Universität Karlsruhe
Fakultät für Informatik
Institut für Logik, Komplexität und Deduktionssysteme
Prof. A. Waibel

WS 2002/03

Diplomarbeit

Erkennung von Zeigegesten basierend auf 3D-Tracking von Kopf und Händen

Kai Nickel

März 2003

Betreuer: Prof. Dr. A. Waibel
Dr. R. Stiefelhagen

Hiermit erkläre ich, die vorliegende Arbeit selbstständig erstellt und keine anderen als die angegebenen Quellen verwendet zu haben.

Karlsruhe, den 27. März 2003 Kai Michel .....

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Der Begriff *pervasive computing* steht für ein neues Konzept bezüglich der Art und Weise, wie Menschen sich die Leistung von Computern zunutze machen. Computer sollen dabei nicht mehr in Form umständlich zu handhabender, ortsgebundener Terminals existieren, sondern ihre Dienste in einer der jeweiligen Situation angepassten Form zu jeder Zeit und an jedem Ort anbieten. Im Idealfall unterstützen diese Geräte den Menschen bei all seinen täglichen Aktivitäten in einer Weise, die ihn vergessen lässt, es überhaupt mit Computern zu tun zu haben.

Um unabhängig vom jeweiligen Aufenthaltsort über die nützlichen Eigenschaften des *personal computers* verfügen zu können, ist inzwischen ein breites Angebot an tragbaren Geräten (Notebook, PDA, Mobiltelefon, Tablet-PC, ...) mit Anschluss an drahtlose Datennetze verfügbar. Vielfach sind diese Geräte bereits in der Lage, auf einfache sprachliche Kommandos zu reagieren oder handschriftliche Eingaben entgegen zu nehmen.

Aktuelle Ansätze in der Forschung gehen noch weiter: *Multimodale Benutzerschnittstellen* ermöglichen es dem Menschen, in der Kommunikation mit Maschinen diejenige Ausdrucksform zu wählen, die der jeweiligen Aufgabe am angemessensten erscheint. Dabei kann es sich um gesprochene Sprache oder Handschrift handeln, aber auch um ein zustimmendes Kopfnicken oder einen Fingerzeig auf ein bestimmtes Objekt im Raum. Jede einzelne dieser natürlichen Modalitäten hat ihre spezifischen Stärken und Schwächen, ihre Kombination jedoch ermöglicht hohe Effizienz bei geringer Einarbeitungszeit.

Die Erkennung von Sprache, Gestik oder Mimik stellt ein großes technisches Problem dar, das allenfalls teilweise als gelöst gelten kann. Besondere Schwierigkeiten entstehen beim Einsatz in natürlichen Umgebungen, in denen zahlreiche Störeinflüsse[1] die Erkennung beeinträchtigen.

Die vorliegende Arbeit beschäftigt sich mit dem Bereich der visuellen Gestenerkennung. Ziel ist es, ein System zur Erkennung menschlicher Zeigegesten zu entwickeln. Dabei soll zum einen das Ausführen der Geste registriert, und zum anderen die Richtung, in die gezeigt wurde, im dreidimensionalen Raum bestimmt

[1] Bei visuellen Verfahren z. B. variierende Lichtverhältnisse oder Verdeckung durch andere Objekte, bei der Spracherkennung z. B. Hintergrundgeräusche oder gleichzeitige Gespräche mehrerer Personen.

werden. Das System soll von verschiedenen Personen genutzt werden können, ohne dass eine manuelle Initialisierung oder Anpassung erforderlich ist. Es soll in Echtzeit auf einem Standard-PC betrieben werden können. Auch der Einsatz in einem mobilen Roboter soll möglich sein.

1.1 Inhaltsübersicht

Das Problem der Zeigegestenerkennung zerfällt in drei Teile: das Verfolgen der Person (*Tracking*), die Gestenerkennung und die Bestimmung der Zeigerichtung.

Kapitel 2 beschreibt den Aufbau eines Systems für das dreidimensionale Verfolgen einer Person anhand der Positionen ihres Kopfes und ihrer Hände. Als Merkmale kommen Hautfarbe sowie Disparitätenwerte aus der Stereobildverarbeitung zum Einsatz: Die hautfarbenen Bildpunkte werden räumlich geballt, aus den entstehenden Clustern werden diejenigen ausgewählt, die am wahrscheinlichsten einen Kopf bzw. eine Hand darstellen.

In Kapitel 3 wird ein Modell für Zeigegesten vorgestellt, das aus drei separaten Hidden-Markov-Modellen (HMMs) für die drei Bewegungsphasen einer Zeigegeste (Beginn, Halten, Ende) besteht. Die Merkmalssequenzen werden mithilfe dieser Modelle fortlaufend klassifiziert, um das Vorkommen einer Geste detektieren zu können.

Zur Bestimmung der Zeigerichtung werden in Kapitel 4 zwei verschiedene Ansätze verfolgt: die Sichtlinie durch Kopf und Hand, sowie die Fortsetzung der Linie des Unterarms. Die Unterarmrichtung wird durch eine Analyse der 3D-Bildpunkte in einem 20cm Radius um die Hand gewonnen.

Die Ergebnisse der Evaluation des Systems finden sich in Kapitel 5. In Experimenten mit 10 Testpersonen wurde die Genauigkeit der Zeigerichtungsbestimmung und die Erkennungsrate des Gestenerkenners gemessen.

In den Anhängen A und B werden Grundlagen der Stereobildverarbeitung bzw. der Hidden Markov Modelle erläutert. In Anhang C wird ein Überblick über den Aufbau des im Rahmen dieser Arbeit entwickelten Programms gegeben.

1.2 Verwandte Arbeiten

Das Verfolgen von Personen mithilfe einer oder mehrerer Kameras ist sowohl für kooperative multimodale Anwendungen als auch für passive Überwachungsaufgaben von großer Bedeutung. Dementsprechend gibt es auch viele unterschiedliche Ansätze, aus Videobildern ein Modell eines Menschen vor der Kamera zu extrahieren.

In [WADP97] stellen Wren et al. das System *Pfinder* vor, welches in Echtzeit ein zweidimensionales Modell einer einzelnen Person bestehend aus Kopf, Händen, Füßen und restlichem Körper aufbaut. Pfinder stützt sich dabei auf das Merkmal Farbe und das Vorhandensein eines statischen Hintergrunds.

Azarbayejani und Pentland beschreiben in [AP96] ein Verfahren, wie Kopf und Hände anhand ihrer Farbe in 3D verfolgt werden können. Die Kalibrierung der Stereokamera erfolgt dabei automatisch durch die Beobachtung der zu verfolgenden Person.

Die aus der Stereobildverarbeitung stammenden Disparitätenbilder werden von Darrell et al. [DGHW98] in Kombination mit Hautfarbe sowie einem neuronalen Netz zur Gesichtsdetektion verwendet, um eine oder mehrere Personen (Silhouette und Kopf) zu verfolgen.

Eine theoretische Grundlage für das gleichzeitige Verfolgen mehrerer Objekte unabhängig von den verwendeten Merkmalen liefert Tao et al. in [TSK99]. Dabei kommt eine Erweiterung des *Condensation*-Algorithmus aus [IB98] zum Einsatz.

Im Bereich der Gestenerkennung werden mit dem Einsatz von Hidden Markov Modellen gute Ergebnisse erzielt. So gelingt es Starner und Pentland in [SP95], Handgesten aus dem Vokabular der *American Sign Language* für Gehörlose mit hoher Genauigkeit zu erkennen.

Mit dem Zusammenspiel von Sprache und Gestik am Beispiel eines Fernseh-Wetteransagers beschäftigen sich Poddar et al. in [PSOS98]. Die Erkennungsgenauigkeit eines HMM-basierten Gestenerkenners konnte durch Kombination mit sprachlichen Schlüsselwörtern gesteigert werden.

Becker stellt in [Bec97] ein System vor, das Sequenzen von Gesten der chinesischen Sportart *T'ai Chi* mithilfe von HMMs klassifiziert. Als Merkmale werden die 3D-Positionen von Kopf und Händen verwendet.

Wilson und Bobick schlagen in [WB98] eine Erweiterung der HMM-Algorithmen vor, um so genannte *parametrische Gesten* besser modellieren zu können. Dabei handelt es sich um Gesten, deren Ausführungsweise von einem inhaltlich bedeutenden Parameter abhängt. Eine Zeigegeste beispielsweise wird je nach Position des Zeigeziels unterschiedlich ausgeführt.

Mit der Erkennung von Zeigegesten und der Bestimmung der Zeigerichtung in Disparitätenbildern beschäftigen sich Jojic et al. in [JBM+00]. Modelliert durch eine Gaußmischverteilung wird der Körper beim Zeigen auf eine Wandtafel in Arm und Rumpf getrennt. Mithilfe der Orientierung des ausgestreckten Arms bzw. der Linie durch Kopf und Hand wird die Zeigerichtung abgeschätzt.

2 Verfolgen von Kopf und Händen

Zahlreiche Ansätze im Bereich des Personenverfolgens (person tracking) stützen sich teilweise oder ausschließlich auf das Merkmal *Hautfarbe* (vergleiche [WADP97] oder [PSOS98]). Vorausgesetzt, dass Kopf und Hände die einzigen unbekleideten Körperteile sind, können mithilfe eines Modells für Hautfarbe mit nur geringem Berechnungsaufwand die gesuchten Bildregionen gefunden werden.

Durch die Verknüpfung des Merkmals Hautfarbe mit Geometrieinformationen aus der Stereobildverarbeitung können im Vergleich zu rein farbbasierten Ansätzen nicht nur die räumlichen Koordinaten von Kopf und Händen gewonnen werden – es wird darüberhinaus auch die Qualität des Trackings erhöht (siehe z. B. [DGHW98]). Letzteres wird unter anderem dadurch erreicht, dass mithilfe von Entfernungsangaben Objekte getrennt werden können, deren Abbilder sich aufgrund der Kameraperspektive überlappen.

Das hier vorgestellte System für das Verfolgen von Kopf und Händen arbeitet auf Bildsequenzen, die von einer Stereokamera geliefert werden (siehe Abbildung 2.1). Nach einer Verknüpfung der Hautfarbpunkte mit ihren 3D-Koordinaten werden Hautfarb-Cluster im dreidimensionalen Raum gesucht. Aus diesen Clustern wählt das System zu jedem Zeitpunkt jene aus, die am wahrscheinlichsten einen Kopf bzw. eine Hand darstellen.

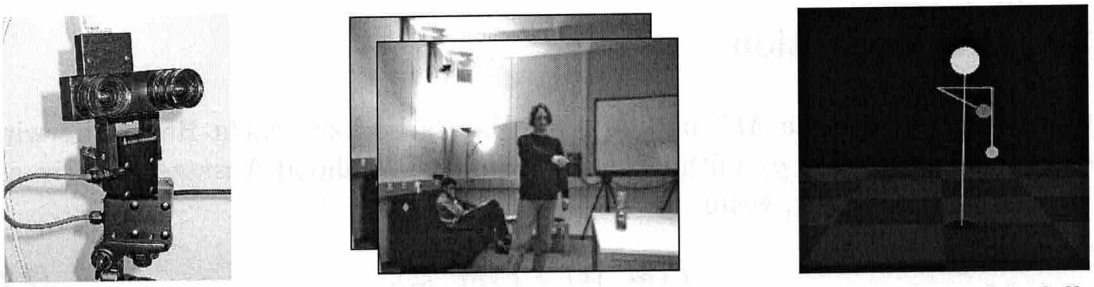


Abbildung 2.1: Eine Stereokamera liefert kontinuierlich Bildpaare, aus denen die Positionen von Kopf und Händen extrahiert werden.

Ein Vorläufer dieses Systems wurde bereits in [Nic02] vorgestellt. Der hier beschriebene Ansatz weist in vielen Bereichen Verbesserungen auf – insbesondere was die Ballung der Hautfarbregionen und die Hypothesenbildung angeht.

2.1 Modellierung der Hautfarbe

Die Grundlage der Suche nach Kopf und Händen ist das Merkmal Hautfarbe. Nach [YLW97] ist bekannt, dass sich die Farbwerte menschlicher Haut in einem engen Bereich des RGB-Farbraums ballen, und dass sich die Streuung der Werte noch reduzieren lässt, wenn man sie in den chromatischen Farbraum transformiert.

Im chromatischen Farbraum (auch: rg-Farbraum) werden zwei Farbvektoren auf denselben Punkt abgebildet, wenn sie – trotz unterschiedlicher Helligkeit – denselben Farbton haben. Der chromatische Farbraum entsteht aus dem RGB-Farbraum durch eine Transformation der Form

$$r = \frac{R}{R+G+B}, \quad g = \frac{G}{R+G+B}. \tag{2.1}$$

Um die Verteilung der Hautfarbwerte im rg-Farbraum zu charakterisieren, ist es erforderlich, ein Modell der Hautfarbe zu erstellen. Prinzipiell ist die Repräsentation der Hautfarbverteilung durch ein parametrisches Modell, wie z. B. eine Gaußmischverteilung, oder durch ein nicht-parametrisches Modell, z. B. ein Histogramm, möglich. In der vorliegenden Arbeit kommt letzteres zur Anwendung.

Ein Histogramm H_+ wird mit Bildausschnitten initialisiert, in denen ausschließlich Hautfarbe zu sehen ist. Zum Vergleich wird ein zweites Histogramm H_- aufgebaut, welches mit Bildbereichen initialisiert wird, die keine Hautfarbe zeigen. Beide Histogramme werden normiert, so dass sich für die Summe ihrer Einträge der Wert 1 ergibt.

2.1.1 Klassifikation

Gegeben zwei Modelle M^+ und M^- für Hautfarbe bzw. Nicht-Hautfarbe wird ein Farbwert $x = (r, g)$ mithilfe des Maximum-Likelihood-Ansatzes positiv als Hautfarbe klassifiziert, wenn

$$P(M^+|x) > P(M^-|x). \tag{2.2}$$

Nach der Regel von Bayes gilt für ein Modell $M = \{M^+, M^-\}$:

$$P(M|x) = \frac{P(x|M) \cdot P(M)}{P(x)} \tag{2.3}$$

Aus 2.2 und 2.3 sowie $P(M^+) + P(M^-) = 1$ folgt für die positive Klassifikation von x:

$$\frac{P(x|M^+)}{P(x|M^-)} > \frac{1 - P(M^+)}{P(M^+)} \tag{2.4}$$

Die Wahrscheinlichkeiten $P(x|M^+)$ und $P(x|M^-)$ lassen sich direkt aus den normierten Histogrammen H_+ und H_- ablesen. Die a-priori Wahrscheinlichkeit $P(M^+)$ des generellen Vorkommens von Hautfarbe in den Beispielbildern ist eine Konstante, weshalb sich folgende Formel für die positive Klassifikation von x ergibt:

$$\frac{H_+(x)}{H_-(x)} > C \tag{2.5}$$

Im Folgenden werden die Bildpunkte aber nicht binär klassifiziert, sondern es wird für jeden Punkt der Wert des Quotienten aus 2.5 berechnet. Dieser ist größer, je wahrscheinlicher der entsprechende Bildpunkt Hautfarbe aufweist. Abbildung 2.2 zeigt das Ergebnis dieser Klassifikation auf einem Videobild.

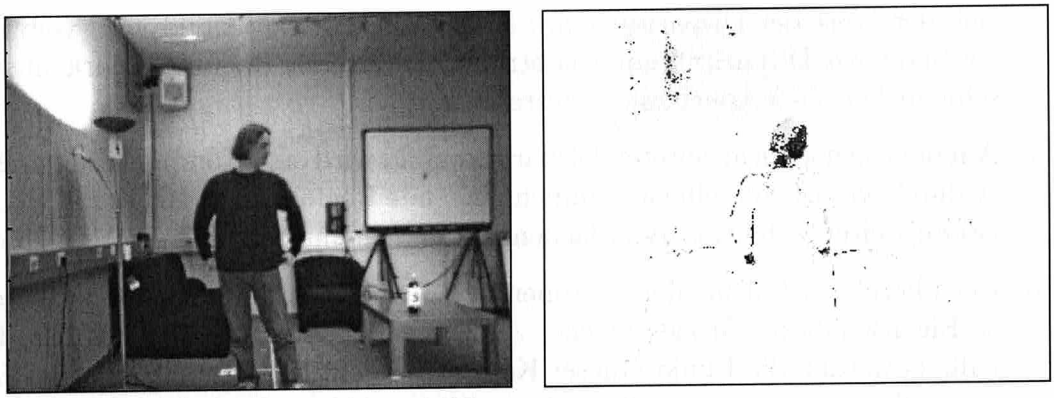


Abbildung 2.2: Klassifikation nach Hautfarbe: Je dunkler ein Punkt im rechten Bild ist, desto wahrscheinlicher weist er Hautfarbe auf.

