Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Bachelorarbeit

Roman Ausländer

Gestenerkennung im Raum mit einer omnidirektionalen Kamera

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Roman Ausländer

Gestenerkennung im Raum mit einer omnidirektionalen Kamera

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

Betreuender Prüfer : Prof. Dr. rer. nat. Kai von Luck
Zweitgutachter : Prof. Dr.Ing. Andreas Meisel

Abgegeben am 23. September 2010

Roman Ausländer

Thema der Bachelorarbeit

Gestenerkennung im Raum mit einer omnidirektionalen Kamera

Stichworte

Mensch-SmartHome-Interaktion, Gestenerkennung, Fish-Eye-Kamera

Kurzzusammenfassung

In dieser Arbeit möchte ich die Möglichkeit der maschinellen Erkennung der dynamischen Gesten ohne Marker in Echtzeit mit einer omnidirektionalen Kamera untersuchen. Es wird ein Überblick über die verfügbaren Algorythmen und Software-Bibliotheken gegeben. Für die Realisierung des Prototyps werden die Anforderungen an diese analysiert und spezifiziert. Eine rudimentäre Gestenerkennnungssoftware wird erstellt und getestet. Die Ergebnisse dieser Tests werden beschrieben und bewertet.

Roman Auslaender

Title of the paper

Gesture recognition in space with an omnidirectional camera

Keywords

Man smarthome interaction, Gesture recognition, Fish eye camera

Abstract

In this work I would like to research on a possibility of computer vision based detection of dynamic gestures without marker in real time with an omnidirectional camera. It is an overview of available algorithms and software libraries. For the realization of the prototype, the requirements of these are analyzed and specified. A rudimentary gesture recognition software is created and tested. The results of the tests are described and evaluated.

Danksagung

An dieser Stelle danke ich meiner Familie für die Unterstützung meines Studiums.

Ich danke auch der Firma MOBOTIX AG für die Zurverfügungstellung einer modernen Fish-Eye-Kamera.

Inhaltsverzeichnis

1. Einführung

Ubiquitous Computing heißt die Zukunfts-Vision für den Einsatz von Computersystemen in privaten Haushalten, Büros und Produktionsstätten. Schon heute wächst die Anzahl der Rechner in unserer Umgebung kontinuierlich. Die Komplexität und die Anzahl der Aufgaben, die diese Systeme übernehmen, um unseren Alltag zu erleichtern, nehmen zu. Damit Menschen mit immer mehr und immer komplexeren Systemen sicher umgehen können, muss die die Interaktion mit diesen Geräten einfacher gemacht werden.

Ansätze der Seamless Interaction gewinnen dabei immer mehr an Bedeutung. Seamless Interaction setzt sich das Ziel, die Interaktion zwischen Mensch und Maschine für Benutzer intuitiv und natürlich zu gestalten. Nicht die Benutzer sollen sich auf die Eingabemethoden der Rechner einstellen, sondern die Computersysteme passen sich ans bekannte Umfeld der Nutzer an.

Der Bruch mit den klassischen Eingabeparadigma der heutigen Computer und Erarbeitung völlig neuer Nutzungskonzepte, ist einer der interessantesten Forschungsgebiete in der modernen Informatik. An vielen Hochschulen und Forschungseinrichtungen wird nach neuen Wegen gesucht, auf natürliche Art und Weise mit Computersystemen zu interagieren. Gestenerkennung bieten uns eine greifbare Möglichkeit mit Computergestützten Systemen auf natürliche Art zu interagieren.

In der menschlichen Kommunikation spielen Gesten neben der Sprache und der Mimik eine wichtige Rolle. Sie können die Sprache im Dialog unterstützen oder sogar völlig ersetzen, was die Gebärdensprache zeigt.

Mit Gesten lassen sich nicht nur Emotionen und Gefühle, sonder auch Hinweise und Befehle zum Ausdruck bringen. Ein Verkehrsposten kann mit seinen Zeichen die Durchfahr auf einer Kreuzung regeln und ein Basketball-Schiedsrichter einen Schrittfehler bekunden.

Wir an der HAW Hamburg versuchen unsere Arbeit an die neuen Interaktionskonzepte zu knüpfen, um ihre Umsetzbarkeit in der Praxis zu evaluieren.

1.1. Motivation

Im Rahmen des Projektes „Living Place Hamburg“ wird an der HAW Hamburg eine Loftwohnung errichtet, die den Ansatz von Ubiquitous Computing verfolgt. Das Ziel ist es, eine Wohnung aufzubauen, die mit ihrem Bewohner interagiert und ihn im Alltag unterstützt.

Für Studenten bietet der „Living Place Hamburg“ eine Probe-Werkstatt für praxisnahe Versuche auf dem Gebiet „Intelligente Wohnung“ an. Viele Verfahren werden erforscht und verschiedene Anwendungen werden entwickelt, um das Ziel zu erreichen.

Unter anderen ubiquitären Sensoren, sind auch moderne omnidirektionale Kameras in der Wohnung installiert. Die Möglichkeit bei diesem Projekt mitzumachen und mich mit der Gestenerkennung mittels dieser Sensoren auseinander zu setzen ist einzigartig. Denn das hat bisher scheinbar noch Keiner versucht. Meine Motivation baut auf dem Pioniergeist auf.

Also was haben wir heute vor? Wir wollen die Welt verändern!

1.2. Gliederung der Arbeit

Diese Arbeit teilt sich im Wesentlichen in vier Teile, die hier kurz umrissen werden sollen.

Kapitel 2 widmet sich den Grundlagen, also der Erläuterung von Begriffen und Zusammenhängen, die für das Verständnis dieser Arbeit vorausgesetzt werden. Neben der Einführung der Begriffe Ubiquitous Computing (Abschnitt 2.1) und Seamless Interaction (Abschnitt 2.2), werden auch menschliche Gesten klassifiziert (Abschnitt 2.3), sowie auch kurz die Funktionsweise der Trackingsysteme vorgestellt (Abschnitt 2.4) und erläutert was unter Freiheitsgraden der Trackingsysteme zu verstehen ist (Abschnitt 2.4.2).

Kapitel 3 Analyse dient dazu die Problemstellung genauer zu untersuchen. Dafür werden Anwendungsszenarien vorgestellt (Abschnitt 3.1) und diese dann ausgewertet (Abschnitt 3.2), um die zugrundeliegenden Fragestellungen herauszuarbeiten. Anschließend werden die Ziele für die Entwicklung eines Prototypen festgelegt (Abschnitt 3.3) und die Anforderungen an diese Implementierung definiert (Abschnitt 3.4). Abschließend wird ein Überblick über die Methoden der Gestenerkennung vorgestellt (Abschnitt 3.5).

Kapitel 4 befasst sich mit dem Entwurf und der Realisierung eines solchen prototypischen Systems. Zunächst werden Verfahren vorgestellt, die auf eine einfache Weise erlauben Bewegungen zu erkennen (Abschnitt 4.1). Anschließend werden die Haupteigenschaften der hemisphärischen Kameras aufgelistet (Abschnitt 4.6), die verwendete Bibliothek (Abschnitt 4.7) und der Aufbau des Prototypen beschrieben (Abschnitt 4.8). Das Kapitel wird mit einer

Evaluation (Abschnitt 4.9) abgeschlossen, welche einen Überblick über den Entwicklungsstand des Prototypen gibt.

In Kapitel 5 wird diese Arbeit mit einer Zusammenfassung und einem Fazit (Abschnitt 5.1) sowie einem Ausblick auf mögliche weiterführende Untersuchungen und Arbeiten (Abschnitt 5.2) abgeschlossen.

2. Grundlagen

Ziel dieser Arbeit ist die Entwicklung eines computergestützten Systems zur Steuerung von Unterhaltungelektronik mittels Körpergesten. Für diesen Zweck sind einige Grundlagen nötig. Als erstes werden die Ideen von „Ubiquitous Computing“ (Abschnitt 2.1) und „Seamless Interaction“ (Abschnitt 2.2) erläutert. Anschließend wird der Begriff Geste definiert und mögliche Klassifikationen von Gesten vorgestellt (Abschnitt 2.3). Abschließend werden Trackingsysteme zur Bewegungserfassung vorgestellt und klassifiziert.

2.1. Ubiquitous Computing

Unter dem Begriff Ubiquitous Computing oder auch allgegenwärtiges Rechnen, versteht man die Allgegenwertigkeit von Computern in unserer Umgebung. Mark Weiser (1999) sagte voraus, dass Computer in unserer Umgebung immer mehr Aufgaben abnehmen und einen immer höheren Anteil einnehmen werden. Dabei schwindet die Anwenderwahrnehmung der Rechner in vielen Geräten. Computer werden in unserer Umgebung benutzt, ohne sie als solche unmittelbar wahrzunehmen.

Beispiele hierfür gibt es heutzutage bereits viele: Der Computer im Fotoapparat, im Satelliten-Receiver oder im neuen Personalausweis sind nur einige Objekte, welche wir täglich um uns haben und benutzen, ohne direkt auf ein Rechnersystem zu schließen.

Eine wichtige Eigenschaft dieser Entwicklung ist die immer steigende Vernetzung dieser Geräte untereinander, wodurch u.a. gemeinsame Daten und Ressourcen effizient genutzt werden können.

Für einen Anwender ist die Bedienung eines Systems viel einfacher, wenn er die Art, wie er mit dem Rechner interagiert, bereits aus anderen Bereichen seines Lebens kennt. Darum sollte das Ziel sein, mit Hilfe von Fähigkeiten und Techniken, die dem Benutzer vertraut sind, ein computergestütztes System zu bedienen. Dieser Ansatz verfolgt „Seamless Interaction“. Ishii u. a. (1994) beschreiben zwei Aspekte, die für „Seamless Interaction“ beachtet werden müssen.

Abbildung 2.1.: Ubiquitous Computing aus Jon Rimmer und Weir (2005)

2.2. Seamless Interaction

An erster Stelle steht die Übertragbarkeit von realen in virtuelle Arbeitsabläufe. Für den Anwender sind Benutzungskonzepte besonders einleuchtend, wenn er sie bereits aus seiner realen Umwelt kennt. Ein Beispiel ist die Anwendung einer Fotometapher für die Darstellung und Manipulation von Bilder-Dateien auf dem Bildschirm. Das Objekt auf dem Bildschirm hat das Aussehen und größtenteils das Verhalten eines Fotos. Der Benutzer bildet sein bereits existierendes mentales Modell eines Fotoabdrucks auf das virtuelle Bild ab. Daher kann der Anwender, ihm bereits bekannte Arbeitsabläufe, wie das Beschriften oder Verschieben in ein Album, auf das virtuelle Objekt übertragen.

An zweiter Stelle steht der Wechsel zwischen verschiedenen Arbeits-Modi und -Bereichen. Dieses muss vor allem bei Gruppenarbeiten beachtet werden, bei denen häufig zwischen einem persönlichem und einem öffentlichem Arbeitsraum gewechselt wird.

Das Ziel von Seamless Interaction ist es, nicht reale Konstellationen „naturgetreu“ auf virtuelle Objekte zu übertragen, sondern vielmehr das beim Anwender existierende mentale Modell mit den Möglichkeiten der Rechner zu kombinieren. Dadurch wird es möglich die Vorteile einer natürlichen Interaktion mit der Schnelligkeit der Computersysteme zu verknüpfen.

Vor allem Gestensteuerung bringt mit sich den Vorteil, dass der Anwender mit der Nutzung verschiedenen Gesten bereits aus seiner realen Welt vertraut ist und diese Art der Menschen-Kommunikation auf die Maschinen-Interaktion leicht übertragen kann. Das er-

möglicht Computersysteme zu entwickeln, die der Anwender auf einfache und natürliche Weise bedienen kann.

2.3. Gesten

2.3.1. Begriffserklärung

In dieser Arbeit geht es um die Steuerung von Unterhaltungelektronik mittels Gesten. Dafür muss geklärt werden, was Gesten eigentlich sind, welche Gesten-Arten es gibt und welche für unser Ziel von Interesse sind.

Laut Wikipedia bezeichnet Geste (lat.: gerere = tragen, ausführen; PPP: gestum) eine mittelbar oder unmittelbar physische Handlung zwecks Kommunikation oder Unterstützung dieser im Sinne der Gestik oder der nonverbalen Kommunikation.

In dieser Arbeit konzentrierten wir uns auf Gesten, welche mittels Körperbewegungen ausgelöst werden. Wir unterscheiden zwischen bewussten und unbewussten Handlungen zur Kommunikation. Bewusste Gesten unterstützen aktiv die Kommunikation, wie die Stop-Haltung der Hand, während unbewusste Gesten meistens andere Kommunikationskanäle bedienen, z.B. auf jemanden Zeigen, von dem gerade gesprochen wird. Unsere Arbeit beschäftigt sich mit bewusst durchgeführten Gesten zur unmittelbaren Kommunikation.

In dieser Arbeit werden die Gesten zur Interaktion mit rechnergestützten Systemen als manipulative Gesten verstanden, sowie bei Heitsch (2008) und Boetzer (2008). Der Anwender hat die Absicht mit seiner Handlung den Zustand des Computersystems zu ändern.

Es kann außerdem nach zwei Gestenarten unterschieden werden. Bei kontinuierlichen Gesten besteht eine direkte Verbindung zwischen der durch den Computer beobachteten Bewegung und einem Zustand im Computer. Beispielsweise kann durch Zeigen auf den Bildschirm ein Zeiger gesteuert werden. Bei diskreten Gesten handelt es sich hingegen um beschränkte Mengen von eindeutigen Gesten, mit denen in der Regel jeweils eine Aktion verknüpft ist. Ein Beispiel für diskrete Gesten ist die Gebärdensprache, bei der jede Gebärde mit einer bestimmten Bedeutung verknüpft ist.

2.3.2. Statische und dynamische Gesten

Für eine funktionierende Gestenerkennung müssen zwei Arten von Gesten unterschieden werden: Statische und dynamische Gesten.

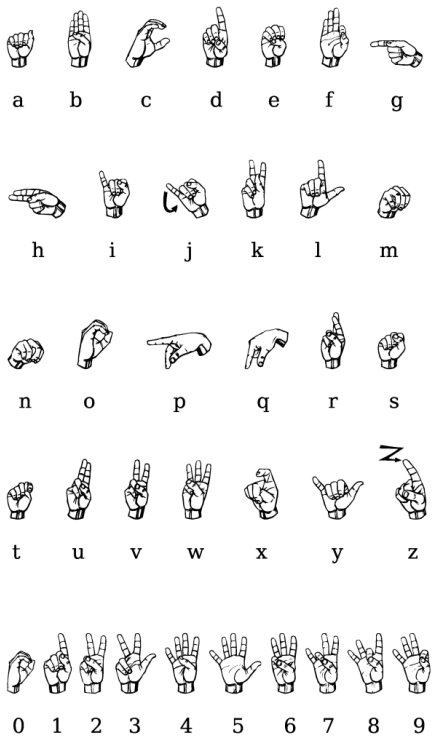


Abbildung 2.2.: American Sign Language alphabet aus Wikipedia (2004a)

Statische Gesten sind hierbei zeitlose Handlungen. Die Einnahme einer bestimmten, vorher definierten, Haltung der Hand ist eine statische Geste.

