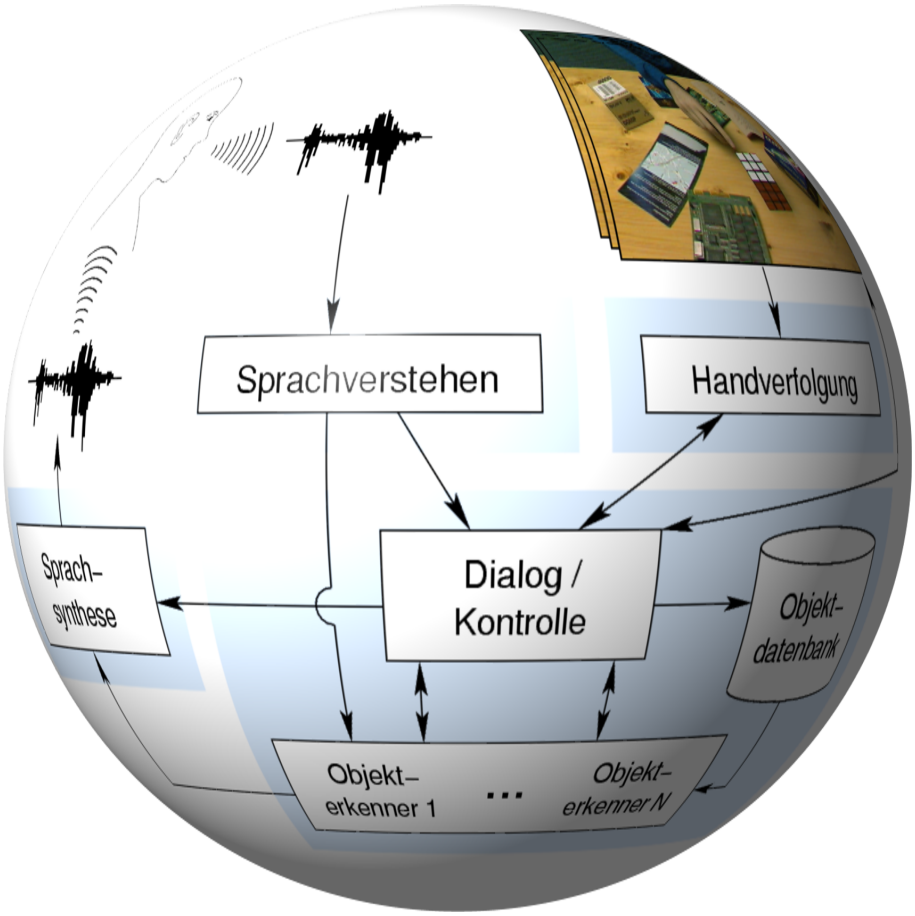
Lernen von Objektbenennungen mit visuellen Prozessen

Frank Lömker



Dipl.-Inform. Frank Lömker
AG Angewandte Informatik
Technische Fakultät
Universität Bielefeld
email: floemker@techfak.uni-bielefeld.de

Abdruck der genehmigten Dissertation zur Erlangung
des akademischen Grades Doktor-Ingenieur (Dr.-Ing.).
Der Technischen Fakultät der Universität Bielefeld
am 3. Mai 2004 vorgelegt von Frank Lömker,
am 9. Juli 2004 verteidigt und genehmigt.

Gutachter:
Prof. Dr. Gerhard Sagerer, Universität Bielefeld
Prof. Dr. Bernt Schiele, Technische Universität Darmstadt

Prüfungsausschuß:
Prof. Dr. Helge Ritter, Universität Bielefeld
Prof. Dr. Gerhard Sagerer, Universität Bielefeld
Prof. Dr. Bernt Schiele, Technische Universität Darmstadt
Dr. Sven Wachsmuth, Universität Bielefeld

Gedruckt auf alterungsbeständigem Papier nach ISO 9706

Lernen von Objektbenennungen mit visuellen Prozessen

Dissertation zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften (Dr.-Ing.)

der
Technischen Fakultät der Universität Bielefeld

vorgelegt von

Frank Lömker

3. Mai 2004

Danksagung

Ohne die Hilfe und Unterstützung vieler Personen ist die wissenschaftliche Arbeit und nachfolgende Erstellung einer Dissertation praktisch nicht möglich. Auch diese Arbeit bildet dabei keine Ausnahme. Diese Stelle möchte ich dazu nutzen, allen zu danken, die mir im Laufe der Zeit mit Rat und Tat zur Seite gestanden haben. Alle diese Personen einzeln aufzuzählen bin ich nicht in der Lage. Nichtsdestotrotz möchte ich zumindest einigen von ihnen hier explizit danken.

Als erstes möchte ich meinem Betreuer Prof. Gerhard Sagerer danken. Er hat es mir überhaupt erst ermöglicht, im Rahmen der Arbeitsgruppe Angewandte Informatik diese Arbeit zu erstellen. Besonders danken möchte ich ihm auch für die Einführung in dieses Thema. Durch die Vielseitigkeit des Themas war die Arbeit immer wieder stimulierend und interessant. Seine fortwährende Unterstützung hat maßgebend dazu beigetragen, daß ich die Arbeit beenden konnte. Auch möchte ich Prof. Bernt Schiele danken, der sich bereit erklärt hat, diese Arbeit zu begutachten.

Jannik Fritsch möchte ich besonders für das Korrekturlesen dieser Arbeit danken. Seine Kommentare haben die Qualität dieser Arbeit sicherlich deutlich verbessert.

Insbesondere für den Sprachteil konnte ich auf umfangreiche Vorarbeiten aufsetzen. ESMERALDA von Gernot A. Fink hat die Integration von Sprache sehr vereinfacht. Auch die Arbeiten von Sven Wachsmuth und Hans Brandt-Pook waren sehr hilfreich.

Dank gilt auch meinen Bürokollegen Sven Wachsmuth, Sascha Hohenner, Thomas Plötz und Nils Hofemann, die immer zu einem sehr angenehmen Arbeitsklima beigetragen haben. Den gleichen Dank möchte ich auch der ganzen Arbeitsgruppe Angewandte Informatik geben. Während der ganzen Zeit war die Arbeit immer spannend und interessant.

Mein besonderer Dank gilt aber meinen Eltern. Ohne ihre durchgehende und jederzeit volle Unterstützung wäre ich niemals so weit gekommen.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

Stellen wir uns eine typische Szene bei einem gemeinsamen Essen vor. Mehrere Personen sitzen an einem Tisch und unterhalten sich. Einer von ihnen, nennen wir ihn Person A, möchte gerne den Zucker haben, der auf dem Tisch auf der ihm gegenüber liegenden Seite steht. Vielleicht sagt er zu jemand anderem am Tisch "Könntest Du mir bitte die Kluntje geben?", womit er eine in Ostfriesland übliche Bezeichnung für großen Kandiszucker benutzt. Der Angesprochene, Person B, wird auf den Tisch schauen, die Kluntje suchen und identifizieren und sie schließlich herüberreichen.

Möglicherweise weiß er aber auch nicht, was Kluntje sind. Oder er hat eine spezielle Vorstellung von Kluntje, beispielsweise daß sie immer weiß sein müssen, und keines der Teile auf dem Tisch deckt sich mit dieser Vorstellung. In diesem Fall muß er erst sein mentales Modell des Gegenstandes aktualisieren oder es sogar komplett neu aufbauen: Er muß *lernen*, was hier mit dem Begriff Kluntje gemeint ist. Sicherlich will er in dieser Situation aber keine mathematisch exakte Definition haben, sein Gegenüber möchte ja nur etwas haben, was er momentan nicht identifizieren kann. Vielleicht fragt er einfach "Welches Teil meinst Du?".

Sein Nachbar weiß eventuell, was gemeint war, schaut auf den Tisch, greift nach den Kluntje und reicht sie herüber. Oder er reicht sie zu Person B und sagt dabei "Der Zucker hier war gemeint.". Person B kann beides beobachten und lernt durch die Beobachtung der Aktion seines Nachbarn, was hier mit dem Begriff Kluntje gemeint war. Er kann mit Hilfe der Aktion sein mentales Modell aktualisieren. Vielleicht antwortet Person A auch mit "Die braunen Kluntje da drüben" und zeigt gleichzeitig auf die Kluntje. Damit schränkt er sowohl visuell durch die Zeigegeste als auch sprachlich durch das Attribut braun den Kontext für die zu lösende Aufgabe, das Finden der Kluntje, ein. Durch diese Informationen ist der Angesprochene nun gegebenenfalls in der Lage, die Aufgabe zu lösen und kann gleichzeitig sein mentales Modell von Kluntje durch die erfolgreiche Durchführung der Aufgabe verbessern.

Eine alltägliche Begebenheit hat sich hier abgespielt, die wir alle schon häufig in der einen oder anderen Form erlebt haben. Ein sprachlich referenziertes Objekt sollte visuell lokalisiert und identifiziert werden, wobei zum Teil eine andere visuelle Modalität, eine Zeigegeste, mit in die Analyse integriert werden mußte. Gleichzeitig wurde in Interaktion mit anderen Personen und unter Einsatz verschiedener Modalitäten das mentale Modell eines bisher unbekannten Objektes neu gelernt oder auch das Modell für ein schon bekanntes Objekt aktualisiert und verbessert. Hierbei wurde auch eine

Aktion, das Greifen und Herüberreichen des zu lernenden Gegenstandes, mit in den Lernvorgang integriert.

All dies sind auch wichtige Eigenschaften bei einem Serviceroboter, wenn er beispielsweise im Haushalt oder in anderen Bereichen erfolgreich unterstützend tätig sein soll. Idealerweise kann man dem Roboter einfach neue Objekte, die während des alltäglichen Einsatzes als Arbeitshilfe immer wieder auftreten, in einer natürlichen und komfortablen Weise zeigen und sprachlich beschreiben. Wie der Mensch sollte er Rückmeldungen geben können oder auch weitere Informationen anfordern können. Dafür muß er in der Lage sein, mit bisher unbekannten Perzepten, Symbolen und Abbildungen zwischen diesen zurecht zu kommen.

Mobile Robotersysteme, die in den letzten Jahren in der Forschung entwickelt wurden, haben diesen Entwicklungsstand bisher noch nicht erreicht [Thr00, Nak02, Wic02, Lan03]. Wichtige Punkte der Forschung sind hier momentan robuste Lokalisation, Wegfindung, Kollisionsvermeidung, Aufmerksamkeitssteuerung und aktuell erste weitergehende Möglichkeiten der Mensch-Maschine-Interaktion. Erste, sehr spezielle Haushalts-Serviceroboter finden inzwischen ihren Weg zum kommerziellen Einsatz, beispielsweise zum Staub saugen in Form des Trilobite von Electrolux oder Rasen mähen in Form des Auto Mower von Husqvarna. Diesen fehlt allerdings noch jegliche Fähigkeit zur komplexeren Mensch-Maschine-Interaktion. Aber die Mensch-Maschine-Interaktion bekommt in den letzten Jahren in der Forschung eine immer größere Bedeutung. So sagen beispielsweise [Bur98]:

> "What lessons did we learn? From hours of watching the robot operate, we concluded that interaction with people is an essential element in the success of future robots of this and similar kinds. We believe that there is a real need for research in the area of human robot interaction."

Der Mensch hat bei all diesen Vorgängen keinerlei Schwierigkeiten. Er kann problemlos verschiedene Modalitäten gleichzeitig bearbeiten und integrieren. Das Erkennen von Objekten ist für uns alltäglich. Und auch das neu Lernen und immer weiter Verbessern von Objektmodellen machen wir ständig. Für den Computer stellen aber all diese Teilgebiete und deren Kombination und Integration eine große Herausforderung da. Der Computer muß mit einer bewegten, sich ständig ändernden, dynamischen Welt zurecht kommen. Objekte können in beliebigen Umgebungen und unter beliebig variablen Beleuchtungsverhältnissen vorkommen. Die Aufnahmesensoren des Systems liefern immer verrauschte und mit andersartigen Fehlern versehene Daten. Die Sprache kann beliebig komplex und unstrukturiert sein und ist insbesondere häufig nicht themengebunden. Viele Störgeräusche wie Verkehrslärm, sekundäre Gespräche und auch Radio- und Fernsehempfang oder andere Hintergrundgeräusche kommen immer wieder vor.

Effektives Lernen erfordert die Auflösung von Korrespondenzen zwischen Sprache und Bild. Für eine sinnvolle Kategorisierung und Hierarchisierung von neuem Wissen ist umfangreiches Weltwissen erforderlich. Sehr häufig ist das Abstrahieren von

konkreten Beispielen zu einer allgemeineren Klasse nötig, wozu ebenfalls umfangreiches Weltwissen erforderlich ist. Die Abstrahierung von einer konkreten Ansicht eines neu zu lernenden Objektes und einer konkreten Umgebung ist fast immer erforderlich [Chr03a].

Es ließen sich leicht weitere große Herausforderungen für die automatische Verarbeitung des skizzierten Problems aufzählen, die wir Menschen relativ problemlos meistern, die für den Computer in der vollen Allgemeinheit aber noch eine nicht überwindbare Hürde darstellen. Erst in neuester Zeit kommen kognitive Systeme auf, die zumindest Teile dieser Probleme angehen [Ovi03, Chr03b]. [Cro03] nennt Gründe:

> "The roots of cognitive vision lie in the explosion of interest in human perception which resulted from developments of artificial intelligence in the 1970's. However, cognitive vision is only now beginning to realise its potential, based partly on developments in our scientific understanding but also, crucially, on improvements in imaging, information and communication technologies."

Trotz der großen Herausforderungen versucht diese Arbeit einen Beitrag zur Lösung der Probleme zu liefern. Es wird ein vollständiges System vorgestellt, welches einen ersten Schritt in die Richtung zu einem interaktiven, multimodalen, visuell basierten Lernen unbekannter Objekte geht. Das System ist domänenunabhängig und damit nicht auf spezielle Objekte beschränkt. Der Benutzer kann gesprochene Sprache benutzen, um auf Objekte zu referenzieren. Deiktische Gesten und Greifaktionen des Benutzers werden zusätzlich ausgewertet.

Das System analysiert die von einer normalen Videokamera aufgenommene Szene und extrahiert verschiedene visuelle Merkmale für die Erkennung von Objekten und das Lernen neuer Objekte. Weiterhin werden Handtrajektorien für die Gestenerkennung extrahiert. Durch Kombination all dieser Informationen versucht das System adäquat auf sprachliche Äußerungen des Benutzers zu reagieren, indem es sprachlich oder deiktisch referenzierte Objekte in der Szene lokalisiert und identifiziert. Unbekannte Objekte werden in Interaktion mit dem Benutzer neu gelernt. Fehlerhaftes Verhalten des Systems kann der Benutzer wiederum interaktiv korrigieren, wobei Objektmodelle des Systems gleichzeitig verbessert werden. Alle erfolgreich durchgeführten Objekterkennungen führen automatisch zu einer Optimierung der gelernten Objektmodelle. Damit wird das System in die Lage versetzt, inkrementell zu lernen und so auf Äußerungen des Benutzers mit verbesserter Leistung zu reagieren.

Gliederung der Arbeit

Kapitel 2 gibt einen Überblick über relevante Forschungsfelder und ordnet die vorgelegte Arbeit dort ein. Andere relevante Arbeiten im Umfeld des Objektlernens werden vorgestellt. Kapitel 3 stellt zuerst das Szenario, welches in dieser Arbeit untersucht wird, genauer vor. Anschließend wird ein Überblick über das vorgeschlagene System

und dessen Architektur gegeben. Alle Teilkomponenten, die in folgenden Kapiteln genauer behandelt werden, werden kurz eingeführt und deren Interaktion wird beschrieben. Das realisierte Verfahren zum Lernen neuer Objekte und neuer Ansichten dieser Objekte wird ebenfalls in diesem Kapitel erläutert.

In den folgenden Kapiteln werden die einzelnen Teile des Gesamtsystems umfassend dargestellt. In Kapitel 4 wird die Sprachverstehens- und Sprachausgabekomponente des Systems vorgestellt. Kapitel 5 stellt das System zur visuellen Erkennung von Gesten und Handlungen und deren Interpretation vor. In Kapitel 6 werden schließlich die visuellen Merkmale und Erkennungsverfahren vorgestellt, die für die Objektidentifikation und Lokalisation benutzt werden. Kapitel 7 geht auf die Nachbearbeitung der durch die Objekterkenner generierten Hypothesen ein. Einerseits werden hierbei durch die Kombination der verschiedenen Erkenner aus dem vorherigen Kapitel neue übergeordnete Hypothesen erzeugt. Andererseits werden weitere sprachliche Informationen sowohl für die Optimierung der Objekthypothesen als auch für die Unterstützung des Lernprozesses ausgenutzt.

Kapitel 8 stellt die Ergebnisse vor, die im Bereich der Performanz des Gesamtsystems mit Hilfe interaktiver Tests mit Benutzern des Systems erreicht wurden. Teilergebnisse verschiedener Komponenten des Systems wurden in verschiedenen vorherigen Kapiteln vorgestellt. Kapitel 9 schließt die Arbeit durch eine Zusammenfassung und einen Ausblick auf sinnvoll erscheinende Anschlußarbeiten ab. Im Anhang wird anschließend noch eine Einführung in das im Rahmen dieser Arbeit entstandene CASE-Tool iceWing gegeben.

2 Problembeschreibung

Diese Arbeit stellt ein komplettes multimodales System vor und berührt dabei verschiedene Forschungsgebiete. Für eine Einordnung der Arbeit im Hinblick auf das Gesamtsystem sind drei Punkte von besonderem Interesse. Erstens referenziert der Benutzer sprachlich auf Objekte, die er und das System visuell wahrnehmen. Eine Betrachtung, wie Objekte benannt werden und wie sie in der Welt verankert sind, findet daher im nächsten Abschnitt statt. Als Zweites ist das Lernen neuer Objekte ein zentraler Punkt. Abschnitt 2.2 gibt daher eine kurze Einführung in das maschinelle Lernen. Schließlich werden in Abschnitt 2.3 verwandte Systeme vorgestellt.

2.1 Objekte aus der Sicht der Linguistik

2.1.1 Das "Symbol Grounding" Problem

Durch anfänglich große Erfolge der künstlichen Intelligenz gefördert, wurde bis in die 70er Jahren hinein ein Modell des Verstandes als ein symbolverarbeitendes System populär [Fod75]. Hier wurde Kognition als reine Symbolmanipulation innerhalb des Systems interpretiert, wobei Systeme aus der KI als Initiatoren dienten. Aber kann so ein symbolverarbeitendes System überhaupt verstehen, was es manipuliert?

Diese Frage wurde von Searle in seinem Problem des chinesischen Raumes gestellt [Sea80]. Dabei stellt er sich vor, in einem geschlossenen Raum zu sitzen, wo er eine Geschichte und Fragen zu dieser Geschichte in Chinesisch bekommt. Zusätzlich bekommt er in Englisch formulierte Regeln zur syntaktischen Manipulation der Fragen. Selber versteht er keinerlei Chinesisch. Durch rein syntaktisches Anwenden der Regeln auf die Fragen kann er die Fragen korrekt beantworten, ohne daß er die Geschichte versteht. Das Verstehen kann durch die Regeln nicht geleistet werden, da die Symbole keine intrinsische Bedeutung haben.

Harnad greift dieses Problem auf und bietet eine Lösung durch *Symbol Grounding*, durch die Verankerung von Symbolen in der Welt, an [Har90]. Dabei wird das klassische symbolverarbeitende System durch Sensoren erweitert, die jeweils mit der Welt verbunden sind. Diese Sensoren können beispielsweise neuronal und damit insbesondere nichtsymbolisch arbeiten. Mit Hilfe der Sensoren kann das ehemals rein symbolische System nichtsymbolische Objekte und Ereignisse unterscheiden und identifizieren und

damit kategorisieren. Basierend auf den nichtsymbolischen Kategorien können nun weitere symbolische Kategorien gebildet werden oder auch symbolisch Schlußfolgerungen gezogen werden. Harnad gibt ein Beispiel, bei dem ein Zebra als ein Pferd mit Streifen definiert wird, wobei das Pferd und die Streifen sensorisch verankert sind und das Zebra rein symbolisch aus diesen gebildet wird.

Weiterhin gibt es die Sicht, daß sich die Erkenntnis über die Umwelt aus der Interaktion des Systems mit seiner Umwelt ergibt und es keine explizite Repräsentation von Wissen innerhalb des Systems geben muß. Das System ist direkt eingebettet in seine Umwelt. [Sha00] stellt dies durch Vergleichen mit dem symbolischen Ansatz genauer vor. Auch wenn dies ein interessanter Ansatz ist, scheint es für ein konkretes System zur Mensch-Maschine-Kommunikation auf Grund von Bootstrappingproblemen nicht geeignet. Hier ist es geschickter, bekanntes Wissen auch explizit zu modellieren.

2.1.2 Benennen von Objekten

Was ist das Ziel einer sprachlichen Benennung eines Objektes? Nach [Her94] spricht jemand, "wenn er ein Ziel hat, das heißt, einen Zustand präferiert, der gerade nicht vorliegt, und wenn er andere als sprachliche Mittel entweder nicht zur Verfügung hat oder diese ihm weniger geeignet erscheinen." Die sprachliche Objektbenennung hat nun häufig das Ziel, ein physikalisch oder auch nur imaginär vorhandenes Objekt für einen Zuhörer identifizierbar zu machen. Aber auch völlig andere Gründe sind denkbar. [Her94, Abschnitt 2.1] gibt ein Beispiel, bei dem das eigentliche Ziel einer Objektbenennung ein Vorwurf an den Zuhörer ist. Der Zuhörer wird hier durch die Benennung eines Objektes an einen Fehler erinnert, den er in der Vergangenheit mit dem Objekt verursacht hat. Um Sprache im Allgemeinen zu verstehen, müssen auch solche Ziele interpretiert werden. Bei dem aktuellen Stand der Forschung in der Mensch-Maschine-Kommunikation ist eine Beschränkung auf das konkrete Benennen physikalischer Objekte aber sicherlich sinnvoll. Schon allein diese sind von einer extremen Komplexität. Auch die linguistische Forschung hat sich intensiv mit diesen Benennungen beschäftigt [Man87, Man03].

Für das Verständnis von Objektbenennungen ist es wichtig, daß Sprecher und Zuhörer von einem gemeinsamen Weltausschnitt, einem gemeinsamen *universe of discourse* ausgehen. Der Sprecher wird jeweils versuchen, seine Benennung informativ zu gestalten, um so sein Ziel auch zu erreichen [Her94, Abschnitt 2.3]. Daher wird er versuchen, die Benennung an das von ihm vermutete Wissen des Zuhörers anzupassen. Auch der Kontext, in dem sich das zu benennende Objekt befindet, hat häufig Einfluß auf die Benennung. In den Versuchen von [Ols70] ergab sich ein auch von [Gri75] beschriebenes Verhalten. Die Versuchsperson sollte in den Versuchen jeweils ein Objekt unter einer Menge verschieden farbiger und verschieden geformter Objekte benennen. Dies tat sie in den meisten Fällen sowohl informativ, das Objekt war eindeutig spezifiziert, als auch minimal. War beispielsweise die Form des Objektes für eine eindeutige Benennung nicht nötig, wurde sie auch nicht mit spezifiziert.

Diese minimale Spezifität tritt allerdings nicht immer auf. [Her94, Abschnitt 2.4] und [Man03] nennen verschiedene Faktoren, die zu einer Überspezifizierung führen können. Überspezifiziert wird insbesondere dann, wenn es das Benennen erleichtert, beziehungsweise wenn die erzeugte Überspezifizierung für den Sprecher schwer erkennbar ist und wenn es den Suchprozeß des Zuhörers unterstützt. Beispielsweise wird Farbe, eine peripher im Auge wahrnehmbare Objekteigenschaft, die sehr schnell ausgewertet wird, besonders häufig zusätzlich mit spezifiziert.

Aber nicht nur die Attributierung einer Benennung ist variabel. Auch die Wahl des Nomens variiert. Häufig werden Objekte auf einer Ebene mittlerer Spezifität benannt. Beispielsweise wird eine Schere meistens nicht als Werkzeug oder Metallschere bezeichnet, sondern eben als Schere [Ros75]. Abweichungen hiervon sind aber häufig, so hängt die Benennung beispielsweise auch von der Typikalität des Objektes und von dessen Kontext ab. Ein Huhn wird meistens nicht als Vogel bezeichnet, da es beispielsweise nicht fliegen kann und damit eher ein untypischer Vogel ist. Eine Rohrzange wird im Kontext von Hämmern seltener als Rohrzange bezeichnet als im Kontext von anderen Zangen. Weitere Einflußgrößen gibt [Her94] an.

Weiterhin können Objekte beispielsweise durch Nebensätze und Präpositionalphrasen genauer spezifiziert werden. Die Betrachtung dieser Phänomene würde in dieser Arbeit allerdings zu weit führen. Benennung von Objekten über Raumrelationen in einem künstlichen Kommunikator werden für eine eingeschränkte Domäne beispielsweise in [Wac01] betrachtet.

2.1.3 Resumé

Wie an dem “Symbol Grounding” Problem zu sehen ist, ist eine Verbindung von Symbolen und den dazugehörigen Perzepten in der Welt von immanenter Wichtigkeit. Ein System zum Lernen von Objekten muß diese daher in der Welt sensorisch verankern.

Bei der Benennung der Objekte muß das System Nominalphrasen verarbeiten können. Häufig angegebene Objektmerkmale, insbesondere die Farbe, sollten neben Nomen für eine erhöhte Flexibilität erlaubt sein. Häufig sind diese wegen minimal durchgeführter Spezifikationen auch für eine korrekte Interpretation nötig. Auch kann sich das System nicht darauf verlassen, das gleiche Objekte immer mit den gleichen Nomen benannt werden. Synonyme als ein erster Schritt in diese Richtung würden daher die Flexibilität des Systems erhöhen.

2.2 Maschinelles Lernen

Ein zentraler Punkt der vorgestellten Arbeit ist das Lernen von Objekten. Für eine Einordnung wird daher eine kurze Einführung in das maschinelle Lernen gegeben.

Definition von Lernen

Bevor man sich tiefer mit dem maschinellen Lernen beschäftigen kann, sollte zuerst definiert werden, was unter Lernen verstanden werden soll. Es wurden schon viele Definitionen vorgeschlagen, eine verbreitete Definition kommt von Simon [Sim83]:

> "Learning denotes changes in the system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficienly and more effectively the next time."

Diese Definition ist nicht unumstritten. So wird beispielsweise argumentiert, daß nicht jede Verbesserung einer Lösung einer Aufgabe auch Lernen ist. Beispielsweise fährt ein repariertes Auto besser. Es hat dabei aber nichts gelernt. Auch muß Lernen nicht immer intentional sein, was in der Definition durch die Angabe der Aufgabe intendiert wird. Auch Minsky gibt eine allgemeine Definition an, führt diese aber gleich mit "If we tried to find a single definition for "learning" to span so many kinds of processes, we'd end up with some phrase too broad to have much use" ein [Min86]:

> " "Learning" is making useful changes in the workings of our minds."

Daher schlägt beispielsweise [Wro00] eine extensionale Definition über präzise definierte Lernaufgaben vor. Durch diese Art der Definition kann aber vermutlich nicht der vollständige Begriff des Lernens definiert werden, da immer nur kleine Teilaspekte durch einen Typ von Lernaufgabe abgedeckt werden. Somit bleibt es unbekannt, ob oder wann das Feld vollständig abgedeckt ist.

Sieht man allerdings die Definition von Simon etwas differenzierter, ist sie zumindest auch für die beiden aufgeführten Problemfälle tragfähig. Im Beispiel des Autos hat das Auto zwar nichts gelernt, aber der die Reparatur initialisierende Autohalter, der nun durch die von Ihm angestoßene Reparatur besser fahren kann. Nicht vorsätzliches Lernen, der zweite Kritikpunkt, wird durch die Definition nicht explizit ausgeschlossen. Daher sehe ich diese Definition als eine gute Grundlage an, die bei Bedarf durch Lernaufgaben in Teilgebieten weiter präzisiert werden kann.

Formen von Lernen

Üblicherweise werden zwei Formen von Lernen unterschieden, der *initiale Wissenserwerb* und die *inkrementelle Verbesserung und Verfeinerung von Fähigkeiten* [Mic83, Abschitt 1.2.3]. Bei dem Wissenserwerb, einem bewußten Prozeß, werden neue symbolische Strukturen und mentale Modelle erzeugt. Es ist beispielsweise das Ergebnis von Lesen in einem Buch oder Lernen von Lehrstoff in der Schule. Neues, bisher unbekanntes Wissen wird akquiriert und kann danach angewendet werden. Je mehr oder je bewußter gelernt wurde, desto mehr Probleme können in besserer Form gelöst werden.

Die Verfeinerung von Fähigkeiten tritt dagegen ständig bei der Anwendung von vorher akquiriertem Wissen auf. Sowohl beim bewußten Training als auch beim unterbewußten Anwenden von Wissen werden ständig Abweichungen vom gewünschten

Ergebnis korrigiert und die Fähigkeiten optimiert. Vieles hiervon wird dabei auch beim bewußten Training unterbewußt gemacht. Erst durch diese ständige Optimierung kann initial erworbenes Wissen effektiv eingesetzt werden. Zusammenfassend ist menschliches Lernen zu großen Teilen eine Mischung aus initialem Wissenserwerb und der inkrementellen Verfeinerung von Fähigkeiten.

Verschiedene Lernstrategien

Systeme zum maschinellen Lernen lassen sich nach verschiedenen Taxonomien in Klassen einteilen. Eine interessante Einteilung ergibt sich dabei durch die eingesetzten Lernstrategien. Diese lassen sich nach der vom Lernenden zu erbringenden Inferenzleistung klassifizieren. [Mic83, Abschitt 1.3.1] gibt folgende fünf Strategien an:

Lernen ohne Inferenz: Dies ist das aufwendigste und komplizierteste Verfahren, da der Lehrende jeden kleinsten Schritt explizit formulieren muß. Der Lernende unterstützt das Lernen nicht durch eigene Leistungen. Daher ist es auch sehr fehleranfällig. Zwei Varianten dieser Lernstrategie können unterschieden werden:

- **Lernen durch Programmieren:** Eine vom Lernenden unabhängige Einheit "implantiert" das Wissen in den Lernenden.
- **Lernen durch auswendig Lernen:** Dem Lernenden werden nur Fakten präsentiert, die er ohne jegliche Inferenz aufnimmt.

Lernen durch Unterweisung: Die in Schulen und anderen Lehranstalten übliche Lernstrategie. Der Lehrende bereitet das Wissen für den Lernenden geeignet auf. Der Lernende muß das Wissen noch in eine interne Repräsentation konvertieren und mit bisherigem Wissen verknüpfen.

Lernen durch Analogien: Durch Anwenden von modifiziertem existierendem Wissen auf Situationen, die Ähnlichkeiten zu einer bekannten Situation haben, wird neues Wissen gewonnen. Wenn man beispielsweise ein Modell eines Videorecorders bedienen kann, kann dieses Wissen auf andere Modelle übertragen werden. Auch ein ähnliches Gerät wie einen Kassettenrecorder wird man mit der Anwendung des Lernens durch Analogie bedienen können.

Lernen durch Beispiele: Dem Lerner stehen positive und negative Beispiele für ein Konzept zur Verfügung. Hieraus generiert der Lerner ein allgemeines Konzept, welches alle positiven Beispiele einschließt und alle negativen ausschließt. Dies ist auch das normale Vorgehen beim Trainieren eines Klassifikators [Sch96, Jai00]. Durch Vorgabe einer klassifizierten Stichprobe wird ein Klassifikator erzeugt, der auch Merkmale außerhalb der Trainingsstichprobe klassifizieren kann. Häufig ist hier das Lernen nach dem initialen Training abgeschlossen. Aber auch ein inkrementelles Verfeinern des Gelernten ist bei dieser Strategie möglich.

Die Lernstrategie durch Beispiele läßt sich nach der Herkunft der Beispiele und der Art der Beispiele weiter differenzieren. Die Beispiele können

- vom Lehrenden generiert und klassifiziert sein. Der Lehrende wird versuchen, für den Lernenden geeignete Beispiele zu wählen.
- vom Lernenden generiert und von einer externen Quelle klassifiziert sein. Der Lernende wird versuchen, ihm unbekannte Gebiete im Merkmalsraum zu explorieren.
- oder von einer externen Quelle stammen. In diesem Fall müssen die Beispiele im allgemeinen Fall als zufällig angesehen werden.

Weiterhin können nur positive Beispiele oder positive und negative Beispiele zur Verfügung stehen.

Lernen durch Beobachtung und Entdeckung: Hierbei steht dem Lerner kein Lehrender, der ihm Informationen über das zu akquirierende Wissen geben könnte, zur Verfügung. Daher wird dieses Lernverfahren auch unüberwachtes Lernen genannt. Der Lernende muß sich auf das Beobachten seiner Umwelt beschränken. Gegebenenfalls hat er zusätzlich die Möglichkeit, seine Umwelt in begrenztem Maße zu beeinflussen und aus der Reaktion der Beeinflussung zu lernen.

2.3 Lernen von Objekten - Eine Übersicht

Da die Mensch-Maschine-Kommunikation in den letzten Jahren eine immer größere Bedeutung erlangt hat, sind auch verstärkt Arbeiten in diesem Bereich durchgeführt worden. Dabei wurde auch das Lernen bisher unbekannter Objekte von verschiedenen Blickwinkeln aus betrachtet. In diesem Abschnitt wird nun ein Überblick über aktuelle in diesem Gebiet durchgeführte Arbeiten gegeben, um im folgenden hieraus Folgerungen und Rückschlüsse für die Konzeption dieser Arbeit ziehen zu können. Dafür werden vier verschiedene Arbeiten, die das Themengebiet jeweils unter unterschiedlichen Gesichtspunkten beleuchten, genauer vorgestellt. Das in Abschnitt 2.3.1 vorgestellte System verankert Objekte mit Hilfe der Sprache auf einer sehr signalnahen Ebene. Das System aus Abschnitt 2.3.2 ist zum Teil vergleichbar, legt allerdings seinen Schwerpunkt auf eine Aufmerksamkeitssteuerung. Abschnitt 2.3.3 stellt ein System vor, in dem die explizite Interaktion mit einem Benutzer von zentraler Bedeutung ist. Das System aus Abschnitt 2.3.4 legt seinen Schwerpunkt schließlich auf die Objekterkennung und schließt Sprache dafür aus.

2.3.1 Lernen von Sprache durch visuelle Verankerung

Roy und Pentland präsentieren ein System zum visuell verankerten Lernen von Sprache [Roy03, Roy02, Roy99]. Dabei gehen sie davon aus, daß Kinder die Bedeu-

tung von Wörtern durch die Beobachtung der Welt durch verschiedene Sinne lernen. Die Syntax der Sprache ergibt sich durch Kombination der Wörter, wodurch Relationen zwischen den zu den Wörtern gehörenden Konzepten ausgedrückt werden können. Durch Benutzung von sprachlichen und visuellen Informationen können Roy und Pentland hierfür ein Modell liefern.

Auch rein pragmatische Vorteile des verankerten Spracherkennens gegenüber dem traditionellen rein sprachgesteuerten Erkennen sehen sie. Einerseits ist das Training vereinfacht. Dies kann mit dem verankerten Erkenner automatisch geschehen und muß nicht durch manuell gelabeltes Trainingsmaterial durchgeführt werden. In dem vorgestellten System werden keinerlei handgelabelten Daten benutzt. Weiterhin kann der verankerte Erkenner Ambiguitäten mit Hilfe des visuellen Kanals leichter auflösen.

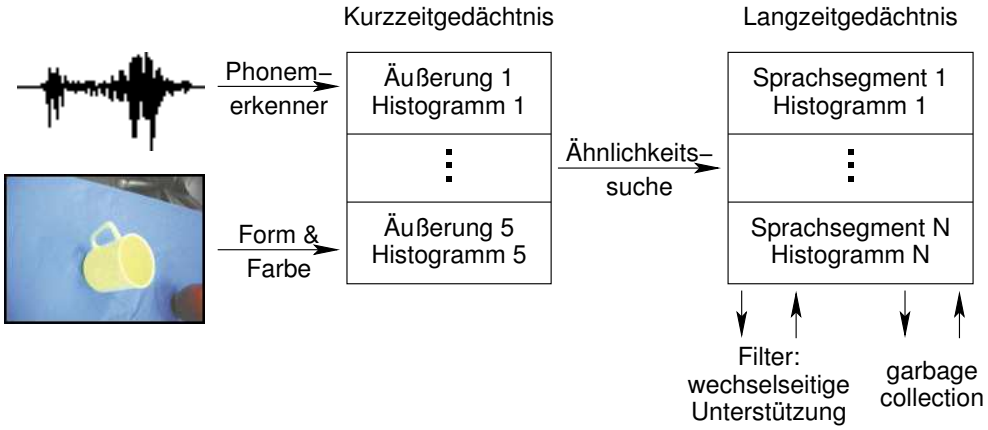


Abbildung 2.1: Architektur des Systems zum Lernen von Sprache durch visuelle Verankerung nach [Roy02] und [Roy03].

Abbildung 2.1 zeigt die Systemarchitektur. Die Eingaben des Systems sind Sprache und Bilder von Objekten. Hieraus wird ein Lexikon von audio-visuellen Objekten (AV-Objekten) bestehend aus einem *Hidden-Markov-Modell* (HMM [Lev83]) eines Wortes und Form- und Farbmerkmalen eines Objektes generiert. Auf der Sprachseite werden hierzu mit Hilfe eines normal trainierten englischen Phonemerkenners Phoneme generiert. Es wird angenommen, daß die zu lernenden Wörter in der Sprache häufiger vorkommen. Um diese häufiger auftretenden Teilsprachsegmente zu finden, ist ein Abstandsmaß d_A zum Vergleichen von zwei Segmenten α_1 und α_2 nötig. Hierzu wird für die beiden aus den Sprachsegmenten α_i generierten Phonemketten ein lineares HMM λ_i mit Zuständen für jedes Phonem generiert. Der Abstand der Segmente ergibt sich nun aus dem log likelihood, daß λ_1 Segment α_2 generiert und λ_2 Segment α_1:

$$d_A(\alpha_1, \alpha_2) = -\frac{1}{2}\left\{\log\left(\frac{P(\alpha_1|\lambda_2)}{P(\alpha_1|\lambda_1)}\right) + \log\left(\frac{P(\alpha_2|\lambda_1)}{P(\alpha_2|\lambda_2)}\right)\right\} \tag{2.1}$$

Die Objekterkennung geht von einem einfarbigen Hintergrund aus. Nicht hintergrundfarbene größere Bereiche werden als Ansichten von Objekten angesehen. Aus Merkmalen der Kante des Objektes und dessen Farbe im rg-Farbraum werden zwei Histogramme berechnet, welche als Objektrepräsentanten dienen. Nähere Informationen zu Histogrammen als Objektrepräsentanten finden sich in Abschnitt 6.2, eine Vorstellung des rg-Farbraums findet sich in Anhang A. Der Abstand d_V zweier Objekte ergibt sich mit Hilfe des χ^2-Testes: $d_V(a,b) = \chi^2(a,b)$. Jedes Objekt wird durch 15 Ansichten aus verschiedenen Richtungen repräsentiert.

Das Lernen von Sprache geschieht nun in 2 Schritten. Im ersten Schritt werden die letzten fünf im Kurzzeitgedächtnis gespeicherten Äußerungen und die dazugehörigen Objekte untersucht. Können mit Hilfe der beiden Abstandsmaße ähnliche Teile gefunden werden, werden diese in das Langzeitgedächtnis übertragen. In einem zweiten Schritt werden diese gefundenen AV-Objekte geclustert. Dazu wird das Maß für die wechselseitige Unterstützung U für einen Prototypen X aus der Menge aller n AV-Objekte maximiert:

$$U(A,V) = \sum_{i\in\{0,1\}} \sum_{j\in\{0,1\}} \frac{|A=i, V=j|}{n} \log\left(\frac{n\cdot|A=i,V=j|}{|A=i||V=j|}\right) \tag{2.2}$$

A beziehungsweise V sind 1, falls der Abstand von X zu einem anderen AV-Objekt unter einer Grenze liegt, 0 sonst. Damit wird U groß, falls X häufig sowohl in der Sprache als auch dem Bild ähnlich zu anderen Objekten ist, aber selten in nur einer Modalität. Hiermit können Wörter wie Artikel beseitigt werden, die in Kombination mit allen Objekten vorkommen. Liegt U über einem Schwellwert, wird von einem neuen verankerten Lexikoneintrag ausgegangen. Alle Objekte, die sowohl in der Sprache als auch dem Bild ähnlich sind, werden entfernt und der Clustervorgang wird fortgesetzt.

Mit Hilfe des generierten Lexikons kann das System nach manueller Auswahl sowohl visuell dargebotene Objekte benennen als auch sprachlich angefragte Objekte finden. Zusätzlich wird bei Wörtern, die nur in der Farbe oder der Form verankert sind, eine Verteilung über deren Anordnung in einer Äußerung geschätzt. Hiermit wird ein erster Schritt in Richtung Lernen von Syntax gegangen.

2.3.2 Aufmerksamkeitsgesteuertes Lernen von Sprache

Die Arbeit von Yu und Ballard ist insbesondere verwandt zu der im letzten Abschnitt vorgestellten Arbeit von Roy und Pentland. Anders als dort werden neben der Sprache und dem Bild aber auch Augenbewegungen, Kopfbewegungen und Handbewegungen, gemessen mit Hilfe spezieller Positionssensoren, als weitere Eingabekanäle betrachtet, um so den perzeptuellen Fähigkeiten eines Menschen noch näher zu kommen [Yu03a, Yu03b]. Abbildung 2.2 (a) zeigt die Sensorik des Systems, Abbildung 2.2 (b) ein aufgenommenes Bild der Kamera. Aus den Daten der Sensoren

lernt das System in der Welt verankerte Objektnamen und Aktionsverben, indem ein Benutzer Tätigkeiten durchführt und diese dabei kommentiert. Dabei wird im Gegensatz zu Roy und Pentland der *focus of attention*, die Position, auf die der Benutzer seine Aufmerksamkeit richtet, als wichtige Komponente mit integriert. Neben der Verbesserung der Mensch-Maschine-Kommunikation sehen die Autoren diesen referentiell intendierten Ansatz auch als ein Modell dafür an, wie Kinder durch die Beobachtung anderer sprechender Personen Sprache lernen.

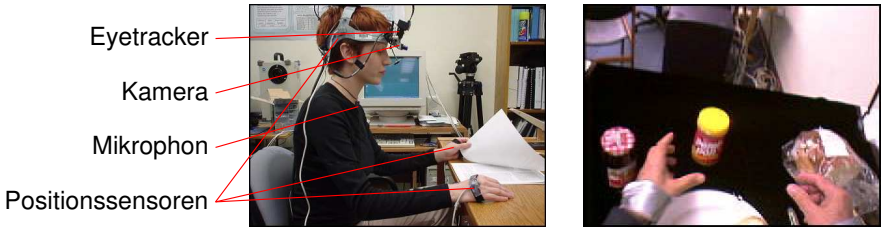


Abbildung 2.2: Aufmerksamkeitsgesteuertes Lernen von Sprache, aus [Yu03a]. (a) Die Sensorik des Systems. (b) Ein typisches Bild der Kamera.

Die Aufmerksamkeitsdetektion geschieht über Augen- und Kopfbewegungen. Fixationspunkte – Punkte, bei denen das Auge verharrt – werden mit Hilfe eines HMM detektiert. Die Geschwindigkeiten der Augen- und Kopfbewegungen gehen dabei als Merkmale ein. Für die Objektdetektion wird eine Farbsegmentierung des Bildes mit den Fixationspunkten als Startpunkte vorgenommen. Ausgehend von den Fixationspunkten findet ein "region growing" statt. Das so segmentierte Objekt wird durch Farb-, Form- und Texturhistogramme repräsentiert. Der sich aus der Konkatenierung der Histogramme ergebene Merkmalsvektor wird durch eine Hauptachsentransformation in der Dimension reduziert. Diese Merkmalsvektoren werden nun geclustert. Jeder dieser Cluster wird schließlich durch einen Vektor repräsentiert.

Um auch menschliches Verhalten zu untersuchen, werden Aktionen basierend auf Handbewegungen mit in die Untersuchung einbezogen. Mit der Annahme, daß Aktionen während Augen- oder Kopffixationen stattfinden, wird eine erste Segmentierung vorgenommen. Hieraus werden die Geschwindigkeit der Hand in zwei Ebenen und die Geschwindigkeit der Handrotation als Merkmalsvektoren extrahiert. Durch einen auf HMMs und *Dynamic-Time-Warping* (DTW [Sak78]) basierenden Clusterprozeß werden diese in sich möglichst stark unterscheidende Klassen eingeteilt.

Sprache wird mit Hilfe eines auf einer normalen Trainingsstichprobe trainierten englischen Phonemerkenners verarbeitet. Zum Vergleichen von Phonemketten werden die einzelnen Phoneme durch 15 binäre artikulatorische Eigenschaften dargestellt. Der Abstand zweier Phonemketten ergibt sich nun mit Hilfe dynamischer Programmierung aus dem Hammingabstand der Merkmale. Für eine erste Kategorisierung der Sprache

werden komplette Äußerungen den Objekten und Aktionen zugeordnet, in deren zeitlicher Nachbarschaft sie auftraten. Durch den Phonemkettenvergleich zwischen den einem Objekt zugeordneten Phonemen werden Kandidaten für Wörter gefunden, die eine in der Welt verankerte Bedeutung haben könnten.