Die Kamerabilder liegen im RGB-Farbraum vor. Eine Eigenschaft dieses Farbraums besteht darin, dass Intensität und Farbton nicht getrennt voneinander repräsentiert werden: je dunkler eine Farbe ist, desto weniger Farbtöne stehen für ihre Darstellung zur Verfügung. Als Konsequenz daraus ergibt sich das Problem, dass die Hautfarbklassifikation für dunkle Farben nur schlechte Ergebnisse liefert. Daher werden nur jene Bildpunkte klassifiziert, die eine gewisse Mindesthelligkeit I_{min} aufweisen[1]:

$$R + G + B > I_{min} \tag{2.6}$$

[1]Für I_{min} hat sich in der Praxis ein Wert von 50 als sinnvoll erwiesen.

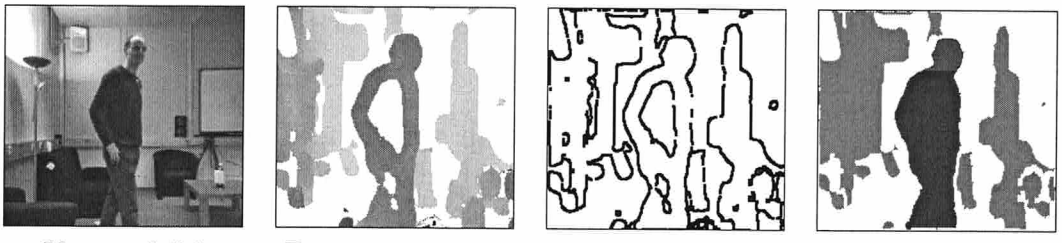
2.1.2 Automatische Initialisierung

Trotz der Helligkeitsnormierung mittels Transformation in den chromatischen Farbraum werden die Messwerte für Hautfarbe immer noch stark von der Beleuchtungssituation beeinflusst[2]. Es ist daher wünschenswert, dass das Modell in der Lage ist, sich ändernden Gegebenheiten anzupassen.

Beschreibung des Verfahrens

Das aus der Stereobildverarbeitung gewonnene Disparitätenbild (siehe Abschnitt A) ermöglicht es, unabhängig vom Merkmal Hautfarbe einen Kopf zu lokalisieren. Mit dessen Bildpunkten kann dann das Hautfarbmodell initialisiert bzw. aktualisiert werden. Das hier beschriebene mehrstufige Verfahren lehnt sich in Teilen an die in [DGHW98] beschriebene Methode an:

1. Auf dem Disparitätenbild wird eine Kantendetektion durchgeführt. Das Ergebnis ist ein Bild, in dem all jene Punkte gesetzt sind, in deren Umgebung sich der Wert der Disparität stark ändert. Dieses Kantenbild wird vom ursprünglichen Disparitätenbild subtrahiert, wodurch Regionen stark unterschiedlicher Tiefe voneinander getrennt werden.
2. Auf dem so nachbearbeiteten Disparitätenbild wird eine Komponentenanalyse durchgeführt, um eine zusammenhängende Region zu finden, die in Höhe, Breite, Seitenverhältnis und Flächeninhalt einem menschlichen Körper ähnelt.
3. Der obere Teil (30cm) der Personenregion wird dahingehend überprüft, ob er Eigenschaften (Größe, Fläche, ...) eines Kopfes aufweist. Ist dies der Fall, so werden die Punkte dieser Kopfregion dem Hautfarbhistogramm H_+ zugerechnet, wohingegen alle anderen Bildpunkte zu H_- beitragen.



Kamerabild | Disparitätenbild | Kantenbild | Personenregion

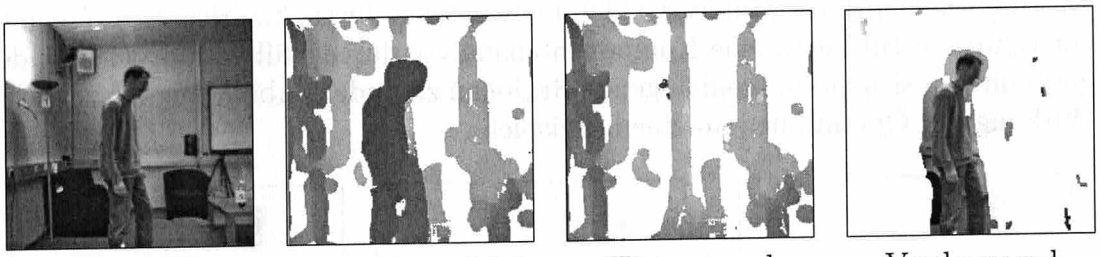
Abbildung 2.3: Vom Disparitätenbild wird das zugehörige Kantenbild subtrahiert, um Bereiche unterschiedlicher Tiefe zu trennen. Unter den so entstehenden Regionen wird die Personenregion identifiziert.

[2]Unterschiedliche Lichtquellen unterscheiden sich nicht nur hinsichtlich ihrer Helligkeit sondern auch in ihrer spektralen Zusammensetzung. Dadurch verschieben sich die vom Kamerasensor wahrgenommenen Farbwerte.

Die durchschnittlichen Werte für Höhe, Breite, Seitenverhältnis und Flächeninhalt von Personen und Köpfen, sowie die Varianzen dieser Werte wurden aus Beispieldaten gewonnen. Abbildung 2.3 verdeutlicht das beschriebene Verfahren.

Hintergrundmodellierung

Verfügt man über eine statische Kamera, oder steht die Kamera auch nur für einen kurzen Moment still (z. B. 1sec), so lässt sich mithilfe von Hintergrundmodellierung das oben beschriebene Verfahren noch verbessern, da ein statischer Hintergrund für die Personensuche ohne Belang ist und aus dem Disparitätenbild entfernt werden kann. In der vorliegenden Arbeit wurde das Verfahren der *gated background subtraction* aus [EKB98] implementiert, welches für jeden Bildpunkt den höchsten gemessenen Entfernungswert speichert. Ändert sich nun die Entfernung signifikant in Richtung Kamera, wird der betreffende Bildpunkt dem Vordergrund zugerechnet (siehe Abbildung 2.4).



Kamerabild | Disparitätenbild | Hintergrund | Vordergrund

Abbildung 2.4: Bei der gated background subtraction wird ausgehend vom Disparitätenbild ein Modell des statischen Hintergrundes aufgebaut. Bildbereiche, die deutlich näher an der Kamera liegen, werden dem Vordergrund zugerechnet.

2.2 Suche nach Kandidaten

Ein mehrstufiges Verfahren wird angewandt, um im Hautfarbbild (Abb. 2.2) Kopf und Hände zu lokalisieren: Zuerst werden zusammenhängende Regionen von Hautfarbe gesucht, die dann mit ihren räumlichen Koordinaten verknüpft werden, um sie mit einem 3D-Ballungsverfahren weiter aufzuspalten. Die so gewonnenen Clusterzentren sind Kandidaten für die Positionen von Kopf und Händen.

2.2.1 Trennung der Hautfarbregionen

Entsprechend ihrer Erscheinung als zusammenhängende Flächen von Hautfarbe werden Kopf und Hände zunächst allein im Hautfarbbild gesucht. Ein Problem des bisherigen Hautfarbbildes besteht darin, dass es eine „pixelige" und keine flächige Strukur aufweist: Selbst innerhalb geschlossener Hautflächen werden einzelne Punkte nicht als Hautfarbe klassifiziert, wodurch „Löcher" entstehen. Umgekehrt tauchen auch in Bereichen ohne Hautfarbe immer wieder einzelne Punkte auf, die fälschlicherweise als Hautfarbe klassifiziert werden.

Um diese störenden Effekte zu eliminieren, wird das Bild mit morphologischen Operatoren behandelt: Zuerst werden durch morphologisches Schließen (Dilatation mit anschließender Erosion) mit einem 3x3-Strukturelement kleinere Lücken in den positiv klassifizierten Bereichen geschlossen. Danach wird das Bild durch morphologisches Öffnen (Erosion + Dilatation, ebenfalls 3x3) von vereinzelten positiven Punkten gereinigt.

Anschließend wird das Grauwertbild mit einem Schwellwert binärisiert. Auf dem so entstandenen Bild wird eine Komponentenanalyse durchgeführt, um voneinander getrennte, in sich zusammenhängende Regionen zu finden. Abbildung 2.5 zeigt die Wirkung der Operationen an einem Beispiel.

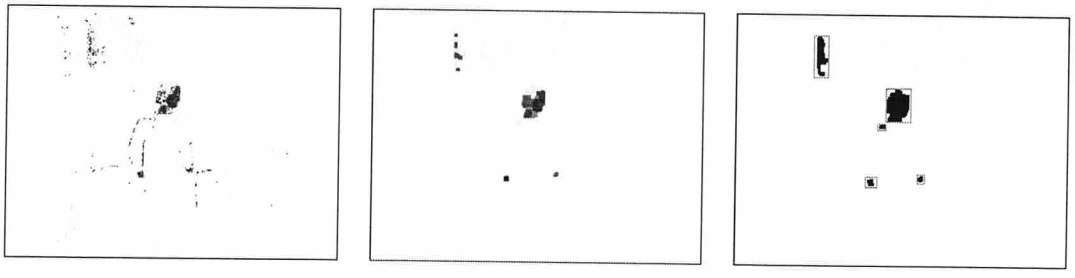


Abbildung 2.5: Das ursprüngliche Hautfarbbild (links) wird morphologisch gefiltert (Mitte) und dann einer Komponentenanalyse unterzogen (rechts). (Die gefundenen Komponenten werden durch ihre bounding box symbolisiert.)

2.2.2 Ballung der Hautfarbpunkte im Raum

Jede der im letzten Abschnitt beschriebenen Hautfarbregionen kann für sich alleine das Abbild einer Hand oder eines Kopfes sein. Eine Region kann aber ebenso das Abbild unterschiedlicher hautfarbener Objekte sein, die sich aufgrund der Kameraperspektive zu einer gemeinsamen Fläche in der Bildebene vereinigt haben.

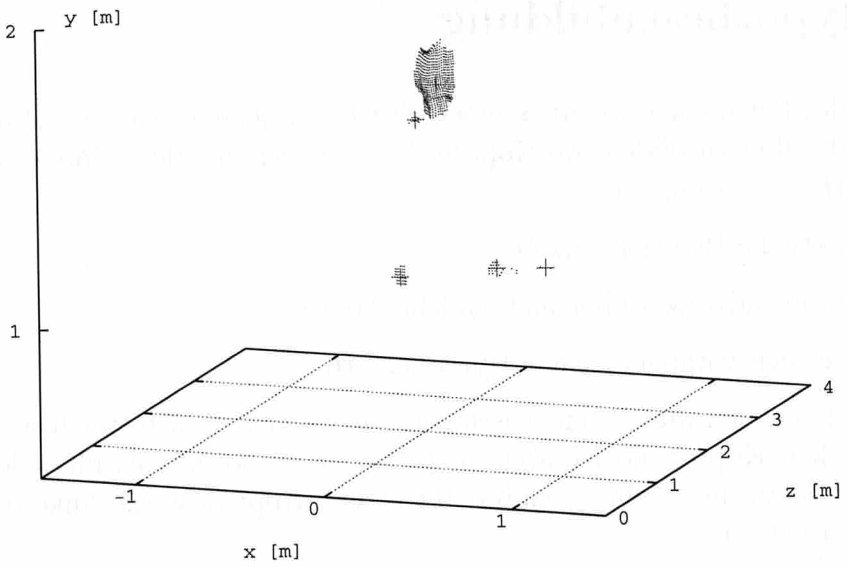


Abbildung 2.6: Die Hautfarbpunkte werden mit einem k-Mittelwerte-Verfahren räumlich geballt. Die daraus resultierenden Clusterzentren sind als Kreuze eingezeichnet.

Um solche Objekte voneinander zu trennen, werden die zu einer Region gehörenden Hautfarbpunkt mit den jeweiligen, aus der Stereobildverarbeitung hervorgehenden räumlichen Koordinaten verknüpft. Die so entstehende Punktwolke wird durch Anwendung eines k-Mittelwerte-Verfahrens zerlegt (clustering). Dabei fließen die Hautfarbwahrscheinlichkeiten der einzelnen Punkte als Gewichte in den Ballungsalgorithmus ein. Seien $x^k_{1..N}$ die 3D-Koordinaten derjenigen Bildpunkte, für die der Mittelwert m_k bei der aktuellen Iteration der nächstgelegene ist, und seien $c_{1..N}$ die Hautfarbwahrscheinlichkeiten dieser Bildpunkte, dann wird m_k folgendermaßen neu geschätzt:

$$m_k = \frac{\sum_{i=1}^{N} c_i \cdot x_i}{\sum_{i=1}^{N} c_i} \tag{2.7}$$

Da die Anzahl der Cluster nicht im Voraus bekannt ist, wird mit einer hohen Zahl von zufällig initialisierten Mittelwerten begonnen (etwa $K = 10$). Im Verlauf des Algorithmus kann es passieren, dass einzelnen Mittelwerten keine Daten mehr zufallen, da sie durch andere Mittelwerte „abgeschattet“ werden. Solche Mittelwerte werden entfernt. Ebenso werden nach der Ballung Mittelwerte entfernt, die zu dicht ($< 0.10m$) an anderen Mittelwerten liegen. Abbildung 2.6 zeigt an einem Beispiel das Ergebnis dieses Verfahrens.

2.3 Hypothesenbildung

Aufgabe des Personenverfolgungssystems ist es, zu jedem Zeitpunkt t eine Hypothese H_t darüber zu bilden, wo Kopf und Hände sich gerade befinden. Grundlage der Hypothesenbildung ist

- die aktuelle Beobachtung O_t,
- Hintergrundwissen über menschliche Anatomie,
- die zeitlich vorangegangene Hypothese H_{t-1}.

Ausgehend von den im letzten Abschnitt gewonnenen Hautfarb-Clustern werden alle möglichen Kopf-Hand-Kombinationen gebildet, so dass es für jeden Cluster eine Arbeitshypothese gibt, gemäß derer er den Kopf bzw. die linke oder rechte Hand repräsentiert.

Für jede Arbeitshypothese H werden drei Maße berechnet:

- Das Beobachtungsmaß $P(O_t|H)$ ist ein Maß dafür, wie gut H zur aktuellen Beobachtung passt.
- Das Anatomiemaß $P(H)$ gibt die Wahrscheinlichkeit für die Körperhaltung an, die durch H repräsentiert wird.
- Das Übergangsmaß $P(H|H_{t-1})$ drückt die Wahrscheinlichkeit des Übergangs von der vorangegangenen Hypothese zu H aus.

Es wird diejenige Arbeitshypothese ausgewählt, die das Produkt der drei oben genannten Maße maximiert:

$$H_t = argmax_H P(O_t|H) \cdot P(H) \cdot P(H|H_{t-1}) \tag{2.8}$$

2.3.1 Beobachtungsmaß

Das Beobachtungsmaß $P(O_t|H)$ drückt die Wahrscheinlichkeit dafür aus, dass O_t beobachtet wird, wenn H gilt. Zur Berechnung dieses Maßes werden über die durch H repräsentierten Positionen Ellipsen gelegt. Die Größe der Ellipsen richtet sich nach der durchschnittlichen Größe eines Kopfes bzw. einer Hand in der entsprechenden Entfernung. Eine gute Übereinstimmung zwischen Hypothese und Beobachtung ist dann erreicht, wenn sich die Hautfarbpunkte größtenteils unter den Ellipsenflächen versammeln. Abbildung 2.7 zeigt den Vorgang an einem Beispiel.

Die Summe der Gewichte der Punkte, die durch die Ellipsen abgedeckt werden, wird einerseits zum Gesamtgewicht aller Bildpunkte und andererseits zur kumulierten Fläche der Ellipsen ins Verhältnis gesetzt. Sei $\varepsilon(x, y) = 1$, wenn der Punkt

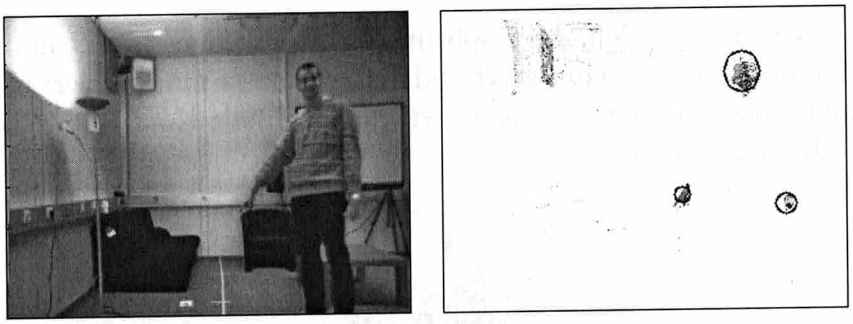


Abbildung 2.7: Um Hypothese und Beobachtung zu vergleichen, werden Ellipsen über die vermuteten Kopf- und Handpositionen gelegt. Je besser diese die Hautfarbpunkte abdecken, desto höher fällt das Beobachtungsmaß aus.