Dynamische Gesten werden über einen gewissen Zeitraum hinweg durchgeführt. In unserer späteren Anwendung ist dieses die Bewegung der Hand in eine bestimmte Richtung. Zur Erkennung dieser Geste ist es erforderlich, die Handbewegung über einen gewissen Zeitraum hinweg durchzuführen.

2.3.3. Erkennung von Start- und -Endpunkten der Gesten

Marcus Rödiger in sener Arbeit Rödiger (2010) beschäftigt sich u. a. mit einer sauberen Start-Stop-Erkennung von Gesten. Um eine einzelne Geste eindeutig erkennen und auswerten zu können, ist es notwendig den Zeitpunkt zu bestimmen, wann die einzelne Geste beginnt und wann sie aufhört. Wichtig ist die Beantwortung dieser Frage, um im weiteren Verlauf die einzelnen Gesten zeitlich voneinander klar trennen und voneinander unterscheiden zu können.

2.4. Tracking von Körperbewegungen

Für die Entwicklung der späteren Anwendung ist es notwendig, die Körperbewegungen des Anwenders zu erfassen um daraus Gesten abzuleiten. S. g. Trackingsysteme übernehmen die Erfassung der Bewegung von Körperteilen. Diese Systeme messen Kennwerte der Bewegung einzelner Körperglieder in einem zeitlichen Zusammenhang und leiten diese Messdaten an das Gestenerkennungsystem weiter. Im Folgenden werden Trackingsysteme allgemein beschrieben.

2.4.1. Trackingsysteme

Für die Erfassung von Körperbewegungen steht eine Vielzahl verschiedenen Trackingsysteme, die sich unterschiedlichen Technologien bedienen, zur Verfügung. Diese Systeme kann man in drei Kategorien unterteilen:

1. **Inside-In Systeme** Bei diesen Systemen befinden sich die Sensoren am Körper des Benutzers und Bewegungen werden direkt in Signale umgewandelt. Ein Beispiel ist die Nutzung eines elektromechanischen Außenskeletts, bei dem die Messung über Potentiometer erfolgt, die über das Bewegen des mechanischen Skeletts am Körper verstellt werden.
2. **Inside-Out Systeme** Sensoren am Objekt/Körper messen "Datenäus externen Quellen Diese Systeme bestehen aus am Probanden befestigten Sensoren, die von externen Quellen gesendete Signale oder generierte Felder messen. Wie z.B. magnetische Feldstärke oder Ultraschal.
3. **Outside-In Systeme** Outside-In Systeme sind eine Umkehrung von Inside-Out- Systemen, Sender befinden sich am Körper des Benutzers, Empfänger außerhalb befestigt. Die Messsignale werden dabei von außen auf den Körper geworfen und die zurück kommenden Reflexionen gemessen. Beispiel hierfür ist das optische Tracking mit Markern.

Für jedes dieser Trackingsysteme gibt es eine Vielzahl von technischen Lösungen. Eine ausführliche Übersicht finden Sie in den Arbeiten von Heitsch (2008) und Boetzer (2008)

Alle gerade vorgestellten Techniken haben gemeinsam, dass sie physikalische Werte dynamisch oder statisch als Daten erfassen und an das verarbeitende digitale System ausliefern. Die jeweils eingesetzten Trackingmethoden unterscheiden sich

- in der Art der erfassten Daten,
- in der Präzision der Erfassung und

- dem verwendeten Datenmodell.

Einen detaillierten Überblick zur Art der gelieferten Daten und den dazugehörigen Genauigkeitsanforderungen in unserem Szenario wird im dazu gehörigen Kapitel „Analyse“ 3 erörtert.

2.4.2. Freiheitsgrade

Für die erfolgreiche Umsetzung unseres Vorhabens ist es wichtig den Begriff und die Bedeutung der unterschiedlichen Freiheitsgrade („Degree of Freedom“; Abgekürzt DOF) zu erklären.

Unter dem Begriff „Degree of Freedom“ (DOF) versteht man die verschiedenen möglichen Freiheitsgrade eines starren Körpers. Dabei umfassen die ersten drei DOFs die Translationen des Schwerpunktes des Starrkörpers in die verschiedenen Richtungsachsen (x, y, z), s.g. 3DOF. Die nächsten drei beschreiben die Drehbewegung um die Richtungsachsen, welche als Gier-, Roll- und Nickwinkel bezeichnet werden. Die Kombination aus 3 Translation und 3 Drehbewegung erlaubt dabei 6 Freiheitsgrade, abgekürzt 6DOF. Unterschiedliche Trackingsysteme erlauben hierbei das Erfassen einer unterschiedlichen Anzahl von Freiheitsgraden. Abbildung 2.3 zeigt die jeweiligen Freiheitsgrade eines Objektes für 6DOF.

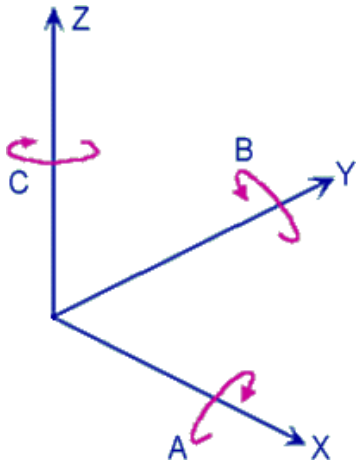


Abbildung 2.3.: Freiheitsgrade eines Starrkörpers aus Prospektiv (2008)

Der Aufbau eines Softwaresystems für die Benutzer-Interaktion mittels Gesten kann sehr Komplex sein, daher ist im Folgenden eine gründliche Analyse der Anforderungen an unser späteres System notwendig.

3. Analyse

Dieses Kapitel beschreibt zunächst das Anwendungsszenario aus der ursprünglichen Vision und macht Vorschläge für weitere Szenarien mit ähnlichen Problemstellungen (Abschnitt 3.1). Anschließend werden diese Szenarien ausgewertet (Abschnitt 3.2), um die Fragestellungen, die sie gemeinsam haben, zu isolieren. Abschließend werden die Ziele des zu entwickelnden Systems (Abschnitt 3.3) sowie dessen formalen Anforderungen (Abschnitt 3.4) festgehalten.

3.1. Anwendungsszenarien

Hier werden die Szenarien beschrieben, die durch Gestenerkennung unterstützt werden könnten. Neben dem ursprünglichen Unterhaltungsgeräte-Szenario werden auch zwei ähnliche, aus anderen Bereichen Szenarien beschrieben.

3.1.1. Steuerung der Unterhaltungsgeräte

Heutzutage verfügt fast jedes Unterhaltungsgerät über seine eigene Fernbedienung. Wenn man einen Fernseher, eine Stereoanlage, einen Satelliten-Receiver, einen DVD-Player, eine Spiel-Konsole, eine Klimaanlage im Wohnzimmer hat, stapelt sich ein Berg von 6 Fernbedienungen auf dem Couch-Tisch.

Eine s.g. universelle Fernbedienung kann beim Abtragen des Berges auch nicht immer helfen. Zuerst muss sie durch eintragen eines Geräte-Codes angelernt werden. Dabei stellt man fest, dass der alte gute Satelliten-Receiver leider nicht unterstützt wird. Ärgerlich wird es, wenn die Bedienung eines Lieblingsgeräts nur teilweise realisiert. Die wichtige Favoriten-Taste ist in einer sehr ungünstigen Ecke oder sogar überhaupt nicht zu finden.

Außerdem haben alle Fernbedienungen eine äußerst unangenehme Eigenschaft, wenn man sie braucht, sind sie weg. Anstatt „Tatort“ am Sonntag zu genießen, muss man erst nach der Fernbedienung suchen.

Abbildung 3.1.: Fernbedienungen aus Mangold (2010)

Wäre es nicht viel einfacher auf die Fernbedienungen zu verzichten und einfach per Gestenerkennung alle Unterhaltungsgeräte zu steuern? Wenn man nicht in jedes Gerät eine Kamera und die teurere Logik einbaut, sondern diese Elemente zentral für alle Geräte zur Verfügung stellt, könnte die Lösung erschwinglich werden.

3.1.2. Aktive Spiele für Kinder

Kinder kommen ungern zum längeren Möbel-Einkauf mit. Mehrstündiges Bummeln durch einen Konsumtempel begeistert kleine Zwerge nicht. Bei Ikea kann man sein Kind kostenlos in einem Småland parken. Andere große Kaufhäuser bieten ähnliche Kinderparadiese an.

In diesen Bereichen können Kinder zwischen 3 und 10 Jahren unter Aufsicht spielen. Üblicherweise bieten man den Kindern unterschiedliche Betätigungsbereiche an: ein Klettergerüst, viele Tafelspiele, ein Minikino, eine Malecke und natürlich eine Lego-Ecke. Der Personalbestand dieser Betreuungsstellen ist gering, die Anzahl der Kinder und die Verantwortung der Aufpasser dagegen hoch.

Um den Stress der Betreuer zu reduzieren oder noch mehr Kinder bei gleicher Anzahl von Aufsichtspersonen aufnehmen zu können, kann man „elektronische Animateure“ einsetzen, die Kinder in die Welt der aktiven Kino-Märchen mitnehmen. Man kann ein Kind zum Nachahmen einer Körperbewegung animieren und anschließend diese überprüfen.

In Bewegung können dann Kinder eine Piratengeschichte zusammen mit Capt’n Sharky miterleben. Oder zusammen mit Tabaluga das Grönland vor der Inversion des bösen Aktros retten. Mehrere Spiel-Szenarien sind möglich.

Abbildung 3.2.: KidsRoom aus Davis (1999)

3.1.3. Elektronischer Fitnesstrainer

Ein „elektronischer Fitnesstrainer“ auf Wii genießt eine hohe Popularität. Mit einem einzigartigen Konzept ist es Firma Nintendo gelungen, Konsolenspiele mit Sport und Bewegung zu verbinden. Um Nintendo zu übertreffen, will Microsoft in ihrer neuen Xbox 360 Spiel-Konsole nicht nur Bewegungen der Spieler erkennen, sondern sämtliche Vorgänge eines Spiels durch Gesten steuern lassen.

Der Trend geht zum Controllerlosen Entertainment. Also kann man einen ëlektronischen Fitnesstraineräuf Basis von Gestenerkennung programmieren. Dem Spieler wird eine Übung zum Nachmachen angeboten. Die Genauigkeit der Ausführung kann kontinuierlich kontrolliert werden.

3.2. Auswertung der Szenarien

Aus o.g. Szenarien ergeben sich nun einige Fragen, denen im Rahmen dieser Arbeit und bei der Implementierung eines Prototypen (siehe Kapitel 4) nachgegangen wurde.

Wo, wann und wie findet die Bewegung statt?

Wie erkennt man eine Bewegung in einer Bildsequenz? Wie isoliert man die Bewegung aus dem Bild? Wie erkennt man den Anfang und das Ende einer Geste auf der Zeitachse? Wo befindet sich der Benutzer? Wie viele Freiheitsgrade soll man betrachten, um eine Bewegung zuverlässig zu erkennen?

Abbildung 3.3.: Interactive Virtual Aerobics Trainer aus Ogden (1999)

Ist diese Bewegung eine Geste?

Wie soll ein Modell einer Person aussehen und die Taxonomie ihrer Bewegungen zusammengestellt sein, um eine Gesten zu erkennen? Reicht alleine die visuelle Information über einen Bewegungsablauf aus, um die Bewegung zu erkennen?

Kann dieser Geste eine Bedeutung zugeordnet werden?

Welche Faktoren, wie Standort, Zeitpunkt, Richtungsvektor müssen wir dem Bild entnehmen, um die Geste korrekt zu deuten? Wie speichert man eine Geste in einer Datenbank? Können wir eine Geste durch unaufwändige Techniken, wie Mustererkennung interpretieren?

3.3. Zielsetzung

Ziel dieser Arbeit ist es, einen Prototyp eines Systems, wie es in Abschnitt 3.1.1 beschrieben ist, beispielhaft zu implementieren. Das System soll also Unterhaltungselektronikgeräte wie Fernseher, Stereoanlagen oder DVD-Player per Gestenerkennung steuern, ohne von Marker, wie spezielle Handschuhe, Gebrauch zu machen. Anwendungen für die Fälle aus den Abschnitten 3.1.2 und 3.1.3 sind in der Literatur bereits beschrieben.

Die Anwender von Markern zu befreien, kann sich positiv auf die Benutzerakzeptanz dieser Systeme auswirken. In den bisherigen Arbeiten an der HAW Hamburg, die im Kontext von Gestenerkennung stehen (siehe Heitsch (2008), Rödiger (2010), Boetzer (2008)), wurde auf ein Modell einer Person und eine Taxonomie ihrer Gesten zurückgegriffen. In dieser Arbeit jedoch soll das Augenmerk auf der Erprobung von robusten Mustererkennungsverfahren liegen, die relativ wenig Rechenaufwand erfordern und als Embedded System in naher Zukunft implementiert werden können. Um dieses Ziel zu erreichen, erscheint es sinnvoll einen Prototyp zu entwickeln, der die Umsetzbarkeit des Vorhabens beweist. Bewusst verzichten wir auf ein Personenmodell und eine Taxonomie der Gesten und beschränkten uns auf nur 3 Freiheitsgrade einer Bewegung.

Unser Gerät sollte ein ganzes Wohnzimmer abdecken und leicht durch einen neuen Benutzer anlernbar sein. Die Anwendung einer 360° Fish Eye Kamera, die an der Decke in der Mitte eines Raums befestigt wird, erscheint daher wirtschaftlich sinnvoll. Als Schnittstellen zu den Unterhaltungsgeräten können Infrarot-Transmitter zum Einsatz kommen, die direkt in das Kameragehäuse integriert sind. Der beschriebene Ansatz soll nicht nur eigenständig genutzt werden, sondern andere Steuerungs- und Tracking-Techniken in einer intelligenten Wohnumgebung, wie z.B. Spracherkennung oder Sensorik unterstützen und erweitern.

Abbildung 3.4.: 360° Fish Eye Kamera von Mobotix (2010)

3.4. Anforderungsanalyse

Im Rahmen dieser Arbeit wird ein Prototyp entwickelt, anhand dessen die Zweckmäßigkeit von markerloser zentralisierter Gestenerkennung zur Gerätesteuerung untersucht werden kann. Dementsprechend ist es nicht das Ziel ein ausgereiftes Produkt zu präsentieren, sondern eine minimale Applikation, die das Szenario aus Abschnitt 3.1.1 hinreichend annähert. Die folgenden Abschnitte beschreiben die funktionalen Anforderungen an den Prototypen, getrennt nach Benutzeranforderungen und Systemanforderungen. Benutzeranforderungen beschreiben die Dienste des Systems aus Sicht des Endanwenders. Die Systemanforderungen stellen eine detailliertere und präzisere Beschreibung der Dienste und Beschränkungen des Systems aus Sicht des Systemarchitekten dar. Auf die Erläuterung der nichtfunktionalen

Anforderungen in Anlehnung an die Normen ISO/IEC 9126 oder DIN 66272 wird verzichtet.