Die finale Zuordnung von Wörtern zu Objekten oder Aktionen und damit die Verankerung der Wörter lehnt sich an der statistischen Übersetzung von Texten an. Hierbei soll von den Wörtern in eine "Bedeutungs"-Sprache bestehend aus den Objekten und Aktionen übersetzt werden. Dazu wird eine Abfolge $S_m^{(i)}$ einer Menge von Äußerungen $i \in \{1, ..., I\}$ bestehend aus einer Menge von "Bedeutungen" und eine Menge von Äußerungen $S_w^{(i)}$ bestehend aus einer Menge von Wörtern betrachtet. Es muß die Wahrscheinlichkeit P maximiert werden, daß die Abfolge $S_m^{(i)}$ aus einer Menge $S_w^{(i)}$ generiert wurde:

$$P(S_m^{(1)}, ..., S_m^{(I)} | S_w^{(1)}, ..., S_w^{(I)}) = \prod_{i=1}^{I} P(S_m^{(i)} | S_w^{(i)}) = \max \tag{2.3}$$

Dies läßt sich auf die Bestimmung von Zugehörigkeitswahrscheinlichkeiten $t(m_i|w_j)$ einer "Bedeutung" m_i und eines Wortes w_j zurückführen. Zur Lösung kommt der *Expectation-Maximization*-Algorithmus [Dem77] zum Einsatz. Selektiert werden schließlich diejenigen Wort-Bedeutungs-Paare, deren Wörter in den Äußerungen genügend häufig vorkommen und deren Zugehörigkeitswahrscheinlichkeiten t genügend hoch sind. Das in den letzten drei Absätzen vorgestellte Clustern von Objekten und Aktionen und das nachfolgende Verankern der Wörter geschieht in einem globalen Optimierungsschritt mit allen zur Verfügung stehenden Daten.

2.3.3 Sozial verankertes Lernen von Sprache

Steels und Kaplan legen in ihrem System einen Schwerpunkt auf das sozial verankerte Lernen von Sprache [Ste01], wo die Interaktion mit einem Menschen entscheidend für den Erfolg angesehen wird. Dabei hat diese Person, die sie Vermittler nennen, die Aufgabe, das Lernen zu lenken und sinnvoll einzuschränken, um so das eigentliche Ziel der Kommunikation zwischen dem Lernenden und dem Vermittler zu erreichen. Auch aktives Lernen, bei dem sich der Lerner an den Mediator wenden kann, ist möglich. Sprache hat dabei nicht nur die Aufgabe, Label für schon vergebene Kategorien zu vergeben. Die Kategorien werden durch Sprache auch beeinflußt und verändert.

Für ihr System benutzen sie einen Sony AIBO™, ein mobiler Roboter in Form eines Hundes, der unter anderem mit einer Kamera, einem beweglichen Kopf und beweglichen Beinen ausgestattet ist. Sprache wird durch einen externen sprecherunabhängigen Spracherkenner erkannt. Das Lernen neuer Objekte läuft in Interaktion mit dem System ab. Ein typischer Dialog ist folgender:

Benutzer: Look
Benutzer: Ball
Aibo : Ball?
Benutzer: Yes

Zur Verstärkung des sozialen Lernens sind die Anweisungen auf den jeweils nötigen Inhalt fokussiert. Vor dem "Look" zeigt der Benutzer dem System einen Ball und benennt ihn, nachdem das System das Objekt trackt. Das System testet das Verstandene durch die Nachfrage an den Benutzer. Auf diese Weise kann das System auch Fragen beantworten und Korrekturen von Fehlern entgegennehmen.

Die Objektklassifikation ist Instanzbasiert und klassifiziert mit Hilfe des Nächsten-Nachbar-Klassifikators. Als Merkmal für ein Objekt kommen farbnormalisierte Histogramme im rg-Farbraum zum Einsatz. Verglichen werden sie mit Hilfe des χ^2-Testes. Dabei wird das Objekt nicht vom Hintergrund segmentiert, das Histogramm wird immer für das komplette aufgenommene Bild erstellt. In einem assoziativen Speicher werden die Objekt-Wort-Paare abgelegt. In die Objekt-Wort-Bindung wird dabei eine positive und negative Rückmeldung des Benutzers mit einbezogen.

Dieses System erreichte mit drei Objekten unter verschiedenen Beleuchtungsverhältnissen eine Klassifikationsrate von circa 82%. Wurden die gleichen Daten dagegen nur durch den unüberwachten EM-Algorithmus geclustert, wurde maximal eine Klassifikationsrate von 47% erreicht. In einem dritten Experiment bewegte sich der Roboter autonom zwischen den drei Objekten. Wenn der Benutzer meinte, der Roboter schaute auf ein Objekt, wurde ihm der Objektname genannt. In diesem Szenario mit stark reduzierter sozialer Interaktion und fehleranfälligen Trainingsdaten, aber überwachtem Lernen, ergab sich eine Klassifikationsrate von 59%.

2.3.4 Objektidentifikation für einen mobilen Roboter

Für einen mobilen Roboter ist eine Objektidentifikation, die mit sich kontinuierlich ändernden Umgebungen und Objekten umgehen kann, eine wichtige Voraussetzung für viele weitere Anwendungen. Bredeche, Zucker und andere präsentieren so ein System [Bre03a, Bre03b]. Es ist in der Lage, vom Roboter aufgenommene Bilder danach zu klassifizieren, ob sie bestimmte Objekte enthalten oder nicht enthalten.

Zur Gewinnung von Trainingsmaterial kann das System, während es sich autonom bewegt, sowohl zufällig als auch auf Aufforderung eines Benutzers Bilder aufnehmen. Diese Bilder werden offline vom Benutzer mit den Namen der zu lernenden Objekte versehen. Aus diesem Trainingsmaterial werden dann Klassifikatoren generiert. Die Klassifikatoren basieren auf dem Vergleich der Farbe von einzelnen Pixeln. Daher ist es nötig, in den Bildern einer Klasse Teilbereiche zu finden, die in anderen Bildern ohne Objekte der Klasse nicht vorkommen. Dafür werden in verkleinerten Bildern einer maximalen Größe von 32x24 Pixeln initial einzelne Pixel im HSV-Farbraum als Klassifikator angesehen und durch Kreuzvalidierung getestet. Weitere Informationen

zum HSV-Farbraum finden sich in Anhang A. Die Klassifikatoren vergleichen dabei Pixel unter Einsatz von individuellen Schwellwerten für die drei Farbkanäle. Iterativ wird versucht, die Klassifikatoren um weitere benachbarte Pixel zu vergrößern, bis eine maximale Größe erreicht ist oder die Klassifikationsrate maximal ist. Mehrere Klassifikatoren eines Objektes werden schließlich für die Objektidentifikation in Kombination eingesetzt.

Die so gelernten Klassifikatoren können noch nicht auf Änderungen in ihrer Umgebung reagieren, Lernen während des normalen Einsatzes ist hierfür nötig. Dies wird auf zweierlei Art erreicht. Erstens werden die einzelnen Klassifikatoren eines kombinierten Klassifikators mit Gewichten versehen, die bei einer fehlerhaften Klassifikation verringert werden. Als zweites werden durchgehend neue mit Objektnamen versehene Bilder akquiriert. Sind hiervon genug vorhanden, werden in einem Offlineschritt neue Klassifikatoren generiert. Der neue kombinierte Objektidentifikator ergibt sich aus einer Teilmenge der besten bisher benutzten Klassifikatoren und weiteren neu trainierten Klassifikatoren. Somit wird eine weitestgehend kontinuierliche Anpassung des Erkenners an neue Gegebenheiten erreicht, ohne bisher gute Klassifikatoren aufgeben zu müssen.

2.3.5 Vergleich der Lernstrategien

In den letzten Abschnitten wurden vier Ansätze präsentiert, die alle unter anderem versuchen, das Problem der Verankerung von anfangs unbekannten Symbolen in der Welt zu lösen. Alle diese Ansätze setzen aber verschiedene Schwerpunkte und gehen auch zusätzliche Probleme an.

Sowohl Roy und Pentland als auch Yu und Ballard versuchen initiales Lernen von Sprache bei Kindern auf einer für ein Mensch-Maschine-System sehr elementaren Ebene zu modellieren. Sie gehen sowohl auf der Sprach- als auch auf der Bildseite von nur minimalem Vorwissen aus. Auf der Sprachseite ist dies ein fester Phonemerkenner und auf der Bildseite fest gewählte Merkmale. Alle Wörter und alle Objekte werden dynamisch gelernt. Bei Yu und Ballard ist dies durch Integration von Kopf- und Augenbewegungen stark aufmerksamkeitsgesteuert. Bei den zu lernenden Objekten gehen allerdings beide Ansätze davon aus, das sich die Objekte leicht vom Hintergrund segmentieren lassen. Der Hintergrund muß entweder statisch und einfarbig sein oder das Objekt muß sich bei bekannten Fixationspunkten durch “region growing” segmentieren lassen. Bei Yu und Ballard wird die Bearbeitung von Objekten und deren Verankerung mit der Sprache immer global über alle Daten behandelt, was dies problematisch für ein Onlinesystem macht.

Steels und Kaplan untersuchen zwar ebenfalls das Lernen bei Kindern, gehen dies aber mit deutlich anderen Schwerpunkten an. Sie untersuchen den Einfluß von sozialem Lernen für die Verankerung von Sprache und können dessen Wichtigkeit zeigen. Ihr System ist eher einfach aufgebaut. Sprache an sich setzen sie für diese Studie als fertig ausgebildet voraus, in ihren Experimenten benutzen sie einen vollwertigen

Spracherkenner. Ihr visuelles System ist mit nur einem globalen Histogramm über das komplette Bild sehr einfach gehalten, mehrere Objekte im Bild können sie beispielsweise nicht bearbeiten. Das Objekt muß auch immer den größten Teil des aufgenommenen Bildes einnehmen.

Bredeche und Zucker setzen im Vergleich zu den vorherigen drei Ansätzen deutlich andere Schwerpunkte. Ihnen geht es nicht um Spracherwerb. Sie wollen bei ihrem System in erster Linie ein technisches Problem lösen, das Identifizieren von Objekten für einen mobilen Roboter. Aber auch hierfür ist das Verankern neuer Symbole in der Welt nötig. Sie realisieren es durch eine Lernstrategie des Lernens durch Beispiele, indem sie in einem regelbasierten Verfahren Klassifikatoren im Batchbetrieb generieren.

2.4 Zusammenfassung und Resumé

In diesem Kapitel wurde das Problem des Lernens von Objekten aus drei verschiedenen Perspektiven beleuchtet:

1. aus der Sicht der Linguistik,
2. aus der Sicht der Informatik in Bezug auf den Lernaspekt und
3. aus der Sicht des Systems in Form bestehender Systeme.

Wie in Abschnitt 2.1.3 dargestellt, ergeben sich aus dem ersten Punkt verschiedene Konsequenzen für die in dieser Arbeit angestrebten Lösung des Lernproblems. Einerseits ist eine Verankerung neuer Objekte Voraussetzung eines erfolgreichen Systems. Und auch aus der Art der Objektbenennung ergeben sich verschiedene im genannten Abschnitt 2.1.3 aufgeführte Konsequenzen für eine Sprachkomponente.

Auch aus den nächsten beiden Punkten ergeben sich verschiedene Konsequenzen, die im weiteren dargelegt werden. Für eine effektive Anwendung von Wissen in einer dynamischen Welt ist sowohl Erwerb neuen Wissens als auch eine ständige inkrementelle Verbesserung nötig, wie dies sowohl durch den Menschen als auch durch verschiedene bestehende Systeme gezeigt wird. Weitere Anforderungen und Möglichkeiten ergeben sich aus dem Szenario. Insbesondere soll

- ein integriertes System entstehen,
- mit dem man intuitiv und natürlich interagieren kann.

Der erste Punkt klingt wenig ergiebig, hat aber wichtige Konsequenzen. Das Ziel ist hier nicht, ein Modell für das Lernen oder den Spracherwerb zu erstellen. Die Wissensbasis des Systems muß damit nicht komplett neu erstellt werden. Es kann jederzeit auf eine initiale vorgegebene Wissensbasis zurückgegriffen werden. Insbesondere im Bereich der Sprache kann dies ausgenutzt werden. Sprache ist sicherlich dynamisch und verändert sich ständig, eine “email” kannte vor wenigen Jahren praktisch niemand.

Auch benutzen Menschen je nach Herkunft, kultureller Einordnung oder auch Umfeld zum Teil außergewöhnliche Begriffe. Das in der Einleitung als Beispiel dienende Wort Kluntje für eine spezielle Art von Zucker ist nicht überall verbreitet. Veränderungen der Sprache finden aber meistens im Bereich von Jahren statt und betreffen jeweils nur kleinere Teile des Wortschatzes. Auch regionale Besonderheiten sind von eher statischer Natur.

Aktuelle Spracherkenner haben inzwischen einen Stand erreicht, bei dem sie auch mit sehr großen Lexika gute Erkennungsraten erzielen. Lexika mit über 20000 Wörtern sind inzwischen nicht mehr außergewöhnlich. Somit wird der wesentliche Grundstock einer Sprache schon abgedeckt. Die Erweiterung des Lexikons um weitere domänenspezifische und regionenspezifische Einträge und die Anpassung an aktuelle Veränderungen ist nach einer phonetischen Transkription der bisher nicht im Lexikon vorhandenen Wörter in einem Offline-Lernschritt relativ einfach möglich. Auch hiermit besteht natürlich immer noch das Problem unbekannter Wörter, die meistens aber in einer nicht zu allgemeinen Domäne selten genug sind, um mit dem Erkenner gut arbeiten zu können. Eine gute Einführung in die Problematik der Spracherkennung gibt beispielsweise [Hua01]. Kapitel 13 behandelt speziell die Erkennung mit großen Lexika.

Für die Sprache bietet es sich daher an, einen Spracherkenner zu benutzen und Wörter nicht explizit zu lernen. Ein Ansatz wie bei Roy oder Yu über einen Phonemerkenner ist damit nicht nötig. Dies hat einerseits den Vorteil, daß es keine Skalierungsprobleme durch unüberwachtes Lernen von Wörtern gibt. Anderseits erhält das System durch den großen vorhandenen Wortschatz die Möglichkeit, einen normalen Dialog mit dem Anwender zu führen.

Durch den zweiten oben aufgeführten Punkt der Intuitivität und Natürlichkeit werden gewisse Lernverfahren ausgeschlossen. Das Lernen ohne Inferenz wie auch das Lernen durch Unterweisung sind in dem gewünschten Szenario unnatürlich und erfordern einen erheblichen Aufwand vom Benutzer, was beides zu einer hohen Fehleranfälligkeit führt. In dieser Anwendung sind sie damit nicht geeignet. Ein erfolgversprechender Ansatz ist dagegen das auch von Steels und Kaplan eingesetzte soziale Lernen in Verbindung mit einem Lernen durch Beispiele. Lernen durch Beispiele wird auch von allen anderen im letzten Abschnitt 2.3 vorgestellten Systemen in verschiedenen Inkarnationen erfolgreich eingesetzt. Ein dialogbasiertes soziales Lernen ermöglicht es dem System, ähnlich wie ein Mensch zu reagieren, indem es beispielsweise weitere zur Erfüllung der jeweiligen Aufgabe nötige Informationen anfordert oder einfach verständliche sprachliche Bestätigungen gibt. Und nicht zuletzt steigert es, wie Steels und Kaplan gezeigt haben, die Lernrate, da sich in vielen Fällen die Qualität der Lernbeispiele erhöht.

Der Punkt der Natürlichkeit schließt neben bestimmten Lernverfahren auch aufwendige und ungewöhnliche mechanische Sensoren wie Eyetracker oder Handpositionssensoren aus. Da diese Sensoren vom Benutzer getragen werden, schränken sie ihn jeweils ein und machen die Kommunikation mit dem System aufwendiger und un-

natürlicher. Trotzdem bietet ein aufmerksamkeitsgesteuertes Lernen klare Vorteile, wie Yu und Ballard mit ihrer Arbeit gezeigt haben. Durch die visuelle Erkennung und Integration von Gesten und Handlungen kann aber eine andere, explizitere Art von Aufmerksamkeitssteuerung mit einer normalen Kamera erreicht werden. Da eine Kamera auch für das visuelle Lernen von Objekten benötigt wird, erhöht sich der Hardwareaufwand nicht. Auch bei einem Einsatz einer zweiten Kamera exklusiv für die Gestenerkennung ist der Hardwareaufwand weiterhin relativ gering.

Die Aufmerksamkeitssteuerung wird bei Yu und Ballard insbesondere auch für die Detektion und Segmentierung von Objekten benutzt. Segmentierung von unbekannten Objekten ist im allgemeinen Fall nur auf Grund von Aufmerksamkeit allerdings nicht möglich, da das zu lernende Objekt beispielsweise zum Teil ähnlich zum Hintergrund sein kann. Die korrekte Grenze zwischen Objekt und Hintergrund kann durch die Segmentierung dann nicht gefunden werden. Daher wird im hier vorgeschlagenen System ein interaktiver Ansatz favorisiert, der auf dem Gedanken des sozialen Lernens beruht. Weiter vorgestellt wird er in Kapitel 3. Auch beim Lokalisieren und Erkennen eines schon gelernten Objektes kann im Allgemeinen keine problemunabhängige Segmentierung zur Lokalisation interessanter Bereiche zum Einsatz kommen. Als Alternative wird hier eine Suche über das komplette Bild mit den gelernten Objektmodellen durchgeführt. Ähnlich wie Yu und Ballard vereinfachen sowohl Roy und Pentland als auch Steels und Kaplan die visuelle Verarbeitung, indem sie von einem einfarbigen Hintergrund oder sogar von nur einem sichtbaren Objekt ausgehen.

Sowohl Roy und Pentland als auch Steels und Kaplan setzen bei ihren Systemen auf ein rein inkrementelles Lernen. Es gibt keine eigenständige Lernphase, in der in einem Offlinebetrieb eine separate globale Exploration des Suchraums stattfindet. Dabei legen beide Gruppen auch großen Wert auf Echtzeitfähigkeit. Beides führt zu einer erhöhten Natürlichkeit bei der Interaktion mit dem System. Das System profitiert sofort von neu akquiriertem Wissen und kann es direkt einsetzen. Die Direktheit der Kommunikation ist sowohl dadurch als auch durch die kurzen Antwortzeiten des Systems erhöht. Der Nachteil gegenüber einer globalen Optimierungsphase wie bei Bredeche und Zucker und auch Yu und Ballard ist allerdings eine nicht optimale Ausnutzung der Trainingsbeispiele. Durch eine gemeinsame Betrachtung aller Beispiele können die maßgebenden Merkmale eines Objektes gefunden werden, falls durch die Beispiele eine gute Abdeckung des kompletten Suchraums gegeben ist. Zur Vermeidung der Akquirierung einer großen Anzahl von Trainingsbeispielen wird im hier vorgestellten System ein rein inkrementelles Lernen eingesetzt. Eine hin und wieder angestoßene globale Optimierung über alle bisher akquirierten Beispiele wäre allerdings als Ergänzung des vorgestellten Systems sinnvoll.

3 Ein System zum Lernen von Objekten

Ziel dieser Arbeit ist das Lernen neuer, dem System bisher unbekannter Objekte. Hierfür wurden im letzten Kapitel verschiedene Konsequenzen für einen Entwurf eines Systems hergeleitet, die sich aus der Sicht der Linguistik und der Informatik und aus schon bestehenden Systemen ergeben. Dabei hat sich ein multimodales System, welches Sprache, Bilder und Gesten verarbeiten kann und integrativ zur Optimierung der Systemperformanz einsetzt, als sinnvoll erwiesen. Eine schnelle Onlineverarbeitung der Daten ist dabei maßgebend für eine natürliche Interaktion mit dem System.

Im nächsten Abschnitt 3.1 wird zuerst das Szenario und die dort eingesetzte Sensorik näher vorgestellt. Im Abschnitt 3.2 wird die Architektur des Systems beschrieben, die sich basierend auf dem Szenario und den Überlegungen aus dem letzten Kapitel ergeben hat. Diese Architektur ist die Grundlage für alle weiteren hier vorgestellten Arbeiten. Der nächste Abschnitt 3.3 stellt das Verfahren zum Lernen neuer Objekte genauer vor. Die folgenden Kapitel gehen schließlich auf die in diesem Kapitel eingeführten Einzelmodule Sprache, Gestik und Merkmalsextraktion genauer ein.

3.1 Szenario

Aus den Überlegungen im letzten Kapitel ergibt sich ein Szenario, welches den Einsatz von Sprache und visuellen Merkmalen erlaubt und möglichst ohne Einsatz von Spezialhardware wie Eyetracker oder Positionssensoren auskommt. Dabei soll es vom Benutzer sowohl im Hinblick auf den Aufwand als auch auf die Komplexität einfach und intuitiv beherrschbar sein. Abbildung 3.1 zeigt ein typisches Szenario, welches aus diesen Überlegungen heraus gewählt wurde. Die Szene mit den zu lernenden Objekten wird von einer einzelnen nicht beweglichen Farbkamera beobachtet, die maximal 25 im YUV-Farbraum kodierte Farbbilder in der Sekunde aufnimmt. Weitere Informationen zum YUV-Farbraum finden sich in Anhang A. Die Bilder haben eine Tiefe von 8 Bit je Farbkanal. Die Kamera wird sowohl für die Objekterkennung als auch für die Gestenerkennung eingesetzt.

Wahlweise kann auch eine zweite Farbkamera speziell für die Gestenerkennung zum Einsatz kommen. In diesem Fall wird die erste Kamera nur für die Objekterkennung

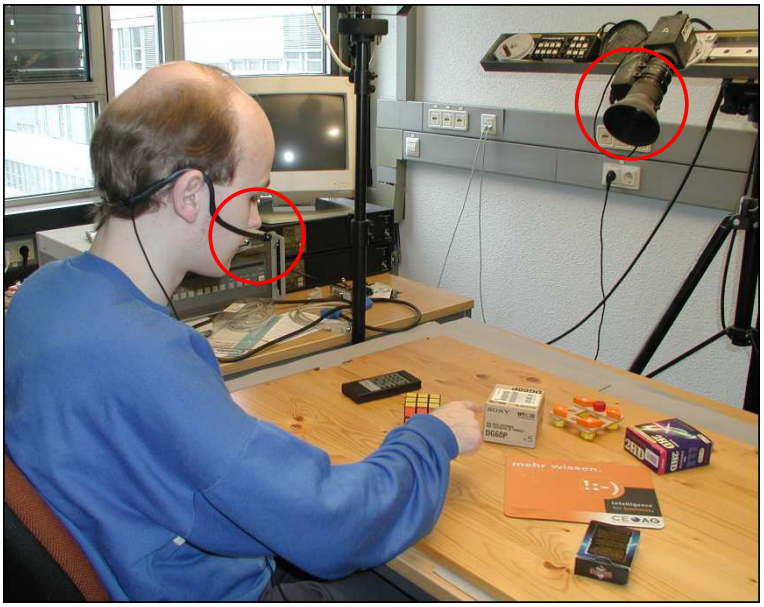


Abbildung 3.1: Beispielszenario für das Objektlernen. Die Szene wird von einer Farbkamera beobachtet. Der Benutzer trägt ein Nackenmikrophon für die Spracherkennung und kann Gesten während der Interaktion einsetzen.

eingesetzt. Durch die zweite Kamera kann ein alternativer und insbesondere größerer Ausschnitt der Szene beobachten werden, der eine sicherere Verfolgung der Hände erlaubt. Die zweite Kamera hat dabei die gleichen technischen Daten wie die erste Kamera, sie liefert maximal 25 im YUV-Farbraum kodierte Bilder von einer Tiefe von 8 Bit. Wie die erste Kamera zeigt auch diese einen festen unveränderlichen Ausschnitt. Der Hintergrund der Szene, in Abbildung 3.1 Holzplatten, ist nicht festgelegt. Er kann sich jederzeit beliebig ändern. Sprache wird über ein Nahbesprechungsmikrophon in Mono mit einer Quantisierung von 16 Bit und einer Abtastfrequenz von 16 Kiloherz aufgenommen. Im Bild ist hierfür ein Nackenmikrophon im Einsatz. Aus Ermangelung eines Manipulators werden Systemreaktionen über ein Paar Lautsprecher und zusätzlich visuell über einen Monitor ausgegeben.

3.2 Architektur des Gesamtsystems

Für ein größeres multimodales System ist die Architektur ein wichtiger Punkt, um eine konsistente und schnelle Verarbeitung der verschiedenen Eingabedaten zu erlauben. Im Laufe der Zeit sind verschiedene Paradigmen für den Entwurf einer Architektur entstanden. Beispiele hierfür sind Blackboard-Architekturen [Eng88] und verhaltensbasierte Systeme [Ark98, Bro91]. Bei Blackboard-Architekturen ist das Black-

board, auf dem inkrementell die Lösung des zu bearbeitenden Problems entsteht, die maßgebliche Einheit. Verschiedene Wissensquellen greifen auf dieses kontrolliert über eine zentrale Kontrolleinheit zu. In verhaltensbasierten Systemen wird dagegen Wissen nicht explizit an einer Stelle abgelegt. Das Verhalten und das Wissen des Systems ergeben sich implizit aus einer Menge von konkurrierend aktivierten Verhaltensmustern, die reaktiv auf Eingaben reagieren. Weiterhin können Formalismen wie Frames, Prädikatenlogik oder auch semantische Netze benutzt werden, um Wissen in homogener und strukturierter Form darzustellen und auf dieses Wissen zuzugreifen [Cha85]. Durch den Einsatz einer problemunabhängigen Kontrollstrategie, wie sie in [Sag97] für semantische Netze vorgestellt wird, vereinfacht sich in so einer homogenen Architektur der Zugriff auf Daten und die Nutzung der Daten.

Aufbau und Funktionalität des Systems

Das hier präsentierte System läßt sich nicht direkt einem dieser Paradigmen zuordnen. Sowohl verteilte als auch lokal zentrale Wissensquellen kommen zum Einsatz. Eine deliberative zentrale Kontrolleinheit steuert einen Teil des Systems. Andere Teile des Systems arbeiten unabhängig und asynchron von der zentralen Kontrolle und stellen nur Ergebnisse auf Anfrage zur Verfügung. Abbildung 3.2 zeigt eine Übersicht über die Architektur des Systems. Ein Sprachmodul untersucht kontinuierlich die Äußerungen des Benutzers und gibt semantisch analysierte Auswertungen der Äußerungen an den Dialog weiter. Details hierzu finden sich in Kapitel 4. Gleichzeitig detektiert ein Handverfolgungsmodul kontinuierlich die Position von Händen im Bild und stellt Trajektorien dieser Hände zur Verfügung. Dieses Modul wird zusammen mit der Gestenauswertung in Kapitel 5 vorgestellt.

Der Dialog reagiert auf zwei Arten von sprachlichen Anfragen. Einerseits kann der Benutzer auf den Zustand des Systems Einfluß nehmen. Realisiert ist hier momentan das interaktive Trainieren des im Modul Handverfolgung benutzten Hautfarbklassifikators. Weitere Details hierzu finden sich in Kapitel 4 und in Abschnitt 5.3.4. Die zweite Anfragemöglichkeit behandelt das eigentliche Thema, das Lernen und Lokalisieren von Objekten. In dieser Form der Anfrage wird ein Objekt spezifiziert. Wurde eines spezifiziert, wird zuerst ein neues Bild von derjenigen Kamera angefordert, die für die Objekterkennung zuständig ist. Innerhalb des Bildes wird ein zu der Anfrage korrespondierendes Objekt gesucht. Dabei wird eine gegebenenfalls ausgeführte deiktische Geste mit integriert, indem bevorzugt in der Umgebung des Ziels der Geste gesucht wird. Für die Suche werden die verschiedenen Module zur visuellen Objekterkennung angestoßen, die daraufhin basierend auf jeweils speziellen visuellen Merkmalen Objekthypothesen generieren. Die Erkenner und die von ihnen eingesetzten Merkmale werden in Kapitel 6 näher vorgestellt. Die Kontrolle hat über diese Objekterkenner nur minimales Wissen. Sie weiß nur, in welcher Reihenfolge die Erkenner angestoßen werden müssen, um ihnen wahlweise ein hierarchisches Arbeiten zu ermöglichen, und wie die zurückgelieferten Hypothesen aussehen. Dies macht es besonders einfach,

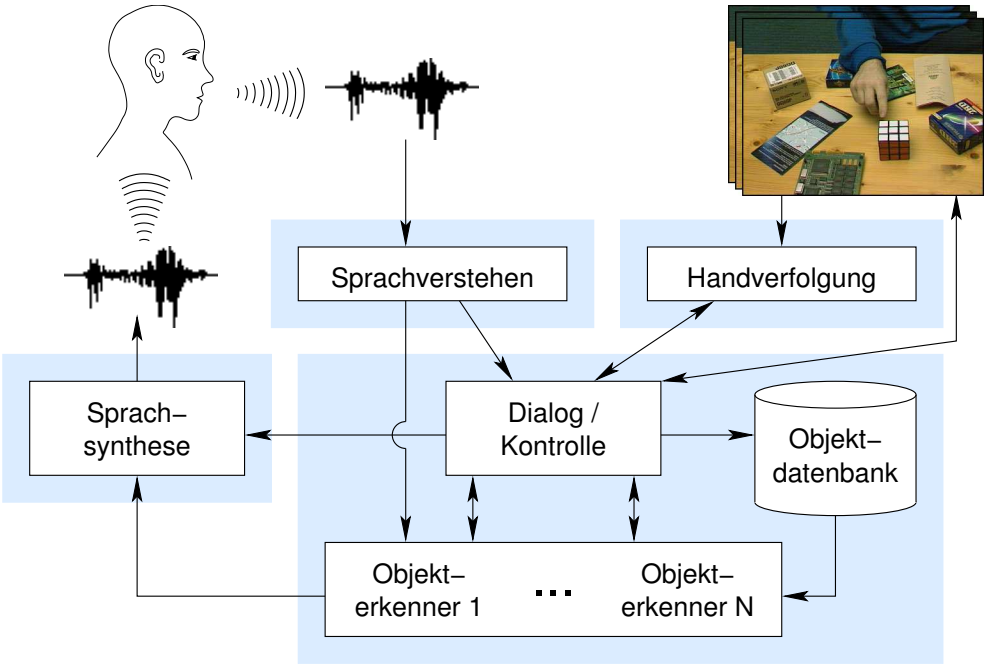


Abbildung 3.2: Architektur des Systems zum Objektlernen. Der Benutzer kann Sprache und Gesten benutzen, um auf Objekte zu referenzieren. Das System lernt neue Objekte durch die Kombination dieser Informationen mit visuell basierten Objekterkennern.

neue Erkenner in das System zu integrieren oder auch dynamisch je nach Situation die Erkenner zu aktivieren, zu deaktivieren oder auch zu modifizieren.

Werden die Objekterkenner angestoßen, holen sie von der Objektdatenbank Merkmale, die das sprachlich intendierte Objekt beschreiben. Mit Hilfe der erhaltenen Merkmale wird das Bild nach dem Objekt abgesucht. Bei Erfolg werden Objekthypothesen erzeugt, die neben einer Bewertung eine Wahrscheinlichkeitsverteilung über die vermutete Objektposition und eine Annahme über die Größe des Objektes enthalten. In der Kontrolle wird versucht, Hypothesen, die von verschiedenen Objekterkennern erzeugt wurden, zu verschmelzen. Ist die Bewertung der besten der sich ergebenen Hypothesen über einem Schwellwert, wird diese Hypothese als das intendierte Objekt angesehen und dem Benutzer mit Hilfe des Synthesemoduls mitgeteilt.

Ist die Bewertung dagegen zu schlecht oder wird das Objekt durch den Benutzer explizit durch eine Anweisung zurückgewiesen, konnte das Objekt nicht gefunden werden. Es muß daher entweder neu gelernt werden oder die in der Datenbank befindliche Repräsentation des Objektes muß verbessert werden. Dazu wird das Objekt im Dialog mit dem Benutzer gelernt, wobei sprachliche, visuelle und aktorische Informationen ausgenutzt werden. Weitere Details hierzu finden sich im nächsten Abschnitt 3.3 und im Kapitel 4. Weiterhin wird die sprachliche Reaktion des Benutzers auf das gefun-

dene Objekt ausgenutzt, um die in der Datenbank befindliche Objektrepräsentation zu modifizieren und zu optimieren. Weiteres hierzu findet sich in Kapitel 7. Während des Lernens wird eine neue Ansicht des Objektes generiert und in der Datenbank abgelegt. Eine Ansicht eines Objektes besteht aus den berechneten Merkmalen aller Objekterkenner. Die Datenbank speichert eine uneingeschränkte Anzahl von Objekten, die jeweils aus einer uneingeschränkten Anzahl von Ansichten bestehen können. Sie bietet damit die nötigen Voraussetzungen, um eine Abbildung zwischen Perzepten und Symbolen auf Basis einer ansichtenbasierten Objekterkennung zu ermöglichen.

Verteilte Systemarchitektur

Das vollständige präsentierte perzeptive System besteht aus verschiedenen zum Teil sehr rechenintensiven Modulen. Trotzdem ist eine schnelle Verarbeitung nötig, um einerseits kurze Latenzzeiten auf Benutzeranfragen zu erreichen und anderseits Sprache und Gestik in Echtzeit verarbeiten zu können. Um dies auch flexibel an vorhandene Hardware anpassen zu können, arbeiten einzelne Teile des Systems als eigenständige Prozesse, die mit Hilfe eines Kommunikationsframeworks interagieren. Dazu kommt DACS, das *Distributed Application's Communication System*, zum Einsatz [Jun98]. Alle Kommunikationsparadigmen innerhalb von DACS basieren auf asynchronem *Message Passing*. Weiterhin bietet DACS *Remote Procedure Calls* (RPC), um eine Funktion systemweit aufrufbar zu machen und *Demand Streams* für eine datengetriebene Kommunikation, bei der ein Prozeß Daten unter einem Namen für eine beliebige Anzahl von anderen Prozessen zur Verfügung stellt. Diese beiden Paradigmen sind intern mit Hilfe des *Message Passing* realisiert.

Alle blau unterlegten Teile in Abbildung 3.2 sind als eigenständige Prozesse realisiert. Die kontinuierlich arbeitenden Module Sprachverstehen und Handverfolgung stellen ihre Daten, analysierte Äußerungen und Handtrajektorien, über Demand Streams zur Verfügung. Das Handverfolgungsmodul bietet zusätzlich ein RPC-Interface, um aktuelle Bilder der Kamera weiterzureichen und ein Neutraining des Klassifikators anzustoßen. Kommen zwei Kameras zum Einsatz, steht ein zweites RPC-Interface für die zweite Kamera zur Verfügung. Das Sprachsynthesemodul nimmt normalen ASCII-Text in Form von Messages entgegen. Damit sind alle netzweit verteilten Daten, die kontinuierlich versandt werden, so weit als möglich durch eine semantisch dichte Repräsentation komprimiert, was eine geringe übertragene Datenmenge und damit einen geringen Zeitverlust durch die Kommunikation zur Folge hat.

Das Handverfolgungsmodul und das Hauptmodul bestehend aus der Kontrolle mit dem Dialog und den Objekterkennern sind mit Hilfe von iceWing realisiert. iceWing ist eine graphische Shell, die mit Hilfe von Plugins erweitert werden kann. Den Plugins stehen neben verschiedenen Hilfsroutinen Funktionen für eine sehr einfache Oberflächengenerierung, zur Visualisierung von Ergebnissen und zur Kommunikation zwischen Plugins zur Verfügung. Damit ist es möglich, sich bei der Softwareentwicklung auf den eigentlichen Algorithmus und seine Optimierung zu konzentrieren. Nebenge-

ordnete Aufgaben, insbesondere die Erstellung einer Oberfläche, werden im wesentlichen automatisch erledigt. iceWing wird in Anhang B näher vorgestellt.

3.3 Lernen neuer Objekte

Konnte eine Anfrage des Benutzers nach einem Objekt nicht korrekt beantwortet werden, versucht das System, seine ansichtenbasierte Repräsentation des Objektes zu verbessern. Auf Grund des Fehlers bei der Lokalisation des Objektes ist bekannt, daß zumindest das korrekte Aussehen des Objektes im zuletzt aufgenommenen Bild nicht bekannt ist. Dies schließt insbesondere auch die korrekte Lage, die Größe und auch die Form des Objektes mit ein. Daher wird auf die Hilfe des Benutzers zurückgegriffen, um korrektes Trainingsmaterial zu bekommen. Der Benutzer könnte auf verschiedene Art helfen:

- Er könnte das zu suchende Objekt näher beschreiben. Dies kann insbesondere dann helfen, falls das Objekt schon bekannt ist. Gegebenenfalls läßt sich durch weitere Informationen in Kombination mit den schon bekannten Ansichten des Objektes auf die im Bild befindliche Ansicht schließen, so daß sich diese aus dem Bild extrahieren läßt. Insbesondere bei neuen und bei komplexer texturierten Objekten und bei der Benutzung abstrakter Objektmerkmale ist dies aber fehlerträchtig und damit im Allgemeinen nicht praktikabel.

- Er könnte auf das Objekt zeigen. Dadurch wird ein Punkt des Objektes bekannt. Die Ausdehnung des Objektes bleibt aber weiterhin unbekannt, da eine korrekte Segmentierung eines unbekannten Objektes von einem ebenfalls unbekannten Hintergrund nicht möglich ist. Auch wenn man sich auf nur fest verbundene Objekte beschränkt, kann ohne weitere Hilfsmittel beispielsweise nicht entschieden werden, ob zwei Teile, die nebeneinander liegen, zu einem Objekt fest verbunden sind oder zu zwei getrennten Objekten gehören.

- Er könnte die Kontur des Objektes beispielsweise mit der Hand in der Szene anzeigen. Dieses Verfahren ist bei vielen Objekten anwendbar, falls eine sehr genaue Gestenerkennung und Hand- beziehungsweise Fingerverfolgung zur Verfügung steht. Werden die Objekte filigraner oder die Erkennung ungenauer, kommt es allerdings zu Fehlern bei der Konturbestimmung. Außerdem ist dieser Ansatz auch eher unbequem und aufwendig für den Benutzer.

- Er könnte das zu lernende Objekt aus der Szene herausnehmen oder es innerhalb der Szene verschieben und dem System durch die Veränderung in der Szene Hinweise geben.

Die letztgenannte Möglichkeit läßt sich nicht auf alle Objekte anwenden – das Objekt muß sich dafür einfach bewegen lassen. Bei vielen Objekten ist die Methode für den

Benutzer aber sehr einfach durchführbar. Durch die Untersuchung der Unterschiede der Szene vor und nach dem Verschieben des Objektes kann dessen Position und Lage damit bestimmt werden. Da diese Methode einen guten Grundstock zum Lernen von Objekten bietet, ist es in dieser Arbeit die Methode der Wahl. Wie in Abschnitt 7.2 beschrieben, kann der Benutzer aber auch zusätzlich sprachliche Attribute zum näheren Einschränken auf mögliche Objekte einsetzen.

Somit wird der Benutzer gefragt, das unbekannte Objekt aus der Szene zu nehmen und Bescheid zu geben, wenn er damit fertig ist. Damit kann er sich beliebig Zeit nehmen, ohne daß es deswegen zu Fehlern kommen kann. Ist er fertig, wird ein zweites Bild von der Kamera angefordert und das Differenzbild D zwischen nachbearbeiteten Versionen der beiden vorhandenen Bilder $I(t-1)$ und $I(t)$ betrachtet. Die Abbildungen 3.3 (a) und (b) zeigen ein Beispiel für die Bilder $I(t-1)$ und $I(t)$. In diesem Fall sollte die Videokassette neu gelernt werden. Durch das Differenzbild sollen nun die Form und die Position des herausgenommenen Objektes bestimmt werden. Da nur die Form und Position des Objektes bestimmt werden soll, kann bei der Nachbearbeitung der Bilder die Textur verändert werden, ohne das Ergebnis negativ zu beeinflussen. Um Rauschen der Kamera zu unterdrücken, kann daher für beide Bilder ein Gaußfilter zum Glätten eingesetzt werden, auch wenn dieser beispielsweise hohe Kontraste nicht erhält [Gon02, Kapitel 4.3.3].

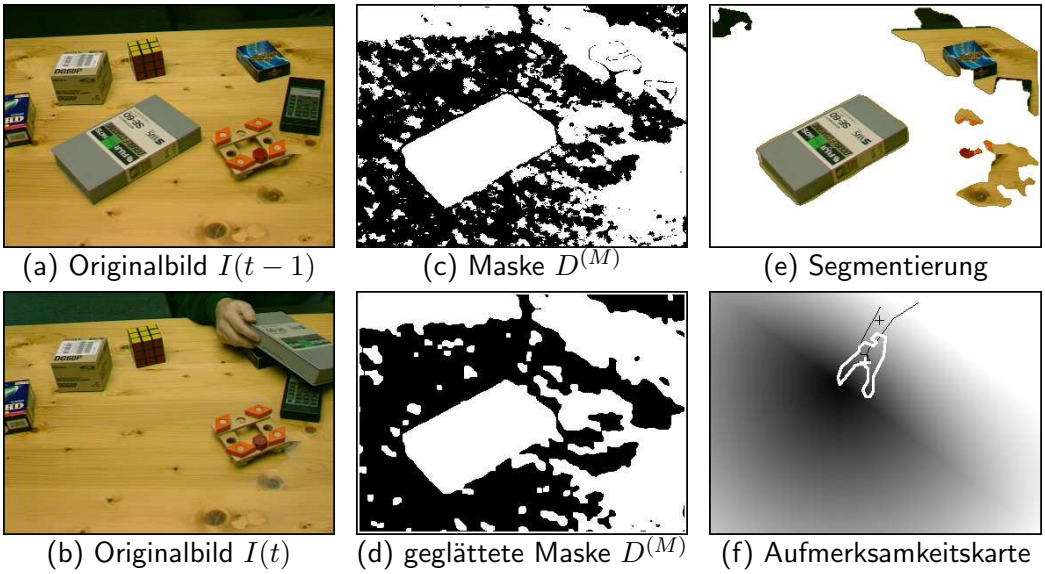


Abbildung 3.3: Beispielbilder zum Lernen eines neuen Objektes. Die Videokassette sollte durch Herausnehmen aus der Szene gelernt werden (a, b). Durch die Bestimmung von Hintergrundpixeln zur Beleuchtungskorrektur (c, d) und der Verfolgung der Greifaktion des Benutzers beim Herausnehmen des Objektes (f) kann das Objekt korrekt segmentiert werden (e).

Weiterhin kann es zwischen der Aufnahme der beiden Bilder zu globalen Helligkeitsveränderungen innerhalb der Szene gekommen sein, die den günstigsten Schwellwert für die Berechnung des Differenzbildes D beeinflussen. Auch bei einer Beschränkung bei der Berechnung des Differenzbildes auf die Farbkanäle U und V wirken sich Helligkeitsveränderungen weiterhin auf das Differenzbild aus. War dies der Fall, kann als erste Näherung angenommen werden, daß alle Teile des Bildes im wesentlichen ähnlich betroffen sind. Abweichungen hiervon können durch Schatten und verschiedene Reflexionseigenschaften unterschiedlicher Materialien entstehen. Unter Vernachlässigung dieser speziellen Probleme sollte es damit Unterschiede zwischen den Bildern geben, die entweder auf die globalen Helligkeitsveränderungen zurückzuführen sind oder auf physikalische Modifikationen in der Szene beruhen. Dabei haben die globalen Änderungen im wesentlichen kleinere, aber merkbare Auswirkungen. Um deren Stärke zu bestimmen, wird davon ausgegangen, daß mindestens 33% der Szene unberührt von Objektverschiebungen bleiben. Sind diese 33% bestimmt, kann aus ihnen die Helligkeitsveränderung bestimmt werden. Zur Bestimmung dieser 33% wird ein Differenzbild $D'^{(M)}$ zwischen den beiden Bildern berechnet, welches abgesehen von einer nachfolgenden Binarisierung auch dem endgültigen Differenzbild D entspricht:

$$D'^{(M)}_{x,y} = |I^{(U)}_{x,y}(t-1) - I^{(U)}_{x,y}(t)| + |I^{(V)}_{x,y}(t-1) - I^{(V)}_{x,y}(t)| \tag{3.1}$$

Ein Schwellwert $T^{(M)}$ wird so gewählt, daß die Werte von 33% der Pixel von $D'^{(M)}$ kleiner $T^{(M)}$ sind. Mit diesem Schwellwert wird eine Binarisierung von $D'^{(M)}$ durchgeführt:

$$D^{(M)}_{x,y} = \begin{cases} 255 & \text{falls } D'^{(M)}_{x,y} \leq T^{(M)} \\ 0 & \text{sonst} \end{cases} \tag{3.2}$$

Dabei ist 255 der Maximalwert bei den im System eingesetzten 8 Bit Bildern. Unter der Annahme, daß das zu findende Objekt eine eher größere zusammenhängende Fläche einnimmt, die sich im Vergleich zum Rest stärker geändert hat, wird diese Maske $D^{(M)}$ mit einem nichtlinearen Medianfilter [Gon02, Kapitel 3.6.2] geglättet. Eine nachfolgende Closing-Operation [Gon02, Kapitel 9.3] – eine morphologische Operation bestehend aus einer Dilatation gefolgt von einer Erosion – erweitert die Maske um weitere Hintergrundpixel. Die größeren Bereiche außerhalb der Maske sind von diesen Operationen nicht betroffen. Die Abbildungen 3.3 (c) und (d) zeigen die aus den Originalbildern (a) und (b) berechnete Maske $D^{(M)}$ vor und nach der Glättung. Alle schwarzen Bereiche sind hier Hypothesen für Hintergrundpixel.