(x, y) sich innerhalb einer Ellipse befindet (und 0 sonst), und sei $s(x, y)$ das Gewicht dieses Punktes, dann berechnet sich das Beobachtungsmaß wie folgt:

$$P(O_t|H) = \frac{\sum_{x,y} s(x,y)\varepsilon(x,y)}{\sum_{x,y} s(x,y)} \cdot \frac{\sum_{x,y} s(x,y)\varepsilon(x,y)}{\overline{s} \sum_{x,y} \varepsilon(x,y)} \tag{2.9}$$

Da die Punkte mit Gewichten versehen sind, die die Stärke ihrer Zugehörigkeit zur Klasse „Hautfarbe“ ausdrücken, ist in obiger Formel auch ein Gewicht $\overline{s}$ für die Ellipsenfläche vorgesehen. Die Gewichte der Punkte bewegen sich in einem Bereich von 0 bis 255, für $\overline{s}$ wurde ein Wert von 30 gewählt.

2.3.2 Anatomiemaß

Das Anatomiemaß $P(H)$ repräsentiert die Wahrscheinlichkeit der Körperhaltung, die durch H ausgedrückt wird. Um dieses Maß zu berechnen, wurden anatomische Maße teils aus Testdaten gewonnen, teils aus Hintergrundwissen formuliert. Dabei handelt es sich um:

- die Größe eines Menschen (repräsentiert durch eine Gaußverteilung),
- die räumliche Verteilung[3] der in Beispieldaten beobachteten Handpositionen (siehe Abbildung 2.8),
- die maximal mögliche Entfernung zwischen Hand und Kopf.

[3] Als Modell für die Aufenthaltshäufigkeit wurde eine dreidimensionale Gaußmischverteilung gewählt.

$P(H)$ wird berechnet, indem die Kopf- und Handpositionen aus H in die entsprechenden Gaußverteilungen eingesetzt und die Ergebnisse multipliziert werden. Verletzt H anatomische Beschränkungen (wie die maximale Kopf-Hand-Entfernung), so wird $P(H)$ auf 0 gesetzt.

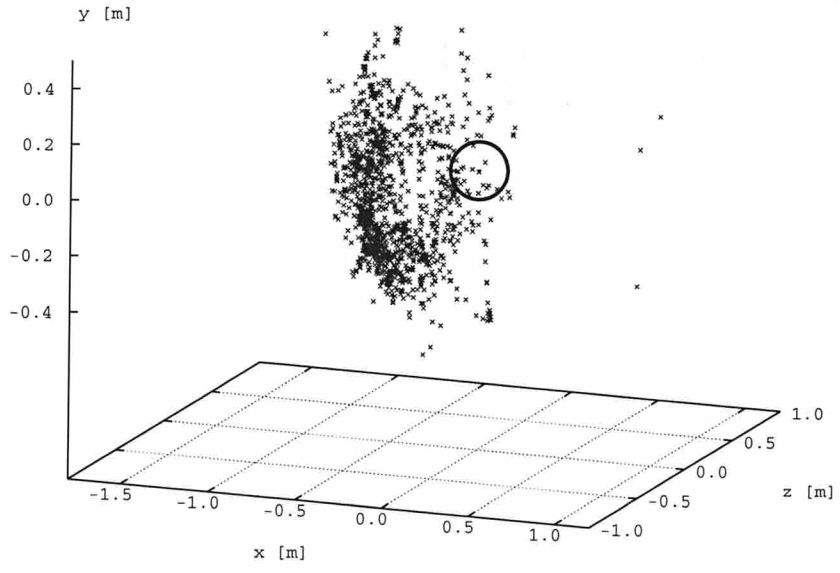


Abbildung 2.8: Aufenthaltsorte der rechten Hand relativ zum Kopf (gemessen über zwei Minuten).

2.3.3 Übergangsmaß

Um das Übergangsmaß $P(H|H_{t-1})$ zu berechnen, werden die Entfernungen von Kopf und Händen in H zu ihren Vorgängern in H_{t-1} betrachtet und zu den vorgegebenen Maximalwerten ins Verhältnis gesetzt. Diese Maximalwerte ergeben sich aus der Tatsache, dass Menschen ihren Kopf und ihre Hände in der kurzen Zeit zwischen zwei Einzelbildern (typisch: $\Delta t = 0.05sec$) nicht beliebig weit bewegen können. Konkret wurden folgende Werte angenommen:

$$\begin{aligned} \Delta Kopf_{max} &= 1.0m/sec \cdot \Delta t + N \\ \Delta Hand_{max} &= 2.0m/sec \cdot \Delta t + N \end{aligned} \tag{2.10}$$

Bei N handelt es sich um einen Zuschlag, der dem Messrauschen Rechnung trägt, und der hier mit $N = 0.1m$ abgeschätzt wurde. Die Berechnung von $P(H|H_{t-1})$ hat die Form

$$P(H|H_{t-1}) = \left(1 - \frac{\Delta Kopf}{\Delta Kopf_{max}}\right) \cdot \left(1 - \frac{\Delta Re.Hand}{\Delta Hand_{max}}\right) \cdot \left(1 - \frac{\Delta Li.Hand}{\Delta Hand_{max}}\right) \tag{2.11}$$

Überschreitet eine Kopf- oder Handbewegung den Maximalwert, so wird $P(H|H_{t-1})$ auf einen Wert nahe 0 gesetzt[4].

2.4 Resultate

Zahlreiche Testsequenzen und Live-Experimente[5] belegen, dass es mit der hier beschriebenen Methode möglich ist, eine Person robust über mehrere Minuten hinweg zu verfolgen. Das Verfolgen des Kopfes gelingt dabei etwas zuverlässiger als das Verfolgen der Hände. Letztere sind aufgrund von Verdeckung, geringerer Größe, höherer Geschwindigkeit und größerer Varianz hinsichtlich Form und Größe ihres Abbildes schwieriger zu vefolgen. Das System ist in der Lage, auch nach Aussetzern oder Verwechslungen bei der nächsten Gelegenheit den korrekten Pfad fortzusetzen.

Ein typisches Problem bei der Verwendung des Merkmals Hautfarbe besteht darin, dass es Oberflächen gibt, die farblich nicht von Haut zu unterscheiden sind. Dazu zählen vor allem Holz und Karton, sowie einige Textilien, welche besonders verwechslungsträchtig sind, da sie direkt am Körper getragen werden. Auch ist es erforderlich, dass Kopf und Hände die einzigen unbekleideten Körperteile sind.

Eine andere Schwierigkeit stellen Bildbereiche dar, die durch Überbelichtung weiß gewordenen sind (z. B. durch ungünstige Voreinstellung der Kamerablende oder durch reflektierende Oberflächen). Generell ist es bei Motiven mit großem Helligkeitsumfang schwer möglich, in den hellen Gebieten Überbelichtung zu vermeiden und gleichzeitig die dunklen Bildteile noch farblich aufzulösen.

Die automatische Initialisierung des Hautfarbmodells sorgt für eine kontinuierliche Adaption an die herrschenden Lichtverhältnisse. Hintergrundmodellierung erleichtert die Initialisierung, ist aber nicht zwingend notwendig.

Auf einem PC mit 2.4GHz Pentium 4 Prozessor erreicht das Verfolgungssystem eine Verarbeitungsleistung von etwa 12 Bildern pro Sekunde (bzw. 18 bei Verzicht auf automatische Initialisierung).

[4]Das Maß sollte niemals ganz auf 0 absinken, damit die Möglichkeit erhalten bleibt, einen sprunghaften Übergang von einer falschen statischen Hypothese zu einer korrekten auszuführen.

[5]Für die Bestimmung der Kopf- und Handpositionen in einem realistischen Testszenario stand keine alternative Messmethode zur Verfügung, so dass auf eine quantitative Leistungsbewertung des Verfolgungssystems verzichtet werden musste.

3 Erkennung von Zeigegesten

Eine Zeigegeste im Sinne dieser Arbeit ist eine Armbewegung[1], durch die die Hand in Richtung eines bestimmten Zieles bewegt wird. Daher erscheint es zweckmäßig, zur Erkennung von Zeigegesten die Trajektorie der zeigenden Hand zu betrachten.

Wie in Kapitel 2 beschrieben, sind die Trajektorien, die das Verfolgungssystem aus den Kamerabildern extrahiert, nicht perfekt, sondern durch den generellen Messfehler und gelegentliche Aussetzer beeinträchtigt. Die Aufgabe besteht also darin, den Bewegungsablauf von Zeigegesten so zu modellieren, dass sie auf derartigen Trajektorien zuverlässig erkannt werden können.

Eine weitere Schwierigkeit der Modellierung von Zeigegesten besteht darin, dass sie in ihrer Ausführung stark variieren. Die konkrete Ausführung der Geste ist unter anderem abhängig von

- der Position der zeigenden Person im Raum,
- der Position und Art des Zeigezieles,
- persönlichen Vorlieben bezüglich der Art zu zeigen.

Im Folgenden wird eine Methode vorgestellt, mit der Zeigegesten personen- und ortsunabhängig erkannt werden können. Dabei wird die Geste durch eine Kombination von *Hidden Markov Modellen* (HMMs) repräsentiert. HMMs werden seit Jahren erfolgreich in der Spracherkennung eingesetzt (siehe [Rab89]), und haben sich auch im Bereich der Gestenerkennung als zweckmäßig erwiesen (siehe z. B. [Bec97]). In Anhang B werden HMMs und die mit ihnen in Verbindung stehenden Algorithmen näher beschrieben.

3.1 Bewegungsphasen

Betrachtet man eine zeigende Person, so lassen sich drei verschiedene Bewegungsphasen erkennen:

[1] Das Zeigen allein durch Fingerbewegungen oder mithilfe eines Zeigestocks wird im Rahmen dieser Arbeit nicht berücksichtigt.

- Beginn (B): Die Hand bewegt sich von einer beliebigen Startposition ausgehend in Richtung Zeigeziel. Im Vergleich zur Zeigeposition ist die Startposition in der Regel näher am Boden und dichter am Körper.
- Halten (H): Die Hand verweilt bewegungslos in Zeigeposition. Um die Zeigerichtung zu bestimmen, ist es von zentraler Bedeutung, diese Phase zeitlich möglichst genau eingrenzen zu können.
- Ende (E): Die Hand verlässt die Zeigeposition. Sie bewegt sich dabei häufig entgegengesetzt der Trajektorie der B-Phase zurück.

Bei der Untersuchung von 89 Gesten, die von 10 unterschiedlichen Personen ausgeführt wurden, ergab sich für die durchschnittlichen Längen der einzelnen Phasen das Ergebnis in Tabelle 3.1. Die Grenzen der Phasen wurden manuell markiert.

	$\mu[sec]$	$\sigma[sec]$
Gesamte Geste	1,75	0,48
Beginn	0,52	0,17
Halten	0,76	0,40
Ende	0,47	0,12

Tabelle 3.1: Durchschnittliche Länge der Zeigegesten und ihrer drei Bewegungsphasen

Es fällt auf, dass die Haltephase die größte Varianz aufweist: bei manchen Gesten beträgt die Haltezeit einige Sekunden, bei anderen ist so gut wie keine Haltezeit festzustellen. Die B-Phase ist im Schnitt länger als die E-Phase, was sich durch die Notwendigkeit des genauen Anpeilens des Zeigeziels in der B-Phase erklären lässt.

3.2 Modellierung der Geste

In der vorliegenden Arbeit wurde der Ansatz gewählt, jede der drei Bewegungsphasen durch ein eigenes HMM zu modellieren. Dies hat den Vorteil, dass die Merkmalssequenzen, die durch das B- und das E-Modell repräsentiert werden, nicht so stark in der Länge variieren, wie wenn ein einziges HMM für die gesamte Geste zuständig wäre. Zudem kann die Haltephase durch die Tatsache, dass sie mit einem dedizierten HMM modelliert wird, genauer identifiziert werden. Ein Nachteil der separaten Modellierung liegt darin, dass die Klassifikation aufwändiger wird, weil die Ausgaben verschiedener Modelle im zeitlichen Verlauf betrachtet werden müssen.

Um einen Referenzwert für die Ausgaben der Phasenmodelle zu erhalten, wird ein weiteres Modell, das so genannte *Null-Modell*, eingeführt. Es dient dazu, Zeigebewegungen von sonstigen Bewegungen abzugrenzen und repräsentiert all die Handbewegungen, die *nicht* einer Zeigegeste zugerechnet werden.

Um geeignete Topologien für die drei Phasenmodelle M_B, M_H und M_E zu finden, wurde mit unterschiedlichen Anzahlen von Zuständen (2-5) und Gaußverteilungen (1-4) pro Zustand experimentiert. Dabei hat sich herausgestellt, dass mit links-rechts Modellen, bestehend aus drei Zuständen und einer Mixtur von zwei Gaußverteilungen, die besten Ergebnisse erzielt werden konnten.

Auch für das Null-Modell M_0 wurden in gleicher Weise Experimente durchgeführt. Die Sequenzen, die durch M_0 repräsentiert werden, sind höchst heterogen. Anschaulich besteht die Aufgabe des Null-Modells weniger darin, einen bestimmten Typ von Bewegungen zu charakterisieren, als vielmehr darin, die Verteilung aller möglichen Bewegungen im Merkmalsraum wiederzugeben. Aus diesem Grund wurde auch mit einer ergodischen Topologie experimentiert, bei der jeder Zustandsübergang erlaubt ist. Diese Topologie – analog zu den anderen Modellen mit 3 Zuständen à 2 Gaußverteilungen – hat sich dann auch als am geeignetsten erwiesen. Abbildung 3.1 zeigt die Topologie aller verwendeten Modelle.

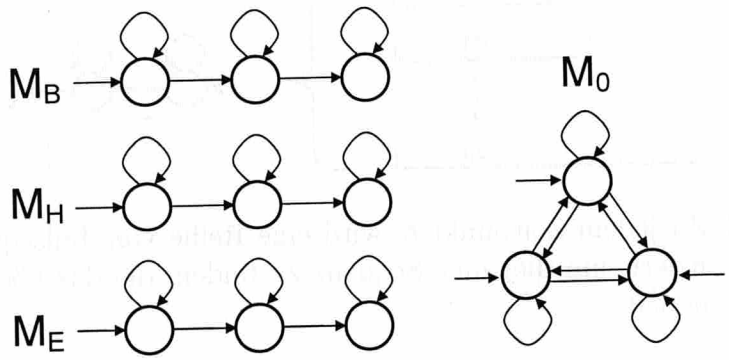


Abbildung 3.1: Drei dedizierte links-rechts HMMs modellieren die einzelnen Bewegungsphasen einer Zeigegeste. Ein ergodisches Null-Modell repräsentiert die Bewegungen, die nicht zu Zeigegesten gehören.

Alle Modelle werden auf Sequenzen trainiert, in denen die einzelnen Phasen manuell gekennzeichnet sind. Zum Training wird der iterative Baum-Welch Algorithmus zur Parameterschätzung angewandt, der in Anhang B beschrieben wird.

3.3 Klassifikation

Jedesmal, wenn das Vefolgungssystem eine neue Kopf- oder Handposition meldet, müssen die zurückliegenden Sekunden der Hand-Trajektorie mit allen Phasenmodellen klassifiziert werden. Dabei tritt folgendes Problem auf: Die Längen der Bewegungsphasen variieren stark. Würde nun stets ein Zeitfenster fester Größe klassifiziert, z. B. 1sec, so würden die Modelle im Falle einer längeren Bewegungsphase nur einen Teil der eigentlichen Sequenz bzw. im Falle einer kürzeren Bewegungsphase eine Sequenz mit teilweise phasenfremden Werten klassifizieren.

Um dieses Problem zu lösen, wird eine ganze Reihe von Teilsequenzen $s_0..s_n$ klassifiziert, die zu unterschiedlichen Zeitpunkten in der Vergangenheit beginnen und mit der aktuellen Zeit t_0 enden (siehe Abbildung 3.2). Die Länge der untersuchten Teilsequenzen richtet sich nach der kleinsten bzw. größten zu erwartenden Länge der entsprechenden Bewegungsphase. Als Grundlage dienen die Messwerte aus Tabelle 3.1.