3.4.1. Benutzeranforderungen

Die Dienste des Systems sollen sich, aus Sicht des Benutzers, folgendermaßen verwenden lassen:

1. Der Anwender lernt das System an.
    - Dabei nimmt er eine Arm-Bewegung auf.
    - Im 2. Schritt richtet er eine Fernbedienung seines Fernsehers auf unser Gerät und nimmt ein IR-Kommando auf.
    - Diesen IR-Befehl seiner Fernbedienung verknüpft der Benutzer mit der davor aufgenommenen Arm-Bewegung.
    - Danach kann der Anwender den gerade aufgenommenen Befehlt mehrmals testen.
    - Der Vorgang wird solange wiederholt, bis alle Gesten und Befehle aufgenommen sind.
2. Der Benutzer befindet sich in seinem Wohnzimmer und kann sich frei im Raum bewegen. Das System erkennt zuverlässig seine Gesten, interpretiert sie als Befehle und schickt diese als IR-Kommandos an den Fernseher.

3.4.2. Systemanforderungen

An dieser Stelle sollen die Teile des Systems, die in den Benutzeranforderungen nur oberflächlich dargestellt wurden, präziser festgehalten werden. Unsere eigentliche Anwendung soll zwei Arbeitsmodi beinhalten:

- Anlernen der Gesten und der IR-Befehle;
- Steuerung der Unterhaltungselektronik durch Gestenerkennung.

Anlernen der Gesten setzt folgende Systemanforderungen voraus:

Kommunikation mit dem Benutzer

Das Gerät soll eine Möglichkeit haben mit dem Anwender zu kommunizieren, um ihn zum Aufnehmen der Gesten aufzufordern.

Aufnahme der Bewegungen und Speicherung dieser in einer Datenbank

Das System soll in der Lage sein Arm-Bewegungen des Anwenders aufzunehmen, eindeutige Merkmale dieser zu extrahieren und platzsparend in einer Datenbank ablegen.

Empfang der IR-Kommandos einer Fernbedienung und Speicherung dieser in einer Datenbank

Das Gerät soll IR-Befehle einer Fernbedienung aufzunehmen und in einer Datenbank ablegen. Da die IR-Kommandos digitale Daten sind und eine Vielfalt von IR-Transceiver auf dem Markt zu finden ist, stellt diese Anforderung keine Herausforderung dar.

Steuerung der Unterhaltungselektronik durch Gestenerkennung erzeugt folgende Systemanforderungen:

Markerlose zentralisierte Erkennung der Gesten

Bei der Gestenerkennung wollen wir auf Marker verzichten. Ein einziges Gerät muss in der Lage sein ein durchschnittliches Wohnzimmer von 12qm zu überblicken, Bewegungen zu erkennen und zu interpretieren.

Zuverlässige Erkennung der Gesten

Definitionen von 2 Arten der Fehlerkennung aus Wikipedia sind für unsere Analyse besonders wichtig:

Falsch Positiv. Ein Testergebnis ist falsch positiv, wenn es fälschlicherweise anzeigt, dass das Testkriterium erfüllt (also positiv) sei. Das Kriterium ist also tatsächlich nicht erfüllt, wird aber vom Test als erfüllt erkannt. Für ein falsch positives Ergebnis wird häufig die englische Bezeichnung "false positive"verwendet. Bei einem statistischen Test mit Nullhypothese führt ein falsch positives Ergebnis zu einem Fehler 1. Art.

Falsch Negativ. Ein Testergebnis ist falsch negativ, wenn es fälschlicherweise anzeigt, dass das Testkriterium nicht erfüllt (also negativ) sei. Das Kriterium ist also tatsächlich erfüllt,

wird aber vom Test nicht erkannt. Für ein falsch negatives Ergebnis wird häufig die englische Bezeichnung "false negative"verwendet. Bei einem statistischen Test mit Nullhypothese führt ein falsch positives Ergebnis zu einem Fehler 2. Art.

Der Algorithmus zur Gestenerkennung muss einfach, robust und zuverlässig sein. D. h. die Anzahl der Fehlerkennungen muss gering sein!

Versenden der IR-Kommandos

Das System soll in der Lage sein, die IR-Befehle mittels IR-Transceiver an die Geräte zu senden. Da unser Gerät an der Decke in der Mitte eines Wohnzimmers befestigt wird, können wir es mit einem IR-Transceiver ausstatten, dessen IR-LEDs in alle Himmelsrichtungen strahlen.

Die Q24 Fish Eye Kamera von Mobotix eignet sich hervorragen für unsere Zwecke. Dieses Gerät verfügt über einen Lautsprecher, ein Mikrofon und eine grafische Web-Oberfläche hat.

In dieser Arbeit konzentrieren wir uns auf Gestenerkennung und vernachlässigen bewusst die Infrarot-Schnittstelle.

3.5. Übersicht der Methoden für Gestenerkennung

Die Abblildung 3.5 zeigt den Ablauf der robusten Gestenerkennung.

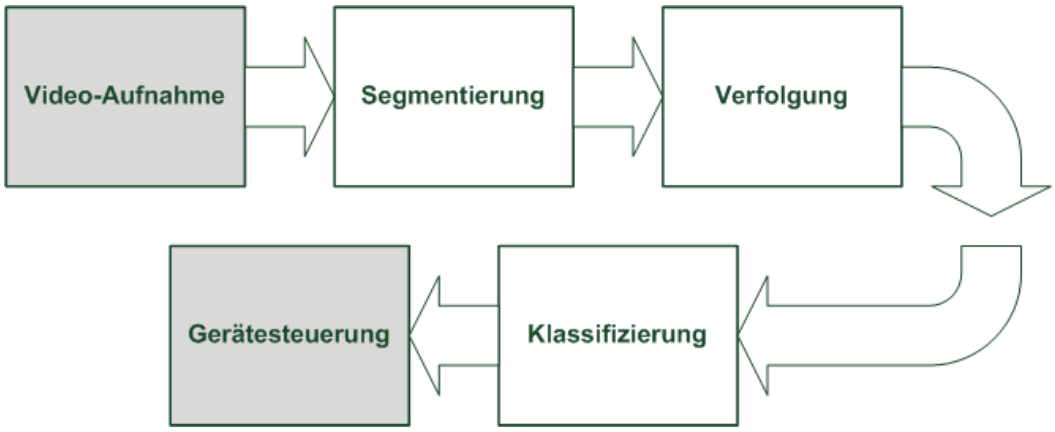


Abbildung 3.5.: Der Grundaufbau der markerlosen Gestenerkennung

Algorithmisch besteht die markerlose Gestenerkennung aus 3 Hauptteilen:

- Gesten-Segmentierung
- Objektverfolgung oder Tracking
- Gestenerkennung durch Klassifikation

Im folgenden Abschnitt werden Methoden und Techniken für diese 3 wichtigen Aufgaben kurz vorgestellt. Wir verzichten hier auf die ausführliche Beschreibung der aufgelisteten Ansätze, weil die meisten davon wegen ihrer Komplexität, Rechenintensivität oder Anforderungen an die Bildquelle sich nur schwer für ein hochauflösendes eingebettetes Echtzeit-System mit Fish-Eye-Objektiv eignen. Vielmehr wollen wir dem Leser die Vielfalt der bestehenden Ansätze präsentieren.

3.5.1. Segmentierung

Ein häufig wiederkehrendes Problem in der Bildverarbeitung ist die Unterscheidung zwischen unwichtigem Hintergrund (z.B. Wände, Tische) und potentiell wichtigem Vordergrund (z.B. Personen) in den Bildern.

Ein Ansatz hierzu, der bei statischen Kameras verwendet wird, ist die Background Subtraction. Dabei wird angenommen, dass Hintergrundobjekte im Gegensatz zu Vordergrundobjekten ihre Position und ihr Erscheinungsbild über die Zeit nicht verändern. Aus diesem Grund wird zunächst anhand von Bildern, auf denen nur Hintergrund sichtbar ist, ein pixelbasiertes Modell des Hintergrunds erstellt. Neue Bilder werden dann mit dem Modell des Hintergrunds verglichen, um zu entscheiden, welche Pixel zum Hintergrund und welche Pixel zum Vordergrund gehören. Um robuster gegenüber Änderungen der Lichtverhältnisse zu sein, wird das Hintergrundmodell in der Regel fortdauernd angepasst.

Beispiel-Methoden aus Bradski und Kaehler (2008) für Background Subtraction sind:

- Averaging Methode
- Connected-Ccomponents Cleanup
- Codebook Methode
- Connected-Component Maske

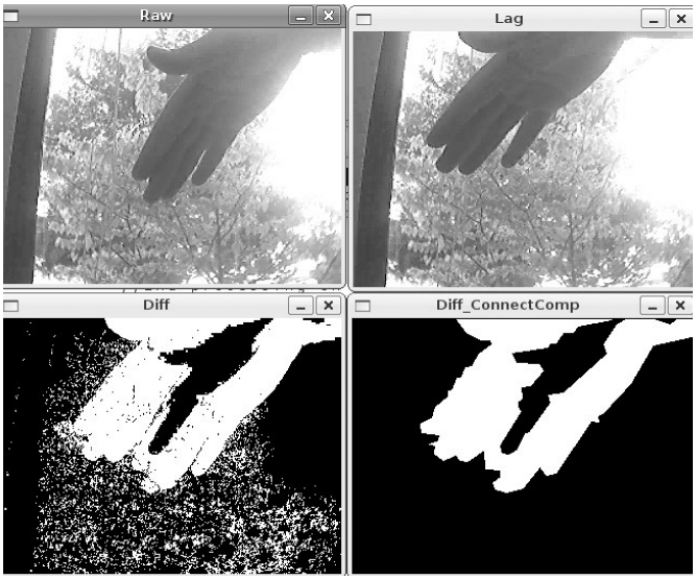


Abbildung 3.6.: 2 verschiedenen Methoden zur Handerkennung aus Bradski und Kaehler (2008)

Segmentierung nach Farbe

Bei der Ähnlichkeitsbestimmung von Bildern werden oft Farbähnlichkeiten ausgenutzt. Stark ähnliche Bilder beinhalten oft eine Anzahl vieler ähnlicher Farben. Um die Farbigkeit von Bildern vergleichbar zu machen, verwendet man Farbhistogramme. Die Idee von Farbhistogrammen ist, dass die Anzahl des Auftretens von festgelegten Referenzfarben in einem Bild bestimmt und verglichen wird. Für jeden Pixel des Bildes wird dann bestimmt welche Farbe es hat und welcher Referenzfarbe diese am nächsten liegt. Die Anzahl der nächstliegenden Referenzfarbe wird dann um 1 erhöht. Am Ende werden die gezählten Häufigkeiten normiert, d.h. durch die Gesamtanzahl der Pixel des Bildes dividiert. Somit erhält man die relative Häufigkeiten der Referenzfarben, die von verschiedenen Bildern miteinander vergleichbar sind.

Für die Gestenerkennung eignet sich Segmentierung nach Hautfarbe.

Segmentierung nach Form

Laut Wikipedia ist die Kantendetektion (engl. edge detection) Teil einer Segmentierung in der Bildbearbeitung, bei der versucht wird, flächige Bereiche in einem digitalen Bild voneinander zu trennen. Man hofft, dass Kantenoperatoren die Übergänge zwischen diesen Bereichen erkennen. Diese Übergänge werden als Kanten markiert. Im Gegensatz dazu soll aber auch

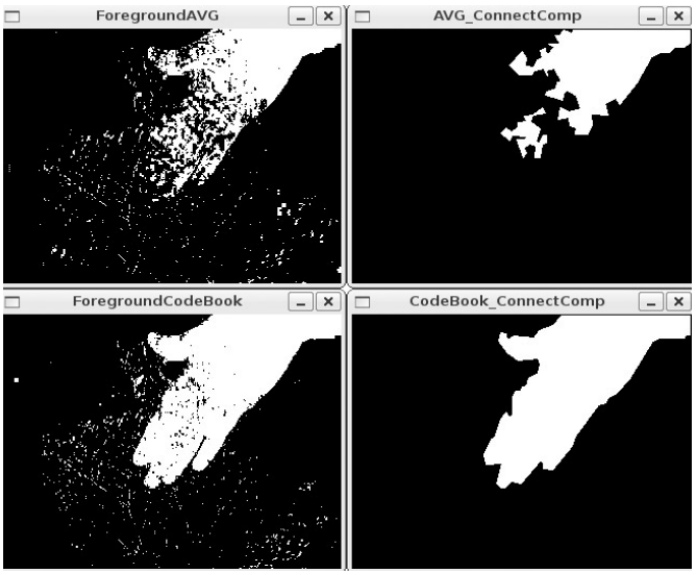


Abbildung 3.7.: 4 verschiedenen Methoden zur Handerkennung im Vergleich aus Bradski und Kaehler (2008)

ein einzelner Bereich als solcher erkannt werden und von einem guten Kantendetektor nicht durch eine Kante in zwei Teile geteilt werden.

Ein Kantendetektor berechnet aus einem Bild sein entsprechendes Kantenbild, in dem alle Kanten entsprechend zu sehen sind. Um dieses zu erreichen, wird jeder Bildpunkt durch eine Berechnung mittels einer Matrix neu gesetzt.

Dies ist jedoch keine Matrix im mathematischen Sinne sondern eine Filtermaske. D.h. man wendet darauf keine Matrizenoperationen an, sondern die diskrete Faltung.

Der Hauptunterschied zwischen den Kantenfiltern liegt darin, dass sie verschiedene Filtermasken benutzen.

Die bekanntesten Kantenoperatoren aus Bradski und Kaehler (2008) sind:

- Sobel-Operator
- Scharr-Operator
- Laplace-Filter
- Prewitt-Operator
- Roberts-Operator
- Kirsch-Operator

Abbildung 3.8.: Segmentierung nach Hautfarbe aus Bradski und Kaehler (2008)

- Canny-Algorithmus
- Marr-Hildreth-Operator
- Kontrastverstärker

Eine Hauptschwierigkeit der Kantendetektion, insbesondere bei dreidimensionalen Objekten, ist die Unterscheidung von Reflexionskanten, die auf Eigenschaften des Objektes beruhen, und Beleuchtungskanten, die auf Eigenschaften der Beleuchtung beruhen (z.B. Schatten, Lichtkegel).

Für den Menschen ist Kantendetektion eine wesentliche Voraussetzung zur visuellen Objekterkennung. Sein Sehapparat leistet diese Aufgabe hauptsächlich mit der lateralen Hemmung.

Viele Grafik-Programme und Bibliotheken verfügen bereits über einen Filter zur Erkennung von Konturkanten (engl. edge detection).

Schwache Klassifikatoren als Detektoren in Graustufenbilder

Man kann nach Händen in den Graustufenbildern mit Hilfe von s.g. schwachen Klassifikatoren suchen.