Die Pixel innerhalb der Maske $D^{(M)}$, das heißt die Menge der Pixel $P^{(M)} = \{(x,y)|D^{(M)}_{x,y} = 255\}$, sind nun mit hoher Wahrscheinlichkeit Hintergrundpixel. Der mittlere Differenzwert dieser Pixel gibt eine gute erste Abschätzung für eine globale Helligkeitsveränderung. Der Schwellwert T für das Differenzbild D ergibt sich damit mit Hilfe eines vorgegebenen Anteils $T^{(S)}$ zu:

$$T = T^{(S)} + \frac{1}{|P^{(M)}|} \sum_{(x,y)\in P^{(M)}} D'^{(M)}_{x,y} \tag{3.3}$$

Bei dem Beispiel aus Abbildung 3.3 hat sich durch diese Berechnung eine Korrektur von fünf des initial mit sieben gewählten Schwellwertes $T^{(S)}$ ergeben. Das für die Objektfindung benötigte finale Differenzbild D wird nun durch Binarisierung aus $D'^{(M)}$ gewonnen:

$$D_{x,y} = \begin{cases} 255 & \text{falls } D'^{(M)}_{x,y} > T \\ 0 & \text{sonst} \end{cases} \tag{3.4}$$

Das vorgeschlagene Verfahren setzt einen globalen Schwellwert für die Binarisierung des Differenzbildes ein. Dieser wird zum Teil fest gewählt und zum anderen Teil problemabhängig angepaßt. Neben diesem Verfahren gibt es viele andere Ansätze. [Ots79] beispielsweise stellt einen Ansatz vor, bei dem ein globaler Schwellwert basierend auf dem Histogramm des zu binarisierenden Bildes automatisch gewählt wird, indem eine Intergruppenvarianz maximiert wird. Diese datengetriebenen Ansätze bringen in dieser Anwendung allerdings schlechtere Ergebnisse als selbst ein fest gewählter Schwellwert, da sie das Wissen, daß es im wesentlichen nur lokale Änderungen in der Szene gab, die lokalisiert werden sollen, nicht integrieren. Auch ein mit zwei Schwellwerten arbeitendes Hystereseverfahren wie es auch bei der Detektion von Kanten eingesetzt wird [Can86] bringt keine Vorteile. Die implizite Glättung und Verkettung, die hier durchgeführt wird, ist in dieser Anwendung durch eine nachgeschaltete Glättung sicherer zu erreichen, da Wissen über die Kompaktheit des gesuchten Objektes vorliegt.

Wie oben angegeben, wird angenommen, daß das zu findende Objekt eine eher größere zusammenhängende Fläche einnimmt. Um damit Segmentierungsfehler zu verringern, werden wieder ein Medianfilter und daraufhin eine Closing-Operation auf das Differenzbild D angewendet. Abbildung 3.3 (e) zeigt das erhaltene Differenzbild als Maske angewendet auf das Originalbild aus Abbildung 3.3 (a). Wie zu sehen ist, wurde die Videokassette korrekt segmentiert. Aber auch andere Teile des Bildes wurden segmentiert. Zum Teil beruhen diese auf zu starken lokalen Beleuchtungsänderungen, die von der konservativ agierenden Schwellwertanpassung nicht erfaßt wurden. Zum Teil gab es aber auch noch andere Änderungen in der Szene.

Normalerweise weiß der Benutzer nicht, welchen Teil der Szene das System exakt sieht. Somit kommt es vor, daß er das zu lernende Objekt nicht komplett aus dem Sichtfeld des Systems entfernt, auch wenn er hierzu aufgefordert wird. Manchmal wird er es nur an eine andere Stelle verschieben oder er verändert die Szene auf andere Weise, beispielsweise durch Hand- oder andere Körperbewegungen. Somit muß aus einer Menge von Szenenänderungen die korrekte Änderung selektiert werden. Als weitere Informationsquelle für diese Selektion neben dem Differenzbild D eignet sich die Greifaktion, die der Benutzer für das Herausnehmen durchführen muß. Dazu wird die Hand des Benutzers zwischen dem Zeitpunkt der Aufforderung, ein Objekt aus der Szene zu nehmen und der Bestätigung des Benutzers verfolgt. Durch Analyse der hierbei gewonnenen Handtrajektorie erhält man eine Hypothese über die Position, an der der Benutzer ein Objekt gegriffen hat und wie er es danach aus der Szene genommen hat. Das weitere exakte Verfahren hierzu wird in Kapitel 5 und

insbesondere Abschnitt 5.4.3 vorgestellt. Das Ergebnis dieses Verfahrens ist eine Aufmerksamkeitskarte [Rae00], die für jede Position des Bildes eine Wahrscheinlichkeit dafür spezifiziert, daß es das Zentrum der Greifaktion war. Abbildung 3.3 (f) zeigt die Aufmerksamkeitskarte für die in diesem Beispiel durchgeführte Greifaktion. Je dunkler die Karte ist, desto wahrscheinlicher lag hier das Zentrum der Greifaktion.

Um die Informationen der Aufmerksamkeitskarte auszunutzen, werden zuerst durch eine Kantenverfolgung zusammenhängender Pixelbereiche Regionen im Differenzbild D bestimmt [Gon02, Kapitel 2.5.2]. Dabei werden Polygonzüge der Regionen, deren Schwerpunkte und deren Größen berechnet. Alle Regionen, die mindestens eine Größe von 40% der größten Region haben, werden als mögliche Kandidaten für die Objektregion angesehen. Für die Auswahl der Region wird die Aufmerksamkeitskarte an der Position der Regionenschwerpunkte betrachtet. Die Region mit dem größten zugeordneten Wert der Aufmerksamkeitskarte wird ausgewählt. Falls mehrere Regionen den gleichen maximalen Wert haben, wird die größte Region unter diesen Regionen genommen. Es wird nun angenommen, daß die ausgewählte Region das zu lernende Objekt enthält und dicht umschließt. Damit werden auf dieser Region die verschiedenen Objekterkenner angestoßen, um neue Objektmerkmale zu berechnen. Näheres hierzu findet sich in Kapitel 6. Die Objektmerkmale werden schließlich innerhalb des Kontrollmoduls zu einer neuen Ansicht des zu lernenden Objektes kombiniert und in der Datenbank gespeichert.

3.4 Zusammenfassung

Dieses Kapitel hat das Szenario und die Architektur eines Systems zum Lernen von bisher unbekannten Objekten vorgestellt. Das Szenario des Systems wurde im Hinblick auf eine einfache und intuitive Beherrschbarkeit durch den Benutzer gewählt. Eine oder zwei Kameras beobachten eine Menge von Objekten. Der Benutzer kann Sprache und Gestik im Rahmen eines Dialogs mit dem System zur Referenzierung der Objekte benutzen. Ein Sprachmodul und eine Gestenerkennung untersuchen dabei kontinuierlich Eingaben des Benutzers. Eine deliberative zentrale Kontrolle reagiert auf Spracheingaben, stößt Objekterkenner an und gibt mit Hilfe einer Sprachsynthese Systemantworten an den Benutzer.

Kann das System eine Anfrage nach einem Objekt nicht beantworten, lernt es in Interaktion mit dem Benutzer eine neue Ansicht des Objektes. Der Benutzer entfernt dazu das Objekt aus der Szene. Durch ein Differenzbild mit einer Adaption an eine globale Helligkeitsänderung kann das Objekt vom Hintergrund segmentiert werden. Die Greifaktion des Benutzers wird als ein weiterer Hinweis auf die korrekte Region mit integriert.

4 Sprache und Dialog

Sprache ist für uns Menschen ein sehr natürliches Kommunikationsmittel. Treten wir in Interaktion mit anderen Personen, ist häufig Sprache als maßgebendes Mittel involviert. Es wird zur Übermittlung von Fragen und Antworten auf die Fragen, zur Vermittlung von Wissen, zur zwanglosen Unterhaltung oder auch einfach zur Gewinnung von Aufmerksamkeit eingesetzt. Sprache kann sowohl in mündlicher als auch in schriftlicher Form benutzt werden. Besteht direkter Kontakt zwischen den Kommunikationspartnern, wird nur in wenigen Ausnahmefällen die Schriftform als alleiniges Mittel gewählt. Üblicherweise wird die einfacher und schneller anwendbare mündliche Form bevorzugt. Im Sinne der Natürlichkeit des Interfaces ist auch bei der Mensch-Maschine-Kommunikation ein Einsatz einer Dialogkomponente mit mündlicher Interaktionsmöglichkeit wünschenswert. Wie in Abschnitt 2.4 dargestellt, ist dies auch aus der Sicht des zu lösenden Problems, dem interaktiven Objektlernen, sinnvoll.

In den folgenden Abschnitten wird das realisierte sprachverarbeitende System vorgestellt. Dazu wird zuerst eine Einführung in die Thematik gegeben. In den folgenden Abschnitten werden dann die Einzelkomponenten Spracherkennung, Semantikanalyse, Dialogführung und Sprachausgabe näher vorgestellt.

4.1 Sprachverarbeitung – Ein Überblick

Für die Verarbeitung des Sprachsignals bis zur Sprachausgabe werden verschiedene Verarbeitungsschritte benötigt. Abbildung 4.1 zeigt einen typischen Aufbau eines sprachverarbeitenden Systems. In einer ersten Phase, der Spracherkennung, wird ein Sprachsignal des Benutzers in eine textuelle Darstellung überführt. Eine syntaktische und semantische Analyse bestimmt domänenspezifisch bedeutungstragende Teile in der Äußerung und klassifiziert diese. Der nachfolgende Dialog bringt die Teile in einen zeitlichen Kontext und verwaltet die Kommunikation mit dem Anwender. Basierend auf dem Zustand des Dialogs wird schließlich eine textuelle Antwort für den Benutzer generiert, die durch eine Sprachsynthese in ein Sprachsignal überführt wird.

Sprachverarbeitende Systeme lassen sich nach ihrer Anwendung in verschiedene Klassen einteilen. *Kommandosysteme* haben meistens nur einen kleinen Wortschatz und erkennen nur Einzelworte. Beispielsweise kann mit einem einzelnen “stopp” ein Wecker ausgeschaltet werden oder mit wenigen Einzelworten ein Menü sprachlich be-

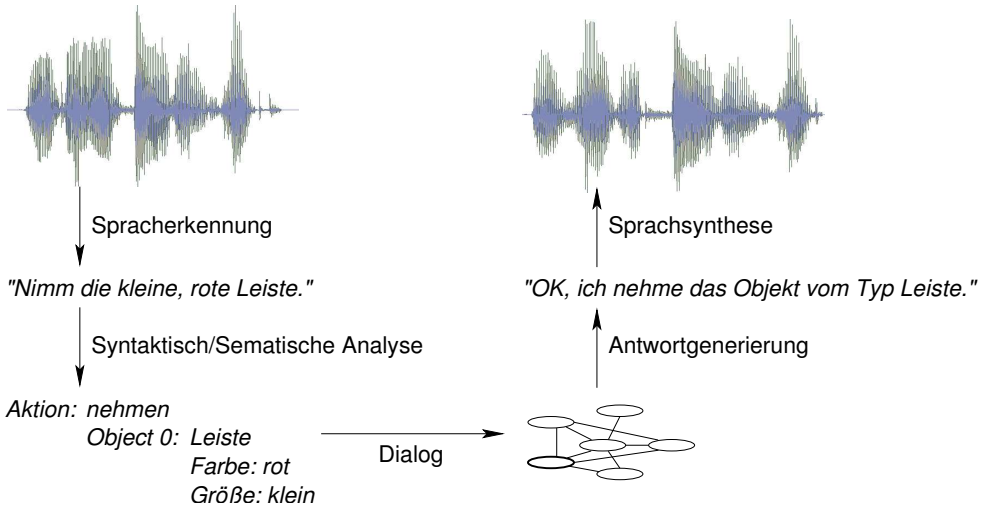


Abbildung 4.1: Typischer Aufbau eines dialogbasierten Systems zur Verarbeitung gesprochener Sprache in der Mensch-Maschine-Kommunikation.

dient werden. Arbeiten an solchen Systemen reichen weit zurück, in [Dav52] wurde noch eine analoge Schaltung zur Erkennung von einzelnen Ziffern eingesetzt. *Diktiersysteme* erlauben das Eingeben von Texten mittels der Sprache und fungieren damit als Ersatz für die Tastatur. *Auskunftssysteme* erlauben es dem Benutzer sprachlich Anfragen an eine Datenbank zu stellen. Dabei sammelt das System in einem Dialog inkrementell alle nötigen Daten für eine Anfrage an und gibt schließlich das Ergebnis der Anfrage zurück. Systeme dieser Art werden beispielsweise in [Zue00] und [Kel97] vorgestellt. [Bra99b] führt zusätzlich den Begriff *integrierte Systeme* für Systeme ein, die in Kooperation mit anderen Modulen in multimodaler Form arbeiten. Im Dialog kann sich der Benutzer hier auf andere Modalitäten beziehen oder diese implizit voraus setzen. Bei "Nimm das Teil." ist sowohl ein Gestikmodul als auch ein Objekterkenner für eine korrekte Auswertung nötig. Außer in [Bra99b] wird auch in [All01] ein Beispiel für so ein System vorgestellt.

4.1.1 Anwendung auf die Domäne

Bei der Interaktion mit dem in Kapitel 3 vorgestellten System kann der Benutzer Objekte benennen und mit ihnen auch gestisch Interagieren und weiterhin direkt auf das System Einfluß nehmen. Dafür ist sowohl eine direkte Kopplung der Gestik als auch der Objekterkennung an den Dialog nötig, um diese zusätzlichen Informationsquellen effektiv zu nutzen. Nach der obigen Taxonomie ergibt sich somit ein integriertes System. Aus diesem Szenario ergeben sich auch verschiedene Anforderungen und Designziele an die Sprachkomponente:

Robustheit: Eine einfache, möglichst natürliche Interaktion war ein wichtiges Kriterium. Um dies zu erreichen, ist insbesondere ein robustes Systemverhalten wichtig. Dies bedeutet natürlich auch, daß eine möglichst hohe Erkennungsrate für die Sprache anzustreben ist. Daneben sollte die Sprachkomponente es aber auch immer vermeiden, mit einer Fehlermeldung auszusteigen. Ist die Eingabe des Benutzers syntaktisch nicht korrekt, sollte dies beispielsweise kein Grund für eine Rückweisung sein. Statt dessen sollte alles identifizierbare weiter genutzt werden. [All95] drückt dies so aus:

> "Going on no matter what: The bottom line for a dialogue system is that it should never give up."

Korrekturmöglichkeiten: Auch wenn Robustheit ein primäres Ziel ist, lassen sich Fehler in der Kommunikation nicht vermeiden. Hat der Benutzer die Möglichkeit, diese einfach zu korrigieren, erhöht dies die Benutzbarkeit. Dabei ist sowohl eine Rücknahme fehlerhafter Schritte als auch eine Abbruchmöglichkeit des aktuellen Dialogschrittes wichtig.

Geschwindigkeit: Für eine möglichst natürliche Kommunikation mit dem System ist auch eine kurze Reaktionszeit des Systems wichtig. Es sollte niemals der Eindruck entstehen, das System reagiere nicht. Dem Benutzer muß immer klar sein, was das System gerade tut.

Anpaßbarkeit an andere Sprachen: Die Realisierung eines kompletten Systems, wie es im Rahmen dieser Arbeit angestrebt wird, ist aufwendig. Durch die Integration einer Sprachkomponente und insbesondere deren direkte Kopplung an das Komplettsystem ist das komplette System normalerweise auf eine Sprache wie Deutsch oder Englisch festgelegt. Ließe sich die Sprachkomponente multilingual beziehungsweise leicht anpaßbar anlegen, wäre die mögliche spätere Nutzergruppe ohne erheblichen Mehraufwand leicht erweiterbar.

Keine linguistische Forschung: Die Dialogkomponente soll kein Beitrag zu aktueller linguistischer Forschung sein. Ein umfangreiches und genaues Parsing der Benutzereingaben, um hiermit sprachliche Besonderheiten zu erkennen und aufzulösen, wird nicht angestrebt. Statt dessen soll es der Dialog dem Benutzer ermöglichen, interaktiv auf möglichst einfache Weise zum Ziel zu kommen. Dafür muß das System gängige und einfache Strukturen aus der Domäne verstehen und verarbeiten können.

In [Bra99a] und [Bra99b] wird eine interessante Dialogkomponente eines integrierten Systems vorgestellt, welche durch eine integrierte Spracherkennung und Semantikanalyse auf Basis einer Phrasenerkennung eine hohe Robustheit erreicht. Durch gleichzeitige Benutzung dieser geparsten Phrasen in der Sprachverstehenskomponente wird eine Vereinfachung und damit Geschwindigkeitserhöhung des Gesamtsystems erreicht.

Abweichend zum klassischen Ansatz der Sprachverarbeitung, der in Abbildung 4.1 dargestellt ist, wird hierdurch eine Kopplung der Spracherkennung und der syntaktisch/semantischen Analyse vorgenommen. Dieser Ansatz bietet eine gute Grundlage, auf der die hier realisierte Komponente beruht.

4.2 Spracherkennung und Grammatik

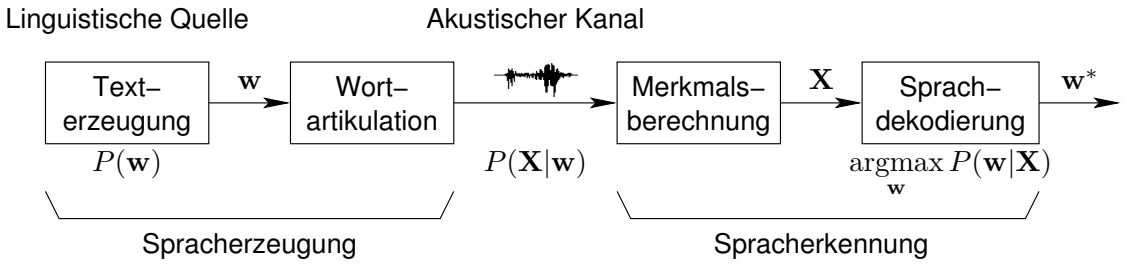


Abbildung 4.2: Das Kanalmodell der Spracherzeugung und -erkennung nach [ST95].

Das zu lösende Problem bei der Spracherkennung zeigt Abbildung 4.2. Ein Sprecher erzeugt eine Wortkette $\mathbf{w}$ und artikuliert diese. Der Spracherkenner nimmt das Signal auf und extrahiert hieraus eine Kette von Merkmalsvektoren $\mathbf{X}$, die durch eine Sprachdekodierung in eine optimale Wortkette $\mathbf{w}^*$ überführt werden soll. Deren Berechnung kann mittels der Bayesregel auf

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmax}} \left[P(\mathbf{w}) \cdot P(\mathbf{X}|\mathbf{w}) \right] \tag{4.1}$$

zurückgeführt werden. Dabei repräsentieren $P(\mathbf{w})$ das zu schätzende Sprachmodell und $P(\mathbf{X}|\mathbf{w})$ das ebenfalls zu schätzende akustische Modell. Für die Modellierung des akustischen Modells werden in den meisten aktuellen Erkennungssystemen *Hidden-Markov-Modelle* [Lev83, Hua01] eingesetzt. Das Sprachmodell wird häufig basierend auf dessen Faktorisierung bestimmt:

$$P(\mathbf{w}) = P(w_1) \cdot P(w_2|w_1) \cdot \ldots \cdot P(w_m|w_1 \ldots w_{m-1}) \quad \text{mit} \quad m = |\mathbf{w}| \tag{4.2}$$

Die hier auftretenden Wahrscheinlichkeiten lassen sich durch Auszählen auf einer Trainingsstichprobe bestimmen. Auf Grund der schnell stattfindenden kombinatorischen Explosion und des damit verbundenen Mangels an genügend Trainingsmaterials beschränkt man sich meistens auf die Unigramme $P(w_i)$ und Bigramme $P(w_i|w_j)$. Zum Teil werden auch noch Trigramme eingesetzt.

Für die Spracherkennung im hier vorgestellten System kommt das von Gernot A. Fink entwickelte ESMERALDA, das *Environment for Statistical Model Estimation*

and Recognition on Arbitrary Linear Data Arrays, zum Einsatz [Fin99]. ESMERALDA ist eine integrierte Umgebung zur Entwicklung von Spracherkennungssystemen. Dabei werden HMMs für die akustische Modellierung und n-Gramme für die Sprachmodellierung benutzt. Beides sind etablierte Techniken, die ihre Leistungsfähigkeit schon häufig gezeigt haben. Um gute Erkennungsraten zu erreichen, benötigen sie allerdings eine große Menge an Trainingsmaterial. Um dies zu minimieren, läßt sich allerdings ausnutzen, daß das System keinen beliebigen Dialog führen muß und damit beliebige Sätze auch nicht sicher erkennen muß. Dies wird durch ESMERALDA durch die Möglichkeit unterstützt, deklaratives Wissen in Form einer Grammatik, die von einem LR(1)-Parser [Aho86] geparsed wird, in die Spracherkennung zu integrieren. Sven Wachsmuth stellt dies in [Wac98] näher vor.

Die Grammatik wird wie ein spezielles Sprachmodell behandelt und erweitert damit das normale n-Gramm. Wortsequenzen, die nicht der Grammatik entsprechen, werden mit einem Strafterm versehen. Dieser ist davon abhängig, ob der Parser eine Konstituente abbrechen muß, ob er neu initialisiert werden muß oder ob ein Wort vorliegt, welches nicht der Grammatik entspricht. Die Grammatik ist dabei eine Konstituentengrammatik, die auch Teile eines Satzes akzeptiert. Dies trägt der Problematik Rechnung, daß man bei natürlicher Sprache nicht von grammatikalisch korrekt formulierten Sätzen ausgehen kann [Bat94]. Hierdurch wird auch vermieden, daß einzelne Fehler in der Erkennung das Parsen eines kompletten Satzes verhindern. Vergleichbar ist dieser Ansatz der Konstituentengrammatik mit anderen Ansätzen zum partiellen Parsing, die jeweils das Detektieren von Schlüsselphrasen zum Ziel haben [Kaw98]. Weiterhin bleibt bei der Benutzung des LR(1)-Parsers auch bei der vom Spracherkenner generierten Ausgabe die analysierte Struktur erhalten.

In denen im vorgestellten System eingesetzten Grammatiken ist sowohl syntaktisches Wissen als auch semantisches Wissen in domänenspezifischer Form integriert. Die in der Erkennung geparsten Strukturen werden im folgenden nicht verworfen, sondern als Ergebnis einer syntaktisch/semantischen Analyse betrachtet. Im Gegensatz zum in Abbildung 4.1 gezeigten klassischen Aufbau der Sprachverarbeitung kommt durch diese Vorgehensweise Domänenwissen in allen Ebenen von der Erkennung bis zum Dialog zum Einsatz. Dies hat verschiedene Vorteile:

- Durch frühzeitigen Einsatz möglichst vieler Restriktionen, die zum Teil auch größere Satzteile als ein Bigramm abdecken, wird die Erkennungsrate der Spracherkennung verbessert [Wac98].

- Die Geschwindigkeit des Gesamtsystems wird einerseits durch die Suchraumeinschränkungen der Grammatik und anderseits durch die nicht mehr nötige syntaktisch/semantische Analyse erhöht.

- Die Anpaßbarkeit an andere Sprachen und auf andere Domänen wird durch eine kleinere Menge benötigten Trainingsmaterials vereinfacht.

Das vorgestellte System ist sowohl in einer deutschen als auch in einer englischen Version realisiert. Die deutsche Version basiert auf dem Spracherkennungskorpus des SFB 360 [Bri95]. Das Lexikon enthält 2157 Einträge. Das dem Erkenner zu Grunde liegende Bigramm wurde auf Trainingsmaterial mit einem Umfang von 32450 Wörtern bestimmt. Das Material wurde bei einem Wizard-of-Oz Experiment aufgezeichnet, bei dem ein Proband einen künstlichen Kommunikator instruieren sollte, ein Spielzeugflugzeug aus einzelnen Holzteilen zusammen zu bauen. Die englische Version basiert auf dem Wall-Street-Journal Korpus [Pau92]. Dieser Korpus besteht aus 15 Stunden vorgelesenen Sätzen aus der amerikanischen Wirtschaftszeitung "The Wall Street Journal". Der hier eingesetzte Erkenner beschränkt sich allerdings auf ein kleines Teillexikon des vollständigen Korpus. Die englische Version dient nicht als ein vollständiges Demonstratorsystem, sondern soll nur ein "Proof-of-concept" sein, der die leichte Anpaßbarkeit des Systems an andere Sprachen zeigt. Daher ist es auch nicht entscheidend, daß der Korpus aus gelesener Sprache besteht, was die Erkennungsrate bei spontaner Sprache erniedrigt, da die akustische und grammatikalische Situation bei spontan gesprochener Sprache im Trainingsmaterial nicht korrekt wiedergegeben wird. In allen anderen Belangen der sprachlichen Verarbeitung ist die englische Version allerdings vollständig.

Abbildung 4.3 zeigt beispielhaft einen Ausschnitt aus der Grammatik des englischen Erkennungssystems. Alle Nichtterminalsymbole, die vom Spracherkenner zur weiteren Verarbeitung mit ausgegeben werden sollen, sind mit einem $ gekennzeichnet, alle nicht auszugebenden Nichtterminalsymbole sind mit zwei $ gekennzeichnet und alle Terminalsymbole sind *kursiv* geschrieben. Beginnend mit dem nichtterminalen Startsymbol $$SEGMENT werden verschiedene Segmente eines Satzes von der Grammatik akzeptiert. Die Segmente entsprechen dabei nicht den normalen syntaktischen Konstituenten wie beispielsweise Nominalphrasen oder Verbalphrasen. Im Gegensatz zu diesen Konstituenten ist in den Segmenten schon domänenspezifisches Wissen modelliert. So gibt es ein eigenes Segment $S_STOP, welches den Abbruch einer Aktion durch Anweisungen wie *stop* oder *halt* modelliert.

Die Segmentdefinition $S_OBJEKT für Objektbenennungen ist ausführlicher dargestellt. Der bei der Benennung benutzte Artikel kann als ein Hinweis für oder gegen eine Zeigegeste interpretiert werden. Ein definiter Artikel wie in dem Satz "Give me the book." ist ein Hinweis auf eine mögliche Geste, ein unbestimmter wie bei "Give me a book." ist ein deutlicher Hinweis dagegen. Dies wird durch das Nichtterminalsymbol $$ARTIKEL abgedeckt. Weiteres zu der Gesteninterpretation findet sich auch in Abschnitt 5.4. Wie in Abschnitt 2.1 gezeigt ist, sind Objektmerkmale wichtig für die Interpretation von Benennungen. Durch $$ADJ werden von der Grammatik beliebige Abfolgen von Farb- und Größenadjektiven akzeptiert. $$OBJEKT akzeptiert schließlich die Objektnamen.

Durch diese nicht auf vollständigen Sätzen basierende Definition der Grammatik führen einzelne Fehler in der Erkennung oder einzelne ungrammatikalische Ausdrücke nicht dazu, daß der komplette Satz nicht mehr geparsed werden kann. Ein Satz wie

```
$$OBJEKT:     bar — book — camera — case — ...;
$$ARTIKEL:    the — this — a — an ;
$$GROESSE:    long — small — big — ...;
$$FARBE:      red — green — blue — ...;
$$ADJ:        $$GROESSE — $$FARBE — $$ADJ ;

$S_OBJEKT:    $$ARTIKEL $$OBJEKT —
              $$ARTIKEL $$ADJ $$OBJEKT ;
...
$S_AKTION:    take — give me — have pointed — ...;
...
$$SEGMENT:    $S_AKTION $S_OBJEKT —
              $S_AGENT $S_AKTION —
              $S_AKTION —
              $S_OBJEKT —
              $S_STOP —
              ...
```

Abbildung 4.3: Ein Ausschnitt aus der Grammatik des englischen Erkennungssystems. Beginnend mit dem Startsymbol $$SEGMENT werden verschiedene für die Domäne relevante Segmente akzeptiert.

> "Computer, please take ... uhm give me the long red book over there."

würde beispielsweise zu der Ausgabe

> ($S_AGENT Computer) please ($S_AKTION take) uhm ($S_AKTION give me) ($_OBJEKT the long red book) over there

führen. Die Struktur und auch die nicht der Grammatik konformen, dem Spracherkenner aber bekannten Wörter bleiben für eine weitere Analyse erhalten.

Die Grammatik des deutschen Erkennungssystems ist vergleichbar zu der englischen Grammatik aufgebaut. Auf Grund der im Deutschen komplizierteren Deklinationsformen werden hier allerdings die unterschiedlichen Fälle für Adjektive, Artikel und Objekte getrennt behandelt, um den Spracherkenner durch weitere grammatikalische Suchraumeinschränkungen zu unterstützen. Dies ist entsprechend der in [Bra99b] beschriebenen Grammatik gelöst.

4.3 Semantikanalyse

Da in der Spracherkennung schon semantisches und pragmatisches Wissen eingeflossen ist, ist die übrig gebliebene Aufgabe bei der Semantikanalyse vergleichsweise

einfach. Je nach von der Spracherkennung erkannter Struktur müssen dessen Einzelworte im wesentlichen nur noch korrekt interpretiert werden. Hierzu wird ein semantisch annotiertes Lexikon benutzt, welches für jedes Wort unter anderem ein Synonym, die Wortart und bei Verben einen Aktionsrahmen enthält. Damit wird es beispielsweise möglich, einen größeren Wortschatz für die Benennung von Objektattributen zu verwenden. Durch die Verwendung der Synonyme kann dieser auf die vom Kontrollmodul verstandenen Objektattribute abgebildet werden. Näheres zur Verarbeitung der Attribute findet sich in Abschnitt 7.2. Die gleiche Abbildung wird auch verwendet, um unterschiedliche Sprachen zu unterstützen. Damit kann beispielsweise das Englische *reddish* auf das Deutsche *rot* abgebildet werden.

Bei der Analyse wird eine interne Repräsentation einer Äußerung aufgebaut, die eine Aktion, einen Agenten und eine Liste von Objektbeschreibungen enthält. Das Füllen der Struktur erfolgt inkrementell durch Einlesen der strukturierten Spracherkennerergebnisse und Betrachten der Synonyme der eingelesenen Terminalsymbole. Das genaue Vorgehen beim Füllen dieser Struktur ist abhängig vom aktuell zu bearbeitenden Segment. Ein Segment $S_STOP wird direkt als eine Aktion *stop* interpretiert. Bei dem Segment $S_AGENT werden Terminalsymbole direkt als der aktuelle Agent übernommen. Verben innerhalb von $S_AKTION-Segmenten werden abhängig von den bisher gesehenen Aktionen in der aktuellen Äußerung behandelt. Wurde bisher keine Aktion gesehen oder hatte die bisherige Aktion keinen im Lexikon eingetragenen Aktionsrahmen, wird das neue Verb als neue Aktion übernommen. Damit werden für die Domäne wichtigere Aktionen, die mit einem Aktionsrahmen versehen sind, bevorzugt.

Durch diese inkrementelle Verarbeitung der Äußerungen werden in der Spontansprache häufig vorkommende Reparaturen in begrenztem Maße implizit unterstützt, indem neuere Segmente ältere ersetzen. Segmente vom Typ $S_OBJEKT ersetzen bisher eingelesene Objekte nicht. Damit kann der Benutzer in einer Äußerung mehrere Objekte benennen. Die einzelnen Terminalsymbole des Segmentes werden gemäß ihrer Wortart in die Objektbeschreibung übernommen. Es werden dabei beliebig viele Farbadjektive und Größenadjektive unterstützt. Bei Nomen wird jeweils das letzte im Segment auftretende Nomen übernommen. Zusammen mit einer Objektbenennung kann zusätzlich eine Zeigegeste auftreten. Als Indikator hierfür wird, wie im letzten Abschnitt und in Abschnitt 5.4 beschrieben, ein definiter Marker in der Sprache angesehen. Wörter, die diese Aufgabe übernehmen, sind im Lexikon markiert. Im Deutschen und im Englischen sind dies definite Artikel. Das Auftreten eines solchen Wortes wird zusätzlich in der Objektbeschreibung hinterlegt.

Wie im letzten Abschnitt beschrieben, kann es jederzeit zu Fehlern in der Erkennung kommen, die dazu führen, daß einzelne Segmente nicht geparsed werden können. Das Gleiche kann auch durch spontansprachliche Phänomene verursacht werden. Dennoch ist es häufig möglich, einzelne Worte des Segmentes korrekt zu erkennen. Durch Wissen über den aktuellen Dialogzustand können zum Teil auch diese Einzelworte verwertet werden. Erwartet der Dialog explizit eine Objektbenennung, werden auch

Nomen, die nicht einem Segment zugeordnet sind, als Objektbenennung interpretiert. Dies ermöglicht beispielsweise auch die Benennung eines Objektes durch ein einzelnes Nomen wie *book*, was nach der Grammatik aus Abbildung 4.3 nicht als korrektes Segment $S_OBJEKT akzeptiert wird. Auf eine Frage wie "Wie heißt dieses Teil?" ist dieses eine Wort aber eine gängige und natürliche Antwort.

Diese einfache Semantikanalyse ermöglicht alles für das angestrebte System nötige. Um das Vorgehen weiter zu verdeutlichen, zeigt Abbildung 4.4 das Ergebnis einer semantischen Interpretation des im Erkennungsabschnitt benutzten Beispielsatzes "Computer, please take ... uhm give me the long red book over there." von Seite 37. Die für das Kontrollmodul nötigen Objektattribute wurden mit Hilfe des Lexikons übersetzt. Der Objektname *book* wird nicht übersetzt, da das restliche visuelle System im Gegensatz zu den Objektattributen keinerlei initiales Wissen über die Objekte hat. Das System kennt initial die visuelle Bedeutung der Farbe *Rot*, die visuelle Verankerung von *book* wird dagegen, genau wie die Verankerung von *Buch* in einem deutschen Satz, gelernt. Initial kennt das System nur dessen sprachliche und linguistische Repräsentation.

Agent:	*Computer*	
Aktion:	*geben*	
Objekt1:	*book*	
	Definit:	¡Zeitpunkt des Artikels *the*¿
	Farbe:	*rot*
	Größe:	*lang*

Abbildung 4.4: Ergebnis der semantischen Interpretation des englischen Satzes "Computer, please take ... uhm give me the long red book over there."

4.4 Dialog

Wie in Abschnitt 3.2 dargestellt ist das Gesamtsystem sprachgesteuert. Gibt der Benutzer eine sprachliche Anweisung, versucht der Dialog diese unter Zuhilfenahme von Gestikinformationen und den vorhandenen Objekterkennern zu bearbeiten. Um dies zu erreichen, stehen nach der im letzten Abschnitt vorgestellten Semantikanalyse ausgewertete Einzeläußerungen zur Verfügung. Diese werden entsprechend ihres Inhaltes ausgewertet und mit Informationen aus dem Dialogverlauf, von der Gestik und den vorhandenen Objekterkennern kombiniert, um schließlich eine passende Systemreaktion an den Benutzer auszugeben.

4.4.1 Ablaufmodell aus einer funktionalen Sicht

Bei der Auswertung der vorhandenen Informationen ist der aktuelle Dialogzustand von zentraler Bedeutung. Abbildung 4.5 zeigt die Zustände, die der Dialog aus einer funktionalen Sicht einnehmen kann. Jeder Weg durch dieses Modell spiegelt den Verlauf eines möglichen Mensch-Maschine-Dialogs wieder. Die Pfeile zwischen den Zuständen geben mögliche Übergänge an, die auf Grund von Sprach-, Objekt- oder Gesteninformationen beschritten werden. Der Zustand START wird immer eingenommen, wenn eine Anweisung abgearbeitet ist. Damit ist dieser Zustand auch der nach einem Systemstart initial eingenommene Zustand. Einen Ausgang aus dem Modell gibt es nicht. Dies spiegelt den Systemansatz des immer anwesenden, fortwährend ansprechbaren Begleiters wieder, der durchgehend versucht, Anfragen zu beantworten und hieraus zu lernen.

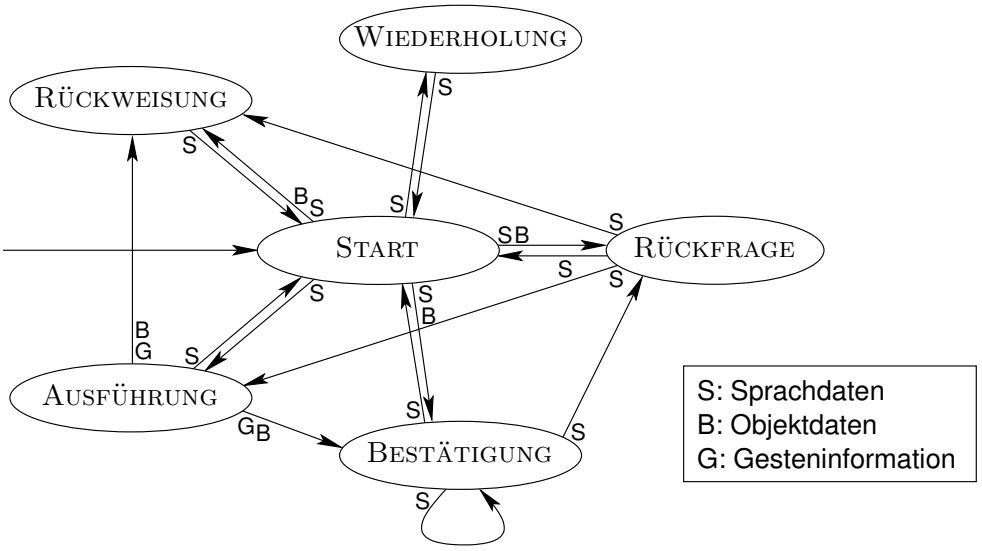


Abbildung 4.5: Ablaufmodell des Dialogs aus einer funktionalen Sicht. Die Wege durch das Modell geben den Verlauf der möglichen Mensch-Maschine-Dialoge an. Übergänge finden auf Grund von Informationen einer der drei angegebenen Modalitäten statt.

Alle Zustände außer START können erst eingenommen werden, wenn eine sprachliche Anfrage des Benutzers vorliegt. Der Zustand WIEDERHOLUNG wird eingenommen, falls die sprachliche Anfrage keine verwertbare Information enthielt. In diesem Fall wird um eine Wiederholung der Anfrage gebeten. Im Zustand RÜCKFRAGE wird der Benutzer nach weiteren benötigten Informationen gefragt. Dies kann einerseits auf Grund rein sprachlicher Informationen geschehen, beispielsweise bei einer Sicherheitsfrage nach der Richtigkeit der semantischen Interpretation. Anderseits kann der Grund

auch das Ergebnis der Objekterkennung sein, etwa bei der Initialisierung des Lernvorgangs, falls ein Objekt nicht gefunden werden konnte. Im Zustand AUSFÜHRUNG gibt der Dialog eine Rückmeldung über eine anstehende Tätigkeit, beispielsweise das Anstoßen der Objekterkennung. Da hier nur der Start einer Tätigkeit angekündigt wird, wird dieser Zustand nur auf Grund von Sprachinformationen eingenommen. Der Zustand BESTÄTIGUNG wird bei erfolgreicher Ausführung einer Tätigkeit eingenommen. Entweder wurde ein Objekt lokalisiert oder der Hautfarbklassifikator wurde erfolgreich neu trainiert. Dem Benutzer wird hierüber berichtet. RÜCKWEISUNG modelliert die im Sinne des Benutzers nicht erfolgreiche Ausführung einer Anweisung. Beispielsweise konnte während des Lernvorgangs kein Objekt gefunden werden oder der Benutzer hat auf die Anfrage nach einem Objektnamen keinen genannt.

Der Benutzer kann jederzeit einen Dialog durch eine Anweisung mit einem Segment vom Typ $S_STOP abbrechen. In diesem Fall wechselt der Dialog in den Zustand START und erwartet eine neue Anfrage. Eine weitere wichtige Anforderung an das System war auch eine Rücknahmemöglichkeit fehlerhaft ausgeführter Anweisungen. Für auf der Datenbank durchgeführte Objektmodifikationen ist dies durch eine sprachliche Korrekturmöglichkeit gegeben. Ist der Dialog im Zustand BESTÄTIGUNG und hat er die Lokalisation eines Objektes bekanntgegeben, kann der Benutzer dem System die fehlerhafte Lokalisation mitteilen. Der Dialog nimmt darauf hin die bisher für die Anweisung durchgeführten Modifikationen an der Datenbank zurück und initiiert einen neuen Lerndialog. Während dessen wird eine neue Objektansicht gelernt, so daß zukünftige Anweisungen wieder korrekt bearbeitet werden können.

4.4.2 Ablaufmodell aus einer prozeduralen Sicht

Im letzten Abschnitt wurde der Dialogablauf aus einer abstrakteren funktionalen Sicht beschrieben. Nun wird er aus einer konkreteren prozeduralen Sicht vorgestellt. Bei der folgenden Beschreibung und den folgenden Ablaufdiagrammen wird auf den jederzeit möglichen Abbruch eines Dialogs aus Gründen der Übersichtlichkeit nicht explizit eingegangen. Dieser ist aber gegeben.

Der Dialog verarbeitet zwei Arten von Anfragen. Einerseits kann der Benutzer das interaktive Trainieren des im Modul Handverfolgung benutzten Hautfarbklassifikators anstoßen. Andererseits kann er Anfragen nach Objekten stellen. Der Dialogverlauf für das Aktualisieren des Klassifikators ist in Abbildung 4.6 dargestellt. Der Benutzer kann im Verlauf eines Dialogs sowohl explizit als auch implizit einen Wunsch äußern. Sagt er beispielsweise “Hey, I have pointed!”, weist er vermutlich auch implizit darauf hin, daß das System die Erkennungsleistung bei Gesten verbessern soll. Um dies zu verifizieren, fragt der Dialog explizit nach und initiiert erst danach das Aktualisieren des Klassifikators. Der Benutzer kann das Aktualisieren aber auch explizit durch eine Anweisung wie “Update the classifier.” anstoßen. In diesem Fall führt diese explizite und damit verläßlichere Information zu einer direkten Ausführung des Aktualisierungsvorgangs, bei dem der Benutzer durch Winken vor der Kamera Trainingsmaterial für ein

Neutraining des Hautfarbklassifikators liefert. Weitere Details zum visuellen Teil des Trainings des Klassifikators finden sich in Abschnitt 5.3.4.

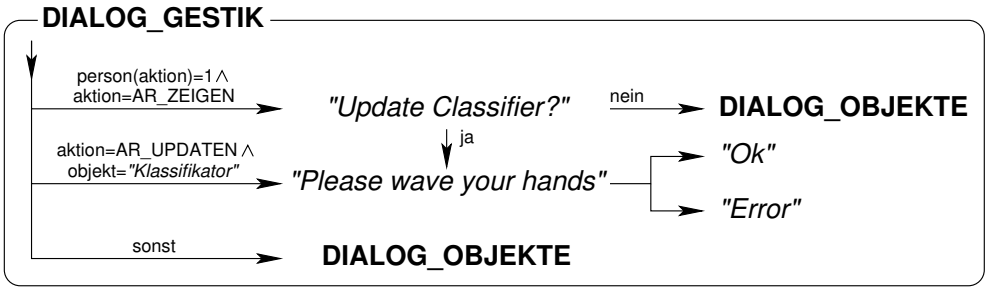


Abbildung 4.6: Der Dialogverlauf bei der Aktualisierung des Hautfarbklassifikators.

Wurde keine Aufforderung zum Aktualisieren des Klassifikators erkannt und sind in der semantischen Beschreibung Objekte angegeben, wird von einer Anfrage zur Lokalisation eines Objektes ausgegangen. Dazu werden zuerst, wie in Abschnitt 3.2 beschrieben, visuell basierte Objekthypothesen generiert und bewertet. Anschließend wird der in Abbildung 4.7 skizzierte Dialog fortgesetzt. Wie in Abschnitt 2.1.3 dargestellt sind Synonyme, die das Benennen von Objekten mit unterschiedlichen Namen erlauben, im Sinne eines flexiblen Dialogs sinnvoll. Daher können diese im Dialog gelernt werden. Hat der Benutzer bei seiner Anfrage auf ein Objekt gezeigt, wird im Bild auch nach Objekten gesucht, die er in der Anfrage sprachlich nicht benannt hat. Mit der Gestik ist in diesem Fall eine zusätzliche Informationsquelle vorhanden, die auch für sich alleine korrekt sein kann. Wurde ein Objekt gefunden, welches der Benutzer sprachlich nicht benannt hat, wird er zur Verifikation gefragt, ob es ein Synonym ist. Falls ja, wird die Objekthypothese akzeptiert, ansonsten wird sie verworfen.