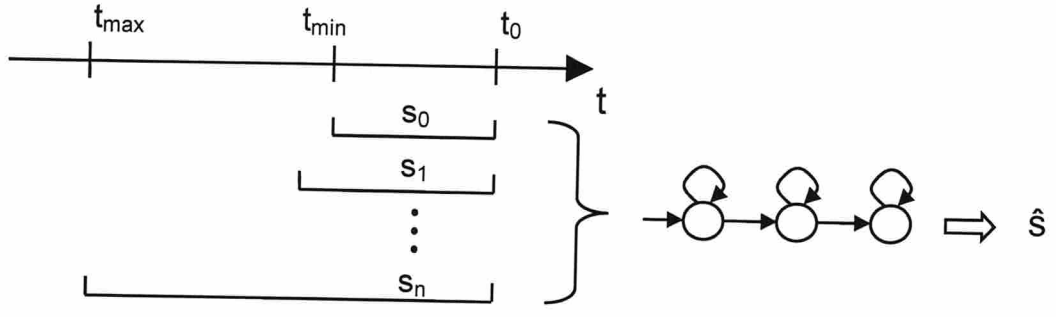


Abbildung 3.2: Zu jedem Zeitpunkt t_0 wird eine Reihe von Teilsequenzen klassifiziert, um diejenige Sequenz zu finden, die das höchste Ergebnis erzielt.

Für jedes der drei Phasenmodelle M_B, M_H, M_E wird die Sequenz $\widehat{s}_{B,H,E}$ gesucht, die die höchste Ausgabewahrscheinlichkeit erzielt. Zur Berechnung von $P(s|M)$ wird der *Forward*-Algorithmus mit *Scaling* (siehe Anhang B) verwendet, der die log-Wahrscheinlichkeit $logP(s|M)$ liefert:

$$\widehat{s}_{B,H,E} = argmax_{s=s_0..s_n} logP(s|M_{B,H,E}) \tag{3.1}$$

In [Bec97] wird eine Methode beschrieben, anhand derer die Klassifikation einer Reihe von Sequenzen effizient vorgenommen werden kann. Diese Methode basiert auf der rekursiven Natur des Forward-Algorithmus[2] und setzt voraus, dass

[2]In [Bec97] wird zur Auswertung der Viterbi-Algorithmus verwendet. Die dort vorgeschlagene Methode lässt sich jedoch auch auf den Forward-Algorithmus übertragen.

die Modelle mit zeitlich invertierten Sequenzen trainiert und ausgewertet werden. Die invertierten Sequenzen beginnen alle zum selben Zeitpunkt t_0 und enden an unterschiedlichen Zeitpunkten. Es ist nun ausreichend, einen einzigen forward-Durchgang mit der längsten Sequenz s_n vorzunehmen. Die Ergebnisse der kürzeren Sequenzen sind dann Zwischenergebnisse der Klassifikation von s_n.

3.4 Suche

Zu jedem Zeitpunkt t werden für die drei Phasenmodelle die Ausgabewahrscheinlichkeiten $P_{B,H,E}(t)$ berechnet, die auf der jeweils besten Sequenz $\widehat{s}_{B,H,E}$ basieren. Die Wahrscheinlichkeiten werden normiert, indem sie jeweils zur Ausgabe des Null-Modells ins Verhältnis gesetzt werden:

$$P_{B,H,E}(t) = logP(\widehat{s}_{B,H,E}|M_{B,H,E}) - logP(\widehat{s}_{B,H,E}|M_0) \tag{3.2}$$

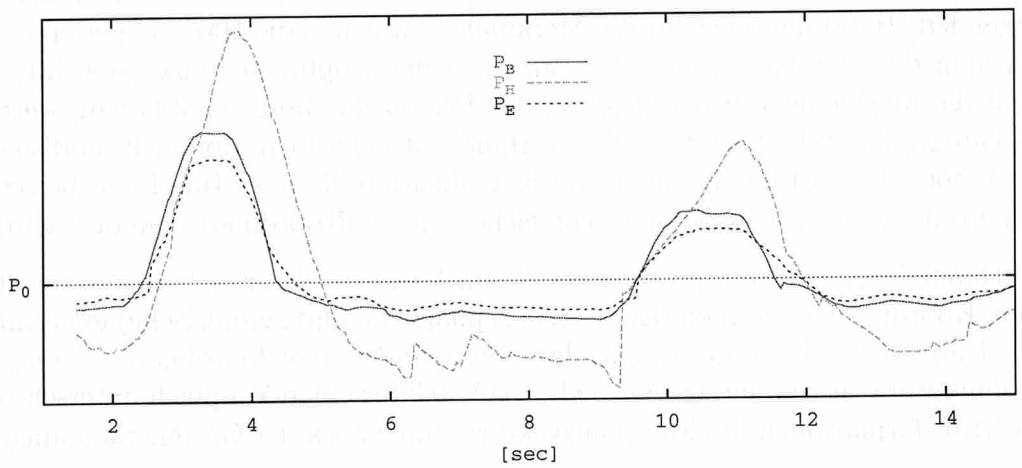


Abbildung 3.3: Normierte Ausgabewahrscheinlichkeiten der Phasenmodelle im Verlauf einer Sequenz mit zwei Zeigegesten. Die Nulllinie entspricht der Ausgabe des Null-Modells.

Um eine Zeigegeste zu erkennen, muss nach drei aufeinander folgenden Intervallen gesucht werden, in denen die Ausgaben von P_B, P_H und P_E der Reihe nach vergleichsweise hoch sind. Idealerweise würde zu diesen Zeitpunkten das jeweilige Modell klar über die beiden anderen Modelle dominieren. Wie Abbildung 3.3 zeigt, sind die Ausgaben der Phasenmodelle jedoch nicht direkt miteinander vergleichbar. So ist die Ausgabe des Mitte-Modells P_M fast über den gesamten Verlauf der Geste höher als die von P_B und P_E.

Um diesem Umstand Rechnung zu tragen, wurde der folgende Ansatz gewählt: Zu jedem Zeitpunkt t_E, zu dem das E-Modell stärker ist als das B-Modell, wird mit der Suche begonnen. Ausgehend von t_E wird rückwärts nach einem Zeitpunkt t_B gesucht, zu dem das B-Modell dominiert. Ist der Abstand $t_E - t_B$ innerhalb des erlaubten Bereichs für eine Zeigegeste (Tabelle 3.1), wird zwischen t_B und t_E nach einem Zeitpunkt t_H gesucht, zu dem das H-Modell akzeptiert wird. Eine Zeigegeste wird also genau dann erkannt, wenn drei Zeitpunkte $t_B < t_H < t_E$ existieren, so dass

$$\begin{aligned} P_E(t_E) > P_B(t_E) \wedge P_E(t_E) > 0 \\ P_B(t_B) > P_E(t_B) \wedge P_B(t_B) > 0 \\ P_H(t_H) > 0 \end{aligned} \tag{3.3}$$

3.5 Wahl der Merkmale

Wie in den letzten Abschnitten angedeutet, sind die räumlichen Koordinaten der (zeigenden) Hand die Basis für die Merkmale, auf denen die HMMs operieren. Als Ursprung des Handkoordinatensystems wird der Kopfmittelpunkt gewählt, wodurch die Merkmale von der Position der Person im Raum unabhängig werden. Um Gesten der rechten und der linken Hand mit denselben Modellen repräsentieren zu können, wird bei Training und Klassifikation die linke Hand auf die rechte Körperseite gespiegelt, indem das Vorzeichen ihrer x-Koordinate geändert wird.

Zur Darstellung der Handposition wurde im Rahmen dieser Arbeit mit verschiedenen Koordinatensystemen (kartesische, sphärische und zylindrische) experimentiert. Dabei hat sich herausgestellt, dass mit *zylindrischen Koordinaten* die besten Ergebnisse erzielt werden. (Siehe auch [CBA+96] für einen Vergleich unterschiedlicher Transformationen des Merkmalsvektors zum Zweck der Gestenerkennung.)

Abbildung 3.4 zeigt eine Hand im zylindrischen Kopf-Koordinatensystem. Zylindrische Koordinaten (r, θ, y) lassen sich aus kartesischen Koordinaten (x, y, z) wie folgt erzeugen:

$$\begin{aligned} r &= \sqrt{x^2 + z^2} \\ \theta &= \arctan \frac{x}{z} \\ y &= y \end{aligned} \tag{3.4}$$

Der Radius r steht für den Abstand der Hand zum Körper, was ihn zu einem wichtigen Merkmal für die Erkennung von Zeigegesten macht. Anders als sein Pendant im sphärischen Koordinatensystem ist r von der Höhe y unabhängig. Der Azimuthwinkel θ bewegt sich im Intervall $[0, 2\pi]$. Bei der Berechnung von

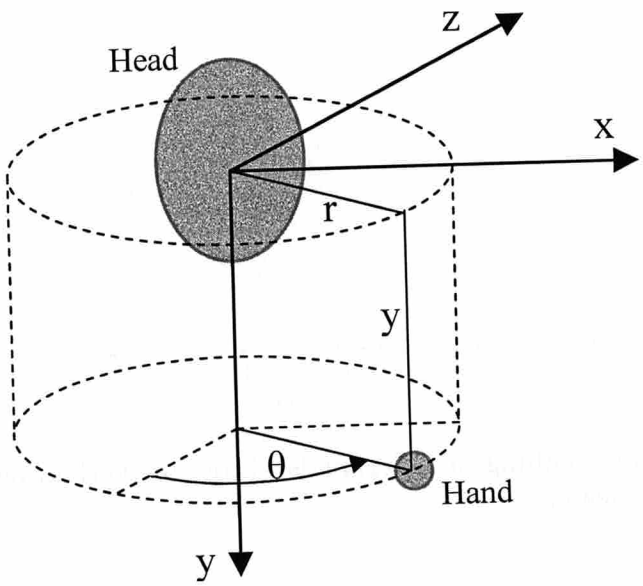


Abbildung 3.4: Hand im zylindrischen Kopf-Koordinatensystem

θ muss darauf geachtet werden, dass der sprunghafte Übergang von 2π auf 0 an einer Stelle liegt, die bei einer normalen Zeigegeste nicht erreicht wird (hier: hinter dem Körper). Eine stetige Handbewegung könnte anderenfalls zu einem Sprung im Merkmalsraum führen.

Würden die HMMs direkt mit den Werten von y und θ trainiert, bestünde die Gefahr, dass sie sich den Positionen der Zeigeziele in der Trainingsmenge anpassen. Um dies zu vermeiden, werden anstatt der Werte von y und θ ihre *Deltas*, also die Werte ihrer Änderung von einem Frame zum nächsten verwendet:

$$\begin{aligned} \Delta\theta &= \theta_t - \theta_{t-1} \\ \Delta y &= y_t - y_{t-1} \end{aligned} \tag{3.5}$$

Der endgültige Merkmalsvektor ist also $(r, \Delta\theta, \Delta y)$. Abbildung 3.5 zeigt die Entwicklung der Merkmale im Verlauf einer typischen Zeigegeste. Zu Beginn der Geste (B-Phase) steigt die Entfernung r der Hand zum Körper an. Die Höhenveränderung Δy ist zur gleichen Zeit negativ - die Hand bewegt sich also nach oben. Der negative Wert von $\Delta\theta$ zeigt eine Bewegung der Hand nach rechts an. Es folgt die Haltephase, in der alle drei Komponenten des Merkmalsvektors zur Ruhe kommen. Die Bewegung in der Ende-Phase verhält sich entgegengesetzt zu jener in der Beginn-Phase.

Hidden Markov Modelle verlassen sich bei der Modellierung von Zeiträumen auf äquidistante Merkmalsvektoren. Das Personenverfolgungssystem aus Kapitel 2

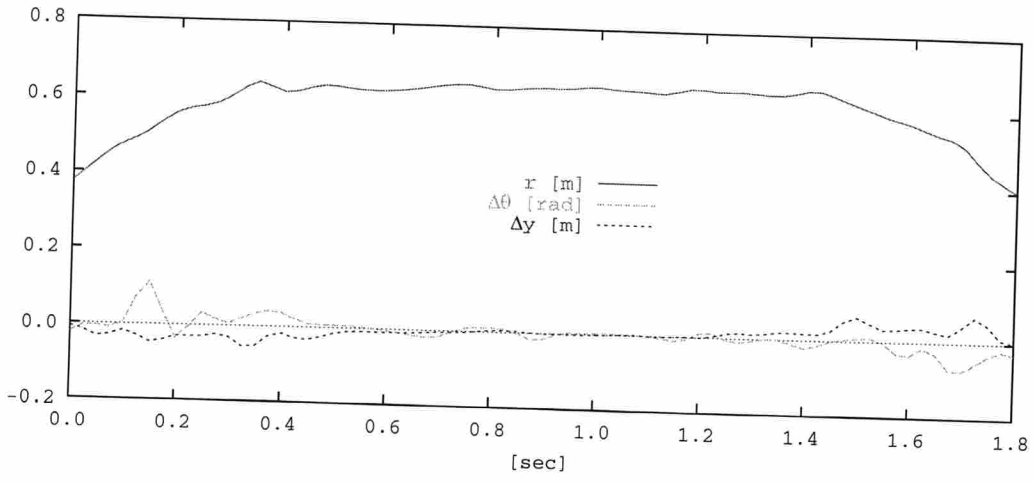


Abbildung 3.5: Entwicklung des Merkmalsvektors im Verlauf einer typischen Zeigegeste.

schwankt jedoch in seiner Leistung zwischen 10 und 20 Hz. In einem Nachbearbeitungsschritt wird die Merkmalssequenz daher durch eine Neuabtastung mittels *spline interpolation* auf konstant 40 Hz gebracht.

4 Bestimmung der Zeigerichtung

Jedes Mal nachdem eine Zeigegeste erkannt wurde, stellt sich die Frage, *wohin* (im Raum) gezeigt wurde. Mit dieser Information kann dann beispielsweise aus einer Reihe möglicher Zeigeziele das nächstliegende ausgewählt werden.

In der vorliegenden Arbeit werden zwei verschieden Ansätze verfolgt, um die Zeigerichtung zu gewinnen (siehe Abbildung 4.1). Der erste Ansatz basiert auf der Sichtlinie zwischen Auge (hier: Kopfmittelpunkt) und Hand, der zweite auf der Fortsetzung des Unterarms.

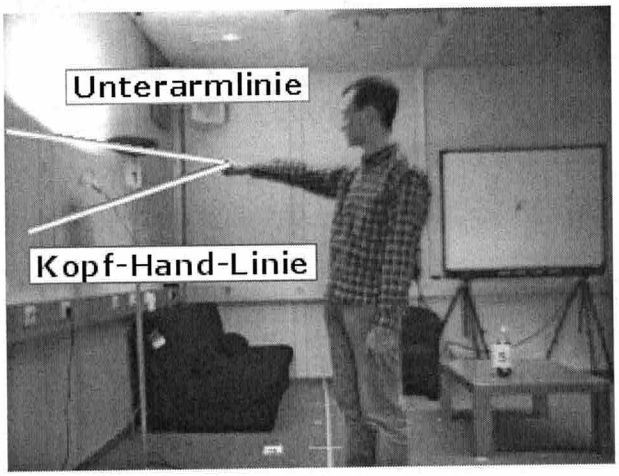


Abbildung 4.1: Zwei Ansätze zur Bestimmung der Zeigerichtung werden verfolgt: die Kopf-Hand-Line und die Unterarmlinie

Ein praktischer Vorteil der Kopf-Hand-Linie besteht darin, dass Kopf- und Handpositionen im Gegensatz zur Orientierung des Unterarms bereits vom Verfolgungssystem geliefert werden. Die Zeigerichtungsbestimmung mittels Unterarmlinie hingegen scheint intuitiv besser geeignet zu sein für den Fall, dass Zeigegesten mit abgewinkeltem Arm ausgeführt werden. Der folgende Abschnitt beschreibt den in dieser Arbeit gewählten Ansatz zur Bestimmung der Unterarmlage.

4.1 Messen der Unterarmlage

Geht man davon aus, dass die Hand während der Haltephase einer Zeigegeste frei im Raum steht, so ist der Unterarm das einzige Objekt, das sich innerhalb einer 40cm durchmessenden Kugel um die Hand befindet. Eine manuelle Überprüfung auf realen Testdaten kam zu dem Ergebnis, dass diese Vermutung im Allgemeinen zutreffend ist.