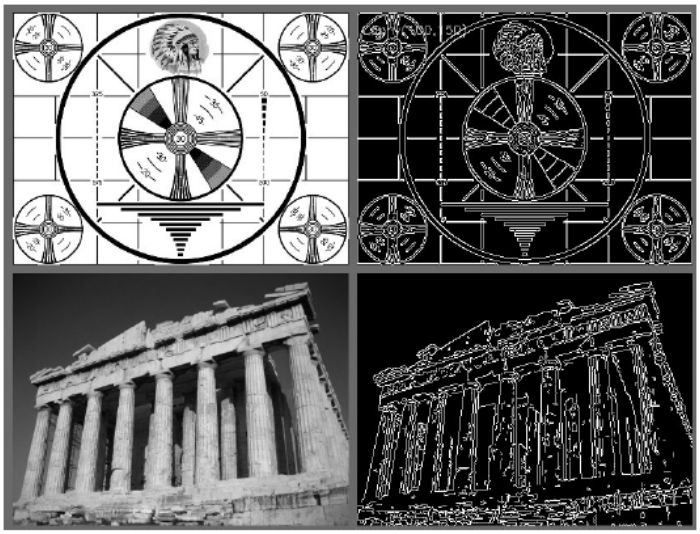


Abbildung 3.9.: Segmentierung nach Hautfarbe aus Bradski und Kaehler (2008)

Vorgegeben ist eine Reihe von Objekten und eine Reihe schwacher Klassifikatoren. Gesucht wird ein Klassifikator, der die Objekte möglichst fehlerfrei in zwei Klassen einteilt. Boosting kombiniert die vorhandenen schwachen Klassifikatoren so, dass der entstehende neue Klassifikator möglichst wenige Fehler macht.

Schwache Klassifikatoren, auch base classifiers (engl. „Basisklassifikatoren“) oder weak learners (engl. „schwache Lerner“) genannt, sind sehr einfach aufgebaut und berücksichtigen meist nur ein einziges Merkmal der Objekte. Für sich genommen liefern sie deswegen einerseits schlechte Ergebnisse, können aber andererseits sehr schnell ausgewertet werden. Boosting führt alle schwachen Klassifikatoren so mit einer Gewichtung zusammen, dass die stärkeren unter den schwachen Klassifikatoren besonders berücksichtigt, die wirklich schwachen hingegen ignoriert werden.

Boosting kann überall dort verwendet werden, wo eine automatische Klassifikation in zwei Klassen benötigt wird, beispielsweise um Bilder von Gesichtern in „bekannt“ und „unbekannt“ oder Produkte auf einem Fließband als „in Ordnung“ oder „fehlerhaft“ einzustufen.

Folgende Boosting-Algorithmen aus Bradski und Kaehler (2008) können als Beispiel dienen:

- AdaBoost
- FloatBoost

Objeketdetektion mittels 3D-Modelle

Dreidimensionale Modelle ermöglichen eine Sicht-unabhängige Detektion. Das verwendete Modell soll über genügen Freiheitsgrade verfügen, um eine Person abzubilden. RAMSIS (Rechnergestütztes Anthropometrisch-Mathematisches System zur Insassen-Simulation) dient als Beispiel für ein solches Modell.

Bei RAMSIS handelt es sich um ein am Lehrstuhl für Ergonomie der Fakultät für Maschinenwesen der TU München entwickeltes dreidimensionales CAD Modell. Es besteht aus einem inneren Model (Skelett) sowie einem äußeren Modell (Oberfläche/Haut), welche jeweils in ihren Ausmaßen (Körpergröße, Körperumfang) relativ exakt an beliebige Versuchsprobanden angepasst werden können (anthropometrische Anpassung). Somit steht dem Programmierer ein detailgetreues 3D Modell der Versuchsperson zur Verfügung.

Durch Änderung der Gelenkwinkel des inneren Models kann RAMSIS beliebige Haltungen einnehmen. Die Gelenke des inneren Models bilden dabei eine kinematische Kette, d.h. jedes nachfolgende Körperelement wird von der Gelenkstellung des Vorgängerelements beeinflusst, z.B. wird bei einer Änderung der Gelenkstellung des Oberarms der Unterarm mit bewegt. Ausgangselement der Transformationskette ist dabei das Becken. Um jede beliebige Haltung beschreiben zu können, sind zusätzlich noch drei Translationsparameter für die Lage des Beckens im Raum notwendig. Somit ist eine Haltung durch die 3D-Rotationen der 28 Gelenke sowie einer initialen Translation komplett beschrieben.

Wikipedia erwähnt auch folgende 3D-Menschenmodelle:

- Jack
- Anybody
- Human Builder (Safework)
- SANTOS

3.5.2. Objekverfolgung

Tracking oder Bild-für-Bild-Verfolgung von segmentierten Bildbereichen oder Merkmalen ist der 2. Schritt in Prozess der Gestenerkennung.

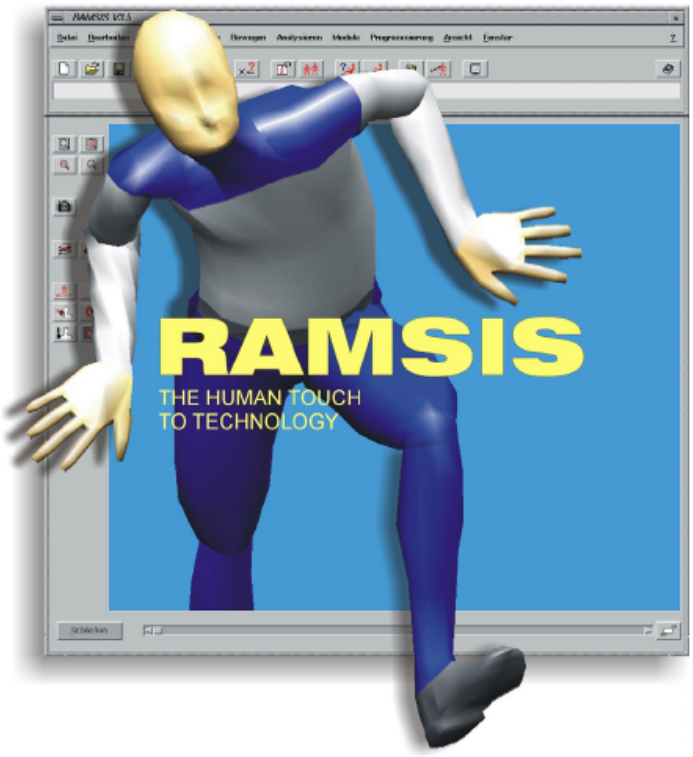


Abbildung 3.10.: RAMSIS (Rechnergestütztes Anthropometrisch-Mathematisches System zur Insassen-Simulation) aus Hudelmaier (2001)

Techniken der optimalen Schätzung

Optimal Estimation Techniques ermöglichen uns die Laufbahn eines Objektes mit großer Wahrscheinlichkeit korrekt vorherzusagen. Ein typisches und weit verbreitetes Beispiel dieser Technik aus Bradski und Kaehler (2008) ist Kalman-Filter.

Laut Wikipedia ist das Kalman-Filter ein nach seinem Entdecker Rudolf E. Kálmán benannter Satz von mathematischen Gleichungen. Mithilfe dieses Filters sind bei Vorliegen lediglich fehlerbehafteter Beobachtungen Rückschlüsse auf den Zustand von vielen der Technologien, Wissenschaften und des Managements zugeordneten Systemen möglich. Vereinfacht gesprochen dient das Kalman-Filter zum Entfernen der von den Messgeräten verursachten Störungen. Dabei müssen sowohl die mathematische Struktur des zugrundeliegenden dynamischen Systems als auch die der Messverfälschungen bekannt sein.

Im Rahmen der mathematischen Schätztheorie spricht man auch von einem Bayes'schen Minimum-Varianz-Schätzer für lineare stochastische Systeme in Zustandsraumdarstellung.

Eine Besonderheit des 1960 von Kálmán vorgestellten Filters bildet seine spezielle mathematische Struktur, die den Einsatz in Echtzeitsystemen verschiedenster technischer Bereiche ermöglicht. Dazu zählen u.a. die Auswertung von Radarsignalen zur Positionsverfolgung von sich bewegenden Objekten (Tracking) aber auch der Einsatz in elektronischen Regelkreisen allgegenwärtiger Kommunikationssysteme wie etwa Radio und Computer.

Abbildung 3.11.: Dilbert wendet Kalman-Filter an aus Dean Oliver (1997)

Tracking mittels Mean Shift Algorithmus

Der Mean Shift Algorithmus findet den Schwerpunkt (mean) von Objekten, indem er den Schwerpunkt eines zu untersuchendes Fensters in einem Bild so lange mit dem Schwerpunkt eines Objekts vergleicht, bis diese annähernd übereinstimmen. Die Objekte werden anhand ihrer Farbwahrscheinlichkeitsverteilung bestimmt, dass heißt je deutlicher sich das Objekt farblich vom Hintergrund unterscheidet, desto besser ist es zu verfolgen.

Camshift ist eine Erweiterung des Mean-Shift-Verfahrens, und zwar steht Camshift für Continously Adaptive Meanshift. Dieser Algorithmus ist in der Lage, sein Suchfenster selbstständig zu vergrößern und zu verkleinern. So kann man z.B. das gesamte Bild als Suchfenster vorgeben, und nach einigen Durchläufen ist dieses auf den größten Massepunkt zentriert und angepasst. Zudem berechnet er die Ausrichtung und die Ausmaße des Objektes, im Beispiel durch ein Kreuz eingezeichnet.

Tracking mit einem Partikel-Filter

Ziel eines Partikel-Filters ist, die gerade aktuelle aber unbekannte Wahrscheinlichkeitsdichte eines Zustandsraumes zu schätzen, um daraus Aussagen über den wahrscheinlichsten

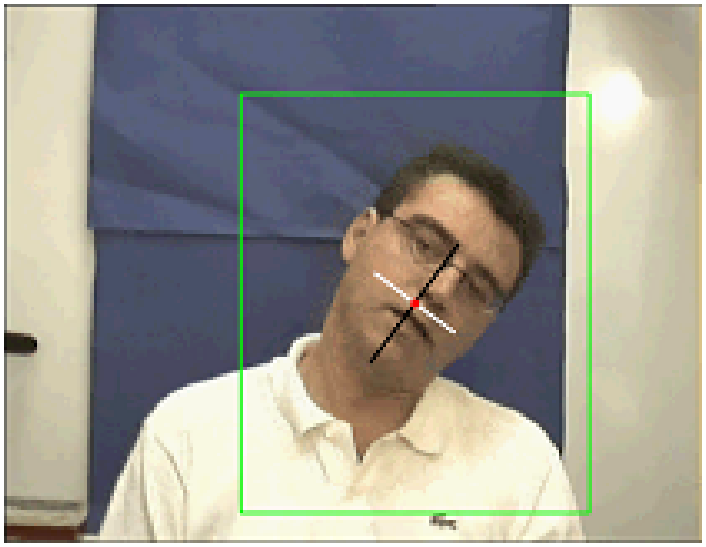


Abbildung 3.12.: Continously Adaptive Mean Shift Algorithmus aus Perales (2003)

Zustand des dynamischen Systems zu einem bestimmten Zeitpunkt treffen zu können. In unserem Szenario ist der Zustandsraum der Raum aller möglichen Haltungen des Menschmodells, und es gilt die wahrscheinlichste Haltung zu einem bestimmten Zeitpunkt in diesem Raum zu bestimmen.

Um den im Allgemeinen nicht normalverteilten Zustandsraum beschreiben zu können, wird eine Menge an Partikeln über den Zustandsraum verteilt. Jedes Partikel stellt dabei eine Hypothese über einen speziellen Zustand dar, und ist mit einem Gewicht verknüpft, welches der Wahrscheinlichkeit des Zustands angenähert ist. Umso höher die Anzahl N der Partikel, umso genauer wird der Zustandsraum beschrieben. Der Vorteil von Partikel-Filtern liegt darin, dass beliebig komplexe Zustandsräume beschrieben werden können (solange genug Partikel verwendet werden), auch solche mit mehreren Maxima. Somit können mehrere Hypothesen, die sich aus der Nichteindeutigkeit gemachter Beobachtungen ergeben, verfolgt werden, bis sich die Mehrdeutigkeiten wieder auflösen.

Das Verfahren kann für jeden Zeitschritt in die folgenden Einzelschritte unterteilt werden. Es wird angenommen, dass der Zustandsraum für den vorhergehenden Zeitschritt bereits durch eine Partikel-Menge mit zugeordneten Gewichten angenähert ist. Dies kann im ersten Zeitschritt z.B. durch die gleichmäßige Verteilung von uniform gewichteten Partikeln über den Zustandsraum erfolgen:

1. Auswählen der neuen Partikel: Es werden aus der alten Partikel-Mengen neue Partikel mit der Wahrscheinlichkeit ihres Gewichtes gezogen (Ziehen mit Zurücklegen).
2. Vorhersage des neuen Objektzustandes: Basierend auf einem Bewegungsmodell wird

für jedes Partikel der neue Zustand geschätzt. Zusätzlich wird ein Rauschen überlagert, um die Unsicherheit des Bewegungsmodells abzubilden.

3. Berechnung der neuen Gewichte: Basierend auf der gemachten Beobachtung werden die Gewichte für jedes Partikel neu berechnet.

4. Um den besten Zustand zum aktuellen Zeitpunkt zu bestimmen, kann man z.B. den gewichteten Mittelwert aller Partikel berechnen oder einfach das Partikel mit dem höchsten Gewicht auswählen.

Ein Beispiel dieser Sorte aus Bradski und Kaehler (2008) ist Condensation Algorithmus (Conditional Density Propagation for Visual Tracking). Der Algorithmus verfolgt mit Hilfe eines probabilistischen Ansatzes, der die konkurrierende Betrachtung verschiedener Handlungs-Hypothesen erlaubt, ein Objekt in einer Videosequenz. Das Ziel ist, auch bei komplexer Objektumgebung eine möglichst sichere Erkennung in Echtzeitverarbeitung zu erreichen. An ausgewählten Stellen wird eine Wahrscheinlichkeitsverteilung repräsentiert. Durch einen Vergleich der Bilddaten mit diesen Stellen wird eine fortwährende Anpassung der Wahrscheinlichkeitsverteilung erreicht.

3.5.3. Objekterkennung

Unter Objekterkennung verstehen wir die Interpretation der Semantik. In unserem Fall ist das die Erkennung der Bedeutung der Gesten.

Hauptkomponentenanalyse

So beschreibt Wikipedia die Hauptkomponentenanalyse: Principal Component Analysis (PCA) ist ein Verfahren der multivariaten Statistik. Sie dient dazu, umfangreiche Datensätze zu strukturieren, zu vereinfachen und zu veranschaulichen, indem eine Vielzahl statistischer Variablen durch eine geringere Zahl möglichst aussagekräftiger Linearkombinationen (die "Hauptkomponenten") genähert wird. Speziell in der Bildverarbeitung wird die Hauptkomponentenanalyse auch Karhunen-Loève-Transformation genannt. Sie ist von der Faktorenanalyse zu unterscheiden, mit der sie formale Ähnlichkeit hat und in der sie als Näherungsmethode zur Faktorenextraktion verwendet werden kann.

Also wird Hauptkomponentenanalyse zur Reduktion der Rohdaten vor der eigentlichen Klassifikation benutzt. Dies bildet die Grundlage für das Anwenden simpler Klassifikatoren und erleichtert die anschließende Erkennung von Mustern.