Ist eine Objekthypothese akzeptiert worden, wird dies dem Benutzer mitgeteilt. Dabei kann es allerdings sein, daß auf Grund eines Fehlers in der Interpretation, der Gestik, der visuellen Objektsuche oder auch bei der Kombination dieser Informationen ein falsches Objekt gefunden wurde. Der Benutzer kann in diesem Fall durch die sprachliche Rückweisung der Hypothese korrigierend eingreifen. Genauso kann es sein, daß die am besten bewertete Objekthypothese zur Akzeptierung zu schlecht bewertet war und daher keine akzeptierte Hypothese gefunden wurde. In beiden Fällen versucht das System sein Objektwissen zu erweitern, indem der in Abbildung 4.8 dargestellte Dialog zum Lernen eines Objektes gestartet wird. Abhängig von der aktuellen Situation, liegt eine Korrektur vor oder hat das System sein Scheitern selber bemerkt, wird dem Benutzer der Fehler mitgeteilt. Zusätzlich wird er aufgefordert, das von ihm benannte Objekt aus der Szene zu nehmen und gegebenenfalls das Objekt zu benennen. Dies ist nötig, falls der Benutzer das Objekt mit einer allgemeinen Benennung wie “das Teil” referenziert hatte. Hat er das Objekt aus der Szene genommen und dies dem System mitgeteilt, wird wie in Abschnitt 3.3 beschrieben eine neue Ansicht des

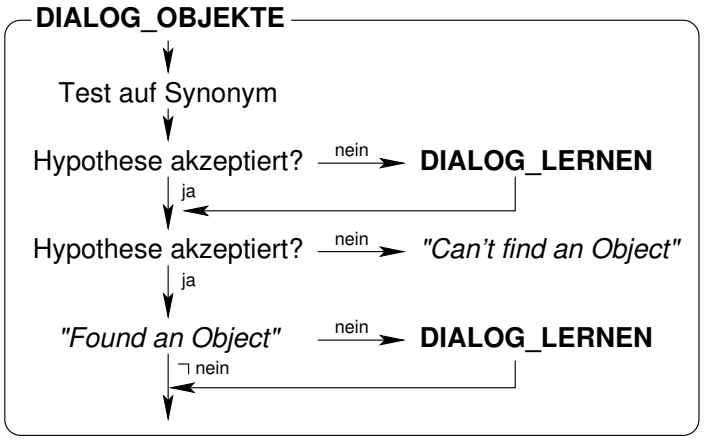


Abbildung 4.7: Der Dialogverlauf bei dem Benennen und Lernen von Objekten.

Objektes gelernt. Nach Abschluß des Lernschrittes wartet der Dialog schließlich auf die nächste Anfrage, die sich wiederum auf Objekte oder das Training des Hautfarbklassifikators beziehen kann.

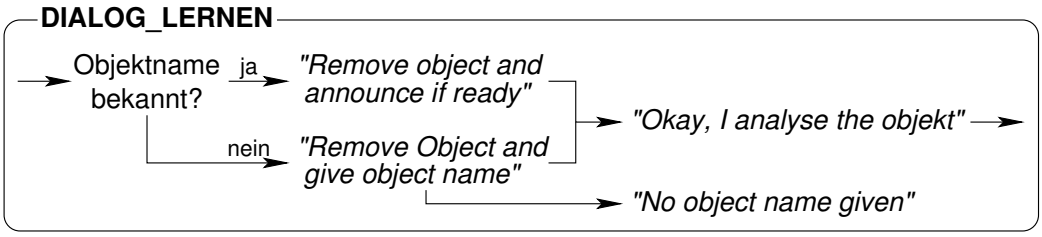


Abbildung 4.8: Der Dialogverlauf beim Lernen eines neuen oder Aktualisieren eines bekannten Objektes.

4.5 Sprachausgabe

Sprachliche Antworten generiert der Dialog durch Textschablonen, in die gegebenenfalls Objektnamen eingefügt werden. Um bei der Textausgabe sprachunabhängig zu werden, können diese statischen Textschablonen sehr einfach durch eine externe Textdatei in andere Sprachen übersetzt werden. Damit liegen nun sprachspezifische Texte in Form von Zeichenketten vor. Um eine möglichst natürliche Mensch-Maschine-Kommunikation zu erreichen, müssen diese Texte mit Hilfe einer Sprachsynthese in ein Sprachsignal überführt werden [Hua01, Teil IV][Dut97].

Im Rahmen dieser Arbeit sollte keine Forschung im Bereich der Synthese durchgeführt werden. Daher kommen fertige Lösungen zum Einsatz. Für das angestrebte Ziel der möglichst weitestgehenden Sprachunabhängigkeit ist allerdings eine Abstraktion von einer konkreten Syntheseimplementierung nötig, um die auszugebende Sprache möglichst einfach zu wechseln. Wünschenswert ist dies auch, um eine größere Flexibilität in Bezug auf die Einbettung in eine vorhandene Soft- und Hardwareumgebung zu erreichen. Abbildung 4.9 zeigt die Architektur des Synthesemoduls, die sich aus diesen Überlegungen heraus ergeben hat. Mit Hilfe eines steuernden Servers werden verschiedene Syntheseeinheiten an das Komplettsystem angebunden. Der Server empfängt über DACS normalen ASCII-Text. Je nach gewählter Einheit gibt er diesen entweder an eine über die serielle Schnittstelle angebundene Hardwaresynthese oder an eine über UNIX Pipes angebundene Softwaresynthese weiter.

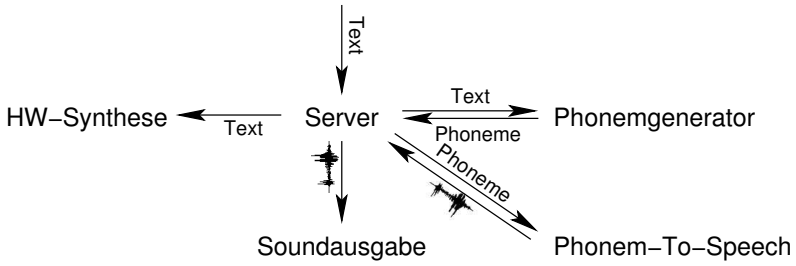


Abbildung 4.9: Architektur des Systems zur Sprachausgabe. Ein Server empfängt Texte und generiert hieraus mit Hilfe verschiedener Untereinheiten Sprachausgaben.

Als Hardwaresynthese wird ein kommerzielles Text-To-Speech System basierend auf einem OKI-Chip unterstützt. Viele auf Software basierende Text-To-Speech Systeme lassen sich in drei unabhängige Einheiten einteilen [Dut97]. In einer ersten orthographisch-phonetischen Transkription werden Phonemketten aus einer Texteingabe bestehend aus Zeichenketten generiert. Gewünschte Intonation und Prosodie werden dabei in der Phonemkette mit kodiert. Ein zweiter Schritt überführt die Phoneme in ein kontinuierliches Sprachsignal, welches schließlich in einem dritten Schritt nur noch ausgegeben werden muß. Diese Dreiteilung ist im Synthesemodul abgebildet. Die Kommunikation zwischen den einzelnen Teilen findet dabei jeweils über UNIX Pipes statt.

Für die Phonemgenerierung kommen im Deutschen txt2pho aus dem HADIFIX-Projekt [Por97] und im Englischen Festival [Bla99] zum Einsatz. txt2pho basiert auf einem Phonemlexikon mit ca. 50000 Einträgen und Flexionstabellen. Prosodische Parameter werden basierend auf einer linguistischen Analyse bedeutungstragender Worte generiert. Festival ist ein komplettes Text-To-Speech System, wird hier aber nur als Phonemgenerator eingesetzt. Nach einer Auflösung von Sonderzeichen und Zahlen und

einer "Part of speech"-Markierung der einzelnen Worte wird in diesem System eine lexikonbasierte Phonemgenerierung durchgeführt. Anschließend werden Intonation und Dauer der Phoneme angepaßt.

Für die Phonem-To-Speech Konvertierung wird sowohl im Deutschen als auch im Englischen MBROLA verwendet [Dut96]. MBROLA, was für *Multi Band Resynthesis OverLap Add* steht, führt die Synthese durch die Konkatenierung von Diphonen durch. Spektrale Diskontinuitäten werden dabei in der Zeitdomäne geglättet, was zu einer relativ kontinuierlich und natürlich klingenden Ausgabe führt. Weitere Syntheseeinheiten lassen sich hinzufügen, sofern sie eine Steuerung über die Standardein- und ausgabe erlauben und sich somit in das über Pipes gesteuerte Synthesemodul einfügen.

4.6 Zusammenfassung

In diesem Kapitel wurde die sprachverarbeitende Komponente eines multimodalen Systems vorgestellt. Das komplette System ist sprachgesteuert, benutzt aber zusätzlich Gestikinformationen und Objekterkennungsergebnisse als steuernde Informationsquellen. Damit läßt sich die Dialogkomponente als ein integriertes System klassifizieren.

Im Gegensatz zum klassischen Ansatz einer sprachverarbeitenden Komponente sind die einzelnen Komponenten stärker miteinander verzahnt. Die Spracherkennung und die syntaktische Analyse arbeiten direkt gekoppelt. Zusätzlich ist hier semantisches Wissen und Domänenwissen integriert. In der Semantikanalyse wird Wissen über den aktuellen Dialogzustand integriert. Durch diese enge Verzahnung wird eine erhöhte Robustheit erreicht. Die Robustheit wird auch durch die Art des Parsings erhöht. Es wird nicht versucht, komplette Sätze zu parsen. Statt dessen wird eine Suche nach Schlüsselphrasen innerhalb einer Äußerung durchgeführt.

Die Sprachkomponente ist leicht anpaßbar an andere Sprachen angelegt. Zur Unterstützung einer neuen Sprache werden nur ein Korpus zum Trainieren eines Spracherkenners, ein Modul zur Sprachsynthese und einige sprachbeschreibende Textdateien benötigt. Fertige Demonstrationssysteme stehen in Deutsch und Englisch zur Verfügung.

5 Gestenerkennung

Gesten werden von Menschen häufig in der Kommunikation eingesetzt. Nach [Arg75] kann bei der menschlichen Kommunikation mehr Information übermittelt werden, falls Gesten benutzt werden können. Andernfalls werden beispielsweise mehr Wörter zur Beschreibung räumlicher Relationen verwendet. Aussagen wie "Nimm dieses Teil." können in vielen Fällen nur mit Hilfe der Interpretation von deiktischen Gesten verstanden werden. Für eine natürliche Mensch-Maschine-Schnittstelle ist eine Integration von Gesten daher immanent wichtig.

Aber was versteht man eigentlich unter Gesten? Abschnitt 5.1 gibt einen Überblick. Abschnitt 5.2 gibt einen Eindruck über verschiedene Ansätze zum automatischen Erkennen von Gesten, die für das in Kapitel 3 vorgestellte System relevant sind. Die beiden folgenden Abschnitte gehen schließlich auf das im Rahmen dieser Arbeit realisierte Verfahren zur Erkennung und Interpretation von Gesten ein. Abgeschlossen wird das Kapitel durch mit diesem Verfahren erzielte Ergebnisse.

5.1 Taxonomie von Gesten

Vor der Aufstellung einer Taxonomie von Gesten ist es sinnvoll festzulegen, was man unter Gesten verstehen möchte. Kendon gibt in [Ken86] eine Definition für Gesten an:

> "...for an action to be treated as a 'gesture' it must have features which make it stand out as such."

Diese Definition zeigt schon das Problem bei Gesten: Es ist schwierig, diese überhaupt zu definieren. Eine weitere Definition liefern beispielsweise Kurtenbach und Hulteen [Kur90]:

> "A gesture is a motion of the body that contains information. Waving good-bye is a gesture. Pressing a key on a keyboard is not a gesture because the motion of a finger on its way to hitting a key is neither observed nor significant. All that matters is which key was pressed."

Hiernach haben Gesten verschiedene Eigenschaften. Sie enthalten (a) eine Bewegung, die (b) vom Körper ausgeführt wird und (c) Information trägt. Aber auch diese Definition ist immer noch äußerst allgemein und damit unspezifisch.

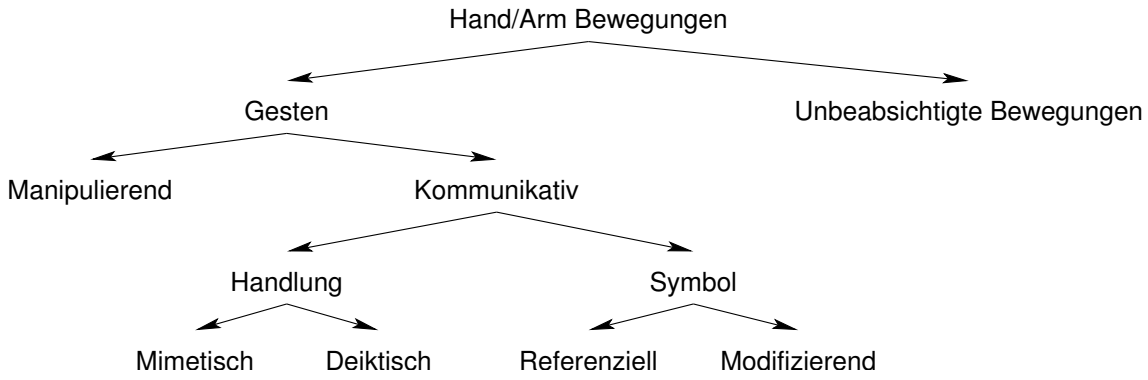


Abbildung 5.1: Taxonomie von Gesten, nach [Pav97] und [Que95].

Eine weitere Spezialisierung ergibt sich durch eine Einteilung von Gesten in Klassen. Abbildung 5.1 zeigt eine von [Que95] vorgeschlagene Taxonomie, die insbesondere in der Mensch-Maschine-Kommunikation als geeignet erscheint. Hier werden nur Hand/Arm Bewegungen betrachtet, andere Bewegungen des Körpers, die nach [Kur90] durchaus Gesten sein können, werden nicht klassifiziert.

Die informationstragenden Gesten werden in manipulierende und kommunikative Gesten eingeteilt. Manipulierende Gesten modifizieren Objekte, indem sie diese beispielsweise bewegen. Kommunikative Gesten übermitteln Informationen dagegen, ohne auf die Umwelt direkt Einfluß zu nehmen. Diese lassen sich in symbolische und 'handelnde' Gesten einteilen. 'Handelnde' Gesten lassen sich direkt durch Analyse der Bewegung interpretieren. Quek teilt diese weiter in mimetische und deiktische Gesten ein. Deiktische Gesten sind Zeigegesten, durch die Geste wird ein anderes Objekt referenziert. Mimetische Gesten imitieren eine Aktion oder ein Objekt. "Auto fahren" kann beispielsweise durch das hoch und runter Bewegen der Hände, als ob diese ein Lenkrad halten, dargestellt werden. Symbolische Gesten haben dagegen eine linguistische Rolle und sind von sich aus arbiträr. Referentielle symbolische Gesten symbolisieren ein Konzept oder eine Aktion. Modifizierende Gesten verändern eine andere Modalität, in den meisten Fällen die Sprache. Beispielsweise kann durch eine Geste bei dem Satz "Schau Dir das Auto an!" angedeutet werden, daß dieses am Schlingern ist.

Neben dieser Klassifikation von Quek gibt es verschiedene weitere, beispielsweise [Ken86] und [McN92]. Diese erscheinen allerdings für die Mensch-Maschine-Kommunikation als weniger geeignet. Sie sind weniger konkret und damit weniger angepaßt an die Mustererkennung. Zusätzlich fehlen wichtige Kategorien, manipulierende Gesten werden beispielsweise gar nicht betrachtet. In [Ken86] gibt es zusätzlich keine Kategorie für die wichtigen deiktischen Gesten.

Für die Domäne des interaktiven Objektlernens wie in Kapitel 3 vorgestellt sind insbesondere deiktische und manipulierende Gesten, die mit den Händen durchgeführt

werden, interessant. Durch deiktische Gesten ist es dem Benutzer möglich, durch eine andere Modalität als der Sprache auf Objekte hinzuweisen. Durch die Beobachtung von manipulierenden Gesten kann das System weitere Informationen über Aktionen des Benutzers gewinnen, die dieser als Reaktion auf eine Systemaufforderung durchführt.

5.2 Ansätze zur Gestenerkennung

5.2.1 Überblick

Für die automatische Erkennung und Interpretation von Gesten gibt es in der Informatik schon eine lange Tradition. Ein frühes System stellt Bolt vor [Bol80]. Es erlaubt, durch deiktische Gesten und Sprache Formen auf einer Projektionsleinwand zu manipulieren. Die Gesten werden dabei durch magnetische Positionssensoren bestimmt. Einen Sensor muß der Benutzer an der Hand tragen, ein Empfänger steht an einer festen Position im Raum. Der Benutzer muß ebenfalls an einer festen Position im Raum sitzen. Wenn er nun mit ausgestreckter Hand zeigt, kann seine Zeigerichtung durch die Bestimmung der Position der Hand im Raum einfach ermittelt werden.

Seit diesem anfänglichen System gab es verschiedenste weitere Entwicklungen sowohl im Hinblick auf die Eingabemodalitäten des Systems als auch auf die Flexibilität der Gesten. Neben den anfänglich eingesetzten Positionssensoren werden häufig Datenhandschuhe und Kameras zur Erfassung der Hände benutzt. Insbesondere bei Virtual-Reality-Anwendungen kommen Datenhandschuhe häufig zum Einsatz [Sow02, LaV99]. Sie messen, je nach Modell, relativ genau Position, Beschleunigung und Winkel der Finger und der Hand und erlauben durch diese gute Datengrundlage einen flexiblen Einsatz verschiedenster Gesten. Allerdings wird der Benutzer durch den Handschuh eingeschränkt. Dies wird durch den Einsatz von Kameras vermieden. Die erhöhte Komplexität durch die nötige Segmentierung der Hände wird zum Teil durch den Einsatz von farbigen Handschuhen [Ass98, Sta95] oder auch durch die Benutzung von Infrarotkameras [Her03, Rei03] vermindert.

5.2.2 Gestenerkennung mit Farbbildern

Kommt eine normale Farbkamera zum Einsatz, kann die Hand oder auch das Gesicht auf Grund von Hautfarbe gefunden werden. Hautfarbe ist im Farbraum in einem relativ kleinen Bereich geclustert. Für den normalisierten rg-Farbraum zeigt dies beispielsweise [Stö99] unter verschiedenen Beleuchtungsbedingungen mit Hilfe eines physikalischen Modells von Hautfarbe. Der rg-Farbraum wird aus dem RGB-Farbraum durch Helligkeitsnormierung gewonnen:

$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B} \tag{5.1}$$

Da die Summe $r + g + b$ immer 1 ist, ist dies ein Farbraum mit zwei Freiheitsgraden. Damit kann beispielsweise b entfallen. In [Hun94] wird zur initialen Hautfarbsegmentierung ein Pixelklassifikator im rg-Farbraum benutzt. An Hand von Trainingsmaterial wird der Hautfarbbereich im Farbraum bestimmt und zur Segmentierung eingesetzt. Unter Einsatz von Bewegungs- und Forminformationen wird schließlich ein Gesicht im vorher identifizierten Hautfarbbereich identifiziert. Anschließend wird das Hautfarbmodell mit Hilfe des gefundenen Gesichts an die aktuelle Beleuchtungssituation angepaßt.

Ein sehr bekanntes System ist Pfinder [Wre97b]. Pfinder erlaubt das Tracken einer Person in Echtzeit vor einem möglichst statischen Hintergrund. In einer Initialisierungsphase ohne Person in der Szene wird zuerst ein Modell des Hintergrundes erstellt, in dem jedem Pixel ein Mittelwert und eine Kovarianzmatrix im YUV-Farbraum zugeordnet werden. Eine Einführung in diesen Farbraum gibt Anhang A. Betritt die zu trackende Person die Szene, wird ein Personenmodell durch Analyse von großen Änderungen zum Hintergrundmodell erstellt. Das Personenmodell besteht aus mehreren Blobs, die durch Clustern in einem kombinierten Orts-/Farbraum mit den Merkmalen (x, y, Y, U, V) gewonnen wurden. Hierbei werden zusätzliche Nebenbedingungen beachtet. Beispielsweise muß ein Blob für die Hände Hautfarbe und eine große Dynamik haben. Jeder dieser Blobs wird durch eine Gaußverteilung modelliert. Eine Klassifikation des Bildes geschieht nun für jedes Pixel durch die Wahl des maximalen log likelihood bezüglich aller Blobs und des Hintergrundes. Daraufhin werden Löcher im segmentierten Vordergrundbereich geschlossen. Eine globale Glättung optimiert die Segmentierung weiter. Die Vordergrund- und Hintergrundmodelle werden an Hand dieser Segmentierung aktualisiert. Zur Behandlung von Verdeckungen können zusätzlich jederzeit neue Blobs generiert oder gelöscht werden. Zum Prädizieren der Position der Blobs im nächsten Bild werden Kalmantracker eingesetzt.

Pfinder wird in verschiedenen Systemen erfolgreich eingesetzt. In [Wre97a] wird es zur Interaktion mit einer virtuellen Spiel- und Entertainmentumgebung benutzt. Dabei werden sowohl verschiedene Gesten, wie deiktische oder auch ikonische, als auch Körperbewegungen im ganzen interpretiert. [Luc98] stellt ein System vor, in dem der Benutzer durch Gesten mit Hilfe von Spracherkennung Objekte manipulieren kann. Dabei werden sowohl deiktische, als auch symbolische und ikonische Gesten unterstützt. Neben Pfinder kommt hier ViaVoice von IBM, ein kommerzielles System zur Spracherkennung, zum Einsatz.

[Tri98] stellt ein System zum Instruieren eines Roboters vor. Der Benutzer kann den Roboter durch Gesten anweisen, ein Objekt auf eine bestimmte Art zu nehmen und an einer anderen Stelle wieder abzulegen. Die Szene wird durch den Roboter mit einer Stereokamera beobachtet. Zur Detektion der Hand wird aus den HSI-Eingangsbildern gewonnene Bewegungs- und Farbinformation verknüpft. Weitere Informationen zum HSI-Farbraum finden sich in Anhang A. Aus der Differenz der beiden letzten Intensitätsbilder ergibt sich eine Bewegungskarte. Durch Segmentierung aller Pixel, die innerhalb einer Ellipse im HS-Farbraum liegen, ergibt sich eine Hautfarbenkarte. Der

Maximalwert im Summenbild der beiden Karten liefert die ungefähre Position der Hand. Zur exakten Positionierung und zur Erkennung der Postur der Hand wird ein Graphmatching mit vorgegebenen Graphmodellen durchgeführt. Hierbei dienen Gaborjets an den Knoten des Graphen als Merkmale.

In [Ver02] wird der CONDENSATION-Algorithmus [Isa98] zum Tracken von Gesichtern eingesetzt. Hierbei wird von einem festen Hintergrund und keinem oder einem zu trackenden Objekt ausgegangen. Als Merkmale kommen Farbe aus dem H- und dem S-Kanal eines Kamerabildes und Bewegungsinformation aus dem Intensitätsdifferenzbild zum Einsatz. Die Merkmale ergeben sich durch Anwendung eines Gitters von Gaußfiltern auf die jeweiligen Kanäle. In einer Initialisierungsphase wird ein festes Hintergrundmodell erstellt. Der Farbteil des Vordergrundmodells wird kontinuierlich durch ein EM-Verfahren aktualisiert. Zum Tracken kommt der CONDENSATION-Algorithmus zum Einsatz.

5.3 Handdetektion

Abbildung 5.2: Beispiel für eine Szene aus der Sicht des Systems: Der Benutzer führt gerade eine deiktische Geste aus.

Nach dem im vorigen Kapitel gegebenen Literaturüberblick wird nun der realisierte Ansatz vorgestellt. Wie in Kapitel 3 beschrieben, steht für die Gestenerkennung durchgehend eine Farbkamera, die auf die Szene gerichtet ist, zur Verfügung. In Abbildung 5.2 ist ein Beispiel einer Aufnahme einer Szene dargestellt, wie sie das System sieht. Der Benutzer kann jederzeit eine sprachliche Anweisung an das System stellen, die durch eine *Zeigegeste* auf ein Objekt begleitet werden kann. Während des Objektlernens ist der Benutzer aufgefordert, ein Objekt aus der Szene zu nehmen. Um dies

sicherer analysieren zu können, ist eine Auswertung dieser *Greifaktion* von Vorteil. Aus diesem Szenario ergeben sich verschiedene Anforderungen an das System:

Echtzeitfähigkeit: Bei beiden Gesten gibt es kein dediziertes Signal über den Start oder das Ende der jeweiligen Zeigeaktion. Das System zur Analyse der Gesten sollte daher kontinuierlich die Szene auf Merkmale hin untersuchen, die schnell auf die zu untersuchenden Gesten schließen lassen. Dies erfordert eine Verarbeitung in Echtzeit. Eine Alternative wäre die Zwischenspeicherung der letzten n Sekunden des aufgenommenen Videos und eine Analyse im nachhinein. Dies hätte allerdings den Nachteil des erhöhten Speicherbedarfs und insbesondere der verzögerten Analyse und damit der verlangsamten Systemreaktion auf Benutzeraktionen.

Keine "Closed World Assumption": Das System sieht durch die Kamera immer nur einen kleinen Ausschnitt der Welt. Somit kann es zu jeder Zeit passieren, daß der Benutzer mit seinen Händen neu ins Bild hineinkommt oder es wieder verläßt. Auch Hände von anderen Personen können zu sehen sein. Damit kann sich die Anzahl der sichtbaren Hände jederzeit ändern — eine *"Closed World Assumption"* gibt es nicht. Dies schließt ein Vorgehen wie bei Pfinder [Wre97b] aus. Hier wird beim Anpassen eines Modells eines Menschen an die Szene von dem Wissen ausgegangen, daß sich genau ein Mensch in der Szene befindet.

Variabler Hintergrund: Es kann weder von einem festen noch von einem sich nicht ändernden Hintergrund ausgegangen werden. Es können jederzeit Objekte aus der Szene heraus genommen werden oder auch herein gelegt werden, was den Hintergrund modifiziert. Natürlich sollte die Gestenerkennung auch nicht an einen speziellen Hintergrund gebunden sein. Insbesondere während der Greifaktion ändert sich der Hintergrund. Um eine stabile Verarbeitung zu erreichen, kann daher kein Hintergrundmodell wie beispielsweise bei Pfinder eingesetzt werden. Auch wenn dies adaptiv wäre, könnte es zu kurzzeitigen Instabilitäten während der Adaptionsphase führen.

Variable Lichtverhältnisse: Ein großes Problem in der Farbbildverarbeitung sind sich ändernde Lichtverhältnisse. Je nach Stärke der Beleuchtung ändert sich die Helligkeit aller Objekte im Bild. Je nach Farbtemperatur der Beleuchtung ändert sich die Farbe aller Objekte im Bild. Indirekte Beleuchtung, die Reflexion von Licht von anderen Objekten, und die Richtung der Beleuchtung beeinflussen ebenfalls die Farbe und Helligkeit der Objekte. Der Mensch nimmt diese Farbveränderungen nur in einem geringen Maße war. Sowohl das Auge als auch das Gehirn trägt zu einer wahrgenommenen Farbkonstanz bei, indem sowohl die Farbe des Umgebungslichtes als auch Wissen über die gesehenen Objekte in die Wahrnehmung mit einfließt [Lev85, Kapitel 7.3 - 7.4][Mon97]. Auch für den automatischen programmgestützten Ausgleich von Farbtemperaturänderungen

gibt es eine große Anzahl von Verfahren [Bar02a, Bar02b]. Diese Verfahren haben allerdings bisher noch nicht die für farbbasierte Objekterkennung nötige Leistungsfähigkeit erreicht [Fun98]. Trotzdem wäre es wünschenswert, wenn das System mit Änderungen der Lichtverhältnisse umgehen könnte.

Um eine stabile Erkennung der Gesten zu erreichen, können bei der Wahl des Verfahrens zwei Punkte in Betracht gezogen werden:

- Beide Gesten werden mit den Händen durchgeführt.
- Nach [Kur90] ist eine Geste eine Bewegung des Körpers, Bewegung ist charakteristisch für eine Geste.

Daher liegt es nahe, eine kontinuierliche Segmentierung des kompletten Bildes nach Hautfarbe vorzunehmen, um so die Hände zu finden. Damit werden jederzeit auch neu im Bild auftauchende Hände gefunden. Allerdings kann es beliebig viele weitere Regionen im Bild geben, die ebenfalls hautfarben sind. Im Erkennungssystem wird daher neben der Farbinformation auch Bewegungsinformation genutzt.

5.3.1 Überblick

Abbildung 5.3 zeigt den prinzipiellen Aufbau des Systems. Ein von der Kamera im YUV-Farbraum erhaltenes Bild wird durch eine pixelbasierte Klassifikation auf Hautfarbe untersucht. Die Klassifikation liefert für jedes Pixel die Wahrscheinlichkeit, ob er hautfarben ist. Anhang A stellt den YUV-Farbraum genauer vor. Abschnitt 5.3.2 erläutert mehr Details über das genaue Vorgehen. Da diese Klassifikation nur auf lokalen Pixelinformationen beruht, die von einer rauschanfälligen Kamera geliefert werden, ist das Klassifikationsergebnis ebenfalls verrauscht. Zur Glättung eignet sich der kantenerhaltende nichtlineare Medianfilter [Gon02, Kapitel 3.6.2]. Nach einer Binarisierung liegen Hypothesen für Hände vor. Nur an Hand der Farbe läßt sich allerdings keine stabile Handsegmentierung vornehmen. Die hintere Karte in Abbildung 5.3 hat beispielsweise die gleiche Farbe wie Hände. Aus der in der Vergangenheit aufgetretenen Bewegung wird daher eine Aktivitätskarte erzeugt, die zur Filterung des vorher erhaltenen Hypothesenbildes dient. Das genaue Verfahren zur Bestimmung der Aktivitätskarte ist in Abschnitt 5.3.3 angegeben.

Aus dem nun erhaltenen Hautfarbpixelbild werden durch eine Kantenverfolgung zusammenhängender Pixelbereiche Polygonzüge für Regionen extrahiert [Gon02, Kapitel 2.5.2]. Gleichzeitig werden verschiedene Regionenmerkmale wie Position des Schwerpunktes, Größe in Pixel oder auch Umfang in Pixel bestimmt. Eine erste Bewertung der Region wird an Hand der in der Region "angehäuften Hautfarbwahrscheinlichkeiten" durchgeführt (Abschnitt 5.3.2). Diese Bewertung wird an Hand von Bewegungsinformationen modifiziert. Die Aktivitätskarte kann nur in einem relativ großen Bereich filtern, um so zu vermeiden, sich nur langsam bewegende Hände mit auszufiltern. Daher werden durch Einbeziehung der Bewegung aus dem letzten Framewechsel

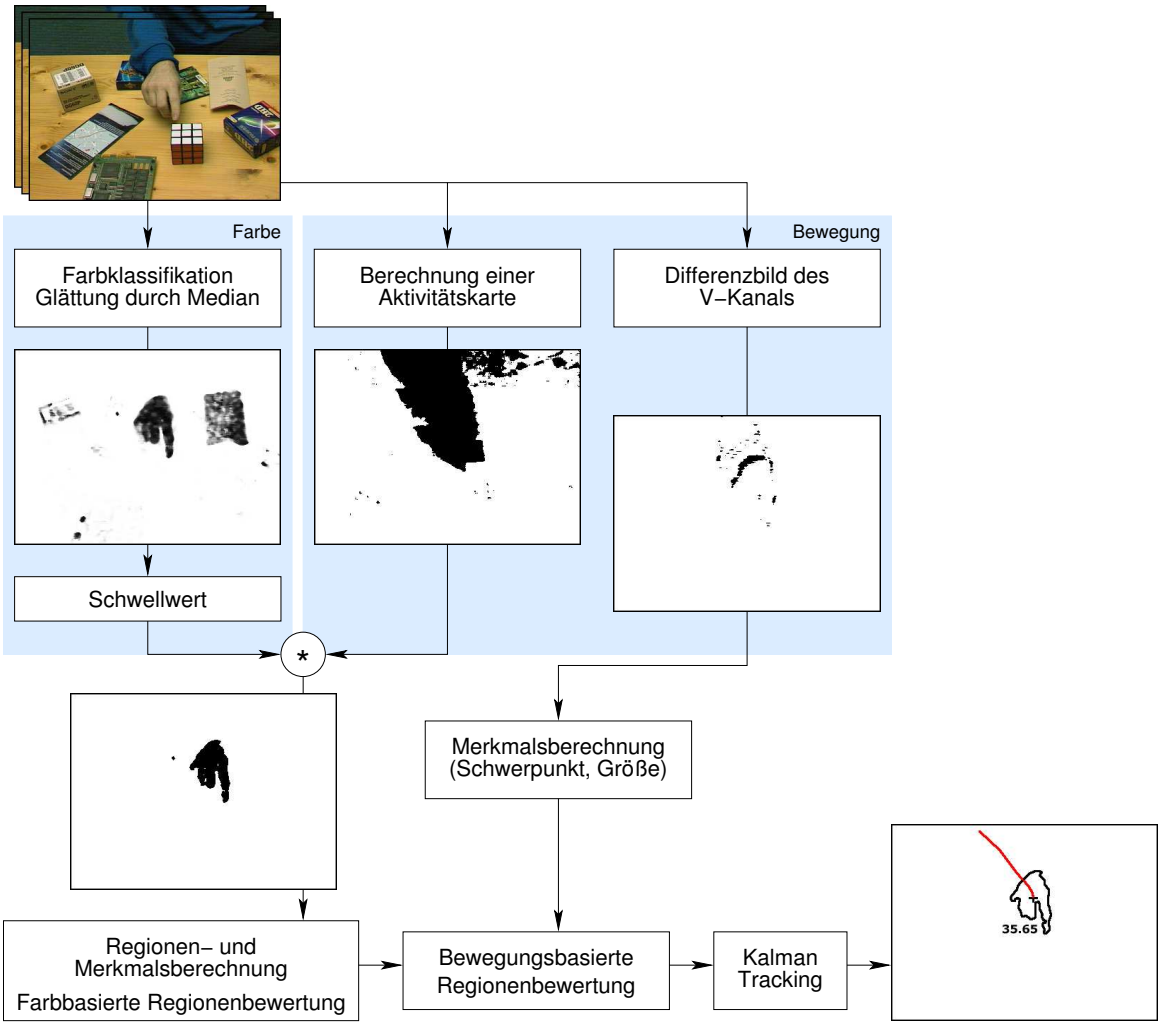


Abbildung 5.3: Flußdiagramm der Handverfolgung. Zur Segmentierung von Händen werden Farb- und Bewegungsinformationen kombiniert.

in die Bewertung insbesondere sich schnell bewegende Regionen weiter bevorzugt. Dies ist genauer in Abschnitt 5.3.3 erläutert. Schließlich werden genügend große und genügend gut bewertete Regionen durch einen Kalmanfilter [Kal60, May79] konstanter Beschleunigung verfolgt. Durch die nun erhaltenen Handtrajektorien und Regioneneigenschaften können die Gesten klassifiziert werden. Dies wird in Abschnitt 5.4 erläutert.

Der erfolgreiche Einsatz dieses Systems in einer anderen Domäne, der Erkennung der Konstruktionshandlungen “Schrauben” und “Stecken” beim Bauen von *Baufix*®-Aggregaten, zeigt die Flexibilität des Ansatzes [Fri00a, Fri00b].

5.3.2 Farbklassifikation

Das Ziel der Farbklassifikation ist die Ermittlung der Wahrscheinlichkeit $P(I_{x,y})$ für jedes Pixel (x, y) eines Bildes I dafür, daß dieses Pixel hautfarben ist. Alternativ kann dies auch als eine Transformation in einen Hautfarb-Farbraum interpretiert werden. Um Echtzeitfähigkeit zu erreichen, muß dies möglichst schnell gehen. Außerdem muß es möglich sein, auf eine Beleuchtungsveränderung reagieren zu können. Der vorgestellte Ansatz geht allerdings davon aus, daß es keine kontinuierlichen Beleuchtungsveränderungen gibt. Diese sollten nur vereinzelt auftreten. Ein adaptives Verfahren zum Klassifizieren von Hautfarbe, welches kontinuierliche Veränderungen verarbeiten kann, wird beispielsweise in [Fri03] vorgestellt. Dieses läßt sich leicht in den hier vorgestellten Ansatz integrieren.

Um flexibel auf die jeweiligen Bedingungen reagieren zu können, kommen zwei verschiedene Klassifikatoren zum Einsatz. Der erste, ein Polynomklassifikator sechsten Grades [Kum98], hat eine große Anzahl von Parametern, was eine detaillierte Modellierung des Eingangsraums erlaubt. Dafür ist das Training allerdings aufwendig und das Trainingsmaterial muß relativ fehlerfrei sein. Dies bedeutet, daß eine Offline-Trainingsphase nötig ist. Als zweiter Klassifikator kommt eine Modellierung von Hautfarbe im rg-Farbraum (siehe Gleichung 5.1, Seite 49 und Anhang A) durch eine Gaußverteilung zum Einsatz. Dieser Ansatz kommt mit wenigen Parametern aus, was das Online-Training des Klassifikators erlaubt. Durch Sprache hat der Benutzer intuitiv die Möglichkeit, dieses Online-Training anzustoßen. Siehe hierzu Kapitel 4.

Der Polynomklassifikator

Ziel des Polynomklassifikators ist es, eine Funktion $\mathbf{d}(\mathbf{c})$, den sogenannten Entscheidungsvektor, zu finden, die bei gegebenen Merkmalsvektor $\mathbf{c}$ den Zielvektor $\mathbf{y}(\mathbf{c})$ möglichst gut approximiert. Möglichst gut heißt dabei, daß der Erwartungswert $E\{\mathbf{y} - \mathbf{d}\}$ minimal werden soll. Die Dimension des Zielvektors ist gleich der Anzahl der Klassen. Gehört der Merkmalsvektor $\mathbf{c}$ zu der Klasse k, hat der Zielvektor an der k-ten Stelle eine 1, an allen anderen Stellen eine 0. Der Polynomklassifikator nutzt den *Approximationssatz von Weierstraß* zum Finden einer Funktion $\mathbf{d}(\mathbf{c})$ aus. Dieser besagt, daß jede stetige Funktion beliebig genau durch ein Polynom angenähert werden kann, sofern der Grad des Polynoms genügend hoch ist. Damit ergibt sich als Ansatz für eine Klasse k:

$$d_k(\mathbf{c}) = a_{0,k} + a_{1,k}c_1 + a_{2,k}c_2 + ... + a_{N,k}c_N + a_{N+1,k}c_1^2 + a_{N+2,k}c_1c_2 + ... \quad (5.2)$$

$$= \mathbf{a}^T \cdot \mathbf{x}(\mathbf{c}) \quad \text{mit} \quad \mathbf{x}(\mathbf{c}) = (1, c_1, ..., c_N, c_1^2, c_1c_2, ...)^T \quad (5.3)$$

N : Dimension des Merkmalsraums

Und somit:

$$\mathbf{d}(\mathbf{c}) = \mathbf{A}^T \cdot \mathbf{x}(\mathbf{c}) \quad \text{mit} \quad \mathbf{A} = (\mathbf{a}_1, ..., \mathbf{a}_K) \quad a_{i,j} \in \mathbb{R} \quad (5.4)$$

K: Anzahl Klassen

$\mathbf{d}(\mathbf{c})$ ergibt sich nun durch Minimierung von $E\{\mathbf{y} - \mathbf{d}\}$. Dies führt auf das lineare Gleichungssystem $E\{\mathbf{x}\mathbf{x}^T\} \cdot \mathbf{A} = E\{\mathbf{x}\mathbf{y}^T\}$, dessen Lösung $\mathbf{A}$ spezifiziert. Die Details und die Herleitung finden sich in [Sch96].

Im vorgestellten System wird ein Polynomklassifikator sechsten Grades eingesetzt, um YUV-Pixel in die Klassen $Hautfarbe$ und $\neg Hautfarbe$ zu klassifizieren. Damit ergibt sich für die Anzahl M der Parameter des Polynomklassifikators (N: Dimension des Merkmalsraums, G: Grad des Polynoms, K: Anzahl der Klassen):

$$M = \binom{N+G}{G} \cdot K = \binom{3+6}{6} \cdot 2 = 168 \tag{5.5}$$

Die Klassifikation wird mit Hilfe der Gleichung 5.4 durchgeführt. Dabei sind die resultierenden $d_k(\mathbf{c})$ allerdings keine Wahrscheinlichkeiten, was aber das Ziel der Klassifikation war. Um dies zu erreichen, wird ein *"confidence mapping"* durchgeführt [Sch96]. Hierbei wird jede Komponente d_k des Entscheidungsvektors $\mathbf{d}$ als Merkmal für ein neues zweidimensionales Klassifikationsproblem angesehen. Die Klasse $eigen$ zeigt dabei die Zugehörigkeit zu Klasse k bezogen auf d_k an, die Klasse $fremd$ Zugehörigkeit zu einer beliebigen Klasse ungleich k. Der Confidence-Wert ergibt sich damit zu $P(eigen|d_k)$. Mit Hilfe der Bayes'schen Regel kann dies aufgelöst werden zu:

$$P(eigen|d_k) = \frac{P(eigen)P(d_k|eigen)}{P(eigen)P(d_k|eigen) + P(fremd)P(d_k|fremd)} \tag{5.6}$$

Diese Werte lassen sich aus einer klassifizierten Stichprobe unter Annahme einer Klasse von Verteilungen schätzen. Hier wird von einer Normalverteilung der d_k ausgegangen, womit Mittelwert und Varianz von $P(d_k|eigen)$ und $P(d_k|fremd)$ geschätzt werden müssen.

Für eine Klassifikation eines kompletten Bildes ist die Berechnung der Gleichungen 5.4 und 5.6 für jedes Pixel zu langsam. Um dies zu beschleunigen, wird eine offline berechnete Lookup-Tabelle zur Klassifikation eingesetzt. Diese enthält das Klassifikationsergebnis für alle Merkmalsvektoren in einer Auflösung von sieben Bit für jede Farbkomponente. Durch die Reduktion der Auflösung von acht auf sieben Bit ergibt sich eine Größe der Tabelle von zwei MByte, was eine gute Ausnutzung von gängigen Second-Level-Caches aktueller Prozessoren erlaubt.

Wie beschrieben wird das Bild durch einen Schwellwert binarisiert und Regionen werden bestimmt. Zusätzlich läßt sich eine Bewertung $J(R_i^{(F)})$ jeder Hautfarbregion $R_i^{(F)}$ vornehmen, indem die mittlere Hautfarbwahrscheinlichkeit aller Pixel der Region bestimmt wird:

$$J(R_i^{(F)}) = \frac{\sum_{(x,y)\in R_i^{(F)}} P(I_{x,y})}{\left|R_i^{(F)}\right|} \tag{5.7}$$

Farbmodellierung durch eine Gaußverteilung

Der Polynomklassifikator erlaubt eine genaue Modellierung des Eingangsraumes, benötigt hierfür aber auch eine gute klassifizierte Stichprobe. Außerdem sind die Berechnungen für das Training relativ aufwendig. Als Alternative hierzu kommt die Modellierung der Hautfarbverteilung durch eine Gaußverteilung zum Einsatz.

Um unabhängiger von der Beleuchtung zu werden und gleichzeitig die Anzahl zu schätzender Parameter weiter zu reduzieren, findet die Klassifikation im normalisierten rg-Farbraum statt (Gleichung 5.1, Seite 49 und Anhang A). Damit müssen nur noch fünf Parameter geschätzt werden, die Mittelwerte μ_r und μ_g der Farben r und g und die aus drei unabhängigen Parametern bestehende symmetrische Kovarianzmatrix Σ. Als weitere Suchraumeinschränkung kommt der in [Sor00] vorgestellte *skin locus* zum Einsatz. Der skin locus, als schwarzer Bereich dargestellt in Abbildung 5.4, stellt den Bereich im Farbraum dar, den Hautfarbe unter verschiedenen Beleuchtungsverhältnissen und von verschiedenen Menschen unterschiedlicher Rassen einnimmt. Je nach eingesetzter Kamera verschiebt und verformt sich dieser Bereich leicht. Zusätzlich von der Klassifikation als Hautfarbe ausgeschlossen wird der im skin locus liegende Weißpunkt, der in Abbildung 5.4 als weißer Punkt dargestellt ist.

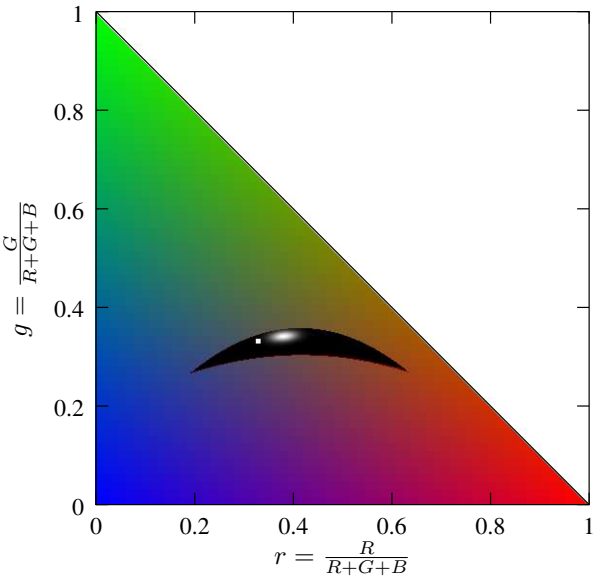


Abbildung 5.4: Der *skin locus* im rg-Farbraum. Innerhalb des skin locus ist eine Normalverteilung dargestellt, die zur Klassifikation eingesetzt wird. Der kleine weiße Punkt im skin locus ist der Weißpunkt.

