anhand von zwei technischen Aspekten. Diese sind die eingesetzte Kamera und der Kameraserver sowie die verwendete Rechnerhardware.

Kamera und Kameraserver

Der Blickwinkel der Kamera beeinflusst den benötigten Abstand zwischen Spieler und Kamera. Der Abstand darf dabei eine Länge von ca. fünf Metern nicht unterschreiten, da dies dazu führen würde, dass der Spieler bei der Ausführung von Posen nicht mehr als Ganzes von der Kamera erfasst werden könnte. Bei einem größeren Blickwinkel wäre es jedoch möglich die Zielanwendung wie andere Spiele auch in kleineren Räumen zu nutzen, z. B. in einem durchschnittlichen Wohnzimmer.

Bei schnell ausgeführten Aktionen liefert die verwendete Kamera teilweise „verwischte“ Bilder, wie z. B. in Abbildung 41 zu sehen. Durch diese *Bewegungsunschärfe* geht die Eindeutigkeit von Kanten, die bei der Berechung der Merkmalsvektoren zum Tragen kommt, teilweise verloren. Eine Kamera, die mit schnellen Bewegungen zurechtkommt, wäre hier von Vorteil, da so eindeutigere Merkmalsvektoren erzeugt würden.

Abbildung 41: Bewegungsunschärfe im Bild

Die von der Kamera gelieferten Bilder besitzen ein leichtes Rauschen, dies beeinflusst die Hintergrundsubtraktion. So könnte durch qualitativ hochwertigere Bilder auch die Qualität der Hintergrundsubtraktion erhöht werden.
Statt aber die einzelnen Parameter der verwendeten Kamera zu verbessern, könnte auch eine zusätzliche Kameratechnologie eingesetzt werden. Eine sogenannte *„Time-of-flight"* Kamera würde zum Beispiel eine sehr saubere Hintergrundsubtraktion ermöglichen. Ein solches Verfahren wird bereits im *Projekt Natal* eingesetzt (siehe Abschnitt 2.5.1). Durch die Tiefensicht der Kamera werden Personen, die sich im Hintergrund des Spielers befinden nicht mit dem Spieler verwechselt. So wäre eine Beeinflussung der Erkennung von Posen und Aktionen, durch Personen im Hintergrund, praktisch nicht mehr vorhanden.

Wie im oberen Teil der Abbildung 42 dargestellt, werden die Bilddaten unkomprimiert vom Kameraserver zum ersten Rechner übertragen. Ein direkter Firewire Anschluss der Kamera an die Rechnerhardware, ohne das Zwischenschalten eines Kameraservers, würde hier die Netzlast deutlich verringern und mögliche Verzögerungen in der Datenübertragung vorbeugen. Die Umsetzung des verbesserten Aufbaus ist im unteren Teil der Abbildung 42 aufgeführt. Dabei wurde nicht nur der Kameraservers entfernt, sondern auch eine Anpassung der Parameter der Rechnerhardware vorgenommen. Diese zur Verbesserung der Zielanwendung führenden Änderungen werden folgend dargestellt.

Rechnerhardware

Die Pipeline der Zielanwendung ist über zwei Rechner verteilt. Auf dem ersten Rechner wird der vordere Teil der Pipeline ausgeführt. In diesem Teil erfolgt die Ermittlung der RoI, die Merkmalsextraktion über das HoG-Verfahren und die Posenerkennung. Die Berechnungen erfolgen dabei direkt auf den Bilddaten. Daher wird hier wesentlich mehr Rechenkapazität benötigt, als im hinteren Teil der Pipeline.

Um die Leistungsfähigkeit des vorderen Teils der Pipeline zu erhöhen, könnten Änderungen der Ausstattung für Rechner I so erfolgen, wie im unteren Teil der Abbildung 42 dargestellt. Statt einen *Core2Duo* Prozessor zu verwenden könnte ein *Core2Quad* Prozessor eingesetzt werden. So bestünde die Möglichkeit alle Module des ersten Teils der Pipeline parallel auszuführen. Dies ist wichtig, da bei der Zielanwendung besonders auf kurze Antwortzeiten zu achten ist, denn sonst wird die intuitive Steuerung des Spieles eingeschränkt. Eine Steigerung der Anzahl der Kerne von zwei auf vier würde somit die Anwendung robuster gegenüber Verzögerungen gestalten.