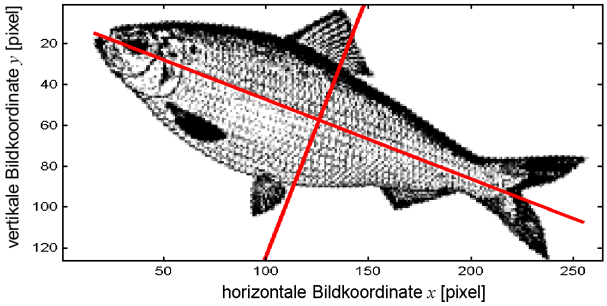


Abbildung 3.13.: Hauptkomponentenanalyse für die Bildverarbeitung: Zwei Hauptkomponenten eines Bildes aus Wikipedia (2004b)

Boosting

Im Abschnitt 3.5.1 haben wir Boosting zur Segmentierung von Bildbereichen vorgestellt. Boosting übernimmt die Aufgabe der Klassifizierung (von lat. classis, „Klasse“, und facere, „machen“), sprich das Zusammenfassen von Objekten zu Gruppen.

Boosting (engl. „Verstärken“) ist ein Algorithmus der automatischen Klassifizierung, der mehrere schwache Lernalgorithmen zu einem einzigen starken Klassifikator verschmilzt.

Abbildung 3.14.: Objekte zum Trainieren von AdaBoost aus Adolf (2003)

Model-basierende Erkennungsmethoden

Dreidimensionale Modelle aus dem Abschnitt 3.5.1 können nicht nur bei der Segmentierung, sondern auch bei der Erkennung gute Dienste leisten. Dabei werden die extrahierten Merkmale mit einem vordefinierten Modell verglichen.

HMMs

Wikipedia beschreibt das Hidden Markov Model (HMM) als ein stochastisches Modell, das sich durch zwei Zufallsprozesse beschreiben lässt. Es ist die einfachste Form eines dynamischen Bayes'schen Netzes.

Der erste Zufallsprozess entspricht dabei einer Markow-Kette, die durch Zustände und Übergangswahrscheinlichkeiten gekennzeichnet ist. Die Zustände der Kette sind von außen jedoch nicht direkt sichtbar (sie sind hidden, verborgen). Stattdessen erzeugt ein zweiter Zufallsprozess zu jedem Zeitpunkt beobachtbare Ausgangssymbole (oder Beobachtungen) gemäß einer zustandsabhängigen Wahrscheinlichkeitsverteilung. Das Attribut verborgen bezieht sich bei einem Hidden Markov Model auf die Zustände der Markow-Kette während der Ausführung, nicht auf die Parameter des Markow-Modells. Die Aufgabe besteht häufig darin, aus der Sequenz der Ausgabesymbole (Beobachtungen) auf die Sequenz der verborgenen Zustände zu schließen.

Im Kontext der Gestenerkennung werden Ausgabesymbole durch Erkennung von Posen (Tokens) in den Bildern geschätzt. Aus diesem Grund und weil Gesten als Folge von Posen erkannt werden können, fanden HMMs eine große Verbreitung für Gestenerkennung. Es ist typisch für jede Geste ihr eigenes HMM zu bauen. Das Erkennungsproblem reduziert man auf die Wahl eines HMM, das die beste Erkennungsrate einer Geste liefert.

4. Design und Realisierung

Im diesem Kapitel wird eine Technik vorgestellt, die es uns auf einfache Weise erlaubt Bewegung darzustellen und zu vektorisieren um die gewonnenen Daten in einer Datenbank zu speichern, oder mit diesen zu vergleichen (Abschnitt 4.1). Wir gehen von einer statischen Kameramontage aus. Anschließend werden hemisphärische Kameras (Abschnitt 4.6) und die Bibliothek OpenCV vorgestellt (Abschnitt 4.7). Im Abschnitt 4.8 wird der Aufbau des Prototypen beschrieben. Abschließend werden die Ergebnisse evaluiert (Abschnitt 4.9).

4.1. Motion-Energy Images

Bei einem Motion-Energy Image (kurz MEI) werden aufeinander folgende Bilder einer Bildsequenz verglichen, um durch Differenzbildung zu erfahren an welchen Stellen die Bewegung stattfand. Die so erfassten Stellen werden über einen festgelegten Zeitraum in einem binären Bild festgehalten, wobei das Bild zu Beginn schwarz ist und die Stellen wo die Bewegung stattfindet weiß gesetzt werden. Dadurch erhält man ein Bild wie es in Abbildung 4.2 dargestellt ist. Mathematisch betrachtet, entsteht das MEI indem man aus einer Bildsequenz $I(x, y, t)$ zum Zeitpunkt t jeweils immer das Differenzbild erstellt und diese Bilder in der Sequenz $D(x, y, t)$ zusammenfasst, welche zeigt wo Bewegung zum Zeitpunkt t stattfand. Das MEI $E_\tau(x, y, t)$ wird berechnet indem alle Differenzbilder folgendermaßen zusammengefasst werden.

$$E_\tau(x, y, t) = \bigcup_{i=0}^{\tau-1} D(x, y, t - i)$$

τ bestimmt dabei über welchen Zeitraum die Bilder betrachtet werden. Da man nicht davon ausgehen kann, dass eine Person immer im gleichen Winkel zur Kamera steht, wird wie in Abbildung 4.2 gezeigt von verschiedenen Winkeln aus das MEI erzeugt.

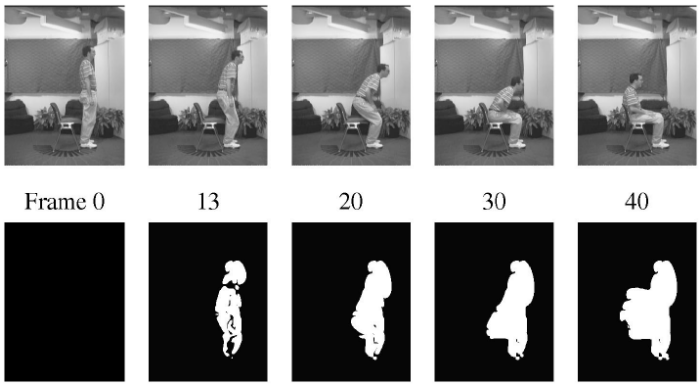


Abbildung 4.1.: MEI bei einer sich hinsetzenden Person aus J.Davis und Bobick (1997)

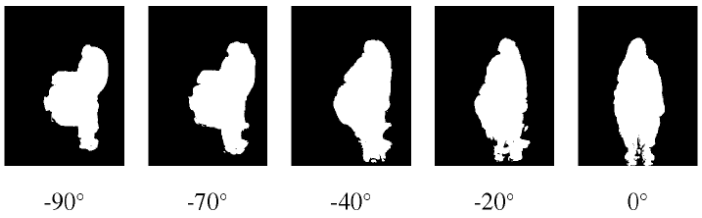


Abbildung 4.2.: MEI bei einer sich hinsetzenden Person aus J.Davis und Bobick (1997)

4.2. Motion-History Images

Da wir nicht nur daran interessiert sind wo, sondern auch wie und in welche Richtung die Bewegung stattfindet, bedienen wir uns des s. g. Motion-History Image (kurz MHI). Bei diesem handelt es sich um ein Graustufenbild, bei welchem die Stellen wo die Bewegung stattfand umso heller sind, desto kürzer die Zeit her ist, in der die Bewegung stattfand. Das Ergebnis ist dann ein Bild, wie in Abbildung 4.3, wo man leicht erkennen kann wie die Bewegung ablief. Mathematisch wird das MHI durch die rekursiv definierte Funktion H_τ berechnet, diese wie folgt definiert ist.

$$H_\tau(x, y, t) = \begin{cases} \tau, & \text{falls } D(x, y, t) = 1, \\ max(0, H_\tau(x, y, t-1) - 1), & \text{falls nicht} \end{cases}$$

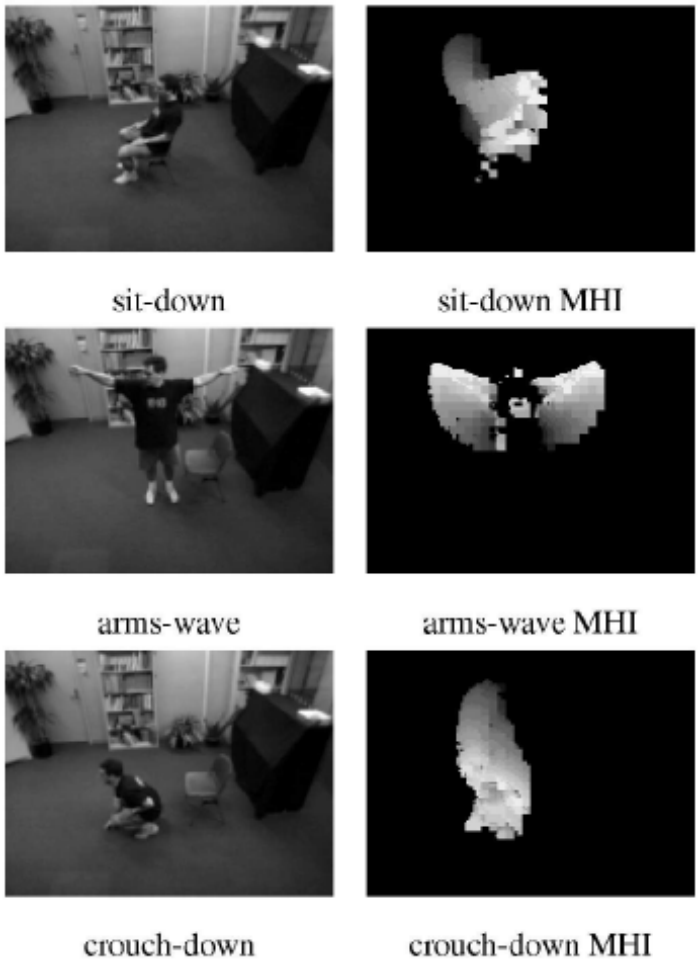


Abbildung 4.3.: MHI bei einer sich hinsetzenden Person aus J.Davis und Bobick (1997)

4.3. Erweiterungen zur Steigerung von Effizienz und Zuverlässigkeit

Mit einfachen MEIs und MHIs alleine ist es noch nicht möglich ein flexibles System zu schaffen, welches uns ermöglicht, verschiedene Bewegungen zu identifizieren, da es vorkommen kann, dass eine Bewegung schneller oder langsamer durchgeführt werden kann, als es beim Anlernen der Datenbank der Fall war. Zudem müssten recht viele MEIs und MHIs berechnet und in der Datenbank abgelegt werden. Richtig kritisch ist jedoch, dass die beobachtete Person woanders im Bild stehen kann, wenn sie z.B. weiter weg ist. Dabei entstehen für die gleiche Bewegung verschiedene Bilder mit unterschiedlicher Skalierung und Position. Letzteres Problem werden wir im Abschnitt 4.4.1 genauer behandeln.

4.3.1. Effizientes Erstellen von MEI und MHI

Praktisch gesehen ist es sehr viel effizienter die aktuellen MEIs und MHIs jeweils mit Hilfe der Vorherigen zu berechnen. So werden zu jedem Zeitpunkt t nicht etwa das MHI immer wieder neu berechnet, sondern von MHI zum Zeitpunkt $t-1$ rekursiv abgeleitet. Ebenso lässt sich das MEI mit Hilfe von MHI berechnen. Dies bringt den Vorteil, dass man ein sehr schnelles und platzsparendes Verfahren erhält.

4.3.2. Zeitliche Anpassung der Bewegung an die Datenbank

Nehmen wir an, dass beim Speichern der Musterbewegungen in die Datenbank wurden alle Bewegungen mit einer festen Geschwindigkeit durchgeführt. Um nun für weitere Bewegungen, welche schneller oder langsamer durchgeführt wurden, brauchbare MEIs und MHIs zu erhalten, gehen wir davon aus, dass eine Bewegung zwischen τ_{min} und τ_{max} Zeiteinheiten andauert. Nun bertachtet man rückwirkend eine schwankende Zeitspanne und verwendet das meist passende Zeitintervall.

Bei dem Algorithmus wird zu jeder Zeiteinheit ein neues MHI $H_\tau(x, y, t)$ berechnet, bei welchem $\tau = \tau_{max}$ gesetzt wird. Zudem wird $\Delta\tau = \frac{(\tau_{max}-\tau_{min})}{n-1}$ gesetzt, wobei n die Anzahl zu berücksichtigender Zeitfenster darstellt.

Wenn wir $\tau_{max} = 10s$ und $\tau_{min} = 7s$ setzen und $n = 7$ dann werden alle Fenster im $\Delta\tau = 0,5s$ Intervall betrachtet.

Durch einfaches Begrenzen der MHI-Werte kleiner als $(\tau - \Delta\tau)$ können wir $H_{\tau-\Delta\tau}$ aus H_τ erzeugen.

$$H_{\tau-\Delta\tau}(x, y, t) = \begin{cases} H_\tau(x, y, t) - \Delta\tau), & \text{falls } H_\tau(x, y, t) > \Delta\tau, \\ 0, & \text{falls nicht} \end{cases}$$

Um nun die Momente zu berechnen wird H_τ mit $\frac{1}{\tau}$ skaliert. Dadurch haben alle MHIs einen Helligkeitswert von 0 bis 1, was uns schließlich eine Invarianz in Bezug auf die Geschwindigkeit der Bewegung liefert.

4.4. Auswertung von Temporal Templates

Sobald wir MEI und MHI einmal erhalten haben, ist es notwendig Merkmale aus diesen zu extrahieren, die uns erlauben verschiedene MEIs und MHIs zu vergleichen. Die grundlegen-

de Idee dabei ist es Daten zu entnehmen, deren Werte unabhängig von Neigungswinkel, Größe oder Position der Bewegung sind.

4.4.1. Invariante Merkmale

Hierbei handelt es sich um numerische Informationen, die aus den Graustufenbildern gewonnen werden. Diese Informationen sollen möglichst invariant gegenüber Skalierung, Position oder Größe sein und jede Bewegung eindeutig identifizieren. So erhält man nach Berechnung der Momente bei den drei MHIs in Abbildung 4.4 jeweils gleiche Werte.

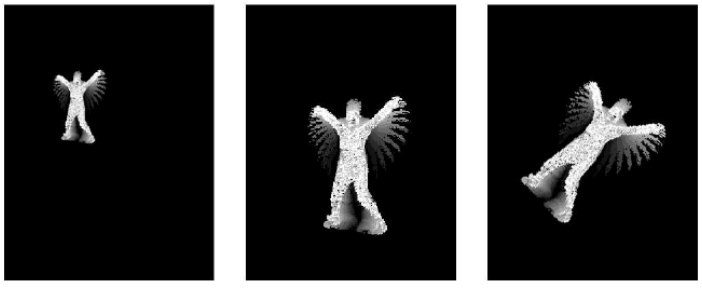
Solche Merkmale könnten z.B. das Verhältnis von Fläche zu Umfang sein.

4.4.2. Die 7 Hu-Momente

Hier kann man auf die Arbeit von L.Hu (1962) zurückgreifen. In seiner Arbeit verwendete er s.g. zentrale Momente. Objektform und Intensitätsverläufe lassen sich durch zentrale Momente eindeutig repräsentieren.