Für die Ermittlung der Wahrscheinlichkeit P dafür, daß ein Punkt (x, y) des in den rg-Farbraum transformierten Kamerabildes I hautfarben ist, ergibt sich damit unter

Zuhilfenahme einer modellierten Hautfarbverteilung $\mathcal{N}_{\mu,\Sigma}$:

$$P(I_{x,y}) = \begin{cases} \mathcal{N}_{\mu,\Sigma}(I_{x,y}) & , \quad \text{PixelInSkinLocus}(I_{x,y}) \wedge \\ & \quad \neg\text{PixelInWhitePoint}(I_{x,y}) \\ 0 & , \quad \text{sonst} \end{cases} \tag{5.8}$$

Das Vorgehen zur interaktiven Schätzung der Normalverteilung $\mathcal{N}_{\mu,\Sigma}$ ist in Abschnitt 5.3.4 dargestellt. Zur Beschleunigung der Klassifikation als auch der Farbraumtransformation wird wie beim Polynomklassifikator eine Lookup-Tabelle mit sieben Bit YUV-Werten als Indizes eingesetzt. Die farbbasierte Bewertung der Regionen ist ebenfalls mit dem Polynomklassifikator identisch.

5.3.3 Einschränkung durch Bewegungsinformation

Wie in Abschnitt 5.3 dargestellt, kann allein durch Farbinformation keine sichere Segmentierung der Hände vorgenommen werden. Allerdings kann ausgenutzt werden, daß sich die Hände beim Ausführen der Gesten bewegen. Andere hautfarbene Regionen im Hintergrund oder auch das Gesicht bewegen sich typischerweise deutlich weniger oder sogar gar nicht. Daher wird Bewegungsinformation auf zwei verschiedenen Zeitskalen in die Segmentierung integriert. Einerseits werden Bereiche, in denen über eine längere Zeitspanne Bewegung stattfand, als Maske für die farbbasierte Segmentierung angesehen. Andererseits werden Änderungen zwischen den letzten beiden Frames zur Bewertung von Handhypothesen herangezogen. Auch der Mensch benutzt Bewegung sowohl zur Aufmerksamkeitssteuerung als auch zur Detektion von Objekten und Ereignissen [Joh73][Tov96, Kapitel 10], was dies zu einem kognitiv adäquaten Ansatz macht.

Allein basierend auf dem Differenzbild zweier Frames kann keine Maske zum Ausfiltern aller sich bewegender Bereiche erstellt werden. Wie in Abbildung 5.5, Spalte (d) zu sehen ist, tauchen im Differenzbild sich bewegende homogene Bereiche nicht auf. Basierend auf der Idee des *motion-history image* (MHI) und des *motion-energy image* [Bob97] werden daher mehrere Differenzbilder summiert, um eine Aktivitätskarte D zu erhalten. Die Karte ist in Abbildung 5.5 in den Spalten (b) und (c) dargestellt.

Bei der Berechnung des MHI wird es auf einen festen Wert τ gesetzt, falls das aktuelle Differenzbild Bewegung anzeigt, ansonsten wird der Wert des alten MHI linear erniedrigt übernommen. Im Gegensatz dazu wird bei der hier realisierten Aktivitätskarte D ein exponentieller Abfall mit Integration des aktuellen Differenzbildes D' benutzt:

$$D'_{x,y} = \begin{cases} 255 & \text{falls } |I^{(Y)}_{x,y}(t-1) - I^{(Y)}_{x,y}(t)| > T_A \\ 0 & \text{sonst} \end{cases} \tag{5.9}$$

$$D_{x,y}(0) = 0 \tag{5.10}$$

$$D_{x,y}(t) = \lfloor D_{x,y}(t-1) * W \rfloor + \lfloor D'_{x,y} * (1-W) \rfloor \tag{5.11}$$

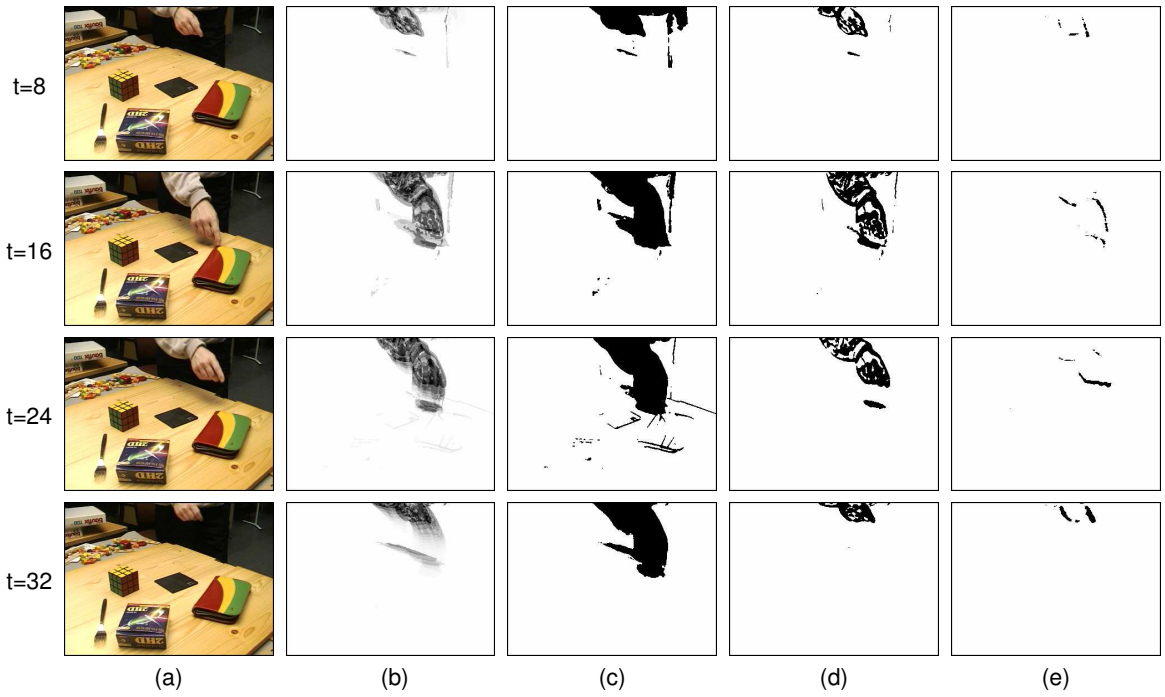


Abbildung 5.5: Vier Frames aus einer Sequenz einer Zeigegeste: (a) Kamerabilder. (b) Aktivitätskarte. (c) Binarisierte Aktivitätskarte. (d) Letztes Differenzbild des Y-Kanals, welches in die Aktivitätskarte integriert wurde. (e) Differenzbild des V-Kanals.

Dabei ist 255 der Maximalwert bei den im System eingesetzten 8 Bit Bildern. T_A wird so gewählt, daß das Rauschen der Kamera bei aufeinanderfolgenden Bildern sicher unterdrückt werden kann. Bei den Bildern in Abbildung 5.5 wurde hierfür 12 gewählt. Die Konstante W, hier 0.7, gibt an, wie groß der in die Aktivitätskarte integrierte zeitliche Kontext sein soll.

Abbildung 5.6 zeigt beispielhaft die Wirkung von Formel 5.11 durch einen Plot von $D_{x,y}$ für vier unterschiedliche Fälle. Zum Zeitpunkt $t = 0$ und zu einem Zeitpunkt $t = 1$, $t = 5$ oder $t = 9$ ist Bewegung aufgetreten und $D'_{x,y}$ war ungleich Null. Wie zu sehen ist, wird der bestehende Wert von $D_{x,y}$ mit in die Berechnung übernommen, was beispielsweise bei dem zweiten Graphen dazu führt, daß $D_{x,y}$ erst 12 statt 10 Iterationen nach dem letzten Auftreten von Bewegung Null wird. Dadurch werden sich stark bewegende Bereiche in der Aktivitätskarte noch einmal verstärkt.

Aus der Aktivitätskarte D läßt sich nun eine Maske M durch einfaches Binarisieren errechnen:

$$M_{x,y}(t) = \begin{cases} 255 & \text{falls } D_{x,y}(t) > 0 \\ 0 & \text{sonst} \end{cases} \tag{5.12}$$

Nach einem Glätten dieser Maske mit einem Medianfilter wird die Maske mit dem farb-

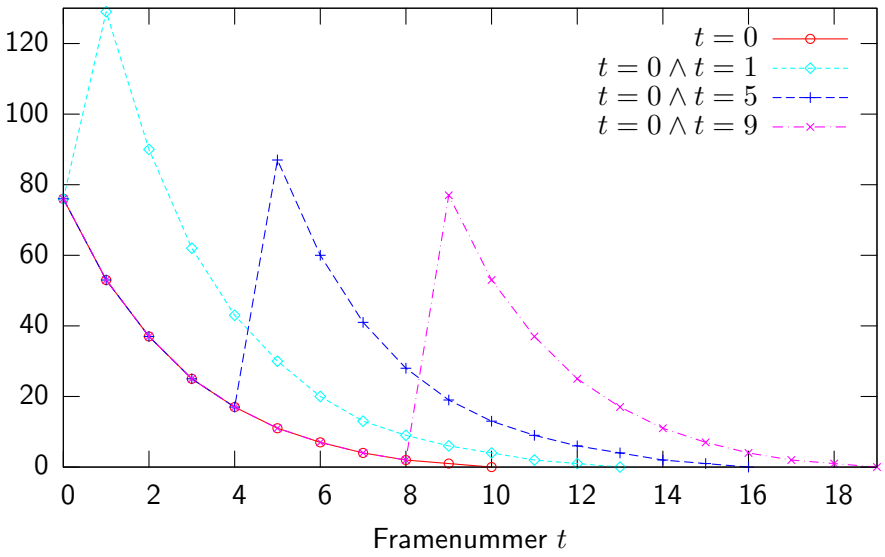


Abbildung 5.6: Darstellung der Funktion $D_{x,y}$ zur Berechnung einer Aktivitätskarte. Zu den Zeiten $t = 0$ und $t = 1$, $t = 5$ bzw. $t = 9$ trat jeweils Bewegung auf.

segmentierten Wahrscheinlichkeitsbild multipliziert. Dadurch werden alle sich nicht bewegenden Bereiche im Bild auf Null gesetzt.

Um zu vermeiden, sich wenig bewegende Objekte mit auszufiltern, muß die Maske allerdings einen relativ großen Bereich erhalten. Außerdem kommt es beispielsweise durch Schatten zu einer weiteren Erweiterung der Maske. Daher wird durch Integration nur der aktuellsten Bewegungsinformation in die Bewertung der Handhypothesen Bewegungsinformation auf einer kleineren Zeitskala integriert. Hierbei kann zusätzlich ausgenutzt werden, daß durch sich schnell bewegende Objekte im Differenzbild größere Regionen entstehen als durch sich langsam bewegende Objekte, um so die Bewertung sich schnell bewegender Hypothesen weiter zu erhöhen. Um Information über die Bewegungen zu gewinnen, wird das Differenzbild $D^{(R)}$ des V-Kanals benutzt:

$$D^{(R)}_{x,y} = \begin{cases} 255 & \text{falls } |I^{(V)}_{x,y}(t-1) - I^{(V)}_{x,y}(t)| > T_B \\ 0 & \text{sonst} \end{cases} \tag{5.13}$$

Der V-Kanal ist von Schatten nicht so stark beeinflußt wie der Helligkeit kodierende Y-Kanal und ist im Farbraum in eine ähnliche Richtung wie die rötliche Hautfarbe ausgerichtet. Spalte (e) in Abbildung 5.5 zeigt diese Differenzbilder. Nach der Generierung des Bildes $D^{(R)}$ werden die Größe und der Schwerpunkt aller Regionen im Differenzbild bestimmt. Für jede Differenzregion $R^{(D)}$ wird der Abstand zwischen deren Schwerpunkt und den Umrandungen aller Hautfarbregionen $R^{(F)}_i$ berechnet. Die

Differenzregion $R^{(D)}$ wird der nächstgelegenen Hautfarbregion $R_*^{(F)}$ zugeordnet, falls dies eindeutig möglich ist. Dies heißt, daß

- der Schwerpunkt der Differenzregion in der Hautfarbregion liegt
- oder
  - $R^{(D)}$ nahe an der Hautfarbregion $R_*^{(F)}$ liegt
  - und der Abstand zur nächsten Hautfarbregion mindestens doppelt so groß ist, d.h.

$$\forall R_i^{(F)} \backslash R_*^{(F)} : |R^{(D)} R_i^{(F)}| \geq 2 \cdot |R^{(D)} R_*^{(F)}| \tag{5.14}$$

Die Bewertung $J(R_*^{(F)})$, bisher gegeben durch die Farbbewertung nach Gleichung 5.7, wird nun gewichtet mit der Größe der Differenzregion erhöht:

$$J(R_*^{(F)}) \leftarrow J(R_*^{(F)}) + \frac{|R^{(D)}|}{\sqrt{|R_*^{(F)}|}} \tag{5.15}$$

5.3.4 Interaktive Modellgenerierung

Eine Anforderung an die Gestenerkennung war die Fähigkeit, auch bei sich ändernder Beleuchtung arbeiten zu können. Wie auf Seite 52 dargestellt, haben Verfahren zum automatischen Ausgleich von Farbänderungen noch nicht die für Farbbildverarbeitung nötige Leistungsfähigkeit erreicht. Daher muß mit einer Anpassung des farbbasierten Klassifikators auf die Veränderungen reagiert werden. Dies kann geschehen, indem interaktiv mit Hilfe des Benutzers der Klassifikator neu initialisiert wird. Der hier vorgestellte Ansatz geht davon aus, daß es keine kontinuierlichen Beleuchtungsveränderungen gibt. Treten diese auf, kann leicht ein adaptives Verfahren zum kontinuierlichen Anpassen des Klassifikators, wie es in [Fri03] vorgestellt wird, integriert werden. Im folgenden wird zuerst ein Überblick über das realisierte Verfahren gegeben. Anschließend werden Details dargestellt.

Überblick

Für die Initialisierung eines Hautfarbklassifikators ist es nötig, sicher Hautfarbe in der Szene zu finden, ohne ein aktuelles Modell von Hautfarbe zu haben. Daher können nur andere Modalitäten genutzt werden. Steht eine sichere texturbasierte Gesichtserkennung zur Verfügung und kann man davon ausgehen, daß sich ein Gesicht in der Szene befindet, kann eine Initialisierung durch ein gefundenes Gesicht durchgeführt werden. Eine Alternative ist eine Interaktion mit dem Benutzer. Dies bietet sich bei dem in dieser Arbeit vorgestellten sehr auf Interaktion ausgelegtem System an. Das einfachste Vorgehen wäre es, wenn der Benutzer seine Hand an eine fest vorgegebene Position im Bild bringen würde und die Initialisierung durch einen Buttonclick oder

ähnliches starten würde. Dieses Verfahren ist im vorgestellten System zwar auch realisiert, ist für ein komfortables und intuitives System aber nicht wirklich praktikabel. Daher wird hier ein alternativer dritter Ansatz favorisiert.

Soll eine Modellinitialisierung gestartet werden, wird der Benutzer aufgefordert, mit seinen Händen vor der Kamera zu winken. Da im Normalfall Gesten des Benutzers analysiert werden sollen, müssen seine Hände während der Durchführung der Geste im Sichtfeld der Kamera sein. Somit muß der Benutzer für die Initialisierung keine besondere Position einnehmen und er braucht auch keine visuelle Rückmeldung wie im oben vorgestellten zweiten Ansatz. Durch die Ausnutzung der beim Winken verstärkt auftretenden Modalität Bewegung können die hautfarbenen Hände gefunden werden, ohne ein gutes Modell von Hautfarbe zu haben. Dazu werden ein sehr allgemeines Modell von Hautfarbe, ein Differenzbild und ein inkrementell erstelltes Hintergrundbild kombiniert eingesetzt. Alle diese Verfahren erlauben jeweils das Ausfiltern einiger Bereiche, die auf Grund von Bewegung, Ähnlichkeit zum Hintergrund oder Farbe vermutlich nicht hautfarben sind. Die hierbei in jedem Bild übrig bleibenden Regionen sind höchstwahrscheinlich hautfarben. Sie werden nach einer weiteren Selektion und akkumuliert über einen Zeitraum von wenigen Sekunden als Trainingsmaterial für einen neuen Hautfarbklassifikator eingesetzt.

Details der Realisierung

Initialisiert wird der in Abschnitt 5.3.2 vorgestellte gaußbasierte Klassifikator. Im Gegensatz zum Polynomklassifikator hat dieser weniger Parameter und ist somit auch weniger anfällig für Fehler in einer Trainingsstichprobe. Für den Klassifikator müssen die für die Normalverteilung benötigten Mittelwerte μ_r und μ_g der Farben r und g und die aus drei unabhängigen Parametern bestehende symmetrische Kovarianzmatrix Σ geschätzt werden. Abbildung 5.7 zeigt den prinzipiellen Ablauf zur Gewinnung des benötigten Trainingsmaterials. Abbildung 5.8 zeigt einige Beispielbilder zu einer durchgeführten Initialisierung. Der initiale Ablauf ist ähnlich zur normalen Handverfolgung. Eine Farbklassifikation und eine Bewegungssuche basierend auf einem Differenzbild nehmen eine initiale Segmentierung des Bildes vor. Da noch kein spezielles Hautfarbmodell vorhanden ist, wird der in Abschnitt 5.3.2 eingeführte skin locus zur Farbklassifikation benutzt. Dieser enthält allerdings Hautfarbe bei beliebigen Beleuchtungsverhältnissen und deckt daher einen großen Bereich im Farbraum ab – es wird bedeutend zu viel als Hautfarbe klassifiziert. Zur Verbesserung wird dieses Bild mit einem Differenzbild des V-Kanals, wie es auch bei der normalen Handverfolgung zum Einsatz kommt, multipliziert. Die Trennung in eine langzeitige Aktivitätskarte und eine kurzzeitige Regionenbewertung ist hier nicht nötig, da

1. der Benutzer explizit schnelle Bewegungen durchführen soll und damit die Differenzregionen größer werden. Um dies sicher zu stellen, wird die Aufnahme eines neuen Bildes um bis zu 0.2 Sekunden verzögert, falls die benötigte Zeit zur Verarbeitung des letzten Bildes sehr kurz war.

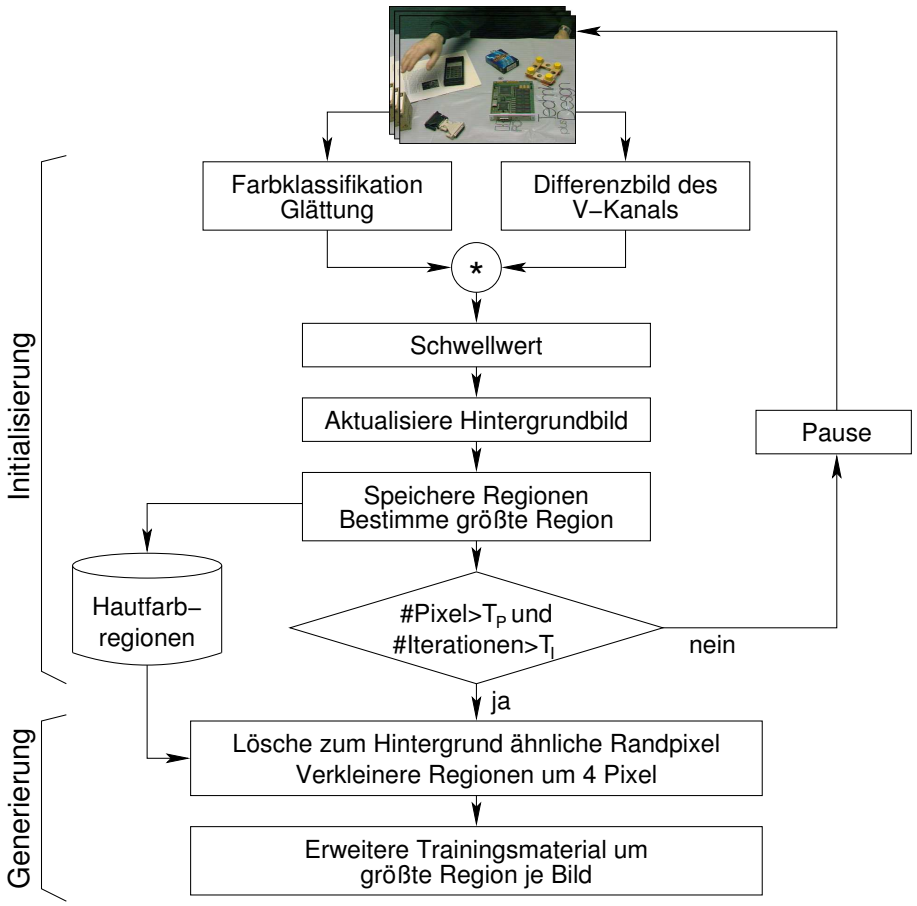


Abbildung 5.7: Flußdiagramm der interaktiven Initialisierung des Hautfarbmodells. In einer ersten Phase werden mögliche Hautfarbbereiche gesucht. In einer zweiten Phase werden diese verifiziert und zum Training genutzt.

2. in dieser Anwendung keine vollständige Segmentierung der Hand nötig ist. Zur Gewinnung von Trainingsmaterial genügt es, jeweils nur einen Teil des Handbereiches mit aufzunehmen.

Aber auch hier bleiben noch viele Bereiche erhalten, die nicht hautfarben sind. Um eine weitere Verbesserung zu erreichen, kann ausgenutzt werden, daß die Modellgenerierung nur eine kurze Zeit in Anspruch nimmt, typischerweise nicht länger als 4 Sekunden, und der Benutzer in dieser Zeit die Szene nicht ändert. Die Annahme eines statischen Hintergrundes für diese kurze Zeit kann daher gemacht werden. Daher wird ein Hintergrundbild iterativ aufgebaut, welches alle sich nicht bewegenden Bereiche des Bildes enthält. Ein Pixel wird als zum Hintergrund zugehörig angenommen ($\mathrm{HG}(x, y) = \mathrm{TRUE}$), falls es in zwei aufeinanderfolgenden Differenzbildern $D^{(H)}$ nicht

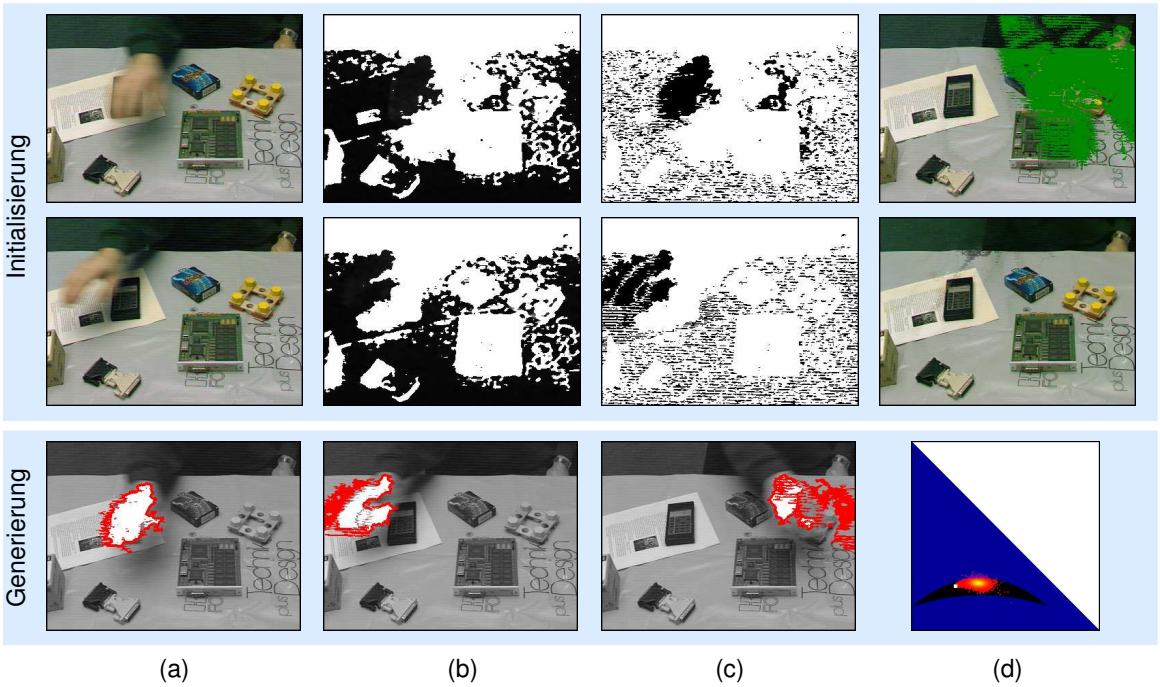


Abbildung 5.8: Beispielbilder zu der in Abbildung 5.7 dargestellten interaktiven Initialisierung des Hautfarbmodells, Initialisierung: (a) Kamerabilder. (b) Farbsegmentierung. (c) Kombiniertes Farb-/Differenzbild. (d) Aktuelles Hintergrundbild. Generierung: (a)-(c) Einige zum Training benutzte Regionen. (d) Gewonnene Trainingsdaten dargestellt im rg-Farbraum.

auftrat:

$$D_{x,y}^{(H)}(t) = \begin{cases} 255 & \text{falls } \frac{1}{3}(\ |I_{x,y}^{(Y)}(t-1) - I_{x,y}^{(Y)}(t)| + \\ & \quad |I_{x,y}^{(U)}(t-1) - I_{x,y}^{(U)}(t)| + \\ & \quad |I_{x,y}^{(V)}(t-1) - I_{x,y}^{(V)}(t)| \ ) > T_H \\ 0 & \text{sonst} \end{cases} \tag{5.16}$$

$$\mathrm{HG}(x,y) := D_{x,y}^{(H)}(t) + D_{x,y}^{(H)}(t-1) = 0 \tag{5.17}$$

Das Hintergrundbild H ergibt sich nun durch Mittelung aller Pixel $I_{x,y}$, die als zum Hintergrund zugehörig angenommen wurden. Abbildung 5.8 zeigt zwei verschiedene Stadien des Hintergrundbildes, einmal nach vier Bildern und einmal nach 13. Für alles Grün dargestellte wurden dem Hintergrund bisher keinerlei Pixel zugeordnet.

Um das Hintergrundbild für die Aufbereitung der Trainingsdaten nutzen zu können, muß es erst erstellt werden. Für die Erstellung werden aber alle Bilddaten, die bis zum Erreichen eines Abbruchkriteriums aufgenommen wurden, eingesetzt. Daher werden

bis zum Erreichen dieses Abbruchkriteriums die aus dem Farb- und Bewegungsbild ermittelten Regionen zwischengespeichert und erst im Anschluß endgültig analysiert. Zur Erfüllung des Abbruchkriteriums müssen genügend Iterationen durchlaufen sein und die Summe der Pixel der jeweils größten Regionen eines jeden Bildes müssen genügend groß sein. Dies stellt Variabilität in der Lernstichprobe durch die Größe der Stichprobe und die leichte Variierung der Aufnahmesituation sicher. Die Anzahl Iterationen T_I war im vorgestellten System auf 20 gesetzt, die Anzahl Pixel T_P auf 1000.

Nach dem Erreichen des Abbruchkriteriums kann die Generierung des Farbmodells beginnen. Zuerst werden von jeder gespeicherten Region alle Pixel entfernt, die

- entweder nicht weiter als vier Pixel vom Rand der Region entfernt sind, womit durch Rauschen der Kamera und durch Bewegung verursachte Fehler verringert werden,
- oder ähnlich zum Hintergrundbild sind und eine Verbindung zum Rand der Region haben, die nur über Pixel verläuft, die ebenfalls ähnlich zum Hintergrundbild sind. Durch die Beschränkung auf Randpixel der Region wird implizit davon ausgegangen, daß die Region keine Löcher enthält, was für Hände korrekt sein sollte. Die Ähnlichkeit $A^{(H)}$ eines Pixels P an der Position (x, y) einer Region ist hierbei vergleichbar zu $D^{(H)}$ definiert:

$$A^{(H)}(P_{x,y}) := \frac{|P^{(Y)} - H^{(Y)}_{x,y}| + |P^{(U)} - H^{(U)}_{x,y}| + |P^{(V)} - H^{(V)}_{x,y}|}{3} \leq T_H \qquad (5.18)$$

Für jedes Bild, daß heißt jede Iteration der Initialisierungsphase, werden alle Pixel der nach der Säuberung größten Region als Daten für die Modellgenerierung aufgenommen. Durch die Beschränkung auf eine Region pro Bild wird die Gefahr der Aufnahme von Hintergrundregionen verringert. Abbildung 5.8 zeigt in der Zeile Generierung in Weiß Pixel, die für das jeweilige Bild in das Modell aufgenommen wurden. In Rot sind die entfernten Pixel dargestellt. Daneben sind in Rot die ausgewählten Pixel innerhalb des skin locus im rg-Farbraum dargestellt. Aus diesen Daten kann nun die für den Klassifikator benötigte Normalverteilung geschätzt werden. Abbildung 5.9 zeigt für zwei Beispielbilder das Klassifikationsergebnis mit dem neu trainierten Klassifikator. Im Gegensatz zur Klassifikation mit dem kompletten skin locus werden nun die Hände sauber segmentiert. Zusätzlich werden Teile der Holzleisten als Hautfarbe identifiziert, was allerdings auf Grund der Farbe korrekt ist. Andere Teile des Bildes, wie das Blatt Papier, werden nun korrekt als nicht Hautfarbe klassifiziert.

5.4 Gesteninterpretation

Das Ziel der Gestenerkennung ist die Interpretation der für das in Kapitel 3 vorgestellte System wichtigen Gesten. Insbesondere sind dies Zeigegesten und eine Analyse

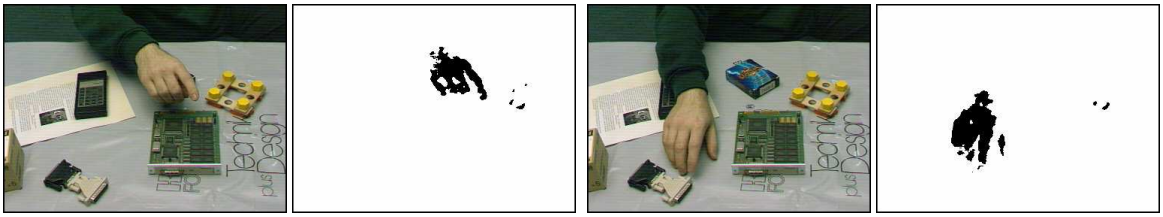


Abbildung 5.9: Klassifikationsergebnis zweier Beispielbilder mit dem in Abbildung 5.8 interaktiv initialisierten Klassifikator.

der beim Herausnehmen von Objekten auftretenden Greifaktion. Bei den Zeigegesten muß festgestellt werden, (a) ob sie auftreten, (b) wann sie auftreten und (c) wohin gezeigt wird. Bei den Greifaktionen ist neben der Erkennung des Auftretens, der Ort, an dem die Greifaktion stattfindet, das Entscheidende.

Mit Hilfe der in Abschnitt 5.3 vorgestellten Handdetektion liegen für jedes neu aufgenommene Bild segmentierte Handhypothesen vor, die zusätzlich mit Hilfe von Farb- und Bewegungsinformationen bewertet sind. Um zeitliche Kontextinformation ausnutzen zu können, findet die Analyse der Gesten basierend auf Trajektorien der Hände statt. Zusätzlich wird weiteres Kontextwissen integriert. Trajektorien von Händen allein sind nicht aussagekräftig genug, um hieraus auf stattgefundene Gesten zu schließen. Abbildung 5.10 zeigt exemplarisch Trajektorien von einigen real durchgeführten Zeigegesten und Greifaktionen. Allein basierend auf der Trajektorie sind diese beiden Klassen nicht zu trennen. Zusätzlich müssen auch Bewegungen der Hände, die nicht diesen beiden Klassen angehören, zurückgewiesen werden. Erwartungen aus der Sprache und der momentanen Dialogsituation werden daher als wichtige Komponenten mit in die Analyse integriert.

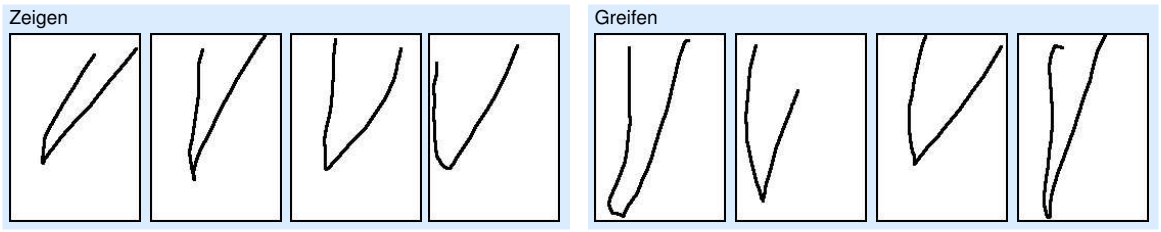


Abbildung 5.10: Handtrajektorien einiger Zeigegesten und einiger Greifaktionen. Allein basierend auf der Trajektorie sind diese im Allgemeinen nicht unterscheidbar.

Zur Gewinnung der Trajektorien kommen Kalmanfilter [Kal60, May79] zum Einsatz, deren Systemmodell Position, Geschwindigkeit und Beschleunigung jeweils eines Regionenschwerpunktes repräsentieren [Wie00]. Nach der Prädiktion des Systemzu-

standes an Hand des Modells wird jedem Kalmanfilter die Handhypothese zugewiesen, die dem prädizierten Schwerpunkt am nächsten liegt, falls die Hypothese innerhalb einer maximalen Entfernung liegt. Andernfalls wird der Kalmanfilter gelöscht. Für nicht zugewiesene Handhypothesen mit hoher Bewertung werden neue Kalmanfilter initialisiert. Nach der Zuordnung der Hypothesen werden die prädizierten Systemzustände an Hand der gemessenen Regionenschwerpunkte korrigiert.

Um eine Auswahl aus der Menge der in jedem Zeitschritt ermittelten Trajektorien treffen zu können, werden für alle Trajektorien zwei Bewertungen bestimmt. Die erste Bewertung J_D ist der Mittelwert der Regionenbewertungen $J(R_i^{(F)})$ der letzten 100 Zeitschritte. Der aus der Farbbewertung stammende Teil der Regionenbewertung ist auf das Intervall [0...1] beschränkt, siehe Gleichung 5.7, Seite 56. Der bewegungsbasierte Teil nach Gleichung 5.15, Seite 61 ist hingegen nicht beschränkt. Daher bewertet diese erste Bewertung J_D die Trajektorie insbesondere im Hinblick auf deren Dynamik. Die zweite Bewertung J_S bewertet zusätzlich die Stabilität der Trajektorien bezüglich vergangener Zeit z und maximal zurückgelegtem Weg w seit dem Start der Trajektorie:

$$J_S = \frac{\tanh\left(\frac{z}{T_Z}\right) \cdot \tanh\left(\frac{w}{T_W}\right)}{\tanh(1) \cdot \tanh(1)} \tag{5.19}$$

T_Z und T_W sind dabei zwei Konstanten. Abbildung 5.11 zeigt einen Plot dieser Funktion. Im folgenden wird nun die Erkennung von Zeigegesten und Greifaktionen basierend auf den bewerteten Trajektorien beschrieben.

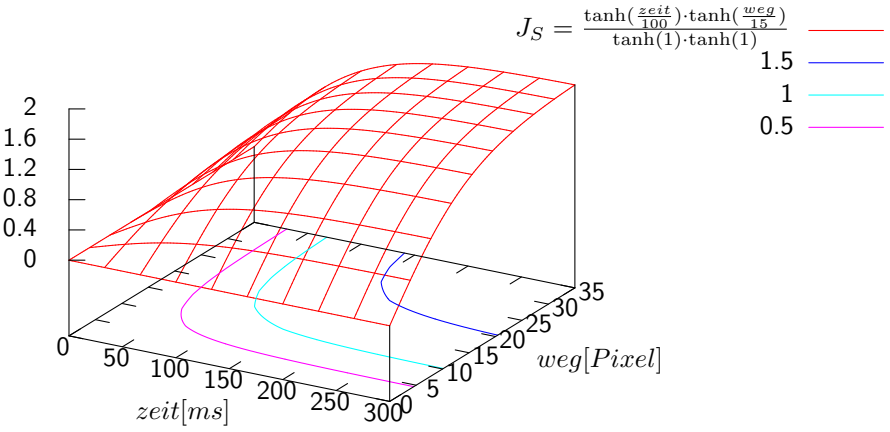


Abbildung 5.11: Die Bewertungsfunktion J_S für Handtrajektorien. Die Konstanten T_Z und T_W aus Gleichung 5.19 sind dabei auf 100 ms bzw. 15 Pixel gesetzt.

5.4.1 Zeigegesten und deren Kontext

Deiktische Gesten haben ohne einen definierten Kontext, wie er beispielsweise durch eine sprachliche Äußerung gegeben sein kann, keine klare Bedeutung [Wex98][McN92, Kapitel 2]. Im Deutschen wie im Englischen ist ein definiter Artikel in der Sprache wie in dem Satz "Nimm den Stift." ein Hinweis auf eine mögliche Geste. Kommt dagegen in der objektbestimmenden Nominalphrase ein unbestimmter Artikel vor, wie in dem Satz "Nimm einen Stift.", kann davon ausgegangen werden, daß der Satz durch keine deiktische Geste begleitet wurde. Im System soll der Benutzer die Möglichkeit haben, durch Gesten auf Objekte hinzuweisen. Daher wird das Auftreten eines definiten Markers in der Sprache, im Deutschen ein definiter Artikel, in der objektbestimmenden Nominalphrase als Voraussetzung für den Start der Gestenerkennung zur Erkennung von Zeigegesten angesehen.

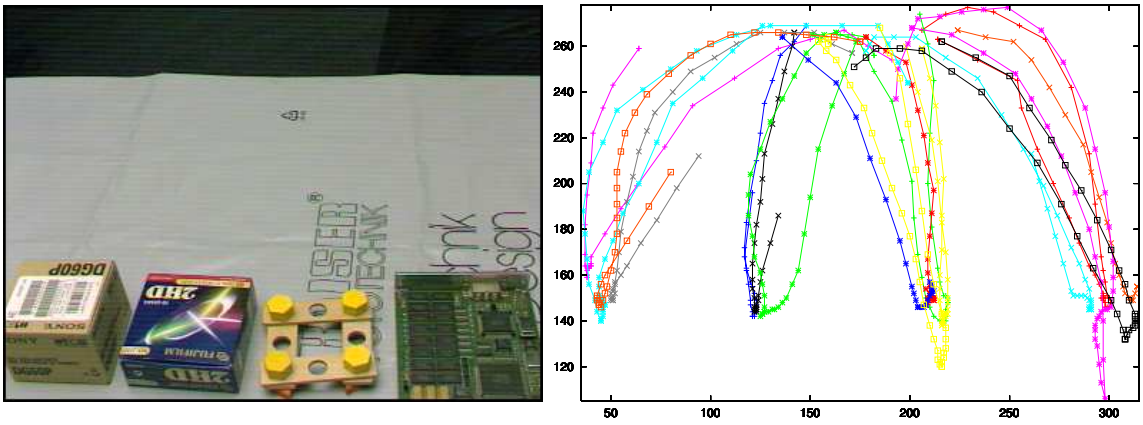


Abbildung 5.12: Eine Studie zur Untersuchung des zeitlichen Verhaltens von Zeigegesten zu Sprache: (a) In der Studie benutzte Szene. (b) Trajektorien einiger durchgeführter Zeigegesten.

Alle durch ESMERALDA erhaltenen sprachlichen Worthypothesen sind mit einem Zeitstempel für jedes Wort versehen. Damit wird eine zeitliche Integration mit der Gestenerkennung, die Zeitstempel durch die Aufnahmezeitpunkte der Kamerabilder hat, möglich. Um diese Integration zu realisieren, muß aber zuerst bekannt sein, in welchem zeitlichen Rahmen eine Zeigegeste zu der Sprache auftritt. Hierzu wurde eine Studie durchgeführt, in der fünf Versuchspersonen auf Objekte in einer Szene gezeigt haben, während sie eine sprachliche Anweisung gaben. Abbildung 5.12 (a) zeigt die in der Studie eingesetzte Szene, einen Tisch, auf dem nahe der Kamera vier Objekte liegen. Die Probanden wurden jeweils aufgefordert, sich vor den Tisch zu setzen. Danach sollten sie abwechselnd auf die Objekte zeigen, während sie das System sprachlich auffordern sollten, das jeweilige Objekt zu nehmen. Dies wurde den

Probanden demonstriert, indem bei dem Satz “Nimm das Teil.” auf ein Objekt gezeigt wurde. Zusätzlich wurde den Probanden der Zweck des Versuches erklärt und darauf hingewiesen, daß das Vorhandensein eines definiten Artikels und einer Trajektorie für die Zeigegeste wichtig ist.

Sowohl die Trajektorien der Handbewegungen als auch die Sprache wurden von dem bisher vorgestellten System automatisch erkannt und zusammen mit den jeweiligen Zeitstempeln protokolliert. Abbildung 5.12 (b) zeigt die Trajektorien einiger der bei der Studie durchgeführten Zeigegesten. In allen sprachlichen Äußerungen wurde jeweils der Zeitpunkt T_S bestimmt, zu dem der in den jeweiligen Sätzen aufgetretene definite Artikel zur Hälfte gesagt wurde. In den Trajektorien wurde der Zeitpunkt T_D bestimmt, an der die Hand am nächsten an den Objekten und damit am nächsten an dem unteren Bildrand in der aufgenommenen Szene war. Durch Rauschen und leichte Veränderungen im aufgenommenen Bild kann die Segmentierung der Hand auch bei nicht dynamischen Szenen leicht schwanken. Um diesem Rechnung zu tragen, wurde T_D bei sich nur noch leicht ändernder Position der Trajektorie gesetzt, auch wenn der minimale Objektabstand noch nicht ganz erreicht war.

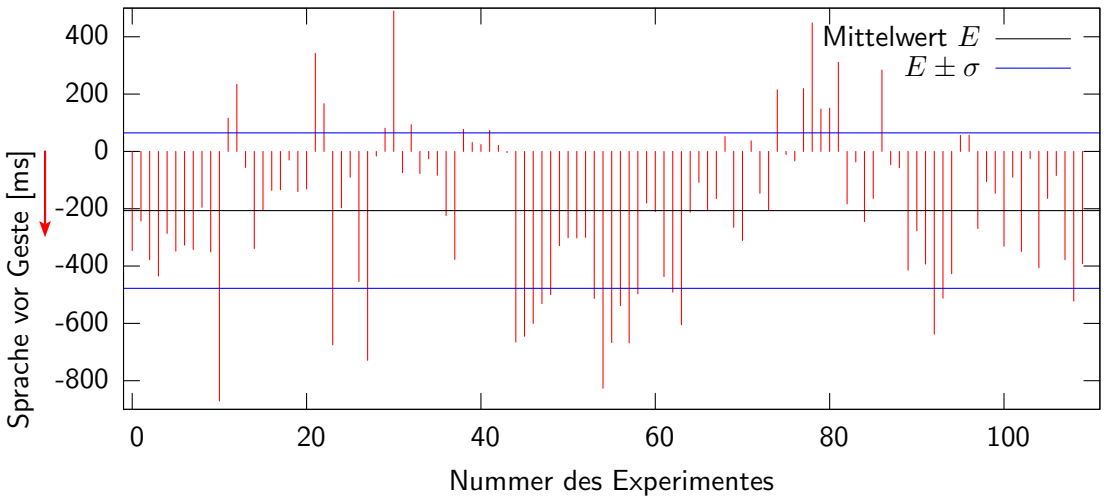


Abbildung 5.13: Der zeitliche Abstand Sprache ⇔ Geste bei allen durchgeführten Experimenten. Durchschnittlich lag der definite Artikel der sprachlichen Äußerung 206 Millisekunden vor der Zeigegeste.

Insgesamt wurden von den fünf Probanden 110 auswertbare Anweisungen mit jeweils einer Zeigegeste an das System gegeben. Bei einigen Anweisungen war entweder die Spracherkennung oder die Handverfolgung zu fehlerhaft, als daß diese Anweisungen in die Auswertung mit einfließen konnten. Abbildung 5.13 zeigt den Wert $T_S - T_D$ und damit den zeitlichen Abstand zwischen der Sprache und der Geste für alle 110 korrekt durchgeführten und erkannten Anweisungen. Im Mittel lag der definite Arti-

kel in der Sprache 206 Millisekunden vor der Zeigegeste. Die Standardabweichung der 110 Messungen betrug 271 Millisekunden. Bei 23 Messungen lag die Geste vor dem Artikel. Damit gibt es zwar eine merkliche Variabilität im Auftreten der Geste, diese ist aber nicht so groß, als daß keine Zuordnung der Geste an Hand des definiten Artikels möglich wäre. Damit wird bei Auftreten eines definiten Artikels eine Geste 0.2 Sekunden nach dem Artikel hypothetisiert und an Hand der Trajektorie in zeitlicher Nähe verifiziert. Dies ist im nächsten Abschnitt genauer beschrieben.