Betrachtet man alle 3D-Punkte $x_{1..n}$, die maximal 20cm vom Handmittelpunkt h entfernt sind und somit zur Hand selbst oder zum Unterarm gehören, dann ergibt sich ein Bild wie in Abbildung 4.2.

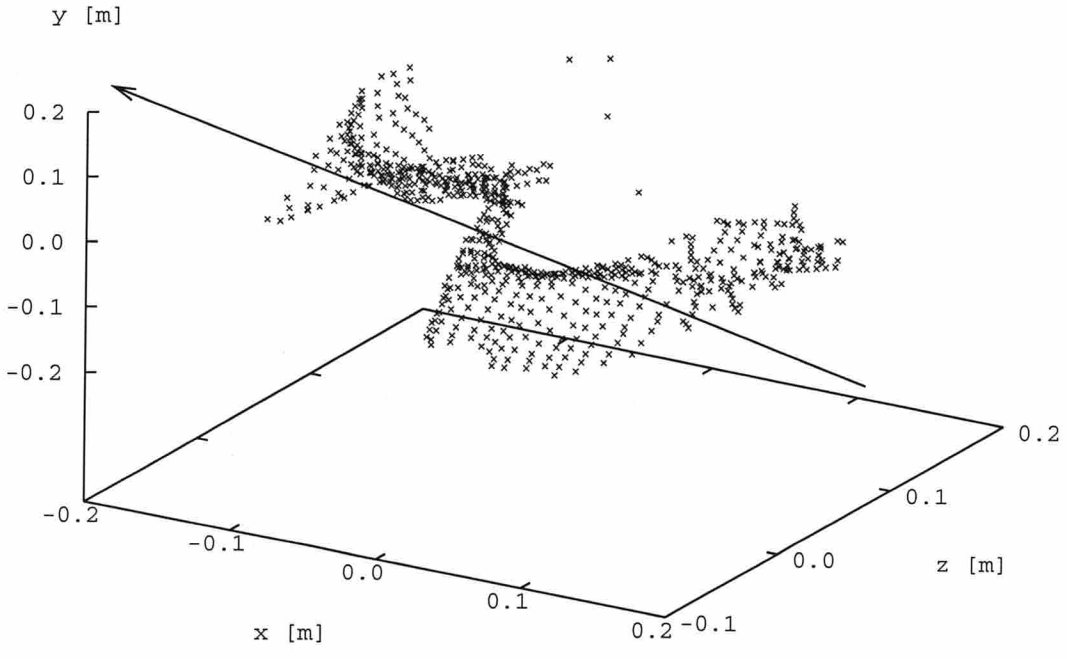


Abbildung 4.2: Die Richtung des Unterarms ergibt sich aus der ersten Hauptkomponente (Pfeil) der 3D-Punktwolke innerhalb einem 20cm-Radius um die Hand.

Da der Unterarm (inklusive Hand) ein längliches Objekt ist, ist seine Orientierung identisch mir der Richtung der größten Varianz der Punktwolke $x_{1..n}$. Diese Richtung lässt sich mithilfe einer Hauptkomponentenanalyse finden. Zunächst wird dazu die Kovarianzmatrix C der Punktwolke um den Handmittelpunkt aufgestellt:

$$C = \frac{1}{n} \sum_{i=1..n} (x_i - h)(x_i - h)^T \tag{4.1}$$

Die erste Hauptkomponente von C, also der Eigenvektor v_1 von C, welcher mit dem größten Eigenwert λ_{max} assoziiert ist, hat die gesuchte Richtung. Um daraus

die Zeigerichtung r zu gewinnen, muss das Vorzeichen von v_1 so gewählt werden, dass der Vektor nicht in Richtung Ellenbogen zeigt. Zu diesem Zweck wird der Mittelpunkt μ der Unterarm-Punktwolke berechnet, der erwartungsgemäß zwischen der Hand h und dem Ellenbogen liegt. Weist v_1 in Richtung von μ, dann gilt $r = -v_1$, ansonsten $r = v_1$.

Das hier beschriebene Verfahren zur Unterarmschätzung wurde inspiriert durch Jojic et al. [JBM+00]. Dort wird im Disparitätenbild ein ausgestreckter Arm mit einem iterativen Verfahren vom Körper getrennt.

4.2 Auswahl des Zeigeziels

Beide Ansätze zur Bestimmung der Zeigerichtung liefern als Ergebnis die Zeigerichtung r als Richtungsvektor relativ zur Handposition h. Die Messwerte von r und h werden über die gesamte Dauer der Haltephase gemittelt, um Störungen durch Ausreißer zu mindern.

Sei z die Position eines potenziellen Zeigeziels, so beträgt dessen Abstand d zur Zeigelinie:

$$d = \frac{|r \times (h - z)|}{|r|} \tag{4.2}$$

Ob nun ein bestimmter Abstand zur Zeigelinie zum Zweck der Annahme des Ziels als eher groß oder eher klein angesehen werden kann, hängt von der Entfernung des Ziels zur Hand ab. Aufschlussreicher als d selbst ist der Fehlerwinkel δ zwischen der gemessenen Zeigelinie und der idealen Linie zwischen Hand und Ziel (siehe Abbildung 4.3):

$$\delta = \arcsin \frac{d}{|z - h|} \tag{4.3}$$

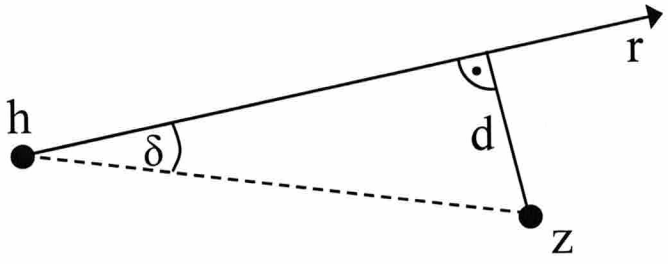


Abbildung 4.3: Der Fehlerwinkel δ ist der Winkel zwischen der *idealen* Linie durch Hand h und Zeigeziel z und der durch die Handposition sowie der Richtung r gegebenen extrahierten Zeigelinie.

Der Fehlerwinkel δ liegt in einem Intervall zwischen 0° und 90°. Für den Fall, dass die Zeigerichtung vom Ziel *weg* weist, wird δ in den Bereich zwischen 90° und 180° verschoben.

Für jedes der potenziellen Ziele wird δ berechnet. Ausgewählt wird jenes Ziel, welches den geringsten Fehlerwinkel aufweist. Wenn der Fehlerwinkel des besten Ziels einen festen Höchstbetrag δ_{max} übersteigt, kann kein Ziel ausfindig gemacht werden. Der Wert von δ_{max} ist anwendungsabhängig und sollte in jedem Fall $< 90°$ gewählt werden.

5 Experimente und Ergebnisse

Um die Leistung des im Rahmen dieser Arbeit entwickelten Systems zu messen, wurden in einem vorbereiteten Szenario Trainings- und Testdaten aufgenommen. Mithilfe dieser wurde die Genauigkeit der Zeigerichtung (Kopf-Hand-Linie im Vergleich zu Unterarmlinie), sowie die Qualität der Gestenerkennung evaluiert. Bei der Gestenerkennung wurde zwischen dem personenabhängigen und dem personenunabhängigen Fall unterschieden.

Die Evaluation erfolgte nach der *leave-one-out* Methode. Dabei wird beim Training der Modelle eine der Testsequenzen ausgelassen. Auf dieser Sequenz wird anschließend evaluiert. Dies geschieht reihum für jede Sequenz aus der Testmenge. Diese Methode gewährleistet, dass die Modelle nicht auf denselben Daten ausgewertet werden, auf denen sie zuvor trainiert wurden, als auch dass die Testdaten optimal ausgenutzt werden.

5.1 Testszenario

Um realistische und vergleichbare Daten zum Training der Modelle und zur Evaluation des System sammeln zu können, wurden in einem möblierten Raum (siehe Abbildung 5.1) mit 10 verschiedenen Testpersonen Aufnahmen gemacht.

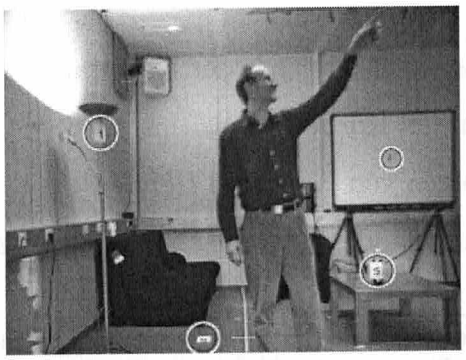


Abbildung 5.1: Das Testszenario aus der Kameraperspektive. Vier der acht markierten Zeigeziele sind im Bild sichtbar.

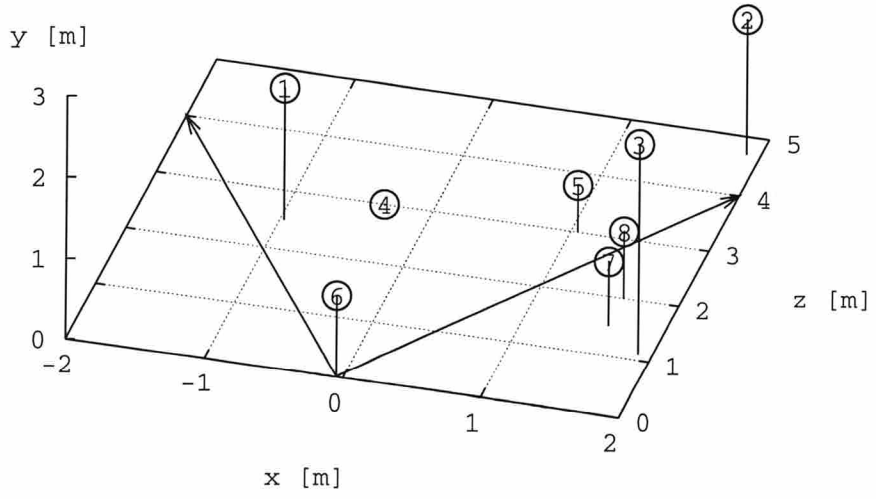


Abbildung 5.2: Positionen der Ziele im Raum. Ziel Nr. 6 ist identisch mit der Kamera. Die beiden Pfeile stehen für die Begrenzung des Kamerasichtfeldes, innerhalb dessen sich die Testpersonen bewegten.

Die Testpersonen wurden gebeten, sich vorzustellen, dass die auf einem Stativ montierte Kamera ein Roboter sei. Ihre Aufgabe bestand nun darin, sich frei im Gesichtsfeld des „Roboters“ zu bewegen und diesen hin und wieder mittels Zeigegesten auf eines der acht markierten Objekte hinzuweisen. Es wurde nicht vorgeschrieben, in welcher Art die Zeigegeste auszuführen sei. Abbildung 5.2 zeigt die Positionen der Ziele im Raum.

Insgesamt wurden 25 Videosequenzen mit einer Gesamtlänge von 24min aufgezeichnet. Jede Sequenz zeigt eine der 10 Testpersonen. In allen Sequenzen wurden die Gestenphasen manuell markiert. Die Gesamtzahl der ausgeführten Zeigegesten betrug 280. Davon mussten 74 aus der Testmenge entfernt werden, weil die Trajektorie der zeigenden Hand nicht vorhanden war. Hierfür gab es vier mögliche Ursachen:

- Verlassen des Erfassungsbereiches der Kamera,
- Verdeckung durch den Körper,
- Fehler beim Verfolgen von Kopf und Händen[1],
- Verschmelzung aufeinander folgender Zeigegesten zu einer einzigen Bewegung.

[1] „Normale“ Fehler beim Verfolgen wie (kurze) Aussetzer und Sprünge wurden toleriert. Gesten wurden nur dann aus der Testmenge entfernt, wenn überhaupt keine Trajektorien vorhanden waren.

5.2 Evaluation der Zeigerichtungsbestimmung

Um die Genauigkeit der gemessenen Zeigerichtung zu quantifizieren, werden zwei verschiedene Maße verwendet:

- Der Fehlerwinkel δ zwischen der extrahierten Zeigerichtung und der Ideallinie zum Ziel (siehe Abschnitt 4.2).
- Der Prozentsatz der Ziele, die mithilfe von δ korrekt zugeordnet werden können.

Abgesehen von den Fehlern, die beim Verfolgen von Kopf und Hand bzw. dem Messen der Unterarmlage entstehen, tragen auch die folgenden Faktoren zum Fehlerwinkel bei:

- Fehler des Gestenerkenners beim Lokalisieren der Haltephase,
- ungenaue Ausführung der Zeigegesten,
- Diskrepanzen zwischen den 3D-Koordinaten des Kamerasystems und den handvermessenen Koordinaten der Ziele im Raum.

In Abbildung 5.2 ist der Fehlerwinkel in Abhängigkeit von der (horizontalen) Position der Zeigehand dargestellt. Tabelle 5.1 zeigt die Ergebnisse der Auswertung in Abhängigkeit vom anvisierten Ziel.

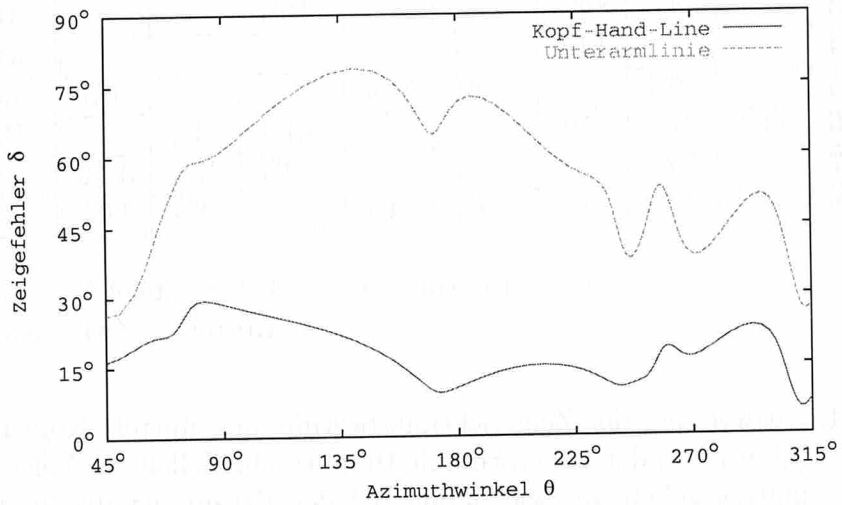


Abbildung 5.3: Zeigefehler δ in Abhängigkeit vom Azimuthwinkel θ der Handposition.

Ziel	1	2	3	4	5	6	7	8	δ
1	97%							3%	24,0°
2	5%	95%							12,6°
3			91%	5%				5%	21,1°
4				100%					14,4°
5				6%	94%				15,7°
6						100%			9,5°
7				4%	7%	4%	82%	4%	23,1°
8				4%	8%		16%	72%	20,3°

Kopf-Hand-Linie Fehlerwinkel δ: 17,3°
Korrektes Ziel: 90,4%

Ziel	1	2	3	4	5	6	7	8	δ
1	67%	20%				7%	7%		55,8°
2	5%	86%				9%			38,5°
3		4%	86%			10%			31,4°
4			12%	88%					34,3°
5		56%		6%	31%			6%	49,3°
6	11%	21%	7%			57%	4%		72,1°
7		19%		4%		4%	37%	37%	58,0°
8	4%	32%	4%	4%	4%		8%	44%	53,8°

Unterarmlinie Fehlerwinkel δ: 47,5°
Korrektes Ziel: 59,9%

Tabelle 5.1: Auswertung der Zeigerichtungsbestimmung mittels Kopf-Hand-Line (oben) und der Unterarmlinie (unten). Die Zellen der Verwechslungsmatrix geben an, wie häufig bei der Erkennung die in den Zeilen angegebenen Ziele mit den in den Spalten angegebenen Zielen verwechselt wurden. Die Hauptdiagonale enthält somit den Prozentsatz der Ziele, die korrekt zugeordnet wurden.

Diskussion der Ergebnisse

Das gute Ergebnis von 17° Abweichung bei Verwendung der Kopf-Hand-Linie zum Abschätzen der Zeigerichtung deutet darauf hin, dass sich Menschen beim Zeigen intuitiv an dieser Linie orientieren. In 90% der Fälle konnte hierbei unter den acht möglichen Zielen das korrekte Ziel identifiziert werden.

Das schlechtere Ergebnis von 48° Abweichung bei Verwendung der Unterarmlinie ist vermutlich sowohl auf ungenaue Messwerte[2], als auch darauf zurückzuführen, dass Menschen nicht direkt mit dem Unterarm auf das Ziel zeigen. Gesten mit stark abgewinkeltem Arm, bei denen die Unterarmlinie der Kopf-Hand-Linie überlegen sein könnte, wurden bei keiner Testperson beobachtet.