Zuerst werden aus der Bildfunktion $B(x, y)$, welche anzeigt welchen Farbwert das MEI- oder MHI-Bild an der Stelle (x, y) besitzt, die Momente $m_{p,q} p + q$-ten grades gebildet. Für die Berechnung verwendet man folgendes Riemann-Integral:

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q \rho(x, y) dx dy$$



(a) (b) (c)

Abbildung 4.4.: Die gleichen MHIs in unterschiedlichen Positionen. Die Hu Momente für alle drei Bilder sind geleich. Aus J.Davis und Bobick (1997)

Mit Hilfe des 0-ten und der beiden 1-ten Momente lässt sich der Schwerpunkt berechnen.

$$\overline{x} = m_{10}/m_{00}$$

$$\overline{y} = m_{01}/m_{00}$$

Dadurch lassen sich dann die zentralen Momente berechnen.

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-\overline{x})^p (y-\overline{y})^q \rho(x,y) d(x-\overline{x}) d(y-\overline{y})$$

In seiner Arbeit hat Hu die Zentralen Momente normiert.

$$\eta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^\gamma}$$

wobei $\gamma = (p+q)/2 + 1$ und $p + q \geq 2$. Dadurch erreichte er die Invarianz gegenüber der Skalierung. Die normierten zentralen Momente nutzte Hu um die ersten sieben orientierungsinvarianten Hu-Momente zu definieren.

$$\upsilon_1 = \eta_{20} + \eta_{02}$$

$$\upsilon_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\upsilon_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2$$

$$\upsilon_4 = (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2$$

$$\upsilon_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})^2[(\eta_{03} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2]$$
$$+(3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

$$\upsilon_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$$

$$\upsilon_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] -$$
$$(\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$$

Diese sieben Momente werden wir benutzen um Merkmale der Bewegungen zu identifizieren.

4.5. Klassifizierung und Vergleich

In der Anlernphase werden Kenndaten der MEIs und MHIs aus Musterbewegungen in eine Datenbank gespeichert, damit wir sie später bei der Gestenerkennung benutzen können.

Dabei werden die jeweiligen Werte den passenden Bewegungen zugeordnet. Für die Erkennung müssen wir die erhaltenen invarianten Momente mit den Momenten aus der Datenbank vergleichen. Davor wollen wir festlegen wie die Daten abgespeichert und wie sie verglichen werden.

4.5.1. Speicherung der Daten

Die sieben Hu-Momente werden in einem Vektor in der Datenbank abgespeichert. wenn man zu jeder Bewegung mehrere Trainingsdaten hat, kann man noch zusätzlich den Mittelwert berechnen und ebenfalls speichern.

4.5.2. Klassifizierung einer Testsequenz

Wenn jemand eine Bewegung durchführt, die durch den Computer erkannt werden soll, wird die erhaltene Bewegung erfasst und die Information aus der Testsequenz klassifiziert. Dafür wird der Momente-Vektor berechnet und mit denen in der Datenbank verglichen. Die Bewegung wird dann einer Muster-Bewegung aus der Datenbank zugeordnet, die am besten passt.

Dies wäre einfach zu realisieren, indem man die euklidische Norm zwischen dem Testvektor und dem Mittelwerts-Vektor der einzelnen Bewegungen berechnet. Anhand der kleinsten Norm könnten wir die Testbewegung zuordnen. Zu unserem Bedauern reicht das noch nicht aus, da die Trainingsdaten ungleich gestreut sein können. Unterschiedliche Testwerte zu einer Bewegung könnten z.B. elliptisch im Raum verteilt sein, oder einfach unterschiedlich stark um den Mittelwert gestreut sein. Also benötigen wir ein Vergleichsverfahren, das diese Verteilung berücksichtigt.

Mit den Testdaten X zu einer Bewegung lassen sich der Mittelwert $E[X]$ und dessen Varianz $var[X]$ berechnen. Mit Hilfe dieser beiden Werte können wir die standardisierte Distanz d berechnen

$$d(t, x_j)^2 = \left[\frac{t_1 - x_{1,j}}{s_{1,j}}\right]^2 + \left[\frac{t_2 - x_{2,j}}{s_{2,j}}\right]^2 + \ldots + \left[\frac{t_7 - x_{7,j}}{s_{7,j}}\right]^2$$

wobei t_i das $i-te$ Moment des Testvektors sei, $x_{i,j}$ das $i-te$ Moment der Bewegung j sei und s_{ij} die Standardabweichung des i-ten Moments der Bewegung j sei.

Diese Berechnung liefert uns eine Distanz, bei welcher die Streuung der Trainingsvektoren keinen Einfluss auf die Klassifizierung hat.

4.5.3. Mahalanobis Distanz

In dem Vektor können verschiedene Momente voneinander abhängig sein. D.h. einige Merkmale könnten stärker ausgeprägt sein, wenn wir sie gleich gewichtet würden. Wir berechnen also zu allen Momenten paarweise die Kovarianz

$$c(i,j) = \frac{[x_{1,i} - m_i] \cdot [x_{1,j} - m_j] + ... + [x_{n,i} - m_i] \cdot [x_{n,j} - m_j]}{n-1}$$

dabei seien $\{x_{1,i}, ..., x_{n,i}\}$ eine Serie von n Trainingswerten zum Moment *i* und $\{x_{1,j}, ..., x_{n,j}\}$ eine Serie von Trainingswerten zum Moment *j*. m_i und m_j seien jeweils die Mittelwerte der Momente.

Diese Kovarianzen kommen dann in eine Kovarianz-Matrix.

$$C = \begin{pmatrix} c(1,1) & \dots & c(1,n) \\ \vdots & \ddots & \vdots \\ c(n,1) & \dots & c(n,n) \end{pmatrix}$$

Mit dessen Hilfe wir die einzelnen Momente gewichten. Durch d^2

$$d^2 = (t - m_x)^T \cdot C^{-1} \cdot (t - m_x)$$

erhalten wir eine Distanz, die unabhängig von der Streuung der Trainingswerte und von der Abhängigkeit der einzelnen Momente voneinander ist. Nun verfügen wir über die geeigneten Algorithmen, um unsere Ziele zu erreichen und können uns auf die Auswahl der geeigneten Optik konzentrieren.

4.6. Die hemisphärische Kamera

Die wichtigsten Bestandteile einer hemisphärischen Kamera sind ein Rundbild-Fisheye-Objektiv, ein hochauflösender Bildsensor und eine in die Kamera integrierte Software zur Bildkorrektur. Die hemisphärische Kamera erfasst über ein sehr weitwinkliges Fisheye-Objektiv eine Halbkugel im Raum (Hemisphäre) und projiziert diese auf einen hochauflösenden CMOS-Bildsensor.

Von der Decke aus deckt so ein hemisphärischer (halbkugelförmiger) Bildbereich den kompletten Raum ab. In der Halbkugel ist das Bild besonders zu den Rändern stark gekrümmt.

Um die Szene in der gewohnten Perspektive betrachten zu können, werden entsprechende Bildausschnitte für den Nutzer durch die integrierte Bildkorrektur-Software entzerrt. Durch Vergrößern oder Bewegen des Bildausschnitts innerhalb der Halbkugel entsteht der Eindruck einer schwenkenden Kamera, ohne dass sich etwas bewegt: der virtuelle Pan-Tilt-Zoom.

Abbildung 4.5.: Abdeckung eines Raums mit einer hemisphärischen Kamera aus Mobotix (2009)

Hemisphärische Kameras wirken durch nur ein Objektiv sehr unscheinbar, da dieses normalerweise auf den ganzen Raum und nicht auf ein spezielles Objekt ausgerichtet ist. Ohne mechanisch bewegliche Teile unterliegt die hemisphärische Kamera keiner Abnutzung und produziert auch keine Geräusche beim Schwenken und Fokussieren auf einen anderen Bildbereich.

Abbildung 4.6.: Deckenmontage einer hemisphärischen Kamera aus Mobotix (2009)

4.7. Bibliothek OpenCV

OpenCV ist eine quelloffene Programmbibliothek unter BSD-Lizenz. Sie ist für die Programmiersprachen C und C++ geschrieben und enthält Algorithmen für die Bildverarbeitung und maschinelles Sehen. Das CV im Namen steht für Computer Vision. Die Entwicklung der Bibliothek wurde von Intel initiiert und wird heute hauptsächlich von Willow Garage gepflegt. Im September 2006 wurde die Version 1.0 herausgegeben, Ende September 2009 folgte nach längerer Pause die Version 2.0.0, welche die Bezeichnung „Gold“ trägt. Die Stärke von OpenCV liegt in ihrer Geschwindigkeit und in der großen Menge der Algorithmen aus neuesten Forschungsergebnissen.

4.7.1. Systemvoraussetzungen

OpenCV wurde für Personal-Computer mit Intel-Architektur entwickelt. Ihre Funktionen verwenden dieselben Strukturen wie die Intel Performance Primitives Bibliothek (IPP), bei der die meisten Funktionen auf Assembler-Ebene optimiert sind.

Zurzeit werden alle gängigen Betriebssysteme unterstützt. Neben dem Betriebssystem wird auf jeden Fall ein C++ Compiler benötigt. Um sämtliche Funktionen der Bibliothek benutzen zu können, müssen die ffmpeg- die IPP-Bibliotheken installiert sein. OpenCV braucht ca. 110 MByte Festplattenplatz. Für eine Installation der IPP werden weitere 50 MByte benötigt.

4.7.2. Dokumentation

Der Bibliothek ist ein umfassendes Handbuch beigefügt, welches aus zwei Teilen besteht. Der erste Teil beinhaltet den nach Themengebieten gegliederten Programmierleitfaden, der eine Übersicht über die Funktionalität von OpenCV bietet. Im zweiten Teil steht die Benutzerreferenz, welche die einzelnen Funktionen mit kurzer Beschreibung auflistet. Der SDK wir mit zahlreichen Beispielen ausgeliefert.

4.7.3. Support

Bei der Lösung von auftretenden Problemen kann man in der Newsgroup Hilfe suchen. Da diese jedoch von den Entwicklern von OpenCV betreut wird, können Probleme sehr schnell gelöst werden.

4.7.4. Struktur und Funktionsumfang

OpenCV ist weitgehend in fünf wichtigsten Komponenten gegliedert, von denen vier in der Abblidung 4.7 zu sehen sind.

- Die CV-Komponente enthält die grundlegenden Algorithmen für die Bildverarbeitung.
- ML ist Bibliothek für maschinelles Lernen, welche viele statistische Klassifikatoren und Clustering-Werkzeuge beinhaltet.
- HighGUI enthält Ein- und Ausgabe-Routinen und Funktionen zum Speichern und Laden Video und Bilder.
- Und CXCore enthält die grundlegenden Daten-Strukturen.
- Die fünfte Komponente CvAux beinhaltet experimentelle Algorithmen, die noch in der Ausarbeitung sind.

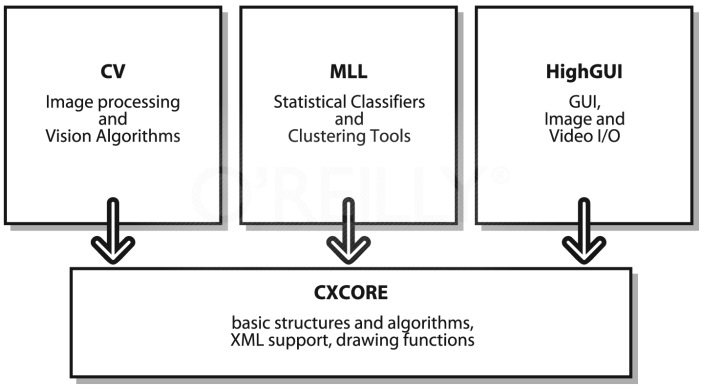


Abbildung 4.7.: Der Aufbau von OpenCV aus Bradski und Kaehler (2008)

Die Bibliothek umfasst unter anderem Algorithmen für Bewegungsanalyse, Objektverfolgung, Objekterkennung, Gesichtsdetektion, 3D-Funktionalität, Haar-Klassifikatoren, verschiedene sehr schnelle Filter und Funktionen für die Kamerakalibrierung.

Für die Entwicklung des Prototypen interessieren uns vor allem Funktionen für die Bewegungssegmentierung, Bildung von MEI und MHI sowie deren Vergleich mittels Distanz der Hu-Momente.

4.8. Aufbau des Prototypen

Der Prototyp besteht aus 2 Programen: Motion Recording `motrec` übernimmt die Aufnahme und die Speicherung der Musterbewegungen. Dabei muss der Anwender lediglich eine Taste

drücken, wenn er ein Abbild der Bewegung speichern will. Motion Tracking *mot* sorgt für die robuste Erkennung der aufgenommenen Bewegung. Der Ablauf der Gestenerkennung bestimmt die Struktur der Software. Die Abblildung 4.8 zeigt den Aufbau des Prototypen.

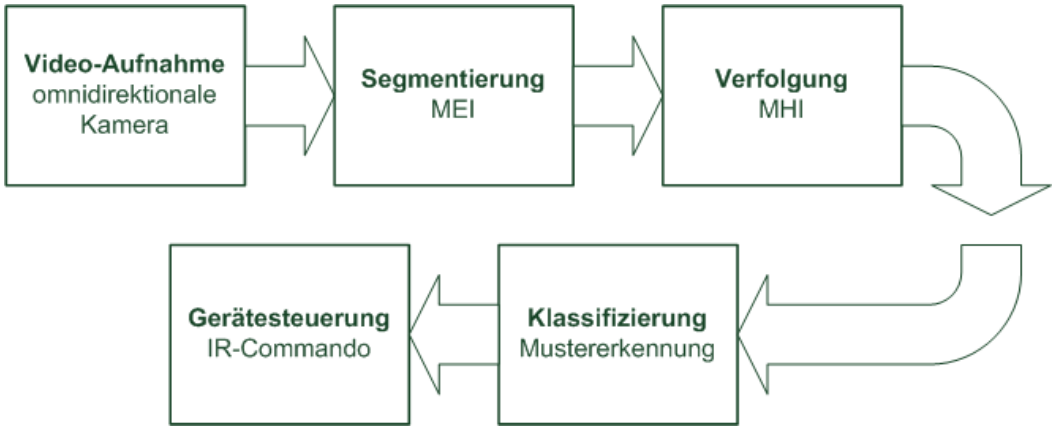


Abbildung 4.8.: Der Aufbau des Prototypen

- Als Quelle der Bildsequenzen dient eine omnidirektionale Kamera.
- Die Bewegungssegmentierung wird mit dem MHI-Verfahren aus dem Kapitel 4.1 vollzogen.
- Die Bewegungsverfolgung und Start- Stop-Erkennung übernimmt MHI aus dem Kapitel 4.2.
- Die Klassifizierung erfolgt über die Mustererkennung mittels Mahalanobis Distanz der Hu-Momente aus dem Kapitel 4.4.

Auf die Realisierung der IR-Schnittstelle wurde verzichtet, da sie nicht im Blickpunkt dieser Arbeit steht. Im Anhang A finden Sie den kommentierten Quellcode der beiden Programme.