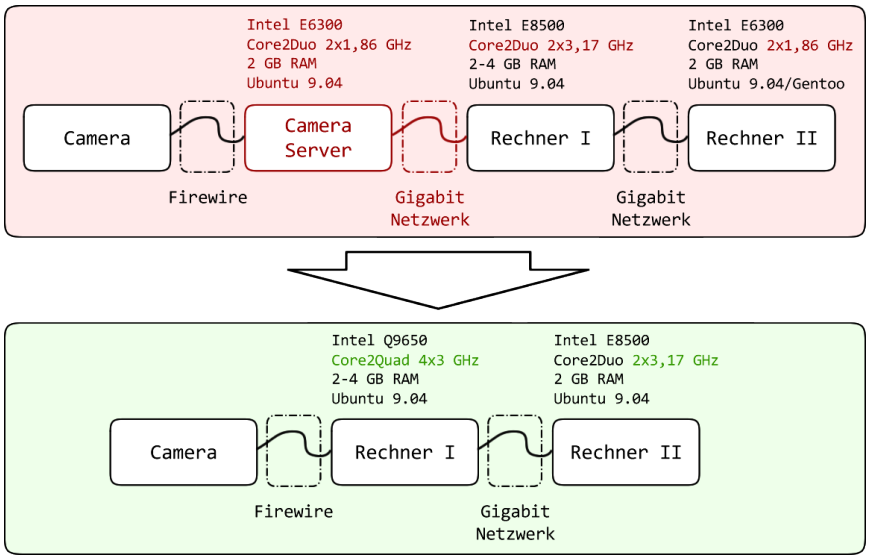

Abbildung 42: Mögliche Verbesserung der Zielhardware

Eine Erhöhung der *Taktung* der Prozessorkerne des Rechners II führt hier ebenfalls zu einer höheren Reaktivität der Zielanwendung. Es werden zwei Module der Pipeline auf Rechner II ausgeführt, die Aktionserkennung und die Emulatorsteuerung.

7 Literaturangaben

Literatur

[1] *Chronik: Die schlimmsten Amokläufe an Schulen*, 2007. Verfügbar unter http://www.spiegel.de/panorama/justiz/0,1518,518191,00.html, besucht am 25.3.2010.

[2] *Gesten von iPhone und iPod touch*, 2007. Verfügbar unter http://www.apfelwiki.de/Main/Gesten, besucht am 30.03.2010.

[3] *Mupen64Plus v2.0 Plugin API*, 2007. Verfügbar unter http://www.emuwiki.com/index.php?title=Mupen64Plus_v2.0_Plugin_API, besucht am 22.3.2010.

[4] *YoFrankie - Erstes Blender Game Engine-Demo veröffentlicht*, 2008. Verfügbar unter http://www.pro-linux.de/NB3/news/1/13188, besucht am 25.3.2010.

[5] *FINCA - A Flexible, Intelligent eNvironment with Computational Augmentation*, 2010. Verfügbar unter http://www.irf.tu-dortmund.de/cms/en/IS/Research/FINCA/index.html, besucht am 30.03.2010.

[6] A. Agarwal, B. Triggs: *3D Human Pose from Silhouettes by Relevance Vector Regression.* In: *Proceedings of the 2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Seiten 882–888, 2004.

[7] A. Azarbayejani, A. Pentland: *Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features.* In: *In Proceedings of 13th ICPR.* IEEE Computer Society Press, 1996.

[8] A. Bissacco, M. Yang, S. Soatto: *Detecting Humans via Their Pose.* In: *Neural Information Processing Systems (NIPS)*, Seiten 169–176, 2006.

[9] A. Yilmaz, M. Shah: *Actions as Objects: A Novel Action Representation.* In: *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Band 1, Seiten 984–989, 2005.