5.4.2 Erkennung von Zeigegesten

Nach dem Auftreten einer sprachlichen Äußerung mit einem definiten Artikel wird die zum Ende der Äußerung als Zeigegeste am geeignetesten erscheinende Trajektorie K_D weiter untersucht. Die Auswahl der Trajektorie K_D ist zweistufig. In einem ersten Schritt wird die zu einem Zeitschritt beste Trajektorie $K^*(t)$ bestimmt. In einem zweiten Schritt wird die global beste Trajektorie K_D bestimmt. Für die Auswahl der Trajektorie werden die auf Seite 67 eingeführten Bewertungen J_D und J_S benutzt. Innerhalb jedes Zeitschrittes t wird die für diesen Zeitschritt beste Trajektorie $K^*(t)$ aus der Menge der für diesen Zeitschritt bestimmten Trajektorien $K(t)$ bestimmt:

$K^*(t) = K_1(t)$		
FOR $K_i \in K(t)$		
	IF	$J_S(K_i) > J_S(K^*) \wedge [J_S(K^*) < 1 \vee J_D(K_i) > 1.2]$
	THEN	$K^*(t) = K_i$

(5.20)

Für einen Zeitschritt werden damit Trajektorien mit möglichst guter Stabilitätsbewertung bevorzugt, falls diese in ihrer näheren Vergangenheit Dynamik aufwiesen. Die gewählte Trajektorie zur Analyse der Zeigegeste sollte möglichst spät aufgenommen sein, um möglichst viel von der Zeigebewegung zu enthalten. Zusätzlich muß sie eine Bewertung aufweisen, die sie als sinnvolle Trajektorie ausweißt. Daher wird die Trajektorie $K^*(t)$ als Kandidat K_D für die Zeigegeste übernommen, falls:

$$K_D \leftarrow K^*(t) : J_D(K^*(t)) > 1 \vee J_S(K_D) < 1 \tag{5.21}$$

Zur weiteren Verifikation des Auftretens einer Zeigegeste wird die Trajektorie K_D untersucht. Abbildung 5.14 zeigt wichtige Parameter dieser Untersuchung an Hand der in Abbildung 5.15 dargestellten Zeigegeste. Der Benutzer hat hier im Zeitraum $t = 0$ bis $t = 970$ den Satz "Nimm das Teil." geäußert und gleichzeitig eine Zeigegeste ausgeführt. Zur Verifikation des Auftretens der Geste wird nun die Umgebung der Trajektorie um den Zeitpunkt T_S betrachtet, wobei T_S den Zeitpunkt des Auftretens der Wortmitte des definiten Artikels in der Sprache angibt. Unter Beachtung der Ergebnisse der im letzten Abschnitt vorgestellten Studie wird die Ableitung der Trajektorie in einem Bereich $A = [T_S - 100ms, T_S + 500ms]$ betrachtet. Das Auftreten einer Zeigegeste wird angenommen, falls der minimale Gradientenbetrag $|\Delta K_D^{(min)}(A)|$

im Bereich A unter einem Schwellwert $T1$ liegt und die Differenz des mittleren Gradientenbetrags $|\overline{\Delta K_D(A)}|$ und des minimalen Gradientenbetrags $|\Delta K_D^{(min)}(A)|$ über einem zweiten Schwellwert $T2$ liegt. Sind diese beiden Bedingungen erfüllt, ist dies ein Hinweis auf eine dynamische Bewegung, innerhalb dessen eine kürzere Ruhepause lag – die Hand hatte sich beispielsweise zum Zeigen nach vorne bewegt, ruhte kurz und wurde wieder zurückgezogen. $|\Delta K_D^{(min)}(A)|$ spezifiziert den Zeitpunkt T_D der Zeigegeste.

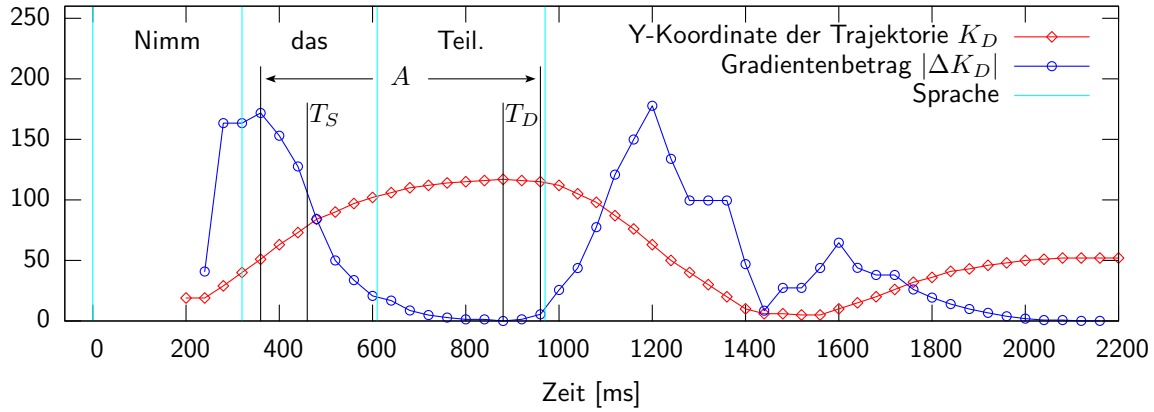


Abbildung 5.14: Verifikation des Auftretens der Zeigegeste aus Abbildung 5.15 mit Hilfe der Sprache und der Handtrajektorie K_D. Ausgehend von dem definiten Artikel "das" zum Zeitpunkt T_S ist eine Zeigegeste bei $t = T_D$ gefunden worden.

Durch diese Spezifizierung des Suchbereiches A wie auch durch die Schwellwerte werden zum Teil gültige Zeigegesten ausgeschlossen. In der Studie des letzten Abschnittes wurden beispielsweise einige Zeigegesten in einem Bereich von $T_S - 490ms$ bis $T_S + 871ms$ durchgeführt. In den meisten Fällen liefert allerdings die Sprache die primäre und auch verläßlichere Information. Die Zeigegeste liefert dazu eine zusätzliche Ortsspezifikation. Um eine Interpretation der vollständigen Äußerung nicht zu verhindern, sollte daher das fälschliche Erkennen einer Zeigegeste vermieden werden. Ungewöhnliche Zeigegesten werden daher zum Teil zurückgewiesen.

Nachdem nun der Zeitpunkt der Zeigegeste und damit auch der Ort der Hand bekannt sind, ist es noch nötig, die Zeigerichtung festzustellen. Hierzu bieten sich zwei Ansätze an. Einerseits ist es möglich, die Postur der Hand zum Zeigezeitpunkt zu analysieren und so beispielsweise die Richtung eines zeigenden Zeigefingers zu bestimmen. Dieser Ansatz wurde beispielsweise in [Nöl98] und [vH01] verfolgt. Eine weitere Möglichkeit ist die Herannahme der Trajektorie, um aus dieser auf die Richtung zu schließen. Dieser Ansatz wurde hier realisiert. Durch die schon vorhandene Segmentierung der Trajektorie ist dieser Ansatz hier deutlich effizienter. Die Robustheit des Ansatzes ließe sich durch eine zusätzliche Analyse der Postur unter Effizienzverlust

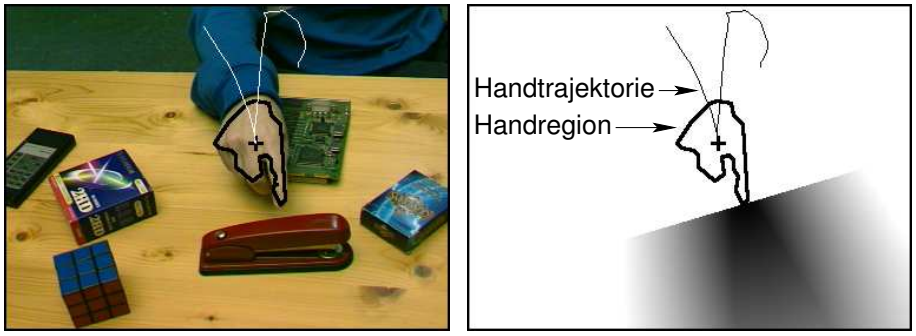


Abbildung 5.15: Beispiel einer ausgewerteten Zeigegeste. (a) Bild der aufgenommenen Szene zum ermittelten Zeitpunkt T_D der Zeigegeste. (b) Die aus der Geste generierte Aufmerksamkeitskarte.

steigern, indem diese als ein weiterer Indikator für das Vorhandensein einer Zeigegeste und zur Bestimmung der Zeigerichtung benutzt würde.

Die Zeigerichtung wird aus der Richtung einiger Punkte "aus der Vergangenheit" der Trajektorie bestimmt. Die Anzahl n der Punkte wird so gewählt, daß der Abstand des maximal in der Vergangenheit liegenden Punktes groß genug ist und n über einem unteren Schwellwert liegt. Damit ist sichergestellt, daß die Richtung sowohl durch die genügend große Anzahl der Punkte als auch deren Abstand voneinander sicher bestimmt werden kann. Zusätzlich muß n unter einem oberen Schwellwert liegen, damit die Trajektorie als eine mögliche Zeigegeste akzeptiert wird. Ist n zu groß, fand für eine Zeigegeste in der unmittelbaren Vergangenheit zu wenig Bewegung statt. Im vorgestellten System muß die Anzahl zwischen 7 und 30 liegen und der Abstand des letzten Punktes muß mindestens 50 Pixel zum Ausgangspunkt T_D betragen.

Durch die ausgewählten Punkte wird nun eine Gerade optimal gelegt. Da beliebig liegende 2D-Punkte vorliegen, kann ein Verfahren zum Berechnen einer Regressionsgeraden [Bro00, Kapitel 16.3] nicht benutzt werden. Dieses würde nur den Abstand zu einer Koordinate optimieren. Statt dessen kommt hier die Singulärwertzerlegung zum Einsatz [Bro00, Kapitel 4.5], mit deren Hilfe der quadratische Abstand aller Punkte zu der zu findenden Gerade minimiert wird. Eine (m, n) Matrix $\mathbf{A}$ kann mit Hilfe der Singulärwertzerlegung immer in 3 Matrizen zerlegt werden,

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T \tag{5.22}$$

wobei $\mathbf{U}$ und $\mathbf{V}$ orthogonale Matrizen sind, die die Eigenvektoren der Matrizen $\mathbf{A}\mathbf{A}^T$ und $\mathbf{A}^T\mathbf{A}$ enthalten und $\mathbf{S}$ eine Diagonalmatrix ist, die auf den ersten Diagonalstellen die Singulärwerte enthält und ansonsten Null ist. Für die Geradenberechnung ergibt

sich die Matrix **A** zu

$$\mathbf{A} = \begin{pmatrix} x_0 - \overline{x} & y_0 - \overline{y} \\ \dots & \dots \\ x_n - \overline{x} & y_n - \overline{y} \end{pmatrix} \tag{5.23}$$

mit

$$\overline{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \overline{y} = \frac{\sum_{i=1}^{n} y_i}{n} \tag{5.24}$$

wobei x_i und y_i die Koordinaten der ausgewählten Trajektorienpunkte sind. Der Spaltenvektor zum kleinsten Singulärwert von **A** aus der Matrix **V** gibt die Richtung der optimalen Gerade an [Gol90].

Um die vollständige Handposition mit in die Auswertung einfließen zu lassen, wird die exakte Zeigeposition entsprechend der maximalen Ausdehnung der Handregion in Zeigerichtung verschoben. Dies bewirkt beispielsweise eine deutliche Verschiebung bei ausgestrecktem Zeigefinger, wie es in der Zeigegeste in Abbildung 5.15 (a) vorkam. Abbildung 5.15 (b) zeigt die ermittelte Zeigerichtung und Position für diese Geste.

Die Auswertung bezüglich Position und Richtung der Geste ist immer mit einem kleinen Fehler versehen. Aber insbesondere zeigt ein Mensch auch nicht exakt auf die von ihm gewünschte Position. Auch kann sich das durch die Zeigegeste referenzierte Objekt unter Umständen in größerer Entfernung befinden – eine Zeigegeste gibt nur eine Richtung und keine Entfernung an. Um all dies zu berücksichtigen, ist eine probabilistische Integration der Richtung und der Position der Geste in das Gesamtsystem unabdingbar. Dies wird durch eine Aufmerksamkeitskarte [Rae00] realisiert, die für jede Position des Bildes eine Wahrscheinlichkeit dafür spezifiziert, daß der jeweilige Punkt im Fokus der Geste war.

In der Aufmerksamkeitskarte wird alles hinter dem ermittelten Zeigepunkt liegende als Zielpunkt der Geste ausgeschlossen. Dem Zeigepunkt näher liegende Punkte werden als wahrscheinlicher angenommen als weiter entfernte Punkte. Die Zeigerichtung wird in der Nähe des Zeigepunktes als genauer angesehen als in der Entfernung. Damit ergibt sich eine Aufmerksamkeitskarte wie sie in Abbildung 5.15 (b) für die Geste aus Abbildung 5.15 (a) dargestellt ist. Es ist ein Kegel, der um 10% der Bildbreite hinter die Zeigeposition verschoben ist. Die Ränder des Kegels sind unscharf, um die wahrscheinlichste Zeigerichtung zu bevorzugen, andere Richtungen aber nicht auszuschließen. Zusätzlich nimmt die Wahrscheinlichkeit linear mit der Entfernung zum Zeigepunkt ab. Die generierte Aufmerksamkeitskarte wird bei der Suche nach Objekten mit berücksichtigt. Das genaue Vorgehen hierfür ist in Kapitel 6 näher erläutert.

5.4.3 Handlungsinterpretation

Bei der Interaktion mit dem in Kapitel 3 vorgestellten System führt der Benutzer während des Lernens eines neuen Objektes eine Greifaktion zum Herausnehmen des Objektes aus der Szene durch. Das System muß das herausgenommene Objekt

sicher vom Hintergrund extrahieren, um so weitere Erkenntnisse über das Objekt gewinnen zu können. Um dies durch zusätzliche Informationen zu unterstützen, wird versucht, die Greifaktion zu detektieren und den Ort der Aktion zu bestimmen. Der Ort der Greifaktion wird dann als eine gute Hypothese für die Position des Objektes angesehen. Näheres zum Lernen eines Objektes wird in Abschnitt 3.3 dargestellt.

Der Benutzer wird während des Lernens aufgefordert, das zu lernende Objekt aus der Szene zu nehmen und dem System mitzuteilen, wenn er damit fertig ist. Damit ist für die Greifaktion ein klarer Kontext durch die Dialogsituation festgelegt. Die Systemaufforderung wird als frühster Start einer Greifaktion interpretiert, die Bestätigung des Benutzers als spätestes Ende. Während dieser Zeit wird die als Greifaktion am geeignetesten erscheinende Trajektorie K_G mit Hilfe der auf Seite 67 eingeführten Bewertungen J_D und J_S bestimmt.

Die Auswahl der für einen Zeitschritt besten Trajektorie $K^*(t)$ aus der Menge der für diesen Zeitschritt bestimmten Trajektorien $K(t)$ geschieht hier analog zu dem in Gleichung 5.20 angegebenen Verfahren für die Auswahl bei Zeigegesten – auch die Greifaktion muß mit einer stabilen Trajektorie, die Dynamik aufweist, durchgeführt werden. Die weitere Auswahl über mehrere Zeitschritte hinaus unterscheidet sich allerdings. Eine günstige Trajektorie einer Zeigegeste liegt nahe dem Ende der zugehörigen sprachlichen Anweisung. Bei der Greifaktion gibt es diese unmittelbare Kopplung mit einer sprachlichen Anweisung hingegen nicht. Daher wird die Trajektorie $K^*(t)$ nur dann als Kandidat K_G für eine Greifaktion übernommen, falls ihre Bewertung besser oder zumindest gleich gut als die bisherige ist:

$$K_G \leftarrow K^*(t) : J_S(K^*) \geq J_S(K_G) \wedge [J_S(K_G) < 1 \vee J_D(K^*) > 1.2] \tag{5.25}$$

Abbildung 5.16 zeigt wichtige Werte für die weitere Analyse der Trajektorie K_G. Als Greifpunkt der Trajektorie wird derjenige Punkt der Trajektorie K_G angesehen, der die größte Entfernung zum Anfangspunkt $G^{(S)}$ und zum Endpunkt $G^{(E)}$ der Trajektorie hat. Der endgültige Greifpunkt P ergibt sich aus diesem durch Verschiebung entsprechend der maximalen Ausdehnung der Handregion zum Zeitpunkt der Greifaktion. Die Entscheidung über das Auftreten einer Greifaktion wird an Hand der Trajektorie gefällt. Die Länge der Trajektorie muß über einem Schwellwert liegen, der Punkt P muß eine Mindestentfernung zu den Punkten $G^{(S)}$ und $G^{(E)}$ haben und der Winkel zwischen den Linien $|G^{(S)}P|$ und $|G^{(E)}P|$ muß genügend klein sein. Durch diese Bedingungen werden einige potentielle Greifaktionen ausgeschlossen. Beispielsweise kann es durchaus sein, daß der Benutzer von links in die Szene langt, das Objekt nimmt und nach rechts die Szene verläßt, was zu einem großen Winkel in der Trajektorie führen würde. Dies wäre allerdings ein eher ungewöhnlicher Verlauf der Aktion – meistens wird die Hand zum Ausgangspunkt der Aktion zurückkehren. Da die Analyse der Greifaktion nur zusätzliche Informationen liefert, sollte die Rate der falsch positiven Fälle bei der Erkennung möglichst gering sein. Somit werden ungewöhnliche Greifaktionen eher ausgeschlossen.

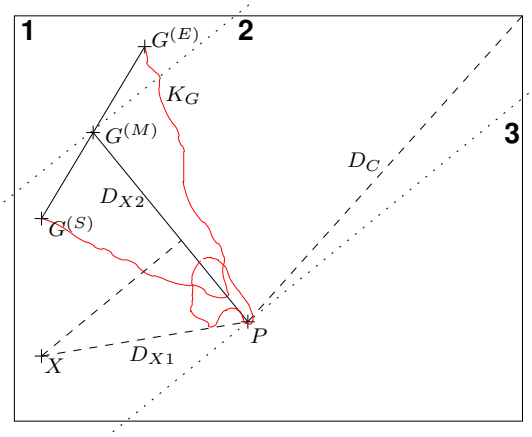
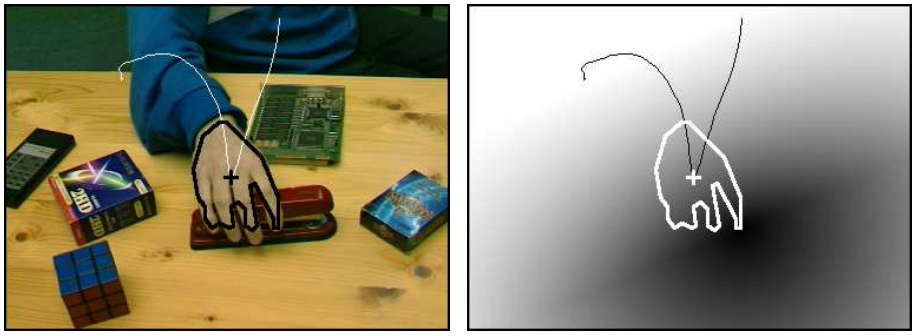


Abbildung 5.16: Berechnung der Aufmerksamkeitskarte für eine Greifaktion. Der Benutzer hat die Greifaktion bei $G^{(S)}$ gestartet, bei P ein Objekt gegriffen und bei $G^{(E)}$ die Aktion beendet.

Entsprechend der Auswertung von Zeigegesten muß auch bei der Greifaktion eine probabilistische Integration in das Gesamtsystem durchgeführt werden. Der Benutzer kann ein Objekt an verschiedensten Stellen greifen, was zu Variationen im exakten Greifpunkt in Relation zum Objekt führt. Und auch die Segmentierung der Hand und damit die exakte Form und Lage der Trajektorie ist wieder mit einem Fehler behaftet. Eine Integration mit Hilfe einer Aufmerksamkeitskarte berücksichtigt dies.

Die Berechnung der Karte teilt sich in drei Bereiche auf, die in Abbildung 5.16 durch gepunktete Linien getrennt dargestellt sind. Bereich 1 liegt hinter dem Start der Trajektorie und wird daher als möglicher Bereich für die Objektlage ausgeschlossen. Die wahrscheinlichste Art, ein relativ kleines Objekt aus der Szene zu nehmen, ist das Greifen des Objektes an der dem Benutzer zugewandten Seite. Das Objekt liegt damit eher vom Benutzer aus gesehen hinter dem Zeigepunkt P und damit im Bereich 3 als vor dem Punkt. Der Punkt P wiederum liegt mit hoher Wahrscheinlichkeit auf dem Objekt. Daher bekommt dieser Punkt die höchste Wahrscheinlichkeit. Im Bereich 3 nimmt die Wahrscheinlichkeit linear mit der Entfernung D_{X1} zum jeweils zu bewertenden Punk X ab. Skaliert wird sie mit D_C, dem maximalen Abstand zu einem der vier Eckpunkte des Bildes. Das sich das Objekt im Bereich 2 befindet ist etwas unwahrscheinlicher. Daher wird der durch D_{X1} und D_C bestimmte radiale Verlauf um P in diesem Bereich zusätzlich durch Multiplikation mit einem in Richtung $G^{(M)}$ abnehmenden linearen Verlauf zwischen P und $G^{(M)}$ erniedrigt. Abbildung 5.17 (b) zeigt eine Aufmerksamkeitskarte, die auf Grund der in Abbildung 5.17 (a) dargestellten Greifaktion generiert wurde.



(a) (b)

Abbildung 5.17: Beispiel einer ausgewerteten Greifaktion. (a) Szene zum Zeitpunkt der Greifaktion. (b) Die aus der Greifaktion generierte Aufmerksamkeitskarte.

5.5 Ergebnisse

Das Ziel des in diesem Kapitel vorgestellten Teilsystems war die Segmentierung und Verfolgung von Händen und die kontextabhängige Auswertung der von den Händen durchgeführten Gesten. Dabei sollte Echtzeitfähigkeit bei trotzdem guter Erkennungsrate erzielt werden. Daher wird in diesem Abschnitt nun sowohl die Geschwindigkeit des Systems als auch dessen Güte ausgewertet. Für die Auswertung der Gesteninterpretation ist jeweils das komplette Gesamtsystem nötig, da hier der durch die Sprache und den Dialog gelieferte Kontext maßgebend ist. Eine genauere Betrachtung der Performanz der Gesteninterpretation wird daher in Kapitel 8 vorgenommen.

5.5.1 Ergebnisse der Farbklassifikation

Für die Auswertung der beiden eingesetzten Farbklassifikatoren, dem Polynomklassifikator und dem gaußbasierten Klassifikator, wurden zwei Versuche durchgeführt. Bei Versuch eins wurden 31 Bilder der Hand einer Person aufgenommen und manuell segmentiert. Für das Training des Polynomklassifikators wurden zusätzlich 22 Bilder für die Rückweisungsklasse – den Hintergrund – aufgenommen. Diese enthielten keine Haut, aber teilweise hautfarbene Bereiche wie beispielsweise Holzteile oder orange Objekte. Abbildung 5.18 (a) zeigt neben einem Beispielbild einer Hand ein Beispiel für ein Hintergrundbild. Die beiden Klassifikatoren benutzen als Merkmale jeweils einzelne Pixel. Insgesamt ergaben sich damit ca. 2.65 Millionen Merkmalsvektoren für die Hand und ca. 5.02 Millionen Merkmalsvektoren für den Hintergrund. Bei Versuch zwei wurden Bilder von Händen von fünf verschiedenen Personen aufgenommen und manuell segmentiert. Abbildung 5.18 (b) zeigt zwei der insgesamt 30 Bilder. Hieraus

ergaben sich ca. 2.6 Millionen Merkmalsvektoren[1]. Für die Rückweisungsklasse wurden die Bilder aus Versuch eins übernommen.

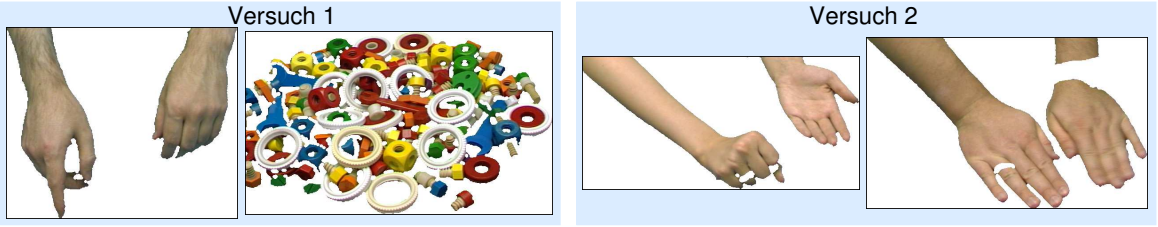


Abbildung 5.18: Einige Beispielbilder, die für das Training der Farbklassifikatoren für die Versuche 1 und 2 eingesetzt wurden.

Um eine optimale Ausnutzung des relativ geringen Bildmaterials zu gewährleisten, wurde die Auswertung mit Hilfe des *leave-one-out* Testes [Lun67] durchgeführt. Diese spezielle Variante der Kreuzvalidierung [Efr83] teilt die n Bilder in eine Trainingsstichprobe der Größe $n-1$ und eine Teststichprobe der Größe 1 ein. Die Klassifikationsrate wird nun gemittelt über alle n möglichen Arten, so eine Partitionierung zu erhalten, bestimmt. Zusätzlich wurde nach der Glättung der klassifizierten Bilder mit einem Medianfilter mit Radius fünf eine zweite Klassifikationsrate für diese Bilder ermittelt.

Beide Klassifikatoren erlauben durch Anpassen eines Schwellwertes die Entscheidung, ab wann ein Pixel als Hautfarbe klassifiziert werden soll. Der Schwellwert wurde für jede Partitionierung jeweils automatisch so bestimmt, daß auf der Trainingsstichprobe eine vorgegebene Klassifikationsrate erreicht wurde. Um auch den Einfluß dieses Parameters zu bestimmen, wurden die Versuche mit vorgegebenen Klassifikationsraten von 53% bis 99% durchgeführt. Für jede vorgegebene Klassifikationsrate wurde somit ein vollständiger leave-one-out Test mit beiden Klassifikatoren jeweils mit und ohne Medianfilter durchgeführt. Abbildung 5.19 zeigt die ermittelten Ergebnisse für beide Versuche. Aus den hier dargestellten Graphen lassen sich auch *Receiver-Operating-Characteristic* Graphen (ROC-Graphen [vT68, Abschnitt 2.2.2]) erstellen. Die Klassifikationsrate für den Hintergrund K_H ist bezüglich des Testes der Handsegmentierung die Rate der korrekt negativ klassifizierten Merkmale. Die Rate der falsch negativ klassifizierten Merkmale ergibt sich damit durch $100 - K_H$. Trägt man dies gegen die Klassifikationsrate für die Hand auf, erhält man einen ROC-Graphen.

Beide Versuche haben konsistente Ergebnisse gebracht. Wie erwartet folgt durch stattfindende Übersegmentierung aus einer steigenden Klassifikationsrate für die Hand eine sinkende Rate für den Hintergrund. Der Polynomklassifikator erreicht durch seine größere Anzahl freier Parameter ein durchgehend besseres Ergebnis. Ab einer vorgegebenen Klassifikationsrate von 98% für die Hand hat der gaußbasierte Klassifikator die

[1] Genau waren es 2653195, 5024326 bzw. 2559392 Merkmalsvektoren.

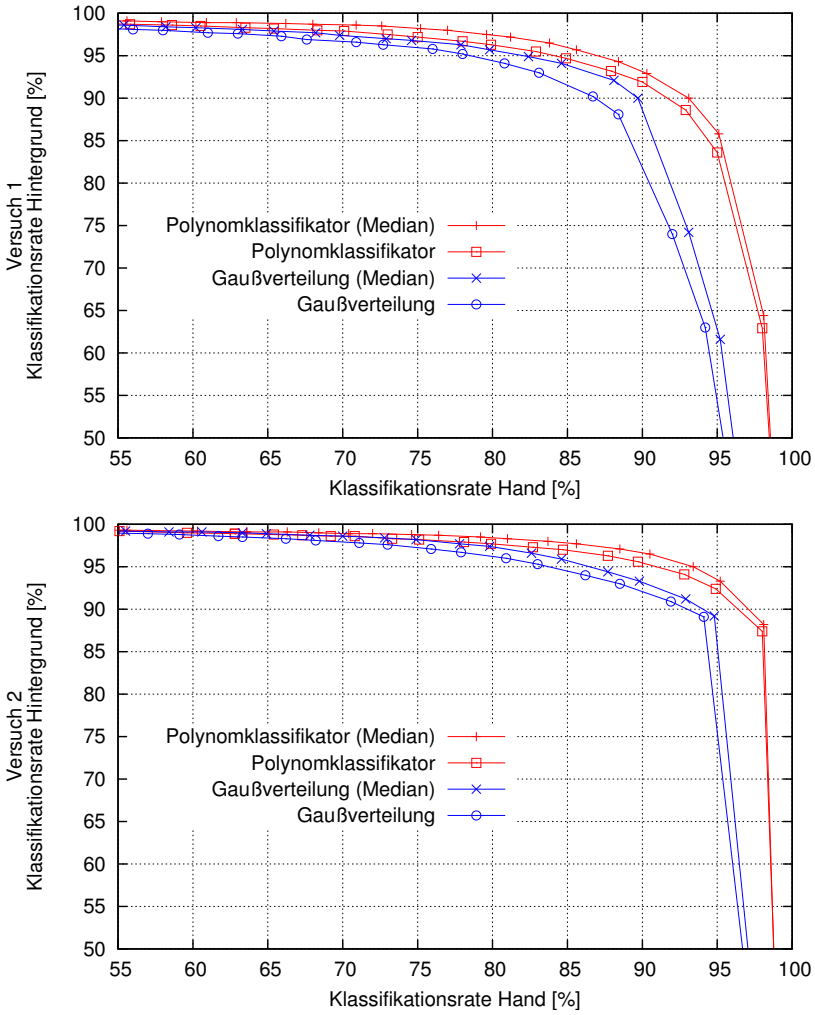


Abbildung 5.19: Ergebnisse der pixelbasierten Farbklassifikation. Dargestellt ist die erzielte Klassifikationsrate für den Hintergrund in Abhängigkeit der Klassifikationsrate für Hände auf der Teststichprobe bei vorgegebener Klassifikationsrate für Hände auf der Trainingsstichprobe.

vollständigen Hintergrundbilder als hautfarben klassifiziert. Der Polynomklassifikator konnte hier noch einige Merkmale korrekt klassifizieren. Durch mit Hilfe des Medianfilters unterdrücktem Rauschen steigt die Klassifikationsrate bei beiden Klassifikatoren. Alle diese Resultate sind jeweils statistisch signifikant. Insgesamt konnte sowohl der Polynomklassifikator als auch der gaußbasierte Klassifikator eine gute Klassifikationsleistung erzielen. Steht korrekt klassifiziertes Trainingsmaterial in genügend großer Menge zur Verfügung, liefert der Polynomklassifikator bessere Ergebnisse.

5.5.2 Ergebnisse der Handverfolgung

Neben der Farbklassifikation besteht die Gestenerkennung aus weiteren 3 Komponenten, in denen die Bewegung eine wichtige oder die zentrale Bedeutung hat. Um eine vollständige quantitative Auswertung dieser bewegungsbasierten Komponenten der Gestenerkennung durchzuführen, ist eine sehr große Stichprobe von Filmsequenzen nötig. Ansonsten ist keine sinnvolle Aussage über die ermittelten Ergebnisse möglich. Die Sequenzen müssen jeweils für jeden Frame manuell ausgewertet werden. Dies ist innerhalb einer Dissertation bei weitem nicht durchführbar. Daher findet neben der quantitativen Auswertung aus Sicht des Gesamtsystems im Kapitel 8 eine qualitative Auswertung an Hand einiger Beispielsequenzen statt.

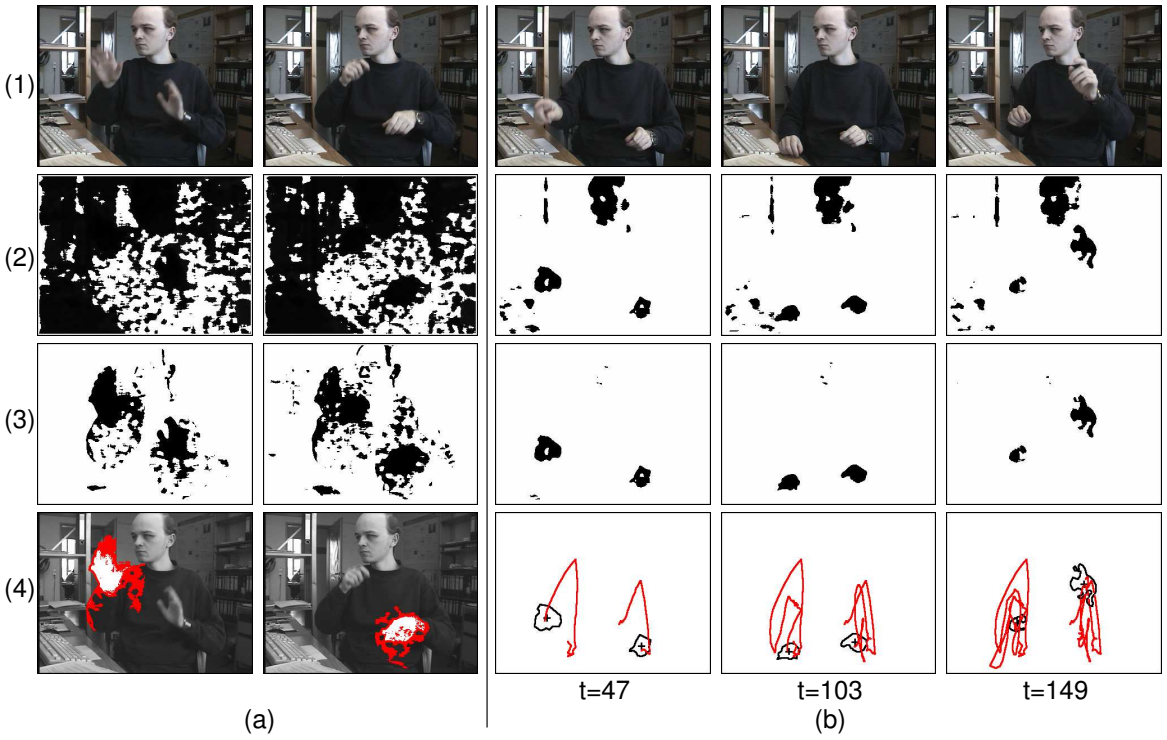


Abbildung 5.20: Einige Ergebnisbilder für die Hautfarbinitialisierung. (a) Interaktive Initialisierug des gaußbasierten Farbklassifikators. (b) Segmentierungsergebnisse aus einer Bildsequenz. (1) Originalbilder. (2) Farbsegmentierung. (3) Kombiniertes Farb-/Differenzbild bzw. Farbbild/Aktivitätskarte. (4) Zum Training benutzte Regionen / Ermittelte Handtrajektorien.

Einige Ergebnisse wurden schon in den Abbildungen 5.5 auf Seite 59, 5.8 auf Seite 64 und 5.9 auf Seiten 66 für die bewegungsbasierte Bewertung und Initialisierung präsentiert. Abbildung 5.20 zeigt weitere Ergebnisse aus einer Büroumgebung bei der

Initialisierung und nachfolgenden Benutzung des gaußbasierten Klassifikators. Teil (a) zeigt Beispiele aus der Initialisierungsphase. Durch Benutzung von Bewegungsinformation kann die initial schlechte Farbsegmentierung bei der Modellgenerierung stark verbessert werden (Zeile (3)). Ein Großteil der restlichen fehlerhaft als Hautfarbtrainingsmaterial ausgewählten Pixel kann durch Benutzung des inkrementell erstellten Hintergrundbildes ausgefiltert werden. In Zeile (4) sind in Weiß 2 finale Beispielregionen für ausgewähltes Trainingsmaterial dargestellt. In Rot sind die mit Hilfe des Hintergrundbildes ausgefilterten Pixel dargestellt.

Teil (b) von Abbildung 5.20 zeigt Ergebnisse der Segmentierung und Regionenverfolgung mit dem in Teil (a) generierten Klassifikator an Hand von drei Frames aus einer längeren Sequenz. Neben Hautfarbe werden durch den Farbklassifikator noch wenige holzfarbene und graue Teile segmentiert (Zeile (2)). Durch den Einsatz der bewegungsbasierten Aktivitätskarte können diese sich nicht bewegenden Teile ausgefiltert werden (Zeile (3)). Auch das sich nicht bewegende Gesicht wird hiermit ausgefiltert, womit nur die für die Gesten wichtigen Hände übrig bleiben. Diese können über einen längeren Zeitraum korrekt mit Hilfe eines Kalmanfilters verfolgt werden (Zeile (4)).

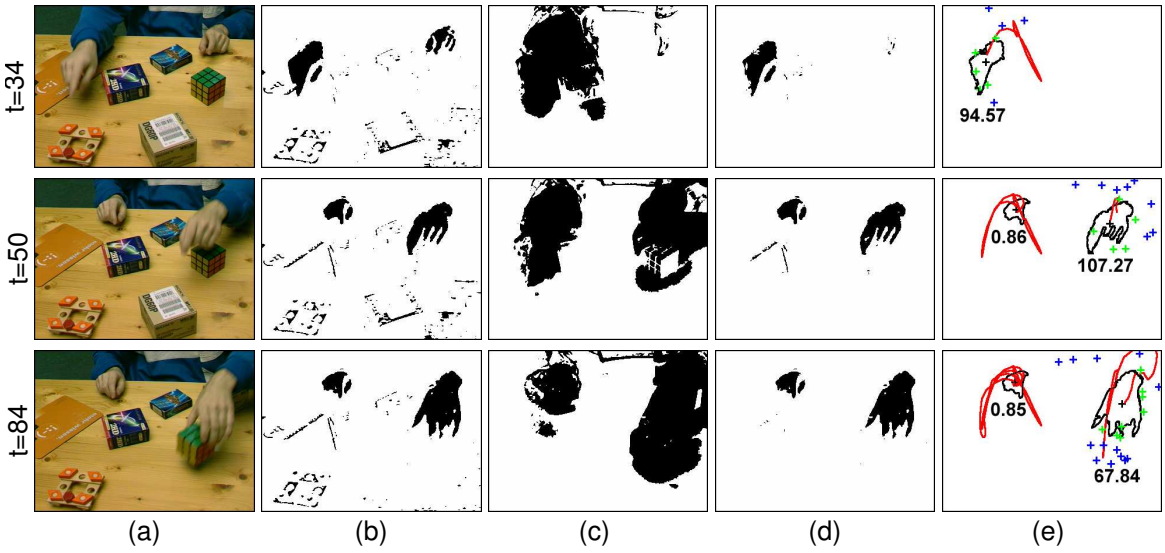


Abbildung 5.21: Einige Ergebnisbilder der Handverfolgung bei Einsatz des Polynomklassifikators. (a) Originalbilder. (b) Farbsegmentierung. (c) Binarisierte Aktivitätskarte. (d) Kombiniertes Farbbild/Aktivitätskarte. (e) Segmentierte Regionen und bisher ermittelte Handtrajektorien.

Abbildung 5.21 zeigt Ergebnisse unter Einsatz des Polynomklassifikators. Für die Hautfarbklassifikation ist dies auf Grund von Holz, Pappe und verschiedenen orangen Teilen eine sehr schwierige Szene. Trotzdem ist der mit einem großen Parametersatz ausgestattete Polynomklassifikator in der Lage, die Szene zu einem großen Teil kor-

rekt zu klassifizieren (Spalte (b)). Durch den Einsatz der Aktivitätskarte können die meisten Segmentierungsfehler behoben werden (Spalten (c) und (d)). Basierend auf dieser Segmentierung können die Hände nun sicher über eine längere Zeit verfolgt werden (Spalte (e)). In dieser Spalte sind in Rot die ermittelten Trajektorien und in Schwarz die Handregionen dargestellt. Die Bewertung der Regionen ist mit angegeben. Durch die Integration von Differenzregionen in die Bewertung der Handregionen kann die jeweils aktive Hand identifiziert werden. Die hierfür eingesetzten Schwerpunkte der Differenzregionen sind als grüne und blaue Kreuze in der Spalte (e) angegeben. Grün sind dabei diejenigen Regionen dargestellt, die einer Handhypothese zugeordnet werden konnten.

5.5.3 Geschwindigkeit der Gestenerkennung

Rechner	CPU		Speicher	SPEC CPU95		SPEC CPU2000	
	Typ	[MHz]	[MB]	CINT95	CFP95	CINT2000	CFP2000
Alpha	Alpha 21164A	433	320	12.1	16.9	?	?
PC	Pentium® 4	2400	1024	?	?	865	872
$Alpha_{Ref}$	Alpha 21164A	500	384	13.7	18.0	161	158

Tabelle 5.1: Technische Daten der bei den Geschwindigkeitstests eingesetzten Rechner. ?: Nicht bekannte Daten.

Eine wichtige Anforderung an das System war die Echtzeitfähigkeit, um so eine schnelle Systemreaktion auf Benutzeraktionen zu erreichen und Speicher durch das Vermeiden des Zwischenspeicherns von Bildsequenzen zu sparen. Um dies zu verifizieren, wurde die Geschwindigkeit auf den zwei in Tabelle 5.1 aufgeführten Rechnern *Alpha* und *PC* getestet. Die Geschwindigkeitswerte SPEC CPU95 und SPEC CPU2000[2] beziehen sich auf das jeweilige Modell. Es sind Herstellerangaben, die auf Referenzmodellen mit zum Teil anderer Ausstattung (beispielsweise anderer Speicherausstattung und insbesondere mit anderen Compilern ausgestattet) ermittelt wurden und stimmen daher unter Umständen nicht exakt mit den Werten der eingesetzten Rechner überein. Der Rechner $Alpha_{Ref}$ ist zusätzlich angegeben, um einen groben Vergleich der SPEC CPU95 und SPEC CPU2000-Werte zu erlauben. An Compilern kamen bei der Alpha der DEC C Version 5.9 und bei dem PC der GCC Version 3.3 [GCC03] zum Einsatz.

Zur Ermittlung der Geschwindigkeit wurde die auch in Abbildung 5.21 zum Einsatz gekommene Bildsequenz verwendet. Bei einer Auflösung der Bilder von 189x139 Pixeln benötigte die Alpha für die Verarbeitung eines kompletten Bildes 35 ms, der PC 11 ms. Für das Erkennen der Gesten ist diese Auflösung genügend groß und die

[2] siehe beispielsweise http://www.specbench.org/osg/

Geschwindigkeit genügend hoch, so daß problemlos in Echtzeit ohne Zwischenspeicherung der Bilder gearbeitet werden kann. Erhöht man die Auflösung der Bilder, sinkt die Geschwindigkeit mit Ausnahme des Grabbens der Bilder und der Regionenverfolgung nahezu linear mit der Anzahl der Pixel. Bilder einer Auflösung von 756x556 Pixeln werden beispielsweise von der Alpha in 535 ms und vom PC in 93 ms verarbeitet.

Tabelle 5.2 gibt die ermittelten Werte aufgeteilt nach den Einzelkomponenten des Systems wieder. Die "Bewegungsdetektion" umfaßt dabei die Differenzbildbestimmung und die Berechnung der Aktivitätskarte. Bei der "Farbklassifikation" ist zusätzlich die Integration der Aktivitätskarte enthalten. Die "Regionenberechnung" bestimmt Polygonzüge für die hautfarbenen Bereiche. Der Teil "Differenzregionen" umfaßt die Berechnung der Bewertung auf Grund von Bewegung. Bei der "Regionenverfolgung" werden schließlich die Hautfarbregionen durch Kalmanfilter verfolgt. Soll das System auf einem langsameren Rechner zum Einsatz kommen, kann wahlweise auf den Einsatz der Aktivitätskarte oder der bewegungsbasierten Bewertung verzichtet werden, womit die Geschwindigkeit bei Verschlechterung der Ergebnisse noch einmal gesteigert werden kann.

	Alpha [ms]		PC [ms]	
Bildauflösung	189x139	756x556	189x139	756x556
Bewegungsdetektion	6	105	1	17
Farbklassifikation	11	184	2	29
Regionenberechnung	7	124	1	22
Differenzregionen	4	75	0.7	10
Regionenverfolgung	<1	<1	0.2	0.2
Komplettes Bild	35	535	11	93

Tabelle 5.2: Geschwindigkeit der Handverfolgung für verschiedene Bildgrößen. Die Zeile "Komplettes Bild" gibt die benötigte Zeit für ein Bild im ms an. Darüber sind die benötigten Zeiten für Teilkomponenten des Systems angegeben.

5.6 Zusammenfassung

In diesem Kapitel wurde ein hierarchisches System zum Erkennen von Gesten vorgestellt. Durch die Hinzunahme weiterer Informationen und der fortlaufenden Einschränkung der Domäne auf jeder Hierarchiestufe können schließlich die gewünschten Gesten, in diesem Fall Zeigegesten und Greifaktionen, sicher erkannt werden. Der Einsatz eines Teils des Systems in anderen Domänen ist durch diesen hierarchischen Ansatz leicht möglich.