Die Untersuchung der Richtungsgenauigkeit in Abhängigkeit von der Handposition[3] zeigt, dass die Messung der Kopf-Hand-Linie desto besser funktioniert, je näher die Hand der Kamera ist. Grund hierfür dürfte sein, dass die Handposition genauer zu messen ist, je größer und somit deutlicher das Abbild der Hand ausfällt. Bei der Unterarmlinie zeigen sich gerade in diesem Bereich besondere Schwächen, da der Unterarm schlechter zu erkennen ist, je direkter er in Richtung Kamera ausgestreckt wird. Die Vermutung wird unterstützt, wenn man das Einzelergebnis von Ziele Nr. 6 betrachtet, bei dem es sich um die Kamera selbst handelt (vgl. Abbildung 5.2): hier erreicht die Kopf-Hand-Linie mit 10° ihr bestes und die Unterarmlinie mit 72° ihr schlechtestes Ergebnis.

5.3 Evaluation der Gestenerkennung

Die Leistung der Gestenerkennung wurde hinsichtlich zweier Maße untersucht:

- *Erkennungsrate* (recall): Anzahl der korrekt erkannten Gesten im Verhältnis zur Anzahl aller Gesten in der Testmenge.
- *Erkennungsgenauigkeit* (precision): Anzahl der korrekt erkannten Gesten im Verhältnis zur Anzahl aller erkannten Gesten inklusive der zu Unrecht erkannten Gesten (false positives).

Im personenunabhängigen Fall bestand die Testmenge aus 10 unterschiedlichen Personen, von denen jeweils eine Sequenz aufgenommen wurde. Im personenabhängigen Fall bestand die Testmenge aus 3 unterschiedlichen Personen, von denen jeweils 5 Sequenzen aufgenommen wurden.

[2]Im Vergleich zu den relativ stabilen Werten von Kopf und Hand ist die Unterarmmessung stark verrauscht.

[3]Die horizontale Handposition wird im Kopfkoordinatensystem durch ihren Azimuthwinkel ausgedrückt. 180° bedeuten, dass die Hand direkt der Kamera entgegen gestreckt wird.

Tabelle 5.3 sowie Abbildung 5.4 zeigen die Ergebnisse der Auswertung. Getrennt nach Personen sind Erkennungsrate, Erkennungsgenauigkeit, Fehlerwinkel der Richtungsbestimmung (mittels Kopf-Hand-Linie) und der Prozentsatz der korrekt zugeordneten Ziele (1 aus 8) angegeben. Die Durchschnittbildung erfolgte gewichtet nach der Anzahl der Gesten, die von den einzelnen Personen ausgeführt wurden. Die 3 Personen der personenabhängigen Testmenge haben auch an der personenunabhängigen Auswertung teilgenommen und sind in beiden Fällen mit den gleichen Nummern bezeichnet.

Tabelle 5.2 zeigt das Ergebnis der Auswertung aufgeschlüsselt nach Zielen. Hierzu wurden alle verfügbaren Sequenzen untersucht.

Ziel	1	2	3	4	5	6	7	8
Erkennungsrate	97%	100%	88%	38%	100%	93%	100%	96%

Tabelle 5.2: Auswertung der Gestenerkennung aufgeschlüsselt nach Zielen

Diskussion der Ergebnisse

Bei der Gestenerkennung zeigt sich sowohl im personenabhängigen als auch im personenunabhängigen Fall eine gleichermaßen hohe Erkennungsrate von etwa 88%. Im personenunabhängigen Fall ist die Anzahl der zu Unrecht erkannten Gesten größer, so dass eine Erkennungsgenauigkeit von 75% einem Wert von 90% im personenabhängigen Fall gegenüber steht. Die Auswahl des korrekten Zieles gelang in beiden Fällen mit einer Zuverlässigkeit von 90% bzw. 98%.

Die nach Zielen getrennte Auswertung zeigt, dass das Ziel Nr. 4 mit einer Rate von nur 38% außergewöhnlich schlecht erkannt wird. Grund dafür ist, dass es sich bei diesem Ziel um einen Punkt auf dem Fußboden handelt. Ein in Richtung Boden ausgestreckter Arm unterscheidet sich kaum von seiner „Ruheposition“, und die Bewegung der Hand auf diese Position ist nicht so signifikant wie beim Zeigen auf höher gelegene Ziele.

Bei der Beobachtung der Testsequenzen fällt auf, dass es bei der Ausführung von Zeigegesten tatsächlich individuelle Vorlieben gibt. Davon betroffen ist insbesondere die Dauer der Haltephase, die bei einigen Personen mehrere Sekunden betrug, wohingegen sie bei anderen so gut wie nicht vorhanden war. Die Anpassung an diesen individuellen Wert dürfte ein Grund dafür sein, weshalb im personenabhängigen Fall die Zeigerichtung mit 12° Abweichung genauer bestimmt werden konnte als im personenunabhängigen Fall, bei dem die Abweichung 21° betrug.

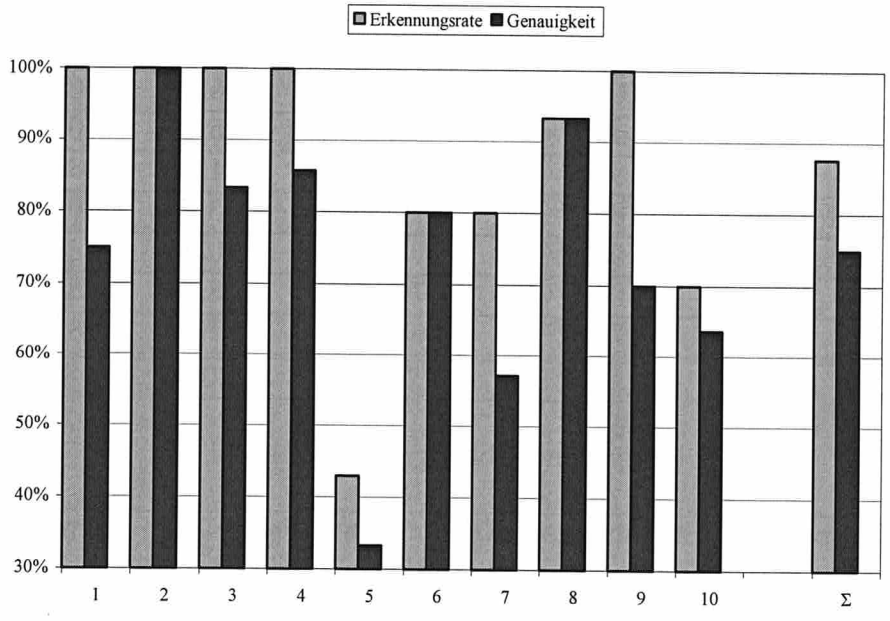
Testperson	Erkennungs-rate	Erkennungs-genauigkeit	Fehler-winkel	Korrektes Ziel
1	100,0%	75,0%	33,5°	75,0%
2	100,0%	100,0%	21,3°	100,0%
3	100,0%	83,3%	11,3°	100,0%
4	100,0%	85,7%	15,1°	100,0%
5	42,9%	33,3%	25,2°	100,0%
6	80,0%	80,0%	21,9°	87,5%
7	80,0%	57,1%	13,5°	100,0%
8	93,3%	93,3%	20,8°	85,7%
9	100,0%	70,0%	28,9°	85,7%
10	70,0%	63,6%	17,3°	85,7%
$\sum$	87.6%	75.0%	20.9°	89.7%

Personenunabhängige Evaluation

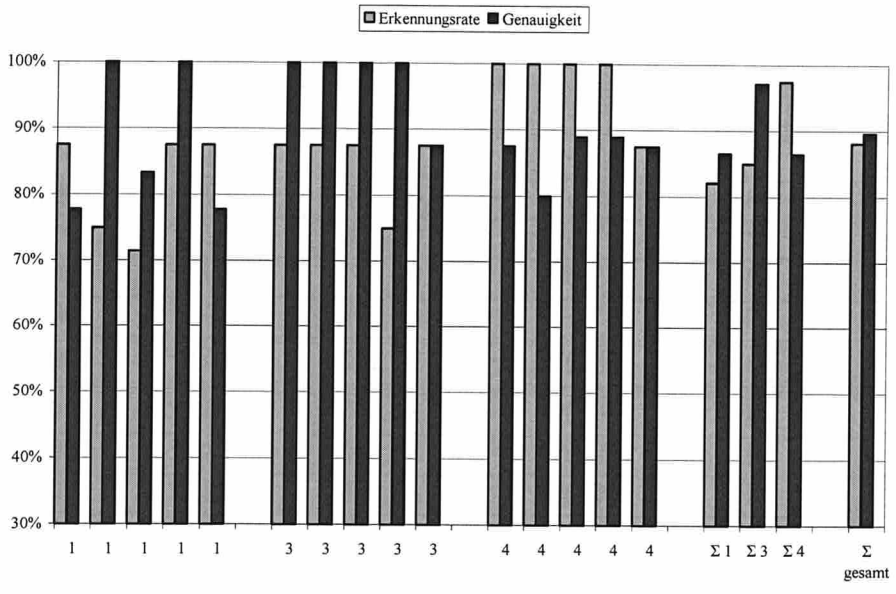
Testperson	Erkennungs-rate	Erkennungs-genauigkeit	Fehler-winkel	Korrektes Ziel
1	87.5%	77.8%	9.2°	100.0%
1	75.0%	100.0%	21.8°	83.3%
1	71.4%	83.3%	9.4°	100.0%
1	87.5%	100.0%	8.6°	100.0%
1	87.5%	77.8%	24.5°	85.7%
$\sum 1$	82.1%	86.5%	14.3°	93.7%
3	87.5%	100.0%	14.0°	100.0%
3	87.5%	100.0%	10.7°	100.0%
3	87.5%	100.0%	12.3°	100.0%
3	75.0%	100.0%	10.0°	100.0%
3	87.5%	87.5%	7.6°	100.0%
$\sum 3$	85.0%	97.1%	10.9°	100.0%
4	100.0%	87.5%	16.4°	100.0%
4	100.0%	80.0%	13.8°	100.0%
4	100.0%	88.9%	10.7°	100.0%
4	100.0%	88.9%	10.7°	100.0%
4	87.5%	87.5%	9.3°	100.0%
$\sum 4$	97.4%	86.4%	12.1°	100.0%
$\sum gesamt$	88.1%	89.6%	12.4°	98.1%

Personenabhängige Evaluation

Tabelle 5.3: Auswertung der Gestenerkennung in Abhängigkeit von der Testperson (siehe auch Abbildung 5.4)



Personenunabhängige Evaluation



Personenabhängige Evaluation

Abbildung 5.4: Auswertung der Gestenerkennung in Abhängigkeit von der Testperson (siehe auch Tabelle 5.3)

6 Zusammenfassung und Ausblick

Im Rahmen der vorliegenden Arbeit wurde ein System entwickelt, das in der Lage ist, mithilfe einer Stereokamera menschliche Zeigegesten zu erkennen und die Zeigerichtung im Raum zu bestimmen. Das System besteht aus zwei Komponenten, die getrennt voneinander betrachtet werden können: der 3D-Personenverfolgung und der eigentlichen Gestenerkennung.

6.1 Personenverfolgung

Das Verfolgen einer Person gelingt in der Testumgebung zuverlässig in Echtzeit (10 bis 20 Messwerte pro Sekunde). Es ist nicht notwendig, eine vorgegebene Startposition einzunehmen oder andere manuelle Initialisierungen vorzunehmen. Während es gelingt, die Kopfposition durchweg stabil zu messen, kommt es beim Verfolgen der Hände hin und wieder zu Aussetzern oder zu Verwechslungen mit anderen Objekten bzw. der Hände untereinander.

Das hier beschriebene System entwickelt zu jedem Zeitpunkt eine einzige – die jeweils beste – Hypothese. Ein Problem dieser Vorgehensweise besteht darin, dass hin und wieder (z. B. durch kurzzeitige Verdeckung) eine zutreffende Hypothese verworfen und durch eine unzutreffende ersetzt wird. Da große Sprünge im Hypothesenraum durch das Übergangsmaß nach Möglichkeit verhindert werden, kann es passieren, dass an der unzutreffenden Hypothese auch dann noch festgehalten wird, wenn die ursprüngliche längst wieder gebildet werden könnte. Eine Verbesserung dieser Situation könnte durch *Multi-Hypothesen-Tracking* erreicht werden. Dabei werden zu jedem Zeitpunkt mehrere verschiedene Hypothesen aufgestellt und gespeichert. Rückwirkend wird dann der optimale Pfad durch alle Hypothesen gesucht. Eine der Herausforderungen solcher Verfahren besteht darin, sie echtzeitfähig zu realisieren.

Obwohl die Abhängigkeit von der Beleuchtungssituation durch das hier beschriebene adaptive Verfahren vermindert werden konnte, bleiben grundsätzliche Probleme mit dem Merkmal Hautfarbe bestehen: Zum einen gibt es Oberflächen,

die farblich nicht von Haut zu unterscheiden sind, zum anderen ist es erforderlich, dass Kopf und Hände die einzigen unbekleideten Körperteile sind. Um die Abhängigkeit der Verfolgung von der Farbe zu reduzieren, wäre es wünschenswert, (zusätzlich) alternative Merkmale wie z. B. Kanten- oder Disparitätenbilder verwenden zu können.

Eine sinnvolle Erweiterung des Programmes wäre das gleichzeitige Verfolgen mehrerer Personen. Zusätzlich zu der linearen Steigerung des Aufwands durch die Erhöhung der Anzahl an Personen werden dabei auch neue Probleme wie z. B. die Zuordnung von Händen zu Köpfen zu lösen sein.

6.2 Gestenerkennung

Die Erkennung von Zeigegesten gelang in einem personenunabhängigen Testszenario mit einer Erkennungsrate (recall) von 88% und einer Erkennungsgenauigkeit (precision) von 75%. Im personenabhängigen Fall verbesserte sich die Genauigkeit bei gleichbleibender Erkennungsrate auf 90%.

Mithilfe der Kopf-Hand-Line konnte die Zeigerichtung im Raum durchschnittlich auf 17° genau bestimmt werden, wodurch in 90% der Fälle das richtige Zeigeziel (1 aus 8) identifiziert wurde. Das alternative Verfahren zur Bestimmung der Zeigerichtung anhand der Unterarmlinie erreichte nur eine mittlere Genauigkeit von 48° und 60% Zielerkennung.

In diesem Zusammenhang wäre es interessant, herauszufinden, inwieweit das Ergebnis der Unterarmlinie auf fehlerhafte Messwerte zurückzuführen ist. Zu diesem Zweck könnten die visuell gewonnenen Messwerte mit gleichzeitig aufgenommenen Messwerten eines am Unterarm befestigen magnetischen Lagesensors[1] verglichen werden.

Bei der Beobachtung der Testaufnahmen fällt auf, dass Menschen in der Regel in die Richtung blicken, in die sie zeigen. Um Erkennungsleistung und Zuverlässigkeit der Gestenerkennung zu steigern, könnten daher neben der Trajektorie der Hand weitere Merkmale wie Blickrichtung oder auch Kopf- bzw. Oberkörperdrehung in die Modellierung aufgenommen werden.

Der Nutzen eines Systems zur Zeigegestenerkennung entfaltet sich erst dann vollständig, wenn es in einen übergeordneten multimodalen Dialog eingebettet ist. Dadurch würde es beispielsweise möglich, sprachliche Platzhalter betreffend eines Objektes oder Ortes durch eine parallel ausgeführte Zeigegeste auszufüllen

[1]Kommerziell erhältliche magnetische Sensorsysteme (wie z. B. der Flock-of-Birds Tracker der Ascension Technology Corporation, Burlington, VT, USA) haben eine räumliche Auflösung im Sub-Zentimeter-Bereich – wenn auch nur innerhalb eines eingeschränkten Operationsradius um den Magnetfeld-Emitter.

(„Schalte *diese* Lampe an!“ oder „Stelle die Tasse *dort* ab!“). Gleichzeitig könnte die Zuverlässigkeit der Gestenerkennung durch Fusion mit sprachlichen Merkmalen (z. B. bestimmten Schlüsselwörtern) gesteigert werden.