4.9. Evaluation

An dieser Stelle findet eine Evaluierung des Gesamtergebnisses statt, das System wird mit den in Abschnitt 3.3 festgelegten Zielen vergleichen und die Praxistauglichkeit wird überprüft. Hierfür wurde ein Prototyp entwickelt und das Verfahren in verschiedenen Extremfällen getestet.

4.9.1. Echtzeitfähigkeit

Mit dem hier dargestellten verfahren, ist es möglich sehr effizient und mit wenig Rechenaufwand eine recht genaue Bewegungserkennung in Echtzeit zu erhalten. Die angewandten Verfahren sind zum Anwenden auf der Hardware von Mobotix Q24 als Embedded System zu empfehlen.

4.9.2. Robustheit

Dennoch gibt es bei der Durchführung Probleme, wie das nicht Erkennen von Bewegung bei Personen mit einfarbiger Kleidung, oder das Erkennen von Bewegung die nicht vorhanden ist, wie z.B. Veränderungen in der Umgebungshelligkeit, oder weitere sich bewegende Objekte in der Nähe des Anwenders. Diese Probleme können jedoch durch eine robustere und effektivere Klassifizierung gelöst werden. Bei den hier vorgestellten Tests wurden ein Rechteck um die zu Untersuchende Bewegung gelegt, und Rechtecke mit einer Fläche kleiner als Schwellwert ignoriert.

4.9.3. Verdeckungen

Die Erkennung ist nicht möglich, wenn eine Person den Probanden verdeckt. Aber durch die Deckenmontage der Kamera tritt dieser Fall äußerst selten auf.

4.9.4. Beleuchtungsveränderungen

Normale Veränderungen der Beleuchtung führen nicht zu Problemen und die Gesten können weiter erkannt werden. Problematisch wird es allerdings, wenn zu wenig Licht vorhanden ist. Die Verwendung einer Infrarotkamera kann die Lösung für dunkle Räume sein.

5. Schluss

In diesem Kapitel sollen die Ergebnisse dieser Arbeit zunächst zusammengefasst und bewertet (Abschnitt 5.1) werden. Der Ausblick (Abschnitt 5.2) zeigt Aufgaben und Themen für mögliche weiterführende Untersuchungen und Arbeiten auf, deren Bearbeitung im Rahmen dieser Bachelorarbeit nicht möglich war.

5.1. Zusammenfassung und Fazit

Im Rahmen dieser Arbeit wurde die Möglichkeit der Implementierung der Gestenerkennung für die Steuerung der Unterhaltungselektronik untersucht. Hierzu wurden zunächst geeignete Szenarien untersucht und Anforderungen an ein solches System formuliert (Kapitel 3). Abschließend wurden verschiedene Verfahren zur Gestenerkennung vorgestellt.

Auf Basis dieser Anforderungen wurde in Kapitel 4 eine prototypische Implementierung eines solchen Systems entwickelt. Unter Einsatz geeigneter Algorithmen wurde eine Anwendungsarchitektur vorgestellt, um die vorgenannten Komponenten zu integrieren. Die Möglichkeiten und Beschränkungen des entwickelten Prototypen werden am Ende des Kapitels 4 Design und Realisierung zusammengefasst.

Insgesamt kann festgestellt werden, dass eine zentrale robuste echtzeitfähige markerlose Gestenerkennung mit einer omnidirektionalen Kamera mit relativ wenig Rechenaufwand auf beschränkter Hardware realisiert werden kann. Verwendete Algorithmen sind zum Implementieren auf der Hardware von Mobotix Q24 als Embedded System zu empfehlen.

Dennoch sind bei der Implementierung einige Probleme und Schwierigkeiten zu Tage gefördert worden. Bei einigen dieser Probleme, wie Fehlerkennung, handelt es sich um Herausforderungen, die durch eine effektivere Klassifizierung von Bewegungen, sowie durch den Einsatz von Stereo behoben werden können. Andere Probleme, wie Beleuchtungsveränderungen, lassen sich möglicherweise durch Anwenden von Infrarot-Kameras bewältigen. Ein weiterer Vorteil von Infrarot wäre die Erhöhung der Anzahl der Freiheitsgrade des Trackers durch Bild-Tiefen-Erkennung.

5.2. Ausblick

Die Zukunft beginnt schon heute. In Museen und auf Messen findet man computergestützte Kunstinstallation, die durch Bewegung, Blick-Kontakt oder Gesten gesteuert oder beeinflusst werden und die Zuschauer zum Mitmachen animieren.

Wir stehen vor der Revolution der Eingabegeräte. Als Erstes ist der Bereich Gaming dran. Microsoft Kinect ist der erste Schritt in diese Richtung. Baldige Nachfolger von diesem System werden die Welt der Konsolen- und Computer-Spiele für immer verändern. Ein Spiel der Zukunft wird ohne jegliche Controller mit vollem Körpereinsatz durch Bewegung des Spielers gesteuert.

Danach findet Gestensteuerung ihre Anwendung in der Küche. Mit dem Projekt iPoint 3D stellt Fraunhofer Institut für Nachrichtentechnik Heinrich-Hertz-Institut eine interessante Lösung vor. Wenn ein Koch gerade Teig knetet, kann er trotzdem seinen Herd ein und ausschalten, den Radiosender wechseln oder im elektronischen Kochbuch berührungslos blättern. Berührungslose Gerätesteuerung spart Zeit und hilft Hygiene-Vorgaben einzuhalten. iPoint 3D lässt sich auf in vielen Anderen Szenarien anwenden, wie z.B. Präsentationen oder 3D-Simulationen. S. Chojecki (2010).

Die israelische Firma PrimeSense (2010) konzentriert sich auf die Entwicklung der Gestensteuerung für die interaktiven Fernsehgeräte der Zukunft. Durch den Einsatz von berührungsloser Steuerung entfällt die Anwendung von Fernbedienungen. Das Zusammenspiel von Fernsehen und Internet auf einem Gerät wird vereinfacht und völlig neue Interaktionsmöglichkeiten mit dem Zuschauer werden geschaffen.

Diese Entwicklungen werden Tag für Tag die Welt der klassischen Eingabe-Geräte verändern. Wir werden Computing für Leute zugänglich machen, die früher aufgrund ihrer Behinderung oder Alters Distanz zu den Computern behielten. Wir werden Computing für Berufsgruppen eröffnen, die bisher auf Rechner verzichten mussten. Und wir werden noch mehr beinah unwahrnehmbare Rechner um jeden Menschen platzieren, die sein Leben sicherer und bequemer machen.

Literaturverzeichnis

[Adolf 2003] ADOLF, Florian: *How-to build a cascade of boosted classifiers based on Haar-like features*. 2003. – URL http://lab.cntl.kyutech.ac.jp/~kobalab/nishida/opencv/OpenCV_ObjectDetection_HowTo.pdf. – Zugriffsdatum: 29.04.2010

[Boetzer 2008] BOETZER, Joachim: *Bewegungs- und gestenbasierte Applikationssteuerung auf Basis eines Motion Trackers*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2008

[Bradski und Kaehler 2008] BRADSKI, Gary ; KAEHLER, Adrian ; LOUKIDES, Mike (Hrsg.): *Learning OpenCV*. O'Reilly Media, Inc., 2008

[Chojecki 2010] CHOJECKI, Paul: *iPoint 3D*. 2010. – URL http://www.hhi.fraunhofer.de/de/abteilungen-am-hhi/interaktive-medien-human-factors/uebersicht/hand-interaction/exponate/ipoint3d0/. – Zugriffsdatum: 19.09.2010

[Davis 1999] DAVIS, Jim: *KidsRoom*. 1999. – URL http://vismod.media.mit.edu/vismod/demos/kidsroom/sponsor.gif. – Zugriffsdatum: 29.04.2010

[Dean Oliver 1997] DEAN OLIVER, Scott A.: *Dilbert uses Kalman*. 1997. – URL http://www.rawbw.com/~deano/images/dilbert.gif. – Zugriffsdatum: 29.04.2010

[Heitsch 2008] HEITSCH, Johann: *Ein Framework zur Erkennung von dreidimensionalen Gesten*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2008

[Hu 1962] HU, L.: Visual Pattern Recognition by Moment Invariants. In: *IRE Trans. on Pattern Analysis and Maschine Intelligence* 8 no. 2 (1962), S. 179–187

[Hudelmaier 2001] HUDELMAIER, J.: *Das Menschmodell RAMSIS*. 2001. – URL http://www.lfe.mw.tum.de/forschung/humanmodeling/Ramsis_Flyer.pdf. – Zugriffsdatum: 29.04.2010

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

[J.Davis und Bobick 1997] J.DAVIS ; BOBICK, A.: The Representation and Recognition of Human Movement Using Temporal Templates. In: *Computer Vision and Pattern Recognition* (1997), S. 928–934

[Jon Rimmer und Weir 2005] JON RIMMER, Ian Wakeman Bill Keller Julie W. ; WEIR, David: *Natural Habitat*. 2005. – URL `http://www.informatics.sussex.ac.uk/research/projects/nathab/images/env.jpeg`. – Zugriffsdatum: 29.04.2010

[Mangold 2010] MANGOLD, Christian: *Fernbedienungen*. 2010. – URL `http://www.beamer.as/images/IMG_5788.jpg`. – Zugriffsdatum: 29.04.2010

[Mobotix 2009] MOBOTIX: *Q24M Kamerahandbuch*. 2009. – URL `https://www.mobotix.com/ger_DE/file/1588407/mx_MLq24m_de_090617.pdf`. – Zugriffsdatum: 29.04.2010

[Mobotix 2010] MOBOTIX: *Mobotix Q24*. 2010. – URL `http://www.mobotix.com/var/storage/images/media/images/products/startseiten/q24_details/1581379-5-ger/Q24_Details_formatVGA.jpg`. – Zugriffsdatum: 29.04.2010

[Ogden 1999] OGDEN, Sam: *Interactive Virtual Aerobics Trainer*. 1999. – URL `http://www.cse.ohio-state.edu/~jwdavis/CVL/Research/VirtualAerobics/pat_photo3_sm.gif`. – Zugriffsdatum: 29.04.2010

[Perales 2003] PERALES, F.J.: *A Colour Tracking Procedure for Low-Cost Face Desktop Applications*. 2003. – URL `http://dmi.uib.es/research/GV/HUMODAN/images/uib_img/camshift1.bmp`. – Zugriffsdatum: 29.04.2010

[PrimeSense 2010] PRIMESENSE: *Nite Middleware*. 2010. – URL `http://www.primesense.com/files/FMF_3.PDF`. – Zugriffsdatum: 19.09.2010

[Prospektiv 2008] PROSPEKTIV: *Freiheitsgrade*. 2008. – URL `http://smerobot-tools.prospektiv.de/glossar/deu/Freiheitsgrade.gif`. – Zugriffsdatum: 29.04.2010

[Rödiger 2010] RÖDIGER, Marcus: *Interaktive Steuerung von Computersystemen mittels Erkennung von Körpergesten*, Hochschule für Angewandte Wissenschaften Hamburg, Diplomarbeit, 2010

[Weiser 1999] WEISER, Mark: The computer for the 21st century. In: *SIGMOBILE Mob. Comput. Commun. Rev.* 3 (1999), Nr. 3, S. 3–11. – ISSN 1559-1662

[Wikipedia 2004a] WIKIPEDIA: *American Sign Language alphabet*. 2004. – URL `http://upload.wikimedia.org/wikipedia/commons/d/d1/Asl_alphabet_gallaudet.png`. – Zugriffsdatum: 29.04.2010

[Wikipedia 2004b] WIKIPEDIA: *Hauptkomponentenanalyse*. 2004. – URL `http://upload.wikimedia.org/wikipedia/commons/e/e6/PCS_fish_ortho.png`. – Zugriffsdatum: 29.04.2010