Initial wird jedes Bild auf Grund von Farbe klassifiziert. Um flexibel auf die Beleuchtungssituation reagieren zu können, stehen zwei Farbklassifikatoren zur Verfügung, ein Klassifikator mit einem großen Parametersatz und ein Klassifikator, der intuitiv neu trainiert werden kann und sich damit leicht an sich ändernde Beleuchtungssituationen anpassen läßt. Eingeschränkt und bewertet wird dieses farbbasierte Ergebnis durch Hinzunahme der Modalität Bewegung. Hiernach liegen vollständig segmentierte hautfarbene Regionen vor, die mit Hilfe eines Kalmanfilters über die Zeit getrackt werden. Unter Zuhilfenahme der Sprache und der Dialogsituation werden aus den Handtrajektorien und den dazugehörigen Handregionen schließlich alle nötigen Informationen zur probabilistischen Integration von Gesten extrahiert.

Verschiedene präsentierte Ergebnisse haben die Adäquatheit des vorgeschlagenen Ansatzes gezeigt. Auf der Ebene der Pixel kann eine gute farbbasierte Klassifikationsrate erreicht werden. Qualitative Ergebnisse haben den Nutzen der Bewegungsintegration und die Güte der Initialisierung des Farbklassifikators in verschiedenen Umgebungen gezeigt.

6 Visuelle Merkmalsextraktion

Für die Verankerung von Objekten in der Welt ist eine Verarbeitung sensorischer Eingabedaten nötig. Der Mensch setzt hierfür verschiedenste Sinne ein. Beispielsweise können wir uns durch den Geruch an eine Rose erinnern oder durch das Geräusch an eine Harfe. Das zentrale Sinnesorgan des Menschen ist hierbei aber sicherlich das Auge. Über das Auge nimmt der Mensch visuell Objekte war, erkennt die Merkmale des Objektes und ist in der Lage, Objekte in einer Szene wiederzufinden und zu verfolgen.

Der Einsatz einer Farbkamera ist daher auch für ein maschinelles System eine naheliegende Entscheidung. Einerseits hat das erfolgreiche menschliche Auge gezeigt, daß dies ein sinnvoller Sensor ist. Andererseits erlaubt aber auch der Einsatz eines vergleichbaren Sensors das leichtere Kommunizieren über die Sensordaten. Dies ist gerade bei dem in dieser Arbeit angestrebten System, welches in hohem Masse auf Interaktion angelegt ist, eine sinnvolle Eigenschaft. Daher kommt eine Farbkamera zur visuellen Erfassung der Szene zum Einsatz. Durch die Beschränkung auf eine Kamera bei der Objekterkennung wird der Hardwareaufwand klein gehalten.

In den folgenden Abschnitten wird die Verarbeitung der Sensordaten bis zu der Generierung von Objekthypothesen vorgestellt. Dazu wird im ersten Abschnitt ein Überblick über die visuelle Objekterkennung und deren Realisierung in der Literatur gegeben. Abschnitt 6.2 stellt anschließend einen Ansatz zur Generierung von Objekthypothesen basierend auf Histogrammen vor. Die mit diesem Ansatz erzielten Ergebnisse werden in Abschnitt 6.3 präsentiert. Abgeschlossen wird das Kapitel durch einen weiteren parallel einsetzbaren Ansatz zur Generierung von Objekthypothesen basierend auf dem Vergleich von Graphen. Zusätzlich werden hier die Ergebnisse dieses Ansatzes präsentiert.

6.1 Objekterkennung – Ein Überblick

Der Mensch ist in der Lage, Objekte sicher in einer visuell dargebotenen Szene zu finden und zu verfolgen. Dies kann er selbst bei großen Verdeckungen, schlechter Sicht oder vielen Störobjekten. Um dies auch nur ansatzweise bei einem maschinellen System zu erreichen, ergeben sich verschiedene vorher zu lösende Fragen, die [Dic99] wie folgt formuliert:

> "How is visual knowledge of an object encoded? What information is recovered from an image in order to recognize an object? And how is this information compared to our stored knowledge in order to recognize the object?"

Diese Fragen spiegeln sich sowohl in der allgemeinen Architektur von Objekterkennungssystemen als auch in konkreten Realisierungen wieder, die in den nächsten Abschnitten jeweils näher vorgestellt werden.

6.1.1 Struktur eines Objekterkennungssystems

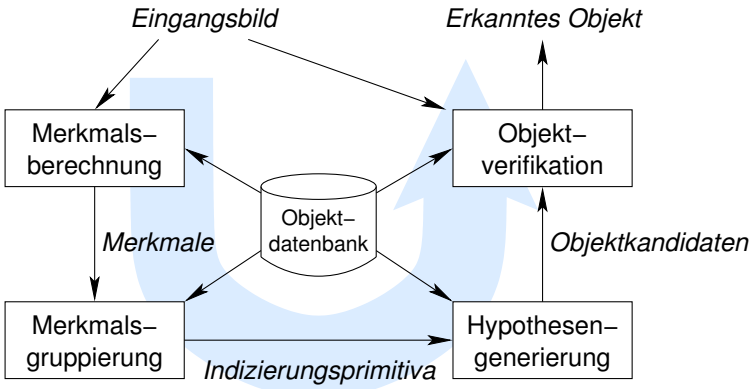


Abbildung 6.1: Typischer Aufbau eines Systems zur Erkennung von Objekten nach [Dic99].

Ein typischer Aufbau eines Objekterkennungssystems, in ähnlicher Form vorgestellt in [Dic99], ist in Abbildung 6.1 zu sehen. Die Eingabe in ein allgemeines Objekterkennungssystem sind beliebige Sensordaten, die zum Teil durch eine Vorverarbeitung modifiziert wurden. Da der Sensor im vorgestellten System auf eine Farbkamera festgelegt ist, sind die Eingangsdaten hier Bilder. Das Ziel des Systems ist das Erkennen und gegebenenfalls Lokalisieren von Objekten innerhalb des Eingangsbildes. Für die zu erkennenden Objekte gibt es jeweils Modelle in der Objektdatenbank. Auf das Wissen in der Datenbank kann in jedem Schritt der Verarbeitung zugegriffen werden. In einem ersten Schritt werden Merkmale aus dem Bild extrahiert. Das Ziel hierbei ist, die Menge der Eingangsdaten zu reduzieren, Merkmale, die ein Objekt beschreiben nach einem Abstandsmaß näher zu gruppieren und Merkmale, die zu verschiedenen Objekten gehören nach dem gleichen Abstandsmaß weiter voneinander zu entfernen. Häufig ist dabei die ideale Merkmalswahl abhängig vom zu erkennenden Objekt.

In vielen Fällen definieren einzelne Merkmale noch keine ganzen Objekte beziehungsweise, falls Teilobjekte identifiziert werden sollen, noch keine wesentlichen Ob-

jektteile. Beispielsweise bestehen Objekte meistens aus mehreren Farbregionen oder mehreren Kanten. In diesem Fall müssen die Merkmale in einem zweiten Schritt gruppiert werden, um geeignete Anfragen für eine Datenbanksuche zu erhalten. Die Suche ist die Aufgabe des dritten Schritts, der Hypothesengenerierung. Basierend auf einer Vergleichsfunktion werden die gruppierten Merkmale mit den Einträgen in der Datenbank verglichen und eine Menge von Objekthypothesen wird generiert. In einem finalen Schritt wird schließlich eine Hypothese aus den generierten Hypothesen ausgewählt und als erkanntes Objekt ausgegeben. Um hierbei gegebenenfalls weitere Merkmale mit einbeziehen zu können, kann dieser Verifikationsschritt auf das Eingangsbild zugreifen.

6.1.2 Invarianz von Merkmalen

Formal gesehen ist ein Merkmal f eine Abbildung von einer Menge S in eine gegebenenfalls andere Menge T, $f : S \to T$. Eine für die Objekterkennung sehr interessante Eigenschaft eines Merkmals ist die Unabhängigkeit des Merkmals von verschiedenen Transformationen des zu erkennenden Objektes. Es kann beispielsweise unter Lage- oder Rotationsänderungen invariant sein. Für die formale Definition der Invarianz von Merkmalen ist die Definition der Transformationsgruppe nötig:

> **Definition Transformationsgruppe:** Sei G eine Gruppe von Transformationen auf einer Menge S, $g \in G$, $g : S \to S$. Falls G unter der Komposition abgeschlossen ist, d.h. $g_1 \in G \wedge g_2 \in G \Rightarrow (g_1 \circ g_2) \in G$, so ist G eine Transformationsgruppe.

Ein Beispiel für eine sehr bekannte Transformationsgruppe ist die Gruppe der affinen Abbildungen:

$$g \in G, \mathbf{s} \in S \qquad g(\mathbf{s}) = \mathbf{A}\mathbf{s} + \mathbf{t} \tag{6.1}$$

Diese Gruppe erlaubt die Translation, Rotation, Skalierung und Scherung von Vektoren. $\mathbf{t}$ gibt dabei die Translation an, $\mathbf{A}$ ermöglicht die restlichen Abbildungsvarianten.

Die Invarianz einer Abbildung läßt sich nun mit Hilfe der Transformationsgruppe definieren:

> **Definition Invarianz:** Sei G eine Transformationsgruppe auf einer Menge S, T eine Menge. Eine Abbildung $f : S \to T$ ist invariant bezüglich G, falls für alle $g \in G$, $s \in S$ gilt: $f(g(s)) = f(s)$.

Ein Merkmal f ist damit beispielsweise invariant bezüglich affinen Abbildungen, falls es für alle $\mathbf{s}$, $\mathbf{A}$ und $\mathbf{t}$ die Eingabe $\mathbf{s}$ und die Ausgabe $\mathbf{A}\mathbf{s} + \mathbf{t}$ auf den gleichen Wert abbildet, das heißt falls immer $f(\mathbf{s}) = f(\mathbf{A}\mathbf{s} + \mathbf{t})$ gilt.

Eine für die Objekterkennung wünschenswerte Eigenschaft ist die Vollständigkeit einer Abbildung:

Definition Vollständigkeit: Sei G eine Transformationsgruppe auf einer Menge S, T eine Menge, $f : S \to T$ eine invariante Abbildung. Dann ist f vollständig, falls für alle $s_1 \in S$, $s_2 \in S$ gilt:

$$f(s_1) = f(s_2) \Rightarrow \exists g \in G : s_2 = g(s_1).$$

Durch die Vollständigkeit werden Mehrdeutigkeiten bei der Zuordnung von Merkmalen ausgeschlossen, was die eindeutige Klassifikation der Merkmale ermöglicht. In den meisten Fällen wird dies nicht für alle Elemente der Menge S erreichbar sein. Eine möglichst große Teilmenge ist in diesen Fällen wünschenswert. Weitere Details zur Invarianz in der Objekterkennung finden sich beispielsweise in [Mun92].

6.1.3 Objekterkennung in der Literatur

Die automatische Erkennung von Objekten hat schon eine lange Tradition. Erste Systeme reichen bis in die 50er Jahre des letzten Jahrhunderts zurück [Gri58] und auch umfangreiche Übersichtsartikel gab es schon in den 60er Jahren [Nag68]. Seit dem hat das Interesse der Forschung an der Objekterkennung nicht nachgelassen. Damit kann hier nur ein sehr kleiner Überblick über die Objekterkennung gegeben werden.

Im Laufe der Zeit sind zwei verschiedene Ansätze zur Objekterkennung entwickelt worden, die ansichtenbasierte Vorgehensweise und die modellbasierte Vorgehensweise [Mun96, Dic99, Cam01]. Bei der ansichtenbasierten Objekterkennung wird jedes Objekt durch eine Menge von einzelnen Ansichten repräsentiert, die das Objekt aus verschiedenen Richtungen, in verschiedenen Skalierungen oder auch unter verschiedenen Beleuchtungen zeigt. Damit reduziert sich die Erkennung eines 3D-Objektes auf das Finden einer 2D-Ansicht eines Objektes im Bild. Zum Teil wird auch die Skalierungs- und Beleuchtungsinvarianz auf diese Weise gelöst. Die modellbasierte Objekterkennung repräsentiert ein Objekt durch ein einzelnes oder einige wenige abstrakte 3D-Modelle. In diesem Fall ist die Darstellung des Objektes in der Objektdatenbank wesentlich kompakter als bei der ansichtenbasierten Vorgehensweise. Der Vergleich des 2D-Eingangsbildes mit dem abstrakten Modell wird allerdings deutlich komplexer, da nun aus dem 2D-Bild Merkmale extrahiert werden müssen, die einen Vergleich mit dem 3D-Modell erlauben.

Insbesondere in den 80er Jahren war der modellbasierte Ansatz sehr verbreitet. Ein relativ spätes System aus dieser Zeit wird beispielsweise von [Hut90] vorgestellt. Es erkennt vielflächige feste Objekte und bestimmt deren Position und Ausrichtung. Bei der Erkennung wird ausgenutzt, daß sich im Objekt schneidende Kanten auch in nahezu jeder beliebigen 2D-Projektion des Objektes auf ein Bild schneiden. Diese Punkte werden im 2D-Bild durch Kantenextraktion bestimmt und mit den Punkten aus einer Projektion eines Modells des Objektes verglichen.

Ein einfacher Ansatz zur ansichtenbasierten Objekterkennung ist Template Matching, wobei ein vorhandenes Muster eines Objektes direkt mit dem Eingangsbild verglichen wird. Für den Vergleich wird häufig die Kreuz-Korrelation eingesetzt [Gon02,

Kapitel 4.6.4]. Um größere Unabhängigkeit von der Beleuchtung zu erreichen, kann eine Normierung bei weiterhin effizienter Berechnung im Frequenzraum durchgeführt werden [Lew95]. Dieser Ansatz ist zwar einfach zu realisieren, aber selbst bei der Berechnung im Frequenzraum relativ rechenintensiv und insbesondere sehr empfindlich gegenüber Rotation, Skalierung, Verdeckungen und auch Rauschen.

Ein anderer bekannter Ansatz zur ansichtenbasierten Objekterkennung wird in [Mur95] vorgestellt. Dieses System identifiziert ein Objekt und bestimmt dessen Ausrichtung. Dazu benutzt es größen- und helligkeitsnormierte Grauwertbilder. In einer Lernphase werden Trainingsbilder aller Objekte in allen Ausrichtungen mit Hilfe der Hauptachsentransformation in zwei verschiedene Räume mit 20 oder weniger Dimensionen projiziert. Durch kubische Interpolation aller Projektionen eines Objektes werden Mannigfaltigkeiten innerhalb der Räume bestimmt. Die Räume werden durch die zu den größten Eigenwerten gehörenden Eigenvektoren aller Trainingsbilder (Raum R_1) beziehungsweise aller Trainingsbilder eines Objektes (Raum R_2) aufgespannt. Für die Klassifikation eines neuen Bildes wird der minimale Abstand der Projektion des Bildes in den Raum R_1 zu einer der Mannigfaltigkeiten bestimmt. Die Bestimmung der Ausrichtung geschieht auf vergleichbare Weise im Raum R_2.

Dieser Ansatz kann nicht mit Verdeckungen umgehen und benötigt eine gute vorherige Segmentierung der zu erkennenden Objekte. Verschiedene Erweiterungen dieses Ansatzes versuchen diese Probleme zu umgehen. Häufig werden hierbei nur Teile des zu erkennenden Objektes betrachten, die jeweils nach verschiedenen Heuristiken ausgewählt werden [Ohb97, Bis98].

[Vio01] stellt ein interessantes System zur Erkennung von Gesichtern vor, welches von verschiedenen anderen Autoren, beispielsweise [Lie02], erweitert wurde. Das System läßt sich auch für andere Objekte einsetzen, falls diese texturiert sind. Es ermöglicht in sehr kurzer Zeit zu bestimmen, ob an einer Stelle im Bild ein zu suchendes Objekt vorhanden ist. Die benutzten Merkmale, Differenzen zwischen summierten Pixeln von kleinen Grauwertrechtecken, sind mit Haar-Wavelets vergleichbar. Durch eine Vorverarbeitung lassen sich diese Merkmale sehr schnell berechnen. Mit Hilfe des Auswahlverfahrens AdaBoost [Fre95] werden aus den möglichen Merkmalen im Training diejenigen ausgewählt, die die Klassifikationsgrenze optimieren. Die Klassifikation findet schließlich mit Hilfe einer Klassifikatorkaskade statt, die es erlaubt, möglichst schnell möglichst viele Bildbereiche zurückzuweisen. Jeder Klassifikator besteht dabei aus einer Menge von Merkmalen, die mit Hilfe eines Schwellwertes die Klassifikation durchführen.

Initiiert durch [Swa91] werden in vielen verschiedenen Systemen Histogramme zum Vergleich von Ansichten von Objekten eingesetzt [Vin97, Sch00, Sig02, Pau00]. Dabei wird die Verteilung von Merkmalen, beispielsweise der Farbe oder verschiedene Texturmerkmale, mit Hilfe einer Quantisierungsfunktion auf einem Trainingsbeispiel geschätzt. Die Erkennung erfolgt durch einen Vergleich dieser Verteilung mit der Merkmalsverteilung in einem Eingangsbild. Diese Verfahren werden in Abschnitt 6.2 ausführlich vorgestellt.

6.1.4 Anwendung auf die Domäne

Wie in Kapitel 3 vorgestellt, können jederzeit sprachliche Anfragen nach Objekten an das System gestellt werden. Die angefragten Objekte sollen dabei in der Szene lokalisiert und identifiziert werden. Falls eine Anfrage nicht beantwortet werden konnte, hat das System durch Lernen die Möglichkeit, eine neue Ansicht eines Objektes zu gewinnen. Durch das Herausnehmen des Objektes aus der Szene während des interaktiven Lernens (siehe Abschnitt 3.3), kann das bisher unbekannte Objekt vom Hintergrund segmentiert werden. Für das Lernen steht damit nach einer Anfrage genau eine neue Ansicht eines Objektes zur Verfügung. Das Generieren eines allgemeinen Modells des Objektes ist hieraus ohne weitere große Einschränkungen nicht möglich. Damit ist für das vorliegende Szenario die ansichtenbasierte Vorgehensweise besser geeignet. Aus diesem Szenario ergeben sich verschiedene weitere Folgerungen für die Objekterkennung:

Unbekannte Objekte: Das System muß mit bisher unbekannten Objekten umgehen können. Damit ist insbesondere auch eine Einschränkung auf eine vorher bekannte Domäne mit bekannten Objektcharakteristika nicht möglich. An der Vielfalt der Ansätze zur Objekterkennung sieht man, daß es im Gegensatz zu der Spracherkennung zumindest bisher kein allgemein akzeptiertes Standardverfahren zur Erkennung beliebiger Objekte gibt. Dies wird beispielsweise auch von [Lei03] durch eine Evaluation unterstrichen. Bei dem Vergleich verschiedener ansichtenbasierter Verfahren zeigten die einzelnen Verfahren bei unterschiedlichen Objektgruppen unterschiedliche Stärken und Schwächen. Eine Kombination aller Verfahren war über alle Objekte gesehen am leistungsfähigsten. Auch im angestrebten System scheint eine flexible Kombination unterschiedlicher Erkenner daher sinnvoll zu sein.

Variabler Hintergrund: Während der Interaktion mit dem System kann sich der Hintergrund der Szene jederzeit ändern. Der Benutzer kann jederzeit in die Szene hineinlangen oder sich innerhalb der Szene bewegen. Um eine möglichst große Flexibilität zu erreichen, sollte es auch keine Abhängigkeit von einem speziellen Hintergrund oder einer speziellen Art eines Hintergrundes geben. Ohne Annahmen über den Hintergrund kann aber auch keine vorherige Segmentierung vorgenommen werden, um die Objekterkennung nur auf diesen segmentierten Bereichen anzustoßen. Insbesondere das Template Matching und der Ansatz von [Mur95] arbeiten mit einer vorherigen Segmentierung und reagieren empfindlich, falls diese nicht korrekt ist.

Direktheit der Kommunikation: Kann das System neu erworbenes Wissen sofort einsetzen, ist die Direktheit und Natürlichkeit der Kommunikation erhöht. Dies bedeutet auch, daß es nach der Akquirierung einer neuen Objektansicht keine langwierige Aufbereitung der neu gewonnenen Daten geben sollte. Außerdem

sollte es möglich sein, mit möglichst wenigen Ansichten eines neuen Objektes auszukommen und trotzdem eine sichere Erkennung zu erreichen. Ein Ansatz wie bei [Vio01] ist damit ausgeschlossen. Hier kommen für jedes zu erkennende Objekt mehrere Tausend positive und negative Trainingsbeispiele zum Einsatz. Das nachfolgende Training dauert dann auf aktuellen Rechnern mehrere Tage. Auch [Mur95] benötigen bei ihrem System viele Trainingsbilder. Diese Ansätze haben jeweils den Nachteil, daß sie beispielsweise gegenüber Rotation nicht invariant sind und sehr empfindlich auf kleine Änderungen am Objekt reagieren. Daher benötigen sie diese Fälle möglichst vollständig im Trainingsmaterial. Dies trifft auch auf das Template Matching zu.

Auch Klassifikatoren mit vielen zu schätzenden Parametern wie Polynomklassifikatoren oder Support-Vector-Maschinen eignen sich eher schlecht für das angestrebte System, da sie viele Trainingsbeispiele benötigen und nach jedem neu hinzugekommenen Beispiel aktualisiert werden müssen [Sch96, Jai00].

Geschwindigkeit: Für eine natürliche Kommunikation mit dem System ist insbesondere auch eine hohe Geschwindigkeit wichtig. Muß der Benutzer lange auf eine Systemreaktion bei der Erkennung, Lokalisation oder auch bei dem Lernen von Objekten warten, kommt keine wirkliche Interaktion mit dem System zustande. Auch aus diesem Grund sollten langwierige Aktualisierungsläufe von Klassifikatoren nach dem Lernen einer neuen Ansicht vermieden werden.

Zusammenfassend sollte die Objekterkennung auf mehrere Merkmale aus verschiedenen Kategorien aufsetzen, invariant oder zumindest möglichst robust gegenüber Transformationen wie Translation, Rotation, Beleuchtungsänderungen oder Verdekkungen sein und insbesondere mit möglichst wenig Trainingsmaterial auskommen. Zusätzlich sollte die Geschwindigkeit möglichst hoch sein. Vollständig ist dies sicherlich nicht möglich. Einen guten Kompromiß stellt allerdings die Objekterkennung auf Basis von Histogrammen dar, die im nächsten Abschnitt näher dargestellt wird. Durch den Einsatz von Textur- und Farbmerkmalen können hier schon Hypothesen mit Merkmalen unterschiedlicher Kategorien erstellt werden. Bei dieser Art der Erkennung werden allerdings Ortsinformationen weitestgehend ignoriert. Um diese Information ebenfalls zu benutzen, kommen als weitere Merkmale Graphen basierend auf segmentierten Farbregionen zum Einsatz, die durch ihre Konnektivität die Nachbarschaft der Farbregionen wiedergeben. Diese werden im Abschnitt 6.4 genauer vorgestellt.

6.2 Histogrammbasierter Merkmalsvergleich

Ein schon häufig eingesetztes Verfahren zur Objekterkennung setzt auf Histogramme für die Objektrepräsentation auf und benutzt dabei verschiedene, beispielsweise auf Farbe oder Textur basierende, Merkmale [Swa91, Vin97, Sch00, Sig02]. Auf Grund

verschiedener positiver Eigenschaften, die im Laufe des Abschnitts näher vorgestellt werden, zeigen diese Histogramme eine gute Erkennungsleistung bei hoher Geschwindigkeit. Daher werden sie auch in diesem System eingesetzt. In den folgenden Abschnitten wird dies näher erläutert.

6.2.1 Dichtefunktionen als Objektrepräsentanten

Im Allgemeinen kann das Vorhandensein eines Objektes innerhalb eines Bildes I mit Pixeln (x, y) bestehend aus mehrdimensionalen Merkmalen $\mathbf{v}$ durch eine Wahrscheinlichkeitsdichte f repräsentiert werden:

$$f = f(\mathbf{v}, x, y) \tag{6.2}$$

Durch die Aufnahme von Ansichten des Objektes stehen Muster dieser Verteilung zur Verfügung, aus denen eine Repräsentation der Verteilung geschätzt werden kann. Diese kann dann als Grundlage für die Erkennung des Objektes in neuen Bildern genommen werden.

Für die Schätzung der Verteilung gibt es verschiedene Ansätze. Die folgende Darstellung wichtiger Methoden lehnt sich an [Sil86] an. Grundsätzlich lassen sich die Methoden in parametrische und nichtparametrische Ansätze aufteilen. Bei der parametrischen Vorgehensweise wird eine Familie von Verteilungen, beispielsweise Normalverteilungen, vorgegeben, für die anschließend die fehlenden Parameter an Hand der Trainingsbeispiele geschätzt werden. Für einen erfolgreichen Einsatz ist hierbei allerdings Vorwissen über die Verteilung oder alternativ eine große Menge an Trainingsmaterial nötig. Bei nichtparametrischen Verfahren wird eine Verteilung nicht explizit vorgegeben, sie wird stattdessen aus den Daten direkt gewonnen. Bekannte Verfahren hierfür sind *Histogramme*, *Kernelverfahren* und der *Nächste-Nachbar Schätzer*.

Histogrammschätzung Das älteste Verfahren für die Schätzung sind Histogramme. Hierbei wird der Merkmalsraum $\mathbf{X}$ in eine Menge von nicht überlappenenden Regionen $\mathbf{X}^{(i)}$ aufgeteilt, die zusammen den kompletten Raum abdecken:

$$\mathbf{X}^{(i)} \subset \mathbf{X} \qquad \bigcup \mathbf{X}^{(i)} = \mathbf{X} \qquad \forall i \neq j : \mathbf{X}^{(i)} \cap \mathbf{X}^{(j)} = \emptyset \tag{6.3}$$

Die Schätzung für die Wahrscheinlichkeit $P(\mathbf{x} \in \mathbf{X}^{(i)})$ kann damit mit Hilfe einer Stichprobe $\mathbf{x}_1, \ldots, \mathbf{x}_n$, $\mathbf{x}_i \in \mathbf{X}$ bestimmt werden:

$$\hat{P}(\mathbf{x} \in \mathbf{X}^{(i)}) = \frac{\left|\left\{\mathbf{x}_i \in \mathbf{X}^{(i)}\right\}\right|}{n} \tag{6.4}$$

Die Schätzung $\hat{f}$ für die Wahrscheinlichkeitsdichte ergibt sich dann zu

$$\hat{f}(\mathbf{x}) = \frac{\left|\left\{\mathbf{x}_i \in \mathbf{X}^{(i)}\right\}\right|}{nV_i} \qquad \mathbf{x} \in \mathbf{X}^{(i)} \tag{6.5}$$

wobei V_i das Volumen der Region $\mathbf{X}^{(i)}$ ist.

Kernelschätzer Ein alternatives Verfahren zur Dichteschätzung basiert auf einer Kernelfunktion κ – auch Parzen Fenster genannt – die

$$\int_{\mathbf{X}} \kappa(\mathbf{x})\, d\mathbf{x} = 1 \qquad \text{und} \qquad \forall \mathbf{x} \in \mathbf{X} : \kappa(\mathbf{x}) \geq 0 \tag{6.6}$$

erfüllen muß. Durch Positionierung dieser Funktion an allen beobachteten Positionen $\mathbf{x}_i$ der Stichprobe ergibt sich die Schätzung für die Dichtefunktion zu

$$\hat{f}(\mathbf{x}) = \frac{1}{nh^D} \sum_{i=1}^{n} \kappa\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right) \tag{6.7}$$

h gibt hier die gewünschte Fensterbreite an, die jeweils problemabhängig gewählt werden muß. D ist die Dimension von $\mathbf{X}$. Für κ wird häufig eine Gaußverteilung oder auch eine Gleichverteilung benutzt. Anschaulich ergibt sich die Dichte durch Verwischen der einzelnen Stichprobenelemente mit Hilfe der Kernelfunktion.

k-Nächster-Nachbar Der k-Nächste-Nachbar Schätzer arbeitet wie der Kernelschätzer mit einem Fenster. Er hält aber nicht die Größe eines Fensters konstant, wie dies der Kernelschätzer tut, sondern die Anzahl der Elemente im Fenster. Die Fenstergröße wird jeweils an die lokale Stichprobendichte angepaßt. Damit erhält man folgende Schätzung für die Dichte:

$$\hat{f}(\mathbf{x}) = \frac{k}{n \cdot R_k(x)^D \cdot V_{Kugel}} = \frac{k \cdot \pi^{D/2}}{n \cdot R_k(x)^D \cdot (D/2)!} \tag{6.8}$$

$R_k(x)$ gibt den minimalen Abstand zwischen x und dem $k$ten nächsten Nachbar aus der Stichprobenmenge x_i an. V_{Kugel} ist das Volumen der Einheitskugel.

Alle diese Verfahren haben verschiedene Vor- und Nachteile. Der k-Nächste-Nachbar Schätzer (kNN-Schätzer) ist durch die Suche nach den Nachbarn sehr aufwendig und liefert keine Dichte, da $\int \hat{f}(\mathbf{x})\, d\mathbf{x}$ divergiert [Sil86]. Der Kernelansatz reduziert wie auch der kNN-Ansatz die Datenmenge nicht und ist damit insbesondere bei großen Stichproben aufwendig. Durch die glättenden Eigenschaften dieser beiden Ansätze werden die Stichprobenelemente unter Umständen auch auf eine für die Objekterkennung nicht gewünschte Art modifiziert. In vielen Anwendungen sind allerdings gerade diese glättenden Eigenschaften erwünscht. Die Parameteranzahl ist sowohl bei dem Kernelansatz als auch bei dem kNN-Ansatz abhängig von der Stichprobengröße, was den Vergleich von verschiedenen Dichten erschwert.

Histogramme lassen sich dagegen durch ihre Einfachheit und datenunabhängige Aufteilung sehr schnell berechnen. Durch die Diskretheit der Darstellung und die unabhängig von der Stichprobengröße immer gleiche Anzahl von Regionen ist ein

Vergleich von Histogrammen einfach möglich. Allerdings sind Histogramme empfindlich bezüglich ihrer Stabilität. Eine sehr kleine Änderung in den Daten kann zu einer großen Änderung in der Dichteschätzung führen, falls die Änderung der Daten zu einem Wechsel der zugehörigen Region führt. Insbesondere bei hochdimensionalen Merkmalen kann auch die datenunabhängige Partitionierung des Merkmalsraumes zu vielen Regionen führen. Für eine sinnvolle Abdeckung der Regionen ist in diesem Fall eine große Menge an Elementen in der Stichprobe nötig.

6.2.2 Histogramme zur Objektrepräsentation

Wie im letzten Abschnitt ausgeführt sind Histogramme insbesondere eine schnelle und einfache Darstellungsmöglichkeit für Objekte, falls die Dimension der Merkmalsvektoren nicht zu hoch wird. Damit sind sie für die Objekterkennung im hier vorhandenen Szenario gut geeignet. Das Problem der Unstetigkeit auf Grund der Diskretheit der Zuordnung von Stichprobenelementen zu Regionen des Histogramms bleibt allerdings. Bezüglich dieses Punktes gibt es zwei Lösungsansätze.

Der erste Ansatz nutzt aus, daß in Aufnahmen von realen Bildern aufgenommen mit Hilfe von Kameras inhärent ein hoher Rauschanteil vorhanden ist. Auch durch den Einfluß von Schwankungen in der Beleuchtung und Reflexionen auf den Objekten kommt es stetig zu kleinen Bildänderungen und Farbänderungen durch Schattierungen. Abbildung 6.2 zeigt den Rauschanteil beispielhaft mit Hilfe von Differenzbildern getrennt nach den drei Kanälen eines YUV-Bildes. Diese Einflüsse bewirken, daß eigentlich einfarbige Bereiche einen größeren Bereich im Merkmalsraum abdecken und somit auf verschiedene Regionen des Histogramms verteilt werden. Durch diese Glättung wird implizit die Stabilität von Histogrammen bei natürlichen Bildern erhöht, ohne daß eine spezielle Vorgehensweise nötig wäre.

Ein zweiter Ansatz wurde von [Swa91] erwähnt und von [Sig02] ausführlich untersucht. Bei einem normalen Histogramm gibt es eine eindeutige Zuordnung eines Merkmalsvektors zu einer Region des Histogramms. Dies ist insbesondere auch unabhängig von der exakten Position des Merkmalsvektors in der Region. Die funktionale Abhängigkeit läßt sich durch eine Zugehörigkeitsfunktion des Merkmalsvektors zu einer Region des Histogramms darstellen. Abbildung 6.3 (a) zeigt diese für ein normales Histogramm für den eindimensionalen Fall. Indem man diese Funktion abhängig vom exakten Merkmalsvektor macht und verschiedene Zugehörigkeitsfunktionen überlappen läßt, läßt sich die Unstetigkeit des Histogramms an Regionengrenzen beseitigen. Ergibt die Summe der Zugehörigkeitsfunktionen für jede Position weiterhin 1, ergibt sich hierbei auch weiterhin eine Dichte, die jedes Stichprobenelement gleichartig berücksichtigt. Abbildung 6.3 (b) zeigt ein Beispiel solcher Zugehörigkeitsfunktionen für eindimensionale Merkmalsvektoren. [Sig02] nennt Histogramme, die auf solchen kontinuierlichen Zugehörigkeitsfunktionen aufbauen, Fuzzyhistogramme.

Die in Abbildung 6.3 (b) dargestellte Zugehörigkeitsfunktion entspricht der Invertierung der linearen Interpolation. Bei der linearen Interpolation sind Werte auf einem

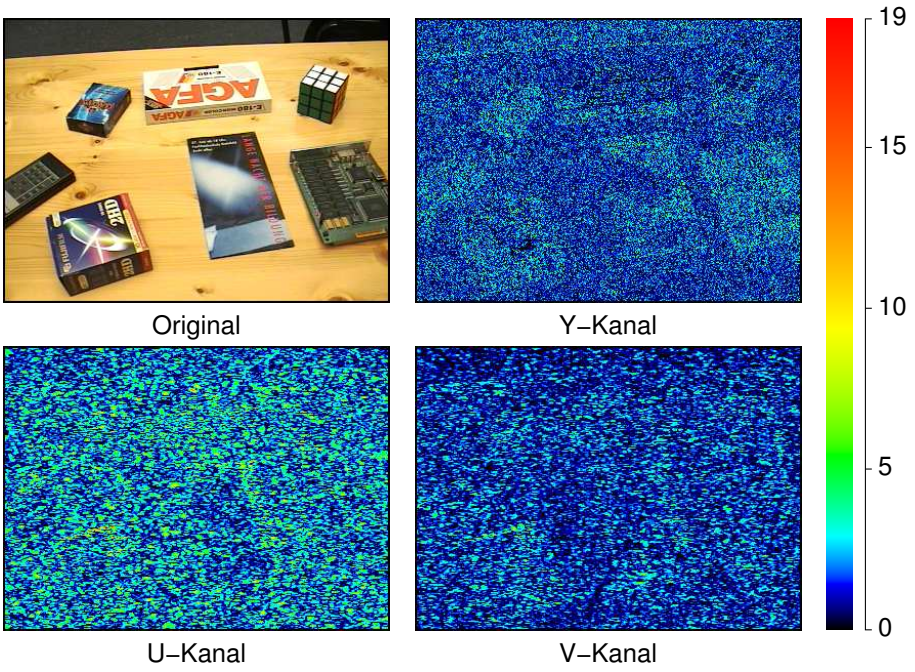


Abbildung 6.2: Differenzbilder der Y-, U- und V-Kanäle zwischen zwei direkt aufeinander folgend aufgenommenen Bildern. Die Farbe gibt den Differenzwert an. Der maximal aufgetretene Wert betrug in diesem Fall 19 bei theoretisch maximal möglichen 255.

Gitter gegeben, aus denen Zwischenwerte bestimmt werden sollen. Bei den Fuzzyhistogrammen sollen aus den Zwischenwerten – den Stichprobenelementen – die Gitterwerte oder auch Regionenwerte berechnet werden. Geht man bei der Regioneneinteilung von Hyperkuben identischer Größe aus, erhöht ein Stichprobenelement $\mathbf{x}$ die Regionenwerte in seiner unmittelbaren Nachbarschaft um einen Wert ΔP, der abhängig ist vom Abstand zum Zentrum des Hyperkubus. [Sig02, Kapitel 4.1] leitet die Berechnung hierfür her: Sei $m_d, d \in \{0, .., D-1\}$ die Anzahl der Hyperkuben in der Dimension d und

$$\tilde{x}_d = \frac{(m_d - 1)(x_d - x_d^{\min})}{x_d^{\max} - x_d^{\min}} \qquad x_d \in [x_d^{\min}, x_d^{\max}) \tag{6.9}$$

das auf die Anzahl der Hyperkuben normierte Stichprobenelement. Die -1 berücksichtigt, daß es in jeder Dimension am Start und am Ende jeweils eine Zugehörigkeitsfunktion halber Breite gibt. Mit dieser Normierung sind

$$\tilde{\mathbf{x}}^{(b_0,\ldots,b_{D-1})} = \sum_{d=0}^{D-1} (\lfloor \tilde{x}_d \rfloor + b_d)\mathbf{e}_d \qquad b_d \in \{0,1\} \quad \mathbf{e}_d : \text{d-te Einheitsvektor} \tag{6.10}$$

die zu $\tilde{\mathbf{x}}$ benachbarten Zentren der Hyperkuben. Damit ergibt sich die Erhöhung der

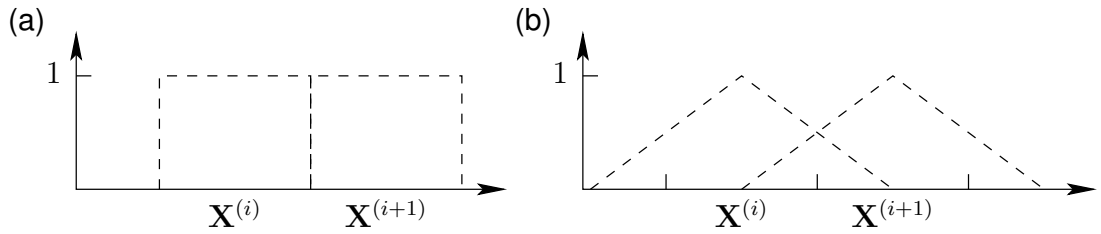


Abbildung 6.3: Zugehörigkeitsfunktion eines eindimensionalen Merkmalsvektors zu einer Region eines Histogramms (a) bei einem normalen Histogramm und (b) bei einem Fuzzyhistogramm, wie es von [Sig02] vorgeschlagen wurde.

Regionenwerte auf Grund eines Stichprobenelementes $\mathbf{x}$ zu

$$\Delta P_{\tilde{\mathbf{x}}^{(b_0,\ldots,b_{D-1})}}(\mathbf{x}) = \frac{1}{n}\prod_{d=0}^{D-1}\left(1-\left|\tilde{x}_d - \tilde{x}_d^{(b_0,\ldots,b_{D-1})}\right|\right) \tag{6.11}$$

für alle 2^D Kombinationen $b_d \in \{0,1\}$. Die Komplexität der Histogrammberechnung nach Gleichung 6.11 ist damit $\mathcal{O}(nD2^D)$ bei einer Stichprobengröße von n, wohingegen die Komplexität einer normalen Histogrammberechnung $\mathcal{O}(nD)$ beträgt.

Beide Ansätze gegen das Unstetigkeitsproblem von Histogrammen scheinen aussichtsreich, sowohl die implizite als auch die explizite Glättung der Histogramme. Damit werden Histogramme zu einem Erfolg versprechenden Ansatz für die Hypothesengenerierung im hier vorgestellten System. Beide Ansätze, sowohl normale Histogramme als auch Fuzzyhistogramme mit Zugehörigkeitsfunktionen entsprechend Abbildung 6.3 (b), wurden im System in der hier dargestellten Form evaluiert. Die Ergebnisse werden in Abschnitt 6.3 näher vorgestellt.

Details der Realisierung

Im Zuge des Lernens einer neuen Ansicht wird durch Herausnehmen eines Objektes aus der Szene eine Bildregion B_1 bestimmt, die das zu lernende Objekt enthält. Abbildung 6.4 zeigt ein Beispiel hierfür. Weitere Details zu diesem Verfahren finden sich in Abschnitt 3.3. Basierend auf dieser ermittelten Bildregion B_1 wird eine neue Ansicht des Objektes bestehend aus Objektmerkmalen generiert. Damit muß aus diesem Bereich ein Histogramm gewonnen werden. Um eine möglichst hohe Geschwindigkeit bei der Histogrammberechnung und dem folgenden Histogrammvergleich zu erreichen, ist eine rechteckige Bildregion vorteilhaft. Das komplette umschließende Rechteck der bestimmten Bildregion enthält allerdings bei vielen Objekten einen großen Hintergrundanteil, der zu Fehlern bei der späteren Suche nach dem Objekt führen kann. Daher wird eine zweite kleinere rechteckige Bildregion B_2 bestimmt, die sich aus der

mittleren Breite W_O und der mittleren Höhe H_O der Bildregion B_1 ergibt. Diese Werte werden durch zeilen- und spaltenweise Abtastung der Bildregion B_1 gewonnen. Positioniert wird die Bildregion B_2 um den Schwerpunkt der Bildregion B_1. Abbildung 6.4 zeigt dies beispielhaft für die Federmappe. Nur der helle Bereich innerhalb des schwarzen Rechtecks wird für die Histogrammberechnung benutzt. Der größte Teil des Hintergrundes ist hier entfernt.

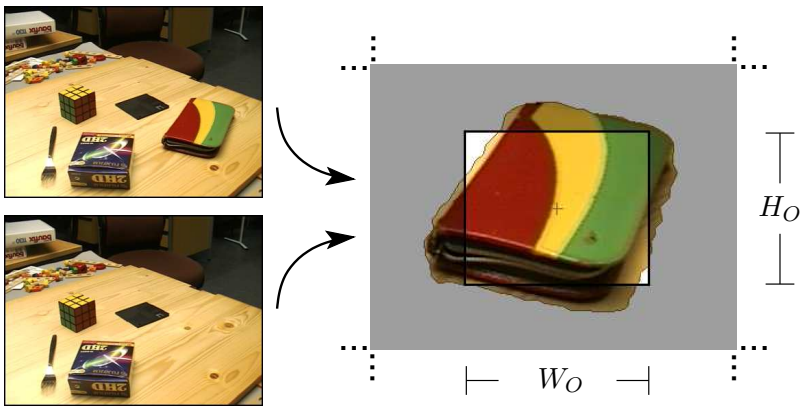


Abbildung 6.4: Lernen einer neuen Objektansicht. Aus der Differenz zweier Bilder wird die Bildregion eines neuen Objektes bestimmt. Der Bereich für die Histogrammberechnung ergibt sich aus der mittleren Breite und Höhe der Differenzregion.

Für die Histogrammberechnung muß entschieden werden, wie die Regionenaufteilung $X^{(i)}$ des Histogramms aussehen soll und insbesondere über welche Eigenschaften des Objektes die Histogrammberechnung durchgeführt werden soll und von welchen Eigenschaften abstrahiert werden soll. Zur Verfügung stehen die Merkmale $\mathbf{v}$ und die Koordinaten (x, y) der Merkmale, siehe Gleichung 6.2, Seite 92. Die Abstrahierung von den Koordinaten bringt für die Objekterkennung Vorteile. Sind die Merkmale $\mathbf{v}$ an sich lage- und rotationsinvariant, wird durch diese Abstrahierung auch das vollständige Histogramm lage- und rotationsinvariant, wodurch für die Objekterkennung merklich weniger Ansichten benötigt werden. Daher werden die Histogramme über den Objektmerkmalen $\mathbf{v}$ berechnet. Für eine schnelle Berechnung und einen schnellen Vergleich der Histogramme ist eine möglichst einfache Histogrammform wünschenswert. Dies wird für alle Histogrammarten durch die für Fuzzyhistogramme schon vorgestellte einheitliche Einteilung in Hyperkuben gleicher Größe erreicht. Als freie Parameter für die Einteilung bleiben damit noch die Anzahl der Aufteilungen $m_d, d \in \{0, .., D-1\}$ in jeder Dimension, wobei D die Dimension der Merkmale $\mathbf{v}$ angibt.

Die allgemeine Berechnung von Fuzzyhistogrammen ist schon auf Seite 95 dargestellt worden. Für die Bestimmung eines neuen Objekthistogramms während der Lernphase wird diese für alle Merkmalsvektoren innerhalb der bestimmten rechtecki-

gen Bildregion B_2 durchgeführt. Bei normalen Histogrammen ist das grundsätzliche Vorgehen identisch. Die Berechnung für jeden Merkmalsvektor unterscheidet sich allerdings. Die Normierung eines Stichprobenelementes $\mathbf{x}$ ist in diesem Fall

$$\tilde{x}_d = \frac{m_d(x_d - x_d^{\min})}{x_d^{\max} - x_d^{\min}} \qquad x_d \in [x_d^{\min}, x_d^{\max}) \tag{6.12}$$

Die Erhöhung eines Regionenwertes ist bei normalen Histogrammen sehr einfach:

$$\Delta P_{\lfloor \tilde{\mathbf{x}} \rfloor}(\mathbf{x}) = \frac{1}{n} \tag{6.13}$$

n gibt die Anzahl der Merkmalsvektoren an und ist somit gleich der Breite W_O mal der Höhe H_O der für das Objekt bestimmten Bildregion B_2.