[10] ADAC: *ADAC testet vier Spurhalteassistenten*, 2007. Verfügbar unter http://www.kfz.net/autonews/adac-testet-vier-spurhalteassistenten-17996/, besucht am 7.3.2010.

[11] auto-reporter.net: *Opel Insignia erkennt Verkehrszeichen*, 2008. Verfügbar unter http://auto.t-online.de/

auto-technik-opel-insignia-erkennt-verkehrszeichen-/id_15384742/index, besucht am 7.3.2010.

[12] Bishop, C.M.: *Neural Networks for Pattern Recognition*. Oxford University Press, Oxford, 1 Auflage, 1 1995.

[13] Bock, H.H.: *Automatische Klassifikation: Theoretische und praktische Methoden zur Gruppierung und Strukturierung von Daten (Cluster-Analyse) (Studia mathematica / Mathematische Lehrbucher)*. Vandenhoeck & Ruprecht, 1974.

[14] Bundesverband Interaktive Unterhaltungssoftware: *Gamesbranche im Jahr 2008 mit neuem Umsatzrekord*, 2009. Verfügbar unter http://biu-online.de/home/news/16-februar-2009-gamesbranche-im-jahr-2008/-mit-neuem-umsatzrekord/, besucht am 28.2.2010.

[15] C. Papageorgiou, T. Poggio: *A Trainable System for Object Detection*. International Journal of Computer Vision, 38(1):15–33, 2000.

[16] C. Thurau, V. Hlavac: *Pose primitive based human action recognition in videos or still images*. In: *Proceedings of the 2008 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Seiten 1–8, 2008.

[17] C.A. Anderson, B.J. Bushman: *Effects of Violent Video Games on Aggressive Behavior, Aggressive Cognition, Aggressive Affect, Physiological Arousal, and Prosocial Behavior: A Meta-Analytic Review of the Scientific Literature*. Psychological Science, 12:353–359, 2001.

[18] CamTrax Technologies: *CamSpace.com: Kiss Your Keyboard, Mouse and Joystick Goodbye! Play Games With Your WebCam*, 2008. Verfügbar unter http://www.camspace.com, besucht am 27.2.2010.

[19] damien.douxchamps.net: *Informations on libVISCA*, 2009. Verfügbar unter http://damien.douxchamps.net/libvisca/, besucht am 30.3.2010.

[20] Davis, J.W.: *Hierarchical Motion History Images for Recognizing Human Motion*. Detection and Recognition of Events in Video, IEEE Workshop on, 0:39, 2001.

[21] EL33TONLINE: *Project Natal runs at 30 FPS, developer video discusses possibilities*, 2010. Verfügbar unter http://www.el33tonline.com/past/2010/1/7/project_natal_runs_at_30/, besucht am 16.2.2010.

[22] Eurogamer: *E3: Post-Natal Discussion Interview*, 2009. Verfügbar unter http://www.eurogamer.net/articles/e3-post-natal-discussion-interview, besucht am 16.2.2010.

[23] F. Suard, A. Rakotomamonjy, A. Bensrhair A. Broggi: *Pedestrian Detection using Infrared images and Histograms of Oriented Gradients.* In: *Intelligent Vehicles Symposium, Tokyo, Japan*, Seiten 206–212, June 2006.

[24] Fengjun Lv, R. Nevatia: *Single View Human Action Recognition using Key Pose Matching and Viterbi Path Searching.* In: *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, Seiten 1–8, June 2007.

[25] Fink, G.A.: *Markov models for pattern recognition.* Springer, 2008.

[26] Funto: *Wiinput64: a wiimote input plugin for Mupen64Plus with Linux.* Internet Forum, April 2009. Verfügbar unter http://www.emutalk.net/archive/index.php/t-47563.html, besucht am 10.08.2009.

[27] G.A. Fink, T. Plötz: *Developing Pattern Recognition Systems Based on Markov Models: The ESMERALDA Framework.* Pattern Recognition and Image Analysis, 18(2):207–215, June 2008.