A Stereobildverarbeitung

Durch die Kombination der Bilder zweier Kameras ist es möglich, die Position eines Objektes im dreidimensionalen Raum zu bestimmen, wenn das Objekt von beiden Kameras gleichzeitig erfasst wird. Die Berechnung von 3D-Koordinaten für jeden Bildpunkt erfolgt in Echtzeit. Sie gelingt nur für solche Bildpunkte, die Teil einer deutlich texturierten Oberfläche sind und sich innerhalb eines bestimmten Entfernungsbereiches befinden.

In der vorliegenden Arbeit wurde zur Stereobildverarbeitung die SVS-Bibliothek eingesetzt, die von der Firma *Videre Design*[1] vertrieben wird. Angaben zur Funktionsweise wurden dem Handbuch entnommen.

A.1 Stereogeometrie

Abbildung A.1 zeigt zwei parallel ausgerichtete Kameras mit den Brennpunkten L und R, deren Bildebenen in einer gemeinsamen Ebene eingebettet liegen. Der horizontale Abstand der Kameras wird als Grundlinie b bezeichnet, f steht für die Brennweite. Beide Kameras beobachten ein Objekt im Punkt P, das sich in der Entfernung z zur Basislinie befindet. Als horizontale Bildkoordinate[2] von P ergibt sich für die linke Kamera der Wert x_L und für die rechte Kamera der Wert x_R. Der Abstand $d = x_L - x_R$ wird als *(horizontale) Disparität* (siehe [Jäh97]) bezeichnet. Mithilfe des Strahlensatzes ergibt sich für die Entfernung z die Beziehung

$$z = \frac{f \cdot b}{d} \tag{A.1}$$

Ist der Abstand z bekannt, lassen sich auch die x- und y-Koordinaten von P im Raum berechnen. In einem Koordinatensystem, dessen Ursprung im Brennpunkt der linken Kamera liegt, gilt:

$$x = \frac{x_L \cdot z}{f}, \quad y = \frac{y_L \cdot z}{f} \tag{A.2}$$

[1] Videre Design, 865 College Ave, Menlo Park, CA 94025, USA. http://www.videredesign.com

[2] Der Nullpunkt befindet sich in der Bildmitte.

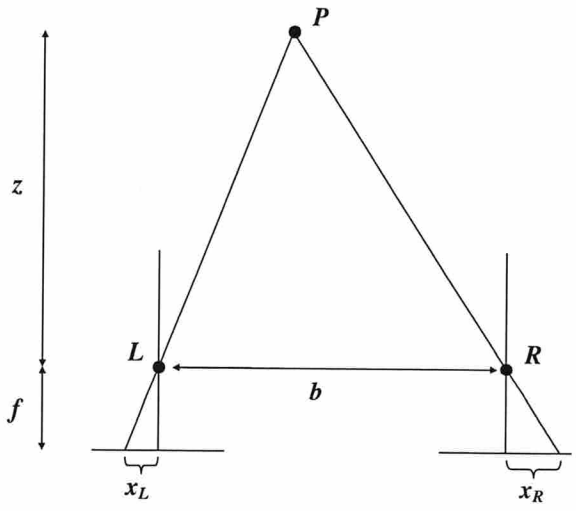


Abbildung A.1: Zwei Kameras mit den Brennpunkten L und R beobachten ein Objekt im Punkt P. Aus der Differenz der Bildkoordinaten x_L und x_R lässt sich die Entfernung z berechnen.

Die Entfernung ist umgekehrt proportional zur Disparität, eine Disparität von 0 steht für unendliche Entfernung. Je näher ein Objekt den Kameras kommt, desto stärker verschieben sich seine Abbilder im linken und im rechten Kamerabild gegeneinander, und die Disparität steigt. Die Disparität kann einen bestimmten Wert nicht überschreiten, da ein näher kommendes Objekt irgendwann den Erfassungsbereich mindestens einer der Kameras verlässt. Um die Tiefeninformation nicht zu verlieren, muss also ein Mindestabstand eingehalten werden. Die Ebene, die den nutzbaren Bereich begrenzt, wird als *Horopter* bezeichnet.

A.2 Disparitätenbild

Das Disparitätenbild (*dense disparity map*) ist ein Graustufenbild, in dem der Helligkeitswert eines Punktes für die Disparität steht, die die beiden korrespondierenden Punkte aus dem linken und rechten Bild zueinander aufweisen. Zur Erstellung des Disparitätenbildes ist es also notwendig, in beiden Kamerabildern jene Teile zu identifizieren, die Abbildungen desselben Objekts sind (*Korrespondenzproblem*).

Es gibt verschiedene Möglichkeiten, Korrespondenzen zu finden, die sich hinsichtlich des Berechnungsaufwands und der Genauigkeit unterscheiden. Die hier zur Disparitätenberechnung eingesetzte SVS-Bibliothek[3] geht nach [Kon97] folgendermaßen vor:

[3] Zusätzliche Informationen über den SVS-Stereoalgorithmus finden sich im SVS User's Manual sowie unter http://www.ai.sri.com/~konolige/

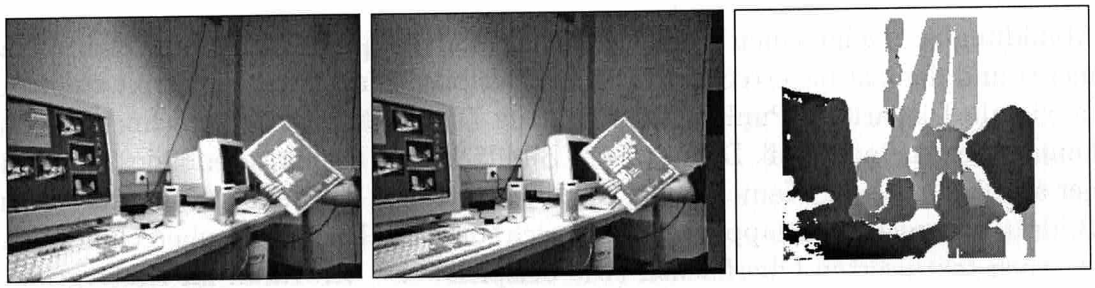


Abbildung A.2: Linkes und rechtes Kamerabild, sowie das daraus errechnete Disparitätenbild

Die Graustufenbilder beider Kameras werden zunächst mit einem Laplacian-of-Gaussian-Filter (siehe [Hor86]) behandelt, um die Kanten hervorzuheben.

Um festzustellen, ob es sich bei einem Punkt im linken und einem Punkt im rechten Bild um Abbildungen desselben Objekts handelt, wird ein quadratisches Suchfenster mit 5-13 Punkten Breite um jeden der Punkte gelegt. Von den Helligkeitswerten der entsprechenden Punkte unter den Fenstern wird die (absolute) Differenz gebildet. Die Summe über alle Differenzen ist ein Maß für die Korrelation der Bildteile unter den Fenstern: je geringer die Summe der Differenzen, desto größer die Korrelation.

Für jedes Fenster im linken Bild wird das maximal korrelierende Fenster im rechten Bild gesucht. Dazu werden die Suchfenster über alle Punkte des linken und des rechten Bildes geschoben. Da die Kameras nur in horizontaler Richtung voneinander versetzt angebracht sind, können beide Suchfenster immer auf der gleichen vertikalen Position bleiben und müssen nur horizontal gegeneinander verschoben werden.

Hat man die maximale Korrelation für ein Fenster um einen Punkt im linken Bild gefunden, so ergibt sich die Disparität dieses Punktes aus dem horizontalen Abstand der korrelierenden Fenster. Zur Begrenzung der Rechenzeit kann der horizontale Suchbereich und somit die maximal mögliche Disparität beschränkt werden.

In einem Nachbearbeitungsschritt werden die Disparitätenwerte an jenen Stellen im Bild wieder entfernt, an denen die Stärke der sichtbaren Textur ein gegebenes Konfidenzmaß nicht erfüllt (confidence filter).

Die Kameras sehen das Bild aus leicht unterschiedlichen Blickwinkeln. Dadurch entstehen an Objekträndern Bereiche, die nur von einer Kamera gesehen werden, was zu fehlerhaften Disparitätenwerten führt. Diese Bereiche können gefunden werden, indem bei der Suche nach Korrelation zunächst das Bild der linken Kamera und daraufhin das Bild der rechten Kamera als fester Ausgangspunkt verwendet wird. Unterscheiden sich an einem Punkt die beiden so gefundenen Disparitätenwerte, so werden sie entfernt (left-/right filter).

Abbildung A.2 zeigt einen Frame aus einer Videosequenz (linke und rechte Kamera) und das daraus errechnete Disparitätenbild. Ein schwarzer Punkt steht für maximale Disparität. Punkte, für die keine Disparitätenwerte errechnet werden konnten, erscheinen weiß. Die nutzbare Bildfläche des Disparitätenbildes ist geringer als die Größe der Kamerabilder, da die Sichtfelder der beiden Kameras an den Bildrändern nicht überlappen. Es zeigt sich, dass die Disparitätenberechnung auf schwach texturierten Oberflächen (wie beispielsweise der Wand im Hintergrund) nicht gelingt. Flächen außerhalb des Horopters (im Beispielbild die linke untere Ecke) zerfallen in zufällige Disparitätenwerte.

A.3 Kalibrierung

Die bisherigen Ausführungen gingen von vollkommen parallel ausgerichteten Lochkameras aus, wie es sie in der Realität nicht gibt. Durch Linsenverzerrungen, kleine Unterschiede in der Brennweite und Abweichungen von der idealen Ausrichtung entstehen fehlerhafte Bilder, die die korrekte Berechnung von Disparitäten verhindern. Daher ist es notwendig, die Kameraanordnung vor Gebrauch zu *kalibrieren.*

Durch die Kalibrierung werden zwei Arten von Parametern gewonnen, so genannte *innere* und *äußere* Parameter. Deren Kenntnis ermöglicht es, die Bilder so zu transformieren, dass sie den Bildern idealer Kameras entsprechen. Zu den inneren Parametern (intrinsics) einer Kamera gehören Brennweite, Linsendezentrierung, radialsymmetrische Verzerrung und das Pixel-Seitenverhältnis des Bildsensors. Die äußeren Parameter (extrinsics) beschreiben die räumliche Anordnung beider Kameras – insbesondere den Grundlinienabstand und die Abweichung der optischen Achsen von der Parallelen.

Die verwendete SVS-Bibliothek unterstützt die Kalibrierung mithilfe eines Schachbrettmusters, das aus fünf verschiedenen Perspektiven aufgenommen wird. Aus diesen Bildern werden die oben genannten Parameter extrahiert, um sie später im Stereo-Algorithmus zur Korrektur der Kamerabilder zu verwenden. Eine kurze Beschreibung des Verfahrens liefert das SVS-Handbuch.

B Hidden Markov Modelle

Ein Hidden Markov Modell (HMM) beschreibt einen stochastischen Prozess, der aus zwei gekoppelten Teilen besteht: Eine endliche Menge von Zuständen (Markov-Kette) wird Schritt für Schritt durchlaufen. In jedem Zustand wird mit einer vom jeweiligen Zustand abhängigen Emissionswahrscheinlichkeit ein Ausgabesymbol produziert. Ein Beobachter sieht nur die dabei entstehende Sequenz von Ausgabesymbolen, nicht jedoch die tatsächliche Abfolge der Zustände.

Eine praxisorientierte Einführung in Hidden Markov Modelle und die mit ihnen in Verbindung stehenden Algorithmen gibt Rabiner in [Rab89]. Die im folgenden verwendeten Formeln basieren auf dieser Quelle.

Ein HMM λ (wie in Abbildung B.1 symbolisiert) wird durch folgende 5 Elemente charakterisiert:

- Die Anzahl der Zustände N.
- Die Matrix $A = \{a_{ij}\}$ der Übergangwahrscheinlichkeiten von Zustand i zu Zustand j:
$$a_{ij} = P(q_t = s_j | q_{t-1} = s_i)$$
Für jeden Zustand i gilt: $\sum_{j=1}^{N} a_{ij} = 1$.
- Die Verteilung der Anfangswahrscheinlichkeiten π_i mit $\sum_{i=1}^{N} \pi_i = 1$.
- Die Menge der Ausgabesymbole $O = \{v_1, \ldots, v_K\}$.
- Die Ausgabewahrscheinlichkeiten $b_j(k)$, die für jeden Zustand j festlegen, welches Symbol mit welcher Wahrscheinlichkeit emittiert wird:
$$b_j(k) = P(a_t = v_k | q_t = s_j),$$

Besteht das Ausgabealphabet aus einer endlichen Menge von Symbolen, so bezeichnet man das HMM als *diskret*. Handelt es sich bei den Beobachtungen hingegen um kontinuierliche Werte, müssen die Ausgabewahrscheinlichkeiten als (kontinuierliche) Wahrscheinlichkeitsdichtefunktionen formuliert werden. In diesem Fall wird auch das HMM als *kontinuierlich* bezeichnet. Ein gebräuchlicher Ansatz für

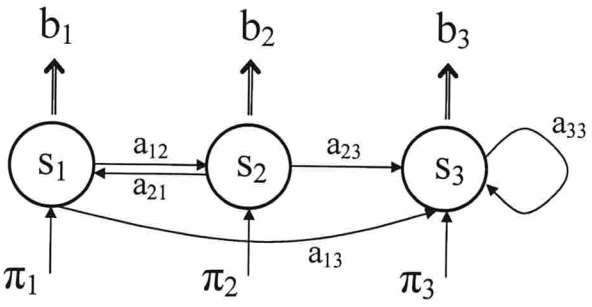


Abbildung B.1: Ein HMM bestehend aus drei Zuständen s_1, s_2 und s_3. Die Übergänge erfolgen mit den Wahrscheinlichkeiten a_{ij}. Die Wahrscheinlichkeit, dass s_i der Anfangszustand ist, wird durch π_i ausgedrückt. In jedem Zustand wird abhängig von der Ausgabewahrscheinlichkeit $b_i(k)$ ein Symbol des Ausgabealphabets produziert.

die Modellierung kontinuierlicher Ausgabewahrscheinlichkeiten ist eine Mixtur aus gauß'schen Normalverteilungen:

$$b_j(O) = \sum_{m=1}^{M} c_{jm} N(O, \mu_{jm}, C_{jm})$$

Jede der Normalverteilungen N ist durch ihren Mittelwert μ_{jm} und ihre Kovarianzmatrix c_{jm} gegeben. Bei O handelt es sich um den Ausgabevektor, c_{jm} ist das Gewicht, mit dem die m-te Normalverteilung zur Mischverteilung beiträgt (Mixturgewicht).

Der Forward-Algorithmus

Verfügt man über ein gegebenes HMM λ und eine Beobachtungssequenz $O = O_1 O_2 \ldots O_T$, so stellt sich bei der Klassifikation die Frage, wie hoch die Wahrscheinlichkeit $P(O|\lambda)$ ist, mit der O von λ produziert wurde. Eine effiziente Lösung für dieses Problem stellt der *Forward*-Algorithmus dar. Dabei handelt es sich um ein rekursives Verfahren, in dem eine Matrix der so genannten Forward-Variablen α aufgebaut wird:

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

Bei $\alpha_t(i)$ handelt es sich also um die Wahrscheinlichkeit, dass die Teilsequenz $O_1 O_2 \ldots O_t$ beobachtet wurde und das Modell sich zum Zeitpunkt t im Zustand s_i befindet. Die Berechnung von $\alpha_t(i)$ geht nach folgendem Prinzip vonstatten:

1. Initialisierung

$$\alpha_1(i) = \pi_i b_i(O_1), \qquad 1 \leq i \leq N$$

2. Induktion

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_j(O_{t+1}), \qquad 1 \leq j \leq N$$
$$1 \leq t \leq T-1$$

3. Terminierung

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

Baum-Welch Parameterschätzung

Es stellt sich nun die Frage nach dem Training der HMMs: Wie müssen die Parameter a_{ij}, c_{jk}, μ_{jk} und C_{jk} eines Modells λ eingestellt werden[1], so dass die Beobachtungswahrscheinlichkeit $P(O|\lambda)$ für eine gegebene Beobachtungssequenz O maximiert wird? Mit den Baum-Welch-Formeln (siehe [Rab89]) zur Parameterschätzung steht ein iteratives Verfahren zur Lösung dieses Problems zur Verfügung.:

$$\begin{aligned}
\overline{a}_{ij} &= \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T} \sum_{j=1}^{N} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)} \\
\overline{\mu}_{jk} &= \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot O_t}{\sum_{t=1}^{T} \gamma_t(j,k)} \\
\overline{C}_{jk} &= \frac{\sum_{t=1}^{T} \gamma_t(j,k) \cdot (O_t - \mu_{jk})(O_t - \mu_{jk})'}{\sum_{t=1}^{T} \gamma_t(j,k)} \\
\overline{c}_{jk} &= \frac{\sum_{t=1}^{T} \gamma_t(j,k)}{\sum_{t=1}^{T} \sum_{k=1}^{M} \gamma_t(j,k)}
\end{aligned}$$

Bei $\gamma_t(j,k)$ handelt es sich um eine Hilfsvariable der Form:

$$\gamma_t(j,k) = \left[\frac{\alpha_t(j)\beta_t(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}\right] \left[\frac{c_{jk} N(O_t, \mu_{jk}, C_{jk})}{\sum_{m=1}^{M} c_{jm} N(O_t, \mu_{jm}, C_{jm})}\right] \tag{B.1}$$

Die so genannte Backward-Variable $\beta_t(i)$ wird analog zur Forward-Variablen α gebildet und steht für die Wahrscheinlichkeit, dass die restliche Beobachtung $O_{t+1} \ldots O_T$ gemacht wird, wenn sich λ zur Zeit t in Zustand i befindet.