A. Quellcode

A.1. Motion recognition

```
1  // mot.cpp ist ein Gesteerkennungsprogram fuer eine
       omnidirektionale Kamera.
2  // Aufgebaut auf motempl.c aus dem OpenCV samples/c Verzeichnis.
3  #include "cv.h"
4  #include "highgui.h"
5  #include <time.h>
6  #include <math.h>
7  #include <ctype.h>
8  #include <stdio.h>
9
10 // Vervolgungsparameter in Sekunden
11 const double MHI_DURATION = 1;
12 const double MAX_TIME_DELTA = 0.5;
13 const double MIN_TIME_DELTA = 0.05;
14 // Anzahl der zyklischen Bild-Puffer fuer die Bewegungsdetektion
15 // abhaengig von der Bildrate
16 const int N = 4;
17
18 // Ringspeicher fur Bilder
19 IplImage **buf = 0;
20 int last = 0;
21
22 // Temporaere Bilder
23 IplImage *mhi = 0; // MHI
24 IplImage *orient = 0; // Ausrichtung
25 IplImage *mask = 0; // Zulaessige Ausrichtungs-Maske
26 IplImage *segmask = 0; // Abblid der Bewegungssegmentierung
27 CvMemStorage* storage = 0; // Zwischenspeicher
28 IplImage *src_img = 0; // Gespeicherte Bewegung
29
```

```
30 // Parameter:
31 //   img – urspruengliches Bild
32 //   dst – resultierendes Bild der Bewegung
33 //   args – Optionale Parameter
34 void  update_mhi( IplImage* img, IplImage* dst, int diff_threshold
       )
35 {
36     double timestamp = (double)clock()/CLOCKS_PER_SEC; // gib
          aktuelle Zeit in Sekunden
37     CvSize size = cvSize(img->width,img->height); // gib aktuelle
          Bildgroesse
38     int i, idx1 = last, idx2;
39     IplImage* silh;
40     CvSeq* seq;
41     CvRect comp_rect;
42     double count;
43     double angle;
44     CvPoint center;
45     double magnitude;
46     CvScalar color;
47
48     // Anlegen der Bilder am Anfang oder
49     // Neuanlegen wenn die Bildgroesse geaendert wird
50     if( !mhi || mhi->width != size.width || mhi->height != size.
          height ) {
51         if( buf == 0 ) {
52             buf = (IplImage**)malloc(N*sizeof(buf[0]));
53             memset( buf, 0, N*sizeof(buf[0]));
54         }
55
56         for( i = 0; i < N; i++ ) {
57             cvReleaseImage( &buf[i] );
58             buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
59             cvZero( buf[i] );
60         }
61         cvReleaseImage( &mhi );
62         cvReleaseImage( &orient );
63         cvReleaseImage( &segmask );
64         cvReleaseImage( &mask );
65
```

```
66          mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );
67          cvZero( mhi ); // Leeren von MHI am Anfang
68          orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );
69          segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
70          mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
71      }
72
73      cvCvtColor( img, buf[last], CV_BGR2GRAY ); // Bild zum
            Graustufenbild umwandeln
74
75      idx2 = (last + 1) % N; // Index vom (letzten - (N-1)). Bild
76      last = idx2;
77
78      silh = buf[idx2];
79      cvAbsDiff( buf[idx1], buf[idx2], silh ); // gib Differenz
            zwischen 2 aufeinander folgenden Bilder
80
81      cvThreshold( silh, silh, diff_threshold, 1, CV_THRESH_BINARY );
            // Umwandeln zum binaeren Bild
82      cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); //
            Aktualisieren von MHI
83
84      // Umwandeln von MHI zu einem Blaubild
85      cvCvtScale( mhi, mask, 255./MHI_DURATION,
86                  (MHI_DURATION - timestamp)*255./MHI_DURATION );
87      cvZero( dst );
88      cvCvtPlaneToPix( mask, 0, 0, 0, dst );
89
90      // Berechnen der Ausrichtung des Bewegungsverlaufs und der
            zulaessigen Ausrichtungsmaske
91      cvCalcMotionGradient( mhi, mask, orient, MAX_TIME_DELTA,
            MIN_TIME_DELTA, 3 );
92
93      if( !storage )
94          storage = cvCreateMemStorage(0);
95      else
96          cvClearMemStorage(storage);
97
98      // cvSegmentMotion: git die Sequence von Bewegungskomponenten
```

```
99      // segmask stellt das Abblid der Bewegungkomponenten dar und
            wird nicht weiter verwendet
100     seq = cvSegmentMotion( mhi, segmask, storage, timestamp,
            MAX_TIME_DELTA );
101
102     // Iteriere durch die Bewegunskomponenten,
103     // Der Iterationsschritt (i == -1) repraesentiert das ganze
            Bild (Gesamte Bewegung im Bild)
104     for( i = -1; i < seq->total; i++ ) {
105
106         if( i < 0 ) { // fuer das ganze Bild
107             comp_rect = cvRect( 0, 0, size.width, size.height );
108             color = CV_RGB(255,255,255);
109             magnitude = 100;
110         }
111         else { // fuer i. Bewegungskomponente
112             comp_rect = ((CvConnectedComp*)cvGetSeqElem( seq, i ))
                    ->rect;
113             if( comp_rect.width + comp_rect.height < 100 ) //
                    Ignoriere sehr kleine Komponenten
114                 continue;
115             color = CV_RGB(0,255,0);
116             magnitude = 30;
117         }
118
119         // Waehle Interessenbereich einer Komponente
120         cvSetImageROI( silh, comp_rect );
121         cvSetImageROI( mhi, comp_rect );
122         cvSetImageROI( orient, comp_rect );
123         cvSetImageROI( mask, comp_rect );
124                 // Auswertung der Distanz zwischen einer Komponente
                        und dem gespeicherten Bild
125         double result;
126         char score[30];
127         CvFont font;
128         // Vergleich der Bewegung mit der Musterbewegung
129         result = 1 / (cvMatchShapes(src_img, mask,
                CV_CONTOURS_MATCH_I1, 0) * cvMatchShapes(src_img, mask,
                CV_CONTOURS_MATCH_I1, 0));
130         sprintf (score,"Recognition: %6.6f", result);
```

```
131         double hScale=0.5;
132         double vScale=0.5;
133         int    lineWidth=1;
134         cvInitFont(&font,CV_FONT_HERSHEY_SIMPLEX|CV_FONT_ITALIC,
               hScale,vScale,0,lineWidth);
135         cvPutText(dst, score , cvPoint(100, 60), &font, CV_RGB
               (255,0,0));
136               // Uebereinstimmung der Bewegung mit der
                     gespeicherten Musterbewegung > 90%
137         if(result > .9) {
138             color = CV_RGB(255,0,0);
139         }
140         // Berechne die Ausrichtung
141         angle = cvCalcGlobalOrientation( orient, mask, mhi,
               timestamp, MHI_DURATION);
142         angle = 360.0 - angle;  // Anpassen fuer Bilder mit dem
               Ausgangspunkt oben links
143
144         count = cvNorm( silh, 0, CV_L1, 0 ); // Berechne die Anzahl
                der Punkte in der Silhouette im Interessenbereich
145         cvResetImageROI( mhi );
146         cvResetImageROI( orient );
147         cvResetImageROI( mask );
148         cvResetImageROI( silh );
149
150         // Ueberpruefung fuer den Fall einer kleinen Bewegung
151         if( count < comp_rect.width*comp_rect.height * 0.05 )
152             continue;
153
154         // Zeichne ein Kreis mit einer Linie, die die
               Bewegungsrichtung darstellt
155         center = cvPoint( (comp_rect.x + comp_rect.width/2),
156                           (comp_rect.y + comp_rect.height/2) );
157
158         cvCircle( dst, center, cvRound(magnitude*1.2), color, 3,
               CV_AA, 0 );
159         cvLine( dst, center, cvPoint( cvRound( center.x + magnitude
               *cos(angle*CV_PI/180)),
160                                       cvRound( center.y - magnitude
```

```
                                                *sin(angle*CV_PI/180)),
                                                color, 3, CV_AA, 0 );
161     }
162 }
163
164 int main()
165 {
166     IplImage* motion = 0;
167     CvCapture* capture = 0;
168     src_img = cvLoadImage("Mo_10007.bmp", CV_LOAD_IMAGE_GRAYSCALE);
            // Lade die Musterbewegung
169     capture = cvCaptureFromFile ("http://10.8.43.167/control/
            faststream.jpg?stream=full&fps=16&error=picture&e=.mjpg");
170     //capture = cvCaptureFromFile ("http://10.8.43.175/control/
            faststream.jpg?stream=full&fps=16&error=picture&e=.mjpg");
171
172
173     if( capture )
174     {
175         cvNamedWindow( "Motion", 1 );
176
177         for(;;)
178         {
179             IplImage* image;
180             if( !cvGrabFrame( capture ))
181                 break;
182             image = cvRetrieveFrame( capture );
183
184             if( image )
185             {
186                 if( !motion )
187                 {
188                     motion = cvCreateImage( cvSize(image->width,
                            image->height), 8, 3 );
189                     cvZero( motion );
190                     motion->origin = image->origin;
191                 }
192             }
193
194             update_mhi( image, motion, 50 );
```

```
195                 cvShowImage( "Motion", motion );
196
197                 if( cvWaitKey(10) >= 0 )
198                     break;
199             }
200             cvReleaseCapture( &capture );
201             cvDestroyWindow( "Motion" );
202         }
203         else
204         {
205             printf("\nFailed␣to␣connect␣to␣the␣camera\n");
206         }
207
208         return 0;
209 }
```

A.2. Motion recorder

```
1  // mot.cpp ist ein Gesteaufnahmeprogram fuer eine omnidirektionale
       Kamera.
2  // Aufgebaut auf motempl.c aus dem OpenCV samples/c Verzeichnis.
3  // Original-Kommentare aus motempl.c auf Englisch beibehalten
4
5  #include "cv.h"
6  #include "highgui.h"
7  #include <time.h>
8  #include <math.h>
9  #include <ctype.h>
10 #include <stdio.h>
11
12 // various tracking parameters (in seconds)
13 const double MHI_DURATION = 1;
14 const double MAX_TIME_DELTA = 0.5;
15 const double MIN_TIME_DELTA = 0.05;
16 // number of cyclic frame buffer used for motion detection
17 // (should, probably, depend on FPS)
18 const int N = 4;
19
20 // ring image buffer
21 IplImage **buf = 0;
22 int last = 0;
23
24 // temporary images
25 IplImage *mhi = 0; // MHI
26 IplImage *orient = 0; // orientation
27 IplImage *mask = 0; // valid orientation mask
28 IplImage *segmask = 0; // motion segmentation map
29 CvMemStorage* storage = 0; // temporary storage
30 int filenamecount = 0; // Counter for the filenames
31
32
33 // parameters:
34 //  img - input video frame
35 //  dst - resultant motion picture
36 //  args - optional parameters
```

```
37 void  update_mhi( IplImage* img, IplImage* dst, int diff_threshold
       )
38 {
39     double timestamp = (double)clock()/CLOCKS_PER_SEC; // get
           current time in seconds
40     CvSize size = cvSize(img->width,img->height); // get current
           frame size
41     int i, idx1 = last, idx2;
42     IplImage* silh;
43     CvSeq* seq;
44     CvRect comp_rect;
45     double count;
46     double angle;
47     CvPoint center, cornerOne, cornerTwo;
48     double magnitude;
49     CvScalar color;
50
51     char imageFilename[50]; // The filename
52
53
54     // allocate images at the beginning or
55     // reallocate them if the frame size is changed
56     if( !mhi || mhi->width != size.width || mhi->height != size.
           height ) {
57         if( buf == 0 ) {
58             buf = (IplImage**)malloc(N*sizeof(buf[0]));
59             memset( buf, 0, N*sizeof(buf[0]));
60         }
61
62         for( i = 0; i < N; i++ ) {
63             cvReleaseImage( &buf[i] );
64             buf[i] = cvCreateImage( size, IPL_DEPTH_8U, 1 );
65             cvZero( buf[i] );
66         }
67         cvReleaseImage( &mhi );
68         cvReleaseImage( &orient );
69         cvReleaseImage( &segmask );
70         cvReleaseImage( &mask );
71
72         mhi = cvCreateImage( size, IPL_DEPTH_32F, 1 );
```

```
73          cvZero( mhi ); // clear MHI at the beginning
74          orient = cvCreateImage( size, IPL_DEPTH_32F, 1 );
75          segmask = cvCreateImage( size, IPL_DEPTH_32F, 1 );
76          mask = cvCreateImage( size, IPL_DEPTH_8U, 1 );
77      }
78
79      cvCvtColor( img, buf[last], CV_BGR2GRAY ); // convert frame to
            grayscale
80
81      idx2 = (last + 1) % N; // index of (last - (N-1))th frame
82      last = idx2;
83
84      silh = buf[idx2];
85      cvAbsDiff( buf[idx1], buf[idx2], silh ); // get difference
            between frames
86
87      cvThreshold( silh, silh, diff_threshold, 1, CV_THRESH_BINARY );
            // and threshold it
88      cvUpdateMotionHistory( silh, mhi, timestamp, MHI_DURATION ); //
            update MHI
89
90      // convert MHI to blue 8u image
91      cvCvtScale( mhi, mask, 255./MHI_DURATION,
92                  (MHI_DURATION - timestamp)*255./MHI_DURATION );
93      cvZero( dst );
94      cvCvtPlaneToPix( mask, 0, 0, 0, dst );
95
96      // calculate motion gradient orientation and valid orientation
            mask
97      cvCalcMotionGradient( mhi, mask, orient, MAX_TIME_DELTA,
            MIN_TIME_DELTA, 3 );
98
99      if( !storage )
100         storage = cvCreateMemStorage(0);
101     else
102         cvClearMemStorage(storage);
103
104     // segment motion: get sequence of motion components
105     // segmask is marked motion components map. It is not used
            further
```

```
106     seq = cvSegmentMotion( mhi, segmask, storage, timestamp,
            MAX_TIME_DELTA );
107
108     // iterate through the motion components,
109     // One more iteration (i == -1) corresponds to the whole image
            (global motion)
110     for( i = -1; i < seq->total; i++ ) {
111
112         if( i < 0 ) { // case of the whole image
113             comp_rect = cvRect( 0, 0, size.width, size.height );
114             color = CV_RGB(255,255,255);
115             magnitude = 100;
116         }
117         else { // i-th motion component
118             comp_rect = ((CvConnectedComp*)cvGetSeqElem( seq, i ))
                    ->rect;
119             if( comp_rect.width + comp_rect.height < 100 ) //
                    reject very small components
120                 continue;
121             color = CV_RGB(0,255,0);
122             magnitude = 30;
123         }
124
125         // select component ROI
126         cvSetImageROI( silh, comp_rect );
127         cvSetImageROI( mhi, comp_rect );
128         cvSetImageROI( orient, comp_rect );
129         cvSetImageROI( mask, comp_rect );
130
131         // calculate orientation
132         //         angle = cvCalcGlobalOrientation( orient, mask,
                    mhi, timestamp, MHI_DURATION);
133         //         angle = 360.0 - angle;  // adjust for images with
                    top-left origin
134
135         count = cvNorm( silh, 0, CV_L1, 0 ); // calculate number of
                points within silhouette ROI
136
137         cvResetImageROI( mhi );
138         cvResetImageROI( orient );
```

```
139             cvResetImageROI( mask );
140             cvResetImageROI( silh );
141
142             // check for the case of little motion
143             if( count < comp_rect.width*comp_rect.height * 0.05 )
144                 continue;
145
146             // Save a sample
147             if( cvWaitKey(10) >= 0 ) {
148                 cvSetImageROI( dst, comp_rect );
149                 sprintf (imageFilename, "Mo_%d.bmp", filenamecount
                        +10000);
150                 filenamecount++;
151                 cvSaveImage(imageFilename, dst);
152                 cvResetImageROI( dst );
153
154             }
155
156             //          // draw a clock with arrow indicating the
                    direction
157             //          center = cvPoint( (comp_rect.x + comp_rect.width
                    /2),
158             //                            (comp_rect.y + comp_rect.height
                    /2) );
159             //
160             //          //       cvCircle( dst, center, cvRound(magnitude
                    *1.2), color, 3, CV_AA, 0 );
161             //          cvLine( dst, center, cvPoint( cvRound( center.x +
                     magnitude*cos(angle*CV_PI/180)),
162             //                                        cvRound( center.y -
                     magnitude*sin(angle*CV_PI/180))), color, 3, CV_AA, 0 );
163             //          // draw just a rectange
164             cornerOne = cvPoint (comp_rect.x,comp_rect.y);
165             cornerTwo = cvPoint (comp_rect.x+comp_rect.width, comp_rect
                    .y + comp_rect.height);
166             cvRectangle(dst, cornerOne, cornerTwo,color, 3, CV_AA, 0  )
                    ;
167         }
168     }
169
```

```
170 int main()
171 {
172     IplImage* motion = 0;
173     CvCapture* capture = 0;
174
175     capture = cvCaptureFromFile ("http://10.8.43.167/control/
            faststream.jpg?stream=full&fps=16&error=picture&e=.mjpg");
176     // capture = cvCaptureFromFile ("http://10.8.43.175/control/
            faststream.jpg?stream=full&fps=16&error=picture&e=.mjpg");
177
178
179     if( capture )
180     {
181         cvNamedWindow( "Motion", 1 );
182         cvNamedWindow( "Capture", 1 );
183
184         for(;;)
185         {
186             IplImage* image;
187             if( !cvGrabFrame( capture ))
188                 break;
189             image = cvRetrieveFrame( capture );
190
191             if( image )
192             {
193                 if( !motion )
194                 {
195                     motion = cvCreateImage( cvSize(image->width,
                            image->height), 8, 3 );
196                     cvZero( motion );
197                     motion->origin = image->origin;
198                 }
199             }
200
201             update_mhi( image, motion, 50 );
202             cvShowImage( "Motion", motion );
203             cvShowImage( "Capture", image );
204
205
206         }
```

```
207             cvReleaseCapture( &capture );
208             cvDestroyWindow( "Motion" );
209             cvDestroyWindow( "Capture" );
210 
211         }
212         else
213         {
214             printf("\nFailed to connect to the camera\n");
215         }
216 
217         return 0;
218 }
```

Listings

Abbildungsverzeichnis

Versicherung über Selbstständigkeit

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

Hamburg, 23. September 2010	
Ort, Datum	Unterschrift