[28] Gavrila, D.: *The Visual Analysis of Human Movement: A Survey.* Computer Vision and Image Understanding, 73(1):82–98, 1999.

[29] H. Jhuang, T. Serre, L. Wolf T. Poggio: *A Biologically Inspired System for Action Recognition.* In: *Proceedings of the 2007 IEEE International Conference on Computer Vision (ICCV)*, Seiten 1–8, 2007.

[30] Handelsblatt: *Spiele-Industrie greift Hollywood an*, 2004. Verfügbar unter http://www.handelsblatt.com/spiele-industrie-greift-hollywood-an;719622, besucht am 28.2.2010.

[31] J. Richarz, T. Plötz, G.A. Fink: *Real-time detection and interpretation of 3D deictic gestures for interactionwith an intelligent environment.* In: *ICPR*, Seiten 1–4, 2008.

[32] J.W. Davis, A.F. Bobick: *The Representation and Recognition of Action Using Temporal Templates.* Seiten 928–934, 1997.

[33] K. Dill, C. Anderson: *Video games and aggressive thoughts, feelings, and behavior in the laboratory and in life.* Journal of Personality and Social Psychology, 78:772–790, 2000.

[34] kfz-auskunft.de: *Endlich - Der neue 5er BMW kommt im März 2010 auf den Markt*, 2009. Verfügbar unter http://www.kfz-auskunft.de/news/7252.html, besucht am 7.3.2010.

[35] libwiimote.sourceforge.net: *Simple Wiimote Library For Linux.* Internet, 2007. Verfügbar unter http://libwiimote.sourceforge.net/, besucht am 11.08.2009.

[36] Logitech: *Bewegungsfreiheit*, 2010. Verfügbar unter http://www.logitech.com/index.cfm/webcam_communications/webcams/devices/5869&cl=de,de, besucht am 16.2.2010.

[37] Microsoft: *Projekt Natal*, 2009. Verfügbar unter http://www.xbox.com/en-US/live/projectnatal/, besucht am 29.1.2010.

[38] Microsoft: *Natural User Interfaces: Voice, Touch and Beyond*, 2010. Verfügbar unter http://www.microsoft.com/presspass/features/2010/jan10/01-06CESNUI.mspx, besucht am 29.1.2010.

[39] Mupen64Plus Project Group: *Mupen64Plus*, 2010. Verfügbar unter http://code.google.com/p/mupen64plus/, besucht am 03.3.2010.

[40] N. Dalal, B. Triggs: *Histograms of Oriented Gradients for Human Detection*. In: *Proceedings of the 2005 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Band 1, Seiten 886–893, 2005.

[41] N. Ikizler, P. Duygulu: *Human Action Recognition Using Distribution of Oriented Rectangular Patches*. In: Elgammal, Ahmed M., Bodo Rosenhahn und Reinhard Klette (Herausgeber): *Workshop on Human Motion*, Band 4814 der Reihe *Lecture Notes in Computer Science*, Seiten 271–284. Springer, 2007.

[42] Nintendo: *WarioWare snapped!*, 2009. Verfügbar unter http://www.nintendo.de/NOE/de_DE/games/dsiware/warioware_snapped_12137.html, besucht am 7.3.2010.

[43] Nintendo: *Whats Wii MotionPlus?*, 2009. Verfügbar unter http://www.nintendo.com/wii/what/accessories/wiimotionplus, besucht am 27.2.2010.

[44] Nissen, S.: *Fast Artificial Neural Network Library (FANN)*, 2008. Verfügbar unter http://leenissen.dk/fann/, besucht am 09.03.2010.

[45] N.J.B. McFarlane, C.P. Schofield: *Segmentation and Tracking of Piglets in Images*. 8(3):187–193, 1995.

[46] P. Viola, M. Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features*. In: *Proceedings of the 2001 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Band 1, Seiten 511–518, 2001.

[47] Pichler, T.: *Google ermöglicht Gestensteuerung auf Android-Smartphones*, 2010. Verfügbar unter http://www.inside-handy.de/news/17658.html, besucht am 30.03.2010.