[1] Die π_j werden nicht neu geschätzt, da bei den in dieser Arbeit verwendeten Modellen immer nur der erste Zustand s_1 als Startzustand erlaubt ist.

Beim Training mit mehreren Beobachtungssequenzen $O = [O^{(1)}, O^{(2)}, \ldots, O^{(k)}]$ wird versucht, die gemeinsame Beobachtungswahrscheinlichkeit zu maximieren:

$$P(O|\lambda) = \prod_k P(O^{(k)}|\lambda)$$

Skalierung (Scaling)

Bei der Implementierung des Baum-Welch-Verfahrens tritt das Problem auf, dass durch die wiederholte Multiplikation kleiner Zahlen (die Wahrscheinlichkeitswerte sind allesamt ≤ 1) der Darstellungsbereich von Gleitkommazahlen verlassen wird. Um dem entgegen zu wirken, werden die Werte mit Skalierungsfaktoren im darstellbaren Bereich gehalten. Bei der späteren Auswertung mithilfe des Forward-Algorithmus müssen die Auswirkungen dieser Faktoren neutralisiert werden. Als Nebeneffekt der Skalierung wird beim Forward-Algorithmus $logP(O|\lambda)$ anstelle von $P(O|\lambda)$ berechnet.

C Implementierung

Das im Rahmen dieser Arbeit entwickelte System zur Zeigegestenerkennung besteht aus einer Stereokamera und einem PC, der die Kamerabilder verarbeitet und die Ergebnisse visualisiert. Als Kamera kommt ein *Mega-D Digital Stereo Head* der Firma Videre Design[1] zum Einsatz, dessen zwei Farbvideokameras über eine gemeinsame IEEE-1394-Schnittstelle (*FireWire*) mit dem PC verbunden werden. Der Kameraabstand beträgt 9cm. Die Videobilder der linken und rechten Kamera werden synchron aufgenommen und haben ursprünglich eine Auflösung von 1288x1032 Punkten, werden hier aber aus Geschwindigkeitsgründen auf eine Größe von 320x240 Punkten reduziert. Auf einem PC mit 2.4GHz Pentium 4 Prozessor wird bei Betrieb des Gesamtsystems eine Framerate von etwa 10 Bildern pro Sekunde erreicht.

Die Software wurde in C++ unter Verwendung von Microsoft Visual C++ V6.0 implementiert und unter Microsoft Windows 2000 ausgeführt. Dank der Verwendung plattformübergreifender Bibliotheken sind bei einer eventuellen Portierung (insbesondere auf Linux) keine prinzipiellen Schwierigkeiten zu erwarten.

C.1 Aufbau des Systems

Das entwickelte Programm besteht aus drei Teilen: dem Personenverfolgungssystem, der Gestenerkennung und einer globalen Ablaufsteuerung. Die Teile sind jeweils aus mehreren Modulen zusammengesetzt. Abbildung C.1 zeigt schematisch den Aufbau des Systems. Jedes Modul verfügt über einen eigenen Bereich innerhalb der graphischen Benutzeroberfläche, in dem seine Parameter eingestellt und (Zwischen-)Ergebnisse visualisiert werden können.

Personenverfolgung

Aufgabe der Personenverfolgung ist es, aus den eingehenden Videosequenzen die Positionen von Kopf und Händen zu extrahieren. Sie setzt sich aus den folgenden fünf Modulen zusammen:

[1] Videre Design, 865 College Ave, Menlo Park, CA 94025, USA. http://www.videredesign.com

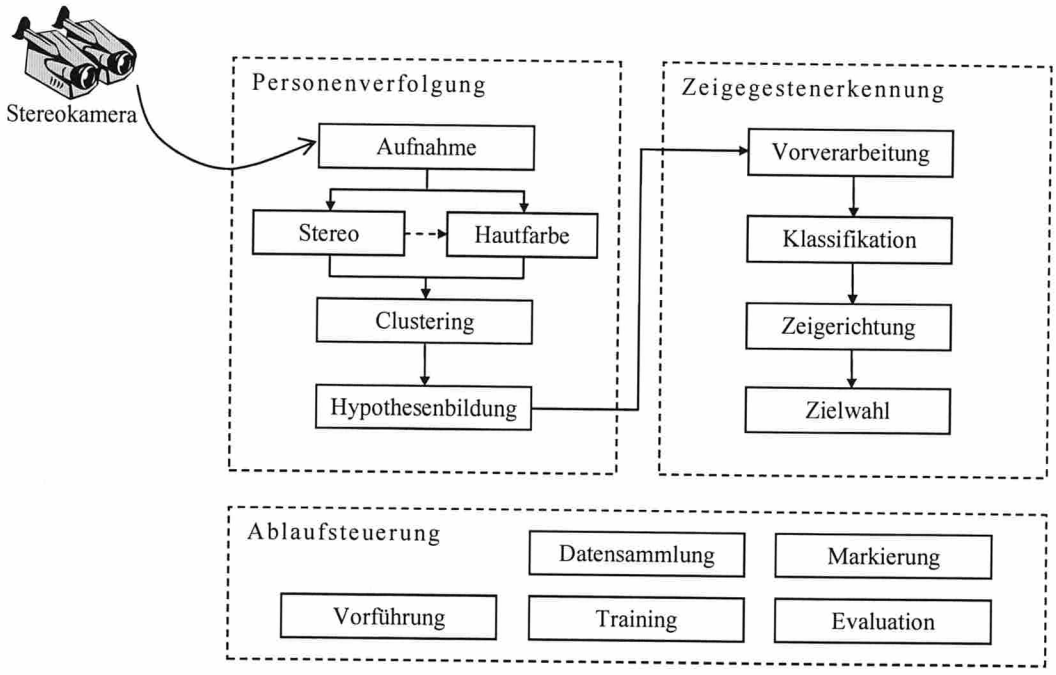


Abbildung C.1: Überblick über die Komponenten des Systems

- Das *Aufnahme*-Modul nimmt die Bildpaare der angeschlossenen Kamera entgegen. Diese Bildpaare können abgespeichert werden, um sie später anstelle der Kamera als Bildquelle zu verwenden.
- Das Modul *Stereo* berechnet das Disparitätenbild sowie die räumlichen Koordinaten für die einzelnen Bildpunkte (siehe Anhang A).
- Im Modul *Hautfarbe* werden die eingehenden Farbbilder mithilfe eines Hautfarbmodells klassifiziert. Dieses Modell wird mithilfe des Disparitätenbildes ständig an die Lichtverhältnisse angepasst (siehe Abschnitt 2.1).
- Im *Clustering*-Modul werden die Hautfarbpunkte mit ihren 3D-Koordinaten verknüpft und in einem kombinierten Verfahren räumlich geballt (siehe Abschnitt 2.2).
- Die *Hypothesenbildung* hat die Aufgabe, zu jedem Zeitpunkt ausgehend von den vorliegenden Hautfarb-Clustern die wahrscheinlichsten Positionen von Kopf und Händen auszuwählen (siehe Abschnitt 2.3).

Gestenerkennung

Die Gestenerkennung arbeitet auf den Positionsdaten, die die Personenverfolgung liefert. Sie detektiert das Vorkommen einer Zeigegeste und bestimmt deren Ziel im Raum. Sie besteht aus den folgenden vier Modulen:

- In der *Vorverarbeitung* werden die Trajektorien von Kopf und Händen durch Interpolation auf eine konstante Abtastrate gebracht. Anschließend findet die Transformation der Positionsdaten in den Merkmalsvektor statt (siehe Abschnitt 3.5).
- Das Modul *Klassifikation* klassifiziert mithilfe von HMMs (siehe Abschnitt 3.3) kontinuierlich die Merkmalssequenz, um eine Zeigegeste direkt nach ihrer Ausführung zu erkennen.
- Wurde eine Geste erkannt, wird im Modul *Zeigerichtung* mithilfe der in Kapitel 4 beschriebenen Verfahren die Zeigerichtung als Linie im Raum bestimmt.
- Das Modul *Zielwahl* verwaltet eine Liste möglicher Zeigeziele und ordnet einer erkannten Geste das am besten passende Ziel zu (siehe Abschnitt 4.2). Abhängig vom gewählten Ziel können auch Aktionen angestoßen werden. Implementiert ist das Ein- und Ausschalten eines elektrischen Gerätes über eine angeschlossene Steckdose.

Ablaufsteuerung

Die Ablaufsteuerung kontrolliert den Programmablauf und den Datenfluss durch die Verfolgungs- und Gestenerkennungsmodule. Sie belegt die Parameter der Module und kombiniert deren Ein- und Ausgaben so, dass eine von fünf verschiedenen Betriebsarten realisiert wird:

- In der Betriebsart *Datensammlung* ist nur das Modul *Aufnahme* aktiv, das die aufgenommenen Bildsequenzen ohne weitere Bearbeitung zur späteren Verwendung abspeichert.
- Im Modul *Markierung* wird dem Benutzer die Möglichkeit gegeben, durch die Einzelbilder einer aufgenommenen Sequenz zu blättern und Markierungen (Kopf- und Handpositionen, Gestenphasen, Zeigeziele) zu setzen.
- Eine oder mehrere markierte Sequenzen werden beim *Training* durchlaufen, um die Parameter der Modelle (Gesten- und Anatomiemodell) neu zu schätzen.
- Die Betriebsart *Evaluation* dient der Leistungsmessung. Hierzu werden auf manuell markierten Sequenzen Personenverfolgung und Gestenerkennung automatisch durchgeführt und die Ergebnisse mit den Markierungen verglichen.
- Bei einer *Vorführung* werden Personenverfolgung und Gestenerkennung in Echtzeit durchgeführt und visualisiert.

C.2 Externe Bibliotheken

Zur Ansteuerung der Kamera, zur Durchführung von Bildverarbeitungsoperationen und zur Erzeugung der graphischen Benutzeroberfläche werden die drei folgenden externen Bibliotheken verwendet:

Small Vision System

Das *SRI Small Vision System (SVS)* der Firma Videre Design ist ein Softwarepaket mit Funktionen zur Ansteuerung unterschiedlicher Kameras (insbesondere des oben genannten Mega-D Stereokopfes), zur Berechnung von Disparitäten und zur Kalibrierung. Die Funktionen zum Einlesen von Videobildern und zur Berechnung von Disparitäten können mithilfe einer dynamischen Bibliothek (DLL) in eigene Anwendungen eingebunden werden. Für die Kalibrierung der Kamera wird ein separates Programm mitgeliefert.

Open Source Computer Vision Library

Die *Open Source Computer Vision Library (OpenCV)*[2] stellt eine große Anzahl von Funktionen zur Verfügung, die bei der Programmierung von Anwendungen aus dem Bereich des Maschinensehens hilfreich sind. Im vorliegenden Programm finden die folgenden Funktionsgruppen aus OpenCV Verwendung: elementare Bildverarbeitungsoperationen, morphologische Operatoren, Kantendetektion, Komponentenanalyse, einfache Zeichenfunktionen.

Fast Light Toolkit

Das *Fast Light Toolkit (FLTK)*[3] ist eine einfache objektorientierte Bibliothek zur Erstellung graphischer Benutzeroberflächen. Es bietet eine Reihe von Dialogbausteinen (Fenster, Schaltflächen, Eingabelfelder, ...), Operationen zum Darstellen von Rasterbildern und dreidimensionalen OpenGL-Szenen, sowie einen einfachen Mechanismus zur Ereignisbehandlung. Die gesamte Oberfläche des in dieser Arbeit entwickelten Programms wurde mit FLTK erstellt.

[2] Open CV wird bereitgestellt von der Firma Intel (http://www.intel.com/research/mrl/-research/opencv/).

[3] http://www.fltk.org/

Literaturverzeichnis

[AP96] AZARBAYEJANI, A. und A. PENTLAND: *Real-time self-calibrating stereo person tracking using 3-D shape estimation from blob features.* In: *Proceedings of 13th ICPR*, 1996.

[Bec97] BECKER, D.A.: *Sensei: A Real-Time Recognition, Feedback and Training System for T'ai Chi Gestures.* Technical Report No. 426, M.I.T. Media Lab Perceptual Computing Group, USA, 1997.

[CBA+96] CAMPBELL, L.W., D.A. BECKER, A. AZARBAYEJANI, A.F. BOBICK und A. PENTLAND: *Invariant features for 3-D gesture recognition.* In: *Second International Workshop on Face and Gesture Recognition*, Killington VT, 1996.

[DGHW98] DARRELL, T., G. GORDON, M. HARVILLE und J. WOODFILL: *Integrated person tracking using stereo, color, and pattern detection.* In: *IEEE Conference on Computer Vision and Pattern Recognition*, Seiten 601–608, Santa Barbara, CA, 1998.

[EKB98] EVELAND, C., K. KONOLIGE und R. BOLLES: *Background modeling for segmentation of video-rate stereo sequences.* In: *In Proceedings IEEE Conf. on Computer Vision and Pattern Recognition*, Seiten 266–271, 1998.

[Hor86] HORN, B.: *Robot Vision.* The MIT Press, Cambridge, Massachusetts, 1986.

[IB98] ISARD, M. und A. BLAKE: *Condensation – conditional density propagation for visual tracking.* International Journal of Computer Vision 29(1), Seiten 5–28, 1998.

[JBM+00] JOJIC, N., B. BRUMITT, B. MEYERS, S. HARRIS und T. HUANG: *Detection and Estimation of Pointing Gestures in Dense Disparity Maps.* In: *IEEE International Conference on Automatic Face and Gesture Recognition*, Grenoble, Frankreich, 2000.

[Jäh97] JÄHNE, B.: *Digitale Bildverarbeitung.* Springer-Verlag, Berlin-Heidelberg, 4. Auflage, 1997.

[Kon97] KONOLIGE, K.: *Small Vision Systems: Hardware and Implementation.* In: *Eighth International Symposium on Robotics Research*, Hayama, Japan, 1997.

[Nic02] NICKEL, K.: *3D-Tracking von Gesicht und Händen mittels Farb- und Tiefeninformation.* Studienarbeit, Fakultät für Informatik, Universität Karlsruhe, 2002.

[PSOS98] PODDAR, I., Y. SETHI, E. OZYILDIZ und R. SHARMA: *Toward Natural Gesture/Speech HCI: A Case Study of Weather Narration.* In: *Proc. Workshop on Perceptual User Interfaces (PUI98)*, San Francisco, USA, 1998.

[Rab89] RABINER, L.R.: *A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition.* In: *Proc. IEEE, 77 (2)*, Seiten 257–286, 1989.

[SP95] STARNER, T. und A. PENTLAND: *Visual recognition of american sign language using hidden markov models.* In: *International Workshop on Automatic Face and Gesture Recognition*, Seiten 189–194, 1995.

[TSK99] TAO, H., H.S. SAWHNEY und R. KUMAR: *A Sampling Algorithm for Tracking Multiple Objects.* In: TRIGGS, B., A. ZISSERMAN und R. SZELISKI (Herausgeber): *Vision Algorithms: Theory and Practice*, Nummer 1883 in *LNCS*, Seiten 53–68, Corfu, Greece, 1999. Springer-Verlag.

[WADP97] WREN, C., A. AZARBAYEJANI, T. DARRELL und A. PENTLAND: *Pfinder: Real-Time Tracking of the Human Body.* In: *IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 19, No. 7*, 1997.

[WB98] WILSON, A.D. und A.F. BOBICK: *Recognition and Interpretation of Parametric Gesture.* In: *International Conference on Computer Vision*, Seiten 329–336, 1998.

[YLW97] YANG, J., W. LU und A. WAIBEL: *Skin-color modeling and adaption.* Technical Report of School of Computer Science CMU-CS-97-146, CMU, USA, 1997.