[48] R. Gonzalez, R. Woods: *Digital Image Processing*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[49] R. Poppe, D. Heylen, A. Nijholt M. Poel: *Towards real-time Body Pose Estimation for Presenters in Meeting Environments.* In: *WSCG (Short Papers)*, Seiten 41–44, 2005.

[50] R. Ronfard, D. Weinland, E. Boyer: *Free Viewpoint Action Recognition using Motion History Volumes.* Computer Vision and Image Understanding, 104(2-3):249–257, 2006.

[51] Rosenblatt, F.: *The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain.* Psychological Review, 65(6):386–408, 1958.

[52] S. Ali, A. Basharat, M. Shah: *Chaotic Invariants for Human Action Recognition.* Seiten 1–8, October 2007.

[53] Seggern, D.H. von: *CRC Standard Curves and Surfaces with Mathematica, Second Edition (Chapman & Hall/CRC Applied Mathematics & Nonlinear Science).* Chapman and Hall/CRC, 2 Auflage, 10 2006.

[54] Shortnews: *Spieleindustrie umsatzstärker als Hollywood*, 2002. Verfügbar unter http://www.shortnews.de/id/359714/Spieleindustrie-umsatzst%C3%A4rker-als-Hollywood, besucht am 28.2.2010.

[55] Sony: *Smile Shutter-Funktion*, 2007. Verfügbar unter http://www.digiklix.de/2007/08/22/news-automatisches-ausloesen-bei-laecheln-sony-/dsc-t70-und-dsc-t200/, besucht am 16.2.2010.

[56] Sony: *EyeToy Play 3*, 2009. Verfügbar unter http://www.eyetoy.com/, besucht am 29.1.2010.

[57] Tavinor, G.: *Towards an ethics of video gaming.* In: *Future Play '07: Proceedings of the 2007 conference on Future Play*, Seiten 1–8, New York, NY, USA, 2007. ACM.

[58] T.F. Coo, C.J. Taylor, D.H. Cooper J. Graham: *Active shape models—their training and application.* Comput. Vis. Image Underst., 61(1):38–59, 1995.

[59] The Financial Times Limited: *Nintendo ‚Äòrejected‚Äô rivals‚Äô choice of technology*, 2009. Verfügbar unter http://www.ft.com/cms/s/0/dfcdde86-513e-11de-84c3-00144feabdc0.html, besucht am 7.3.2010.

[60] The Linux Game Tome: *The Linux Game Tome*, 2010. Verfügbar unter http://happypenguin.org/, besucht am 28.2.2010.

[61] The Seattle Times: *E3: New info on Microsoft's Natal – how it works, multiplayer and PC versions*, 2009. Verfügbar unter http://

seattletimes.nwsource.com/html/technologybrierdudleysblog/2009296568_e3_new_info_on_microsofts_nata.html, besucht am 7.3.2010.

[62] V. Parameswaran, R. Chellappa: *View Invariants for Human Action Recognition.* Computer Vision and Pattern Recognition, IEEE Computer Society Conference on, 2:613–619, 2003.

[63] Wikipedia: *Wii MotionPlus*, 2010. Verfügbar unter http://en.wikipedia.org/wiki/Wii_MotionPlus, besucht am 27.2.2010.

[64] W.S. McCulloch, W. Pitts: *A Logical Calculus of Ideas Immanent in Nervous Activity.* Bulletin of Mathematical Biophysics, (5):115–133, 1943. Reprinted in *Neurocomputing: Foundations of Research, ed. by J. A. Anderson and E Rosenfeld. MIT Press 1988.*

[65] Y. Freund, R. Iyer, R.E. Schapire Y. Singer: *An efficient boosting algorithm for combining preferences.* International Conference on Machine Learning (ICML), 4:933–969, 2003.

[66] Zuljevic, S.: *Playstation 3: Motion-Controller angekündigt (Update)*, 2010. Verfügbar unter http://www.netzwelt.de/news/80781-playstation-3-motion-controller-angekuendigt-update.html, besucht am 27.2.2010.