6.2.3 Vergleich von Histogrammen

Durch das im letzten Abschnitt vorgestellte Verfahren liegen Histogramme vor, die eine Schätzung für eine Wahrscheinlichkeitsdichte eines Objektes darstellen. Die eigentliche Aufgabe der Objekterkennung ist allerdings die Lokalisation und Identifikation von Objekten. Dies wird durch eine Vergleichsfunktion, die die Ähnlichkeit zweier Histogramme $\mathbf{O}$ und $\mathbf{I}$ bestimmt, möglich. In den letzten Jahren wurden verschiedene Abstandsmaße für diesen Zweck vorgeschlagen. [Rub01] und [Sch97] geben einen Überblick über wichtige Maße. Eine Auswahl wird nun näher vorgestellt. Um die Notation einfacher zu halten, werden im folgenden die mehrdimensionalen Histogramme $\mathbf{O}$ und $\mathbf{I}$ in linearisierter und damit eindimensionaler Form betrachtet. Dabei gibt

$$m = \prod_{d=1}^{D} m_d \tag{6.14}$$

die Anzahl der Regionen eines Histogramms an. D ist hier die Dimension der Merkmale und m_d die Anzahl der Regionen in jeder Dimension d.

Minkowski-Metrik: Ein häufig eingesetztes Maß ist die Minkowski-Metrik, die auch in der Mustererkennung häufig benutzt wird:

$$L_p(\mathbf{O}, \mathbf{I}) = \sqrt[p]{\sum_{i=0}^{m-1} |O_i - I_i|^p} \qquad p \in \mathbb{N} \tag{6.15}$$

Für $p = 1$ ergibt sich die City-Block-Metrik, für $p = 2$ die euklidische Metrik.

Histogrammschnitt: Ein klassisches Maß für den Histogrammvergleich ist der Histogrammschnitt [Swa91]. In der Literatur ist er üblicherweise unter der Bezeichnung "Histogram Intersection" zu finden:

$$\cap(\mathbf{O}, \mathbf{I}) = \sum_{i=0}^{m-1} \min\{O_i, I_i\} \tag{6.16}$$

Anschaulich läßt er sich als die Berechnung des gemeinsamen Teils der zwei Histogramme interpretieren. Wie [Swa91] gezeigt hat, läßt sich der Histogrammschnitt auf die L_1-Norm zurückführen, falls die Summe über alle Regionen der beiden Histogramme identisch ist:

$$\cap(\mathbf{O},\mathbf{I}) = 1 - \frac{L_1(\mathbf{O},\mathbf{I})}{2} \quad \text{falls} \quad \sum_{i=0}^{m-1} O_i = \sum_{i=0}^{m-1} I_i \tag{6.17}$$

χ^2-Test: Ein in der Statistik weit verbreiteter Test zum Vergleichen zweier Verteilungen ist der χ^2-Test:

$$\chi^2(\mathbf{O},\mathbf{I}) = \sum_{i=0}^{m-1} \frac{(O_i - I_i)^2}{O_i + I_i} \tag{6.18}$$

Neben dieser Definition gibt es auch noch verschiedene andere Varianten dieses Testes [Sch97].

Quadratisches Abstandsmaß: Alle bisher betrachteten Maße haben den Nachteil, daß sie die Histogrammregionen unabhängig voneinander betrachten. Abbildung 6.5 zeigt ein Beispiel des resultierenden Problems. Um dem entgegen zu wirken, benutzt das quadratische Abstandsmaß eine zusätzliche Matrix $\mathbf{A}$, deren Einträge die Ähnlichkeit von Regionen anzeigen:

$$\mathrm{QA}(\mathbf{O},\mathbf{I}) = \sqrt{(\mathbf{O}-\mathbf{I})^T \mathbf{A} (\mathbf{O}-\mathbf{I})} \tag{6.19}$$

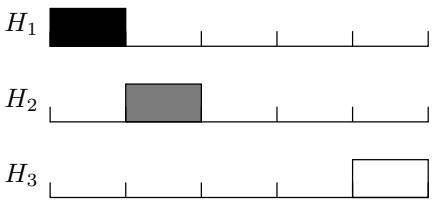


Abbildung 6.5: Durch einen direkten Regionenvergleich sind die Farbhistogramme H_1 und H_2 und die Farbhistogramme H_1 und H_3 gleich ähnlich, obwohl nach der perzeptuellen Wahrnehmung der in den Regionen repräsentierten Merkmale die ersten beiden Histogramme ähnlicher sind als H_1 und H_3.

Earth Movers Distance: Das quadratische Abstandsmaß beachtet beim Vergleich noch nicht die Füllmenge der verschiedenen Regionen. Damit kann ein Eintrag in einem Histogramm gleichzeitig für verschiedene Vergleiche mit unterschiedlichen Regionen des zweiten Histogramms zum Einsatz kommen. Dem wirkt der

"Earth Movers Distance" entgegen, indem er den Vergleich als ein Transportproblem der Füllung von Histogramm **O** nach Histogramm **I** ansieht:

$$\mathrm{EMD}(\mathbf{O},\mathbf{I}) = \min_f \frac{\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} d_{ij} f_{ij}}{\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} f_{ij}} \tag{6.20}$$

d_{ij} gibt den zu überwindenden Abstand zwischen den Histogrammregionen O_i und I_j an. Die Matrix f_{ij} gibt den Fluß zwischen zwei Histogrammregionen an. Dieser muß unter folgenden vier Beschränkungen minimiert werden:

$$f_{ij} \geq 0 \qquad \sum_{j=0}^{m-1} f_{ij} \leq O_i \qquad \sum_{i=0}^{m-1} f_{ij} \leq I_j \tag{6.21}$$

$$\sum_{i=0}^{m-1}\sum_{j=0}^{m-1} f_{ij} = \min\left\{\sum_{i=0}^{m-1} O_i, \sum_{i=0}^{m-1} I_i\right\} \tag{6.22}$$

Das fundierteste Abstandsmaß der hier vorgestellten Maße ist der "Earth Movers Distance". Allerdings hat dieses Maß eine Komplexität von $\mathcal{O}(m^3 \log m)$ [Rub01] und ist damit für die benötigte Aufgabe bei weitem zu rechenintensiv. Auch die Komplexität des quadratischen Abstandsmaßes ist mit $\mathcal{O}(m^2)$ zu hoch. Hilfreich ist es, wenn das Abstandsmaß die Dreiecksungleichung der Metrikdefinition erfüllt. Danach muß für eine Menge X bei einer Metrik d gelten:

$$\forall x, y, z \in X : d(x,z) \leq d(x,y) + d(y,z) \tag{6.23}$$

Ist diese Gleichung erfüllt, läßt sich sowohl die Lokalisation als auch die Identifikation von Objekten durch Ausnutzung von Ähnlichkeiten zwischen Histogrammen beschleunigen. Näheres hierzu wird im nächsten Abschnitt vorgestellt. Sowohl bei dem Histogrammschnitt als auch bei der Minkowski-Metrik ist die Dreiecksungleichung erfüllt. Bei dem χ^2-Test ist dies nicht der Fall.

Die verschiedenen Abstandsmaße wurden unter verschiedenen Bedingungen und Aufgabenstellungen schon von verschiedenen Autoren verglichen. [Sch97] hat drei Varianten des Maßes $\cap$, drei Varianten von χ^2 und zwei Varianten von L_2 für die Objektidentifikation verglichen. Die besten Ergebnisse erzielte das hier präsentierte χ^2 dicht gefolgt vom normalen Histogrammschnitt. Bei partiellen Objektverdeckungen, eine wichtige Eigenschaft in beliebigen Szenen, war der Histogrammschnitt mit Abstand am leistungsfähigsten. Bei [Pau00] kamen die Maße $\cap$, eine hier nicht präsentierte Variante von χ^2, L_2 und EMD für die Objektlokalisation zum Einsatz. Das beste Ergebnis erreichte hier der Histogrammschnitt.

Neben diesen guten Erkennungsraten ist der Histogrammschnitt mit nur zwei Operationen pro Region zusätzlich am schnellsten zu berechnen. Daher wird dieses Abstandsmaß im folgenden eingesetzt.

6.2.4 Schnelle Objektlokalisation

Im hier vorliegenden Szenario sollen Objekte in einer Szene bestehend aus verschiedenen Objekten und einem beliebigen Hintergrund lokalisiert und identifiziert werden. Eine Möglichkeit der Objektidentifikation mit Hilfe von Histogrammen besteht darin, einen Vergleich des Objekthistogramms mit dem Histogramm der vollständigen Szene durchzuführen. Hierbei wird also ein Histogramm, das in einem kleinen Bereich eines Bildes gewonnen wurde, mit einem Histogramm des vollständigen aktuellen Bildes verglichen. Abbildung 6.6 und auch Abbildung 6.4 auf Seite 97 zeigen an Hand zweier Beispiele den Bereich, aus dem ein typisches Objekthistogramm gewonnen wird. Durch den Histogrammschnitt werden bei diesem Vergleich Merkmale, die im Objekthistogramm nicht vorkommen, auch im Szenenhistogramm ignoriert. In der Szene werden aber sehr häufig einzelne Merkmale verstreut über die Szene vorkommen, die zwar vergleichbar im Objekt vorhanden sind, aber innerhalb der Szene nicht zu diesem Objekt gehören. Damit wird es bei dieser Art der Identifikation häufig zu falsch positiven Ergebnissen kommen. Eine Lokalisation ist bei dieser Art des Vergleichs überhaupt nicht möglich.

Daher findet die Objektsuche durch das Schieben eines Fensters in der Größe des Objekthistogramms über das vollständige Szenebild statt. An jeder Stelle des kompletten Bildes wird ein Vergleich des Objekthistogramms mit einem Histogramm generiert aus einem lokalen Ausschnitt des Bildes vorgenommen. Jeder Vergleich gibt eine Wahrscheinlichkeit für das Auftreten des Objektes an der getesteten Bildposition an. Insgesamt ergibt sich damit eine Karte, die für jede Position die Wahrscheinlichkeit des Vorhandenseins des Objektes auf Grund des hier betrachteten Merkmals spezifiziert. Abbildung 6.6 zeigt zwei Beispiele solcher Wahrscheinlichkeitskarten, die auf Grund einer Objektanfrage generiert wurden. Je dunkler diese Karten sind, desto wahrscheinlicher ist jeweils das Vorhandensein des Suchobjektes. Der weiße Rand kommt auf Grund eines bei der Suche ausgelassenen Bereiches zu Stande, der jeweils die halbe Breite beziehungsweise Höhe des Suchobjektes hat. Es wird also nur nach Objekten gesucht, die vollständig innerhalb der Szene liegen.

Auf Seite 73 im Abschnitt 5.4.2 wurde die Berechnung einer Aufmerksamkeitskarte vorgestellt, die für jeden Punkt eine Wahrscheinlichkeit dafür spezifiziert, daß der jeweilige Punkt im Fokus einer Zeigegeste lag. Hiermit, und durch die Benutzung einer allgemeinen sprachlichen Benennung wie “das Teil”, ist es dem Benutzer möglich, die Aufgabe des Systems auf die Identifikation eines Objektes zu fokussieren. Die durch das Schieben des Objekthistogramms berechnete Wahrscheinlichkeitskarte wird mit der Gestik basierten Aufmerksamkeitskarte multipliziert, um eine endgültige Karte für das Merkmal zu erhalten. Liegt der Maximalwert der Karte über einem Schwellwert, ist das Objekt an einer Position im Bild gefunden worden. Andernfalls konnte das Objekt durch dieses Merkmal und unter Beachtung einer eventuellen Geste nicht gefunden werden.

Ein großes Problem ist allerdings die hohe Komplexität dieses Ansatzes bei einer

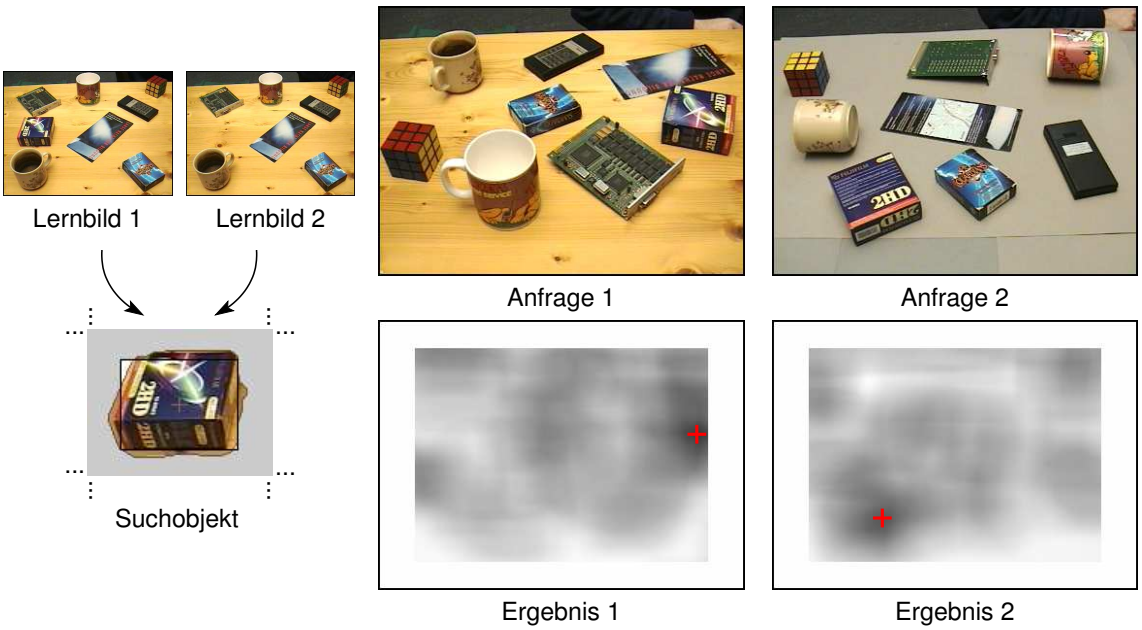


Abbildung 6.6: Beispiele für Wahrscheinlichkeitskarten, die durch eine histogrammbasierte Suche gewonnen wurden. Das Suchobjekt, gewonnen aus den Bildern links, wurde in den Anfragebildern gesucht. Die wahrscheinlichste Position ist in den Ergebnissen jeweils mit einem roten Kreuz markiert.

direkten naiven Realisierung. Bei den Tests des Systems kamen Bilder der Größe $W_I = 378$ mal $H_I = 278$ Pixel zum Einsatz. Bei einer Objektgröße von $W_O = 77$ mal $H_O = 59$ Pixeln, der Größe des Suchobjektes in Abbildung 6.6 links, muß beispielsweise an insgesamt 66440 unterschiedlichen Positionen ein Histogrammvergleich durchgeführt werden. Dies ist auch mit dem schnellen Histogrammschnitt für ein interaktives System bedeutend zu aufwendig. Es gibt allerdings zwei Optimierungsansätze, das gleitende Fenster und die aktive Suche, die im folgenden näher vorgestellt werden.

Beschleunigung durch ein gleitendes Fenster

Ein Ansatz zur Beschleunigung ergibt sich durch ein Verfahren ähnlich dem, das in [Hua79] für die schnelle Berechnung eines Medianfilters vorgestellt wurde. Beim Verschieben des Fensters über das Bild kann ausgenutzt werden, daß das Histogramm eines Bildausschnittes und das Histogramm des um ein Pixel verschobenen Bildausschnittes sehr ähnlich sind. Für die Bestimmung des Ausschnitthistogramms I beispielsweise beim Verschieben von links nach rechts müssen jeweils nur die auf den H_O links herausfallenden Pixeln basierenden Merkmale aus dem Histogramm entfernt werden und die H_O rechts neu hereinkommenden Merkmale neu in das Histogramm

eingetragen werden. Abbildung 6.7 zeigt dies schematisch. Wahlweise kann die Verschiebung des Fensters auch von oben nach unten erfolgen. In diesem Fall müssen jeweils $2 \cdot W_O$ Pixel betrachtet werden. Vergleichbar zur Bestimmung des Median in [Hua79] muß auch der Histogrammschnitt nicht an jeder Stelle neu berechnet werden. Er kann entsprechend der herausfallenden und hereinkommenden Merkmale mit aktualisiert werden. Damit wird die Berechnung unabhängig von der Histogrammgröße.

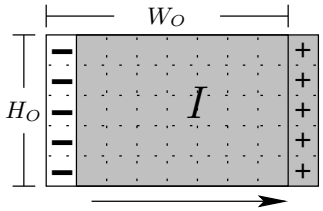


Abbildung 6.7: Beschleunigung der Histogrammsuche durch ein gleitendes Fenster. Beim verschieben eines Fensters müssen nur die sich ändernden Pixel betrachtet werden.

Gilt $W_O \ll W_I$ und $H_O \ll H_I$, ist das zu suchende Objekt also deutlich kleiner als das Bild der Szene, kann bei der Komplexitätsbetrachtung das Überspringen des Bildrandes ignoriert werden. In diesem Fall ergibt sich eine Komplexität von $\mathcal{O}(W_I \cdot H_I \cdot \min\{W_O, H_O\})$. Beim naiven Algorithmus ergibt sich dagegen $\mathcal{O}(W_I \cdot H_I \cdot (W_O \cdot H_O + m))$, wobei m die Anzahl der Histogrammregionen angibt. Bei diesem Ansatz muß an jeder Bildposition ein Histogramm berechnet und ein Histogrammvergleich durchgeführt werden. Der optimierte Algorithmus hat zusätzlich die wünschenswerte Eigenschaft, daß er in der Hauptschleife nur wenige Operationen benötigt: Eine Subtraktion und eine Addition zum Anpassen des Histogramms auf Grund eines herausfallenden und hereinkommenden Merkmals, zwei Vergleiche um festzustellen, ob der Histogrammschnitt auf Grund dieser beiden Merkmale angepaßt werden muß und im Durchschnitt eine Anpassung des Histogrammschnitts durch eine Addition oder Subtraktion.

Bei aktuellen Rechnern ist allerdings nicht nur die Komplexität des Algorithmus bedeutend, da auf Grund langsamen Hauptspeichers eine gute Cacheausnutzung entscheidend ist. Beim Verschieben des Fensters von oben nach unten liegen die herausfallenden Merkmale und die neu hereinkommenden Merkmale bei üblicher Matrixanordnung jeweils linear im Speicher. Eine separate Behandlung zuerst aller herausfallenden Merkmale und dann aller neu hereinkommenden Merkmale ist daher am schnellsten. Da hier die Cacheausnutzung besonders gut ist, wird diese Richtung auch bei Fenstern leicht größerer Breite als Höhe bevorzugt. Im realisierten System ist hier mit einem Faktor von $1.7 \cdot H_O$ ein fester Wert gesetzt. Am effizientesten wäre eine Anpassung zur Laufzeit an die jeweils benutzte Hardware. Beim Verschieben des Fensters von links nach rechts ist das abwechselnde Behandeln der herausfallenden und hereinkom-

menden Merkmale in einer Schleife am günstigsten, da die Merkmale dann in linearer Form angesprochen werden.

Beschleunigung durch eine aktive Suche

Ein weiterer Ansatz zur Optimierung ergibt sich durch den von [Vin97, Vin98] vorgestellten Ansatz der aktiven Suche. Die Idee hierbei ist, daß Histogramme aus sich überlappenden Bildregionen, die daher ähnliche Merkmale enthalten, auch eine vergleichbare Ähnlichkeit zum Objekthistogramm aufweisen. Voraussetzung dabei ist die Gültigkeit der Dreiecksungleichung für das eingesetzte Vergleichsmaß. Beim Histogrammschnitt ist diese Gleichung bei Vergleich gleich großer Histogramme gültig. Damit können ausgehend von einem Histogrammvergleich an einer Position im Bild für die örtliche Nachbarschaft Schwellwerte für eine maximal zu erreichende Ähnlichkeit eines Histogrammvergleichs an der jeweiligen Position berechnet werden. Liegt der Schwellwert unter einem bisher erreichten Maximalwert, kann ein Histogrammschnitt an dieser Stelle kein neues Maximum erreichen. Daher muß an dieser Stelle auch kein Histogrammvergleich mehr durchgeführt werden.

[Vin97] leitet eine Abschätzung für den maximal zu erreichenden Histogrammschnitt mit einer Bildregion B her, falls der Histogrammschnitt mit einer Bildregion A bekannt ist:

$$\cap(O, I_B) \leq \frac{\min\{\cap(O, I_A)|A|\,,\ |A \cap B|\} + |B - A|}{|B|} \tag{6.24}$$

I_A und I_B geben die Histogramme der Bildregionen an, $||$ die Größe einer Bildregion in Pixeln. Die restlichen Bezeichner werden in Abbildung 6.8 erläutert. Im hier betrachteten Fall gilt $|A| = |B| = W_O H_O$. Bei der Durchführung der aktiven Suche wird nun der Histogrammschnitt in einem Ausschnitt der Größe W_O mal H_O der linken oberen Ecke des Bildes berechnet. Mit Hilfe von Gleichung 6.24 werden die Schwellwerte in dessen Umgebung bestimmt. Daraufhin wird die Suche durch pixelweises Verschieben der aktuell betrachteten Position im Bild fortgesetzt. Ist der jeweilige lokale Schwellwert kleiner als der bisher beste erreichte Histogrammschnitt, wird mit dem nächsten Pixel fortgesetzt. Ansonsten muß der Histogrammschnitt an der aktuellen Position neu berechnet werden. Die Schwellwerte in der Nachbarschaft werden in diesem Fall auf Grund des neu berechneten Histogrammschnitts aktualisiert.

Durch diesen Algorithmus werden zwar viele Histogrammvergleiche und Histogrammberechnungen gespart, dafür kommen allerdings neue Berechnungen für die Bestimmung der Schwellwerte hinzu. Dadurch wird dieser Algorithmus im Durchschnitt im Vergleich zu einer optimierten Implementierung des gleitenden Fensters sogar langsamer. Allerdings gibt es noch Optimierungsmöglichkeiten, die [Vin97] nicht aufgeführt haben. Zum einen kann ein gleitendes Fenster genutzt werden, falls für zwei direkt benachbarte Pixel ein Histogrammschnitt berechnet werden muß. Auch bei dicht beieinander liegenden Pixeln wäre dies möglich. Entfernungen größer einem

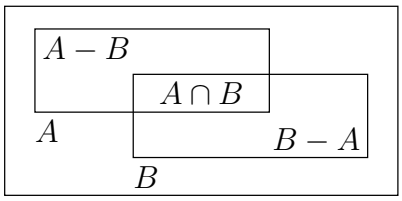


Abbildung 6.8: Zwei sich überschneidende Fenster und die sich daraus ergebenen Bildregionen; eingesetzt in der aktiven Suche.

Pixel werden im realisierten System allerdings noch nicht ausgenutzt. Bei der Aktualisierung der Schwellwerte kann ausgenutzt werden, daß Gleichung 6.24 ausgehend von der Position $A = B$ monoton steigend ist. Damit können die Pixelpositionen bestimmt werden, ab denen Gleichung 6.24 eins wird. Ab hier müssen die Schwellwerte nicht mehr aktualisiert werden, sie würden keine Einschränkung mehr bedeuten.

Als drittes bietet es sich an, die Schwellwerte auf einem verkleinerten Bild zu berechnen. Bei einer Verkleinerung um den Faktor n müssen jeweils n^2 weniger Schwellwerte aktualisiert werden. Bei diesem Verfahren werden mehrere Bildpixel auf einen Schwellwert abgebildet. Um Fehler bei der Abschätzung zu vermeiden, müssen von diesen Pixeln die von der zentralen Bildregion A am weitesten entfernten Pixel für die Schwellwertberechnung genommen werden. Damit wird die Abschätzung schlechter und es müssen mehr Histogrammschnitte berechnet werden. Die gesparten Schwellwertberechnungen wiegen dies allerdings im Durchschnitt wieder auf. Im eingesetzten System kam eine Verkleinerung der Breite und Höhe um den Faktor acht zum Einsatz.

Mit diesen Optimierungen kann zum Teil eine merkliche Beschleunigung im Vergleich zur Berechnung mit einem gleitenden Fenster erreicht werden. Die genauen Ergebnisse sind in Abschnitt 6.3 näher dargestellt. Zwei nicht zu lösende Nachteile bleiben allerdings. Einerseits ist dieser Ansatz nicht unabhängig von der Histogrammgröße m, da der Histogrammschnitt zum Teil neu berechnet werden muß. Außerdem liefert dieser Algorithmus nur die Position und den Wert des maximalen Histogrammschnitts des Objektes im Bild. Es wird keine vollständige Wahrscheinlichkeitskarte erstellt, da durch den Schwellwert Positionen bei der Berechnung übersprungen werden. Abbildung 6.9 zeigt Beispiele für mit der aktiven Suche erstellte Wahrscheinlichkeitskarten für die beiden Bilder, die auch in Abbildung 6.6 für die Demonstration der vollständig berechneten Wahrscheinlichkeitskarte zum Einsatz kamen.

6.2.5 Merkmale für die Histogrammberechnung

Im bisherigen Abschnitt wurde die Generierung einer Wahrscheinlichkeitskarte für die Spezifikation einer Position eines Objektes mit Hilfe des Histogrammschnitts vorgestellt. Die Histogramme werden dabei über Objektmerkmalen $\mathbf{v}$ berechnet, von den Bildkoordinaten (x, y) wird durch die Histogrammerstellung abstrahiert. Weitere De-

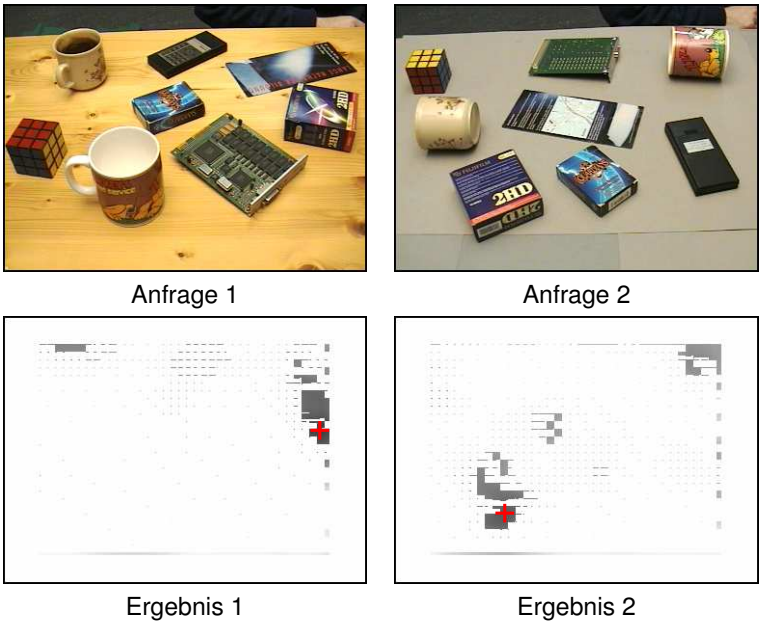


Abbildung 6.9: Beispiele für Wahrscheinlichkeitskarten, die bei einer aktiven Suche mit dem Histogrammschnitt entstanden sind. Das Suchobjekt war wie in Abbildung 6.6 der Diskettenkasten. Die wahrscheinlichste Position ist jeweils mit einem roten Kreuz markiert.

tails hierzu finden sich auf Seite 97. Auf die konkreten Merkmale $\mathbf{v}$ wurde bisher allerdings nicht eingegangen.

In den letzten Jahren wurden bei der Benutzung von Histogrammen in der Mustererkennung verschiedene farb- oder texturbasierte Merkmale eingesetzt. [Swa91] setzen in ihrem Objekterkennungssystem beispielsweise Farbe als Merkmale ein. Dabei nehmen sie einen Farbraum basierend auf Gegenfarben, der durch eine einfache lineare Transformation des RGB-Farbraums gewonnen wird:

$$rg = R - G \qquad by = 2B - R - G \qquad wb = R + G + B \tag{6.25}$$

[Pau00] haben verschiedene Farbräume für die Eignung bei der Objektsuche mit Histogrammen verglichen. In ihren Experimenten getestet haben sie die Farbräume RGB, rg, HSI, YUV, CIE L*a*b* und die drei Varianten H, UV und a*b* der vorherigen Farbräume, die durch Weglassen der Luminanzkomponenten entstanden sind. Einen Überblick über diese Farbräume gibt Anhang A. Bei ihren Tests haben sie die Erkennungsraten bei der Suche nach 17 verschiedenen Objekten unter vier verschiedenen Beleuchtungsverhältnissen untersucht. Unter diesen Bedingungen hat der perzeptuell lineare und rein chrominanzbasierte a*b*-Farbraum die besten Ergebnisse erzielt.

Eine positive Eigenschaft von Farbmerkmalen ist deren inhärente Rotationsinvarianz, da sie keinerlei rotationsabhängige Strukturinformationen enthalten. Auch die

Berechnung der Merkmale ist einfach und schnell. Außer einer Farbraumtransformation, die nur von der Bildgröße abhängt und damit eine Komplexität von $\mathcal{O}(W_I H_I)$ hat, sind keine weiteren Berechnungen nötig. Dafür sind allerdings unterschiedliche Beleuchtungssituationen ein ungelöstes Problem beim Einsatz von Farbe. Näheres zu dieser Problematik findet sich auf Seite 52. Die besten Ergebnisse durch die Benutzung des a*b*-Farbraums und damit die Ignorierung der Helligkeitsinformationen bei [Pau00] ergaben sich aus diesem Problem heraus.

Um beleuchtungsunabhängiger zu werden, wird in [Fun95] nicht direkt mit den Farben gearbeitet, sondern mit Quotienten von benachbarten Farbpixeln. Damit setzen sie texturbasierte Merkmale ein. Für die Berechnung logarithmieren sie die einzelnen Farbkanäle und nehmen verschiedene Ableitungen als Merkmale. Getestet haben sie den Laplaceoperator, den "Laplacian of Gaussian"-Operator und eine Summe über verschiedene einfache Ableitungen. Alle diese Maße haben bei einer nichtlinearen Einteilung der Histogramme eine gute Erkennungsrate unter verschiedenen Beleuchtungen erreicht.

In [Sch97] und [Sch00] sind insbesondere Ableitungen von Gaußfunktionen die Basis der verwendeten Merkmale. Aus diesen werden verschiedene Varianten von Merkmalen berechnet. Im einzelnen sind dies die erste Ableitung in x- und y-Richtung, der Betrag und die Richtung des Gradienten der Gaußfunktion, der Laplaceoperator und ein spezieller, auf einer Summe aus ersten und zweiten Ableitungen bestehender Operator. Eine nützliche Eigenschaft dieser Merkmale ist deren Ausrichtbarkeit in eine gewünschte Richtung durch eine Linearkombination einiger Basisrichtungen der Merkmale. Die Operatoren wurden in unterschiedlichen Experimenten in verschiedenen Kombinationen erfolgreich eingesetzt.

Eine interessante Möglichkeit der Erzeugung translations- und rotationsinvarianter Texturmerkmale verwendet [SM95]. Durch die Integration und damit Mittelung über eine Transformationsgruppe G, die auf Bildern I arbeitet, wird hier eine Funktion f in eine bezüglich der Gruppe G invariante Funktion F überführt:

$$F(I) = \frac{1}{|G|} \int_G f(gI)\, dg \tag{6.26}$$

Für die Gruppe G der Rotationen und Translationen ergibt sich damit unter der Annahme zyklischer Bildränder:

$$F(I) = \frac{1}{2\pi W_I H_I} \int_{t_0=0}^{W_I} \int_{t_1=0}^{H_I} \int_{\varphi=0}^{2\pi} f(gI)\, d\varphi\, dt_1\, dt_0 \tag{6.27}$$

mit

$$(gI)(\mathbf{n}) = I_{\mathbf{n}'} \quad \text{mit} \quad \mathbf{n}' = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \mathbf{n} + \mathbf{t} \tag{6.28}$$

Anschaulich bedeutet Gleichung 6.27, daß die Funktion f zuerst durch Rotation um 360° um alle Punkte (t_0, t_1) des Bildes I ausgewertet werden muß. Alle Werte aller

Punkte werden anschließend addiert. Hat f nur eine begrenzte Ausdehnung, wird damit die Mittelung einer lokalen Unterstützung durch f berechnet. In diesem Fall wird F auch relativ robust gegenüber vereinzelten Objektverformungen, da nur die berechneten Werte im Bereich der Verformungen abweichen. Beispielsweise liefern Bilder einer verschieden weit geöffneten Schere sehr ähnliche Werte für F. An Funktionen f werden in [SM95] Monome der Form $f(I) = \prod_k I_{\mathbf{n}^{(k)}}$ eingesetzt. Ein Beispiel wäre $f(I) = I_{0,0}I_{0,2}$. Bei dieser Funktion werden die Pixel auf einem Kreis mit Radius 2 addiert und mit dem Zentralpixel multipliziert.

[Sig02] benutzt diesen Ansatz, ersetzt dabei aber die Mittelung über die x- und y-Koordinaten durch eine Histogrammberechnung. Damit hat er rotations- und translationsinvariante texturbasierte Merkmale. Monome kommen wie bei [SM95] definiert zum Einsatz, werden allerdings zusätzlich normiert:

$$f(I) = \sqrt[K]{\prod_{k=0}^{K-1} I_{\mathbf{n}^{(k)}}} \tag{6.29}$$

Durch die Normierung bleibt der Dynamikbereich erhalten und die Linearität der Merkmale wird erhöht. Weiterhin werden von [Sch01] vorgeschlagene relationale Funktionen eingesetzt, die jeweils zwei Pixel miteinander vergleichen:

$$f(I) = \mathrm{rel}\left(I_{\mathbf{n}^{(0)}} - I_{\mathbf{n}^{(1)}}\right) \tag{6.30}$$

mit

$$\mathrm{rel}(x) = \begin{cases} 1 & , \quad x < -c \\ \frac{1}{2c}(c - x) & , \quad -c \leq x \leq c \\ 0 & , \quad c < x \end{cases} \qquad c \in \mathbb{R} \tag{6.31}$$

Durch die kontinuierliche Rampenfunktion werden die Merkmale im Vergleich zu einer einfachen Schwellwertfunktion stabiler gegenüber Rauschen.

Ein weiteres Merkmal läßt sich aus dem Sobeloperator, einem häufig benutzten Kantendetektor [Gon02, Kapitel 10.1.3], gewinnen. Dieser Operator führt nicht nur eine Ableitung aus, sondern glättet das Bild zusätzlich orthogonal zu der Richtung der Ableitung durch eine Gaußfilterung. Die Faltungsmasken für den Sobeloperator in x- und y-Richtung sind

$$D_X = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad D_Y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{6.32}$$

Durch Anwendung des Sobeloperators in x- und y-Richtung und Summation ergibt sich ein relativ rotationsinvariantes und beleuchtungsinvariantes Merkmal $v_{x,y}$, welches die lokalen Kantenstärken betrachtet:

$$v_{x,y} = \frac{|I_{x,y} * D_X| + |I_{x,y} * D_Y|}{6} \tag{6.33}$$

Eingesetzte Merkmale

Um die benötigte Menge an Objektansichten gering zu halten, ist die Rotationsinvarianz eine interessante Eigenschaft. Alle im System zum Einsatz kommenden Merkmale, sowohl farbbasierte als auch strukturbasierte, erfüllen diese Eigenschaft zumindest näherungsweise. Die im System eingesetzten farbbasierten Merkmale sind der YUV-Farbraum und der perzeptuell lineare CIE L*u*v*-Farbraum. Einen Überblick über diese Farbräume gibt Anhang A. An strukturbasierten Merkmalen kommen die Monome nach Gleichung 6.29, relationale Merkmale nach Gleichung 6.30 und der Sobeloperator nach Gleichung 6.33 zum Einsatz. Diese Merkmale werden jeweils auf dem L*-Farbkanal berechnet. Einerseits ist dieser Kanal perzeptuell linear. Zusätzlich hat der Helligkeitskanal bei vielen von Kameras und Grabbern gelieferten Bildformaten eine höhere strukturelle Auflösung als die Farbkanäle. Die freien Parameter für die Monome und die relationalen Merkmale wurden in Übereinstimmung mit [Sig02] eingeschränkt. Die Konstante c der relationalen Merkmale ist auf 25 gesetzt und die Koordinate $n^{(1)}$ der relationalen Merkmale auf $(0, 0)$. Damit bleibt hier noch ein freier Parameter. Bei den Monomen ist die Anzahl K der Faktoren auf 2 gesetzt. Der Phasenversatz der Koordinaten $n^{(0)}$ und $n^{(1)}$ ist auf 90° gesetzt, so daß für diese Koordinaten noch 2 freie Parameter bleiben. Die optimale Wahl dieser Parameter wird durch eine im nächsten Abschnitt vorgestellte Evaluierung vorgenommen.

Eine Möglichkeit der Benutzung der Merkmale ist die Integration aller Merkmale in ein hochdimensionales Histogramm. Dies hat allerdings den Nachteil, daß insbesondere bei nicht zu geringer Regioneneinteilung je Dimension die Anzahl der Histogrammregionen sehr stark ansteigt. Die Folge ist ein nur spärlich besetztes Histogramm, welches gegen Rauschen besonders empfindlich wird. Ein weiterer Nachteil ist, daß die Einflußnahme auf die Gewichtung zwischen farb- und strukturbasierten Merkmalen komplizierter wird. Eine freie Gewichtung zwischen verschiedenen Merkmalen wird im vorgestellten System sowohl durch den interaktiven Lernprozeß als auch durch die Möglichkeit der Benutzung von Adjektiven in der Sprache benötigt. Daher werden Farbhistogramme und Texturhistogramme getrennt bestimmt und verglichen. Beide generieren unabhängig voneinander Hypothesen mit getrennten Wahrscheinlichkeitskarten, die nachträglich durch Addition kombiniert werden. Dies wird in Abschnitt 7.1 näher vorgestellt.

6.3 Ergebnisse der Histogrammsuche

Das Ziel des im letzten Abschnitt vorgestellten Teilsystems war die Lokalisation und Identifikation von Objekten durch den Einsatz von Histogrammen. Neben einer hohen Erkennungsrate, die bei der Objekterkennung natürlich immer erwünscht ist, gab es weitere Ziele. Insbesondere sollte das System mit möglichst wenig Trainingsmaterial auskommen und gleichzeitig sowohl beim Lernen einer neuen Objektansicht als auch bei der Objektsuche schnell sein. Daher wird in diesem Abschnitt nun sowohl

die Geschwindigkeit des Systems als auch dessen Güte bei minimalem Trainingsmaterial ausgewertet. Weiterhin wird der Einfluß verschiedener im System modifizierbarer Parameter auf die Geschwindigkeit und auf die Erkennungsleistung untersucht.

6.3.1 Evaluierungsstichproben

Zur Auswertung der Histogrammsuche wird eine Stichprobe an Daten benötigt. Dazu gibt es die Möglichkeit, eine bestehende Stichprobe wie die "Columbia object image library" (COIL-100, [Nen96]) zu benutzen oder eine neue Stichprobe aufzunehmen. Insbesondere die COIL-Stichprobe, eine der wenigen weithin akzeptierten Stichproben, ist für das angestrebte System ungeeignet, da es eine Stichprobe einzelner Objekte vor einem schwarzen Hintergrund ist. Um möglichst realistische Daten zu erhalten, wurde eine neue Stichprobe mit dem System selbst aufgenommen.

Für die Stichprobe wurden insgesamt 73 Bilder von acht Objekten aufgenommen, 16 Bilder für das Training und 57 Bilder für den Test. Alle Bilder hatten eine Größe von 378x278 Pixeln. Die Beleuchtung der Szene blieb konstant. Acht der 16 Trainingsbilder enthielten acht unterschiedliche Objekte. Die restlichen acht Trainingsbilder wurden nach dem Entfernen je eines der Objekte aufgenommen. Dabei wurde darauf geachtet, daß außer dem Entfernen des Objektes keine weiteren Szenenmodifikationen auftraten. Diese Bilder bildeten die Grundlage für das Lernen einer neuen Objektansicht mit der in Abschnitt 3.3 beschriebenen Methode. Das Lernen einer Ansicht geschah damit mit Hilfe des vorgestellten Systems. Mit dem vorhandenen Trainingsmaterial konnte pro Objekt eine Ansicht generiert werden. Zwei der 16 Trainingsbilder sind zusammen mit dem Ergebnis der aus allen Trainingsbildern gewonnenen Segmentierung aller acht Objekte in Abbildung 6.10 dargestellt. Auf Grund der automatisch durchgeführten Segmentierung gibt es jeweils verschiedene kleine Fehler in den generierten Ansichten. Viele Objekte enthalten beispielsweise einen durch Schatten verursachten schmalen Rand von Hintergrundpixeln.

Neben den 16 Bildern des beschriebenen Trainingsmaterials wurden weitere 57 Bilder mit jeweils acht Objekten als Testmaterial aufgenommen. Die Teststichprobe enthielt damit $57 \cdot 8 = 456$ verschiedene Objektansichten. Auf den verschiedenen Bildern wurden die Objekte nach verschiedenen Vorgaben von Hand in der Szene angeordnet. Die Objekte standen immer auf einem Tisch mit horizontaler Platte. Abbildung 3.1 auf Seite 22 zeigt das Szenario. Die jeweilige Anordnung wurde auf drei verschiedenen Hintergründen wiederholt. Durch die Anordnung der Objekte von Hand kam es zwischen den unterschiedlichen Hintergründen zu Variationen in der exakten Positionierung der Objekte, was die Variationen in der Teststichprobe weiter erhöht hat. An Objektanordnungen kamen zum Einsatz:

1. Vier Bilder, in denen die Objektposition im Vergleich zum Trainingsbild 6.10 (a) weitestgehend beibehalten wurde, die Objekte aber jeweils um 90° um ihre vertikale Achse rotiert wurden. Abbildung 6.11 (a) zeigt drei Beispielbilder.

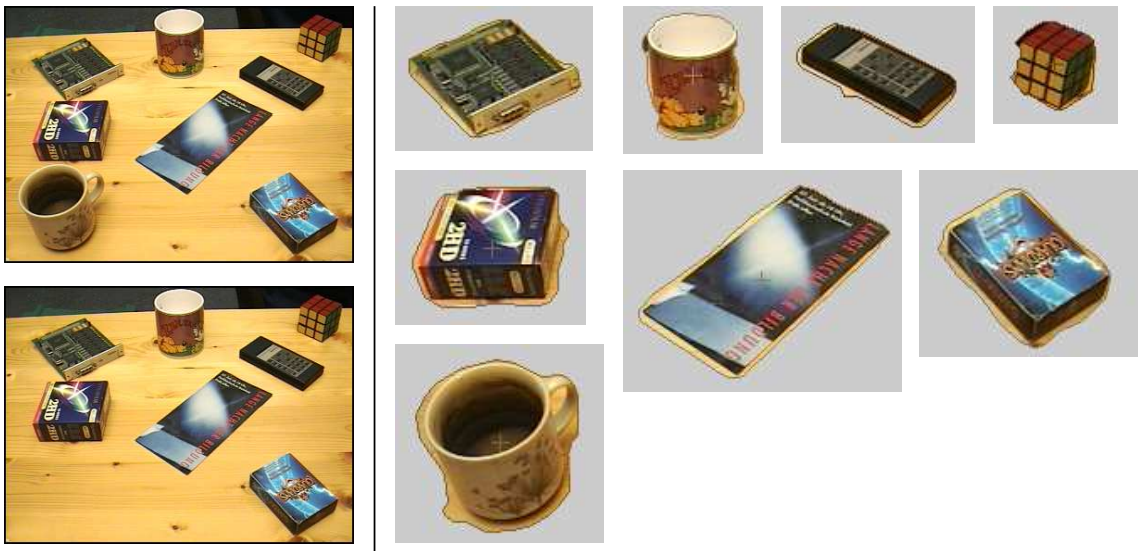


Abbildung 6.10: Trainingsmaterial und Modelle für die Histogrammsuche. (a) Zwei der 16 zum Lernen der Objekte benutzten Bilder. (b) Segmentierte Bildregionen, aus denen Objekthistogramme bestimmt wurden.

2. Neun Bilder, in denen die Objekte beliebig verschoben und um beliebige Winkel um ihre vertikale Achse rotiert wurden. Abbildung 6.11 (b) zeigt drei Beispielbilder.

3. Sechs Bilder, in denen die Objekte beliebig in der Szene verschoben und um beliebige Achsen rotiert wurden. Abbildung 6.11 (c) zeigt drei Beispielbilder.

6.3.2 Geschwindigkeit der Histogrammsuche

Eine wichtige Anforderung an das System war eine hohe Geschwindigkeit, um bei der Kommunikation mit dem System ein möglichst natürliches interaktives Verhalten zu erreichen. Zur Verifikation wurde die Geschwindigkeit auf den zwei in Tabelle 6.1 aufgeführten Rechnern *Alpha* und *PC* getestet. Wie bei der Auswertung der Gestenerkennung schon erläutert, sind die hier angegebenen SPEC-Werte Herstellerangaben, die auf Grund abweichender Testsysteme unter Umständen nicht exakt mit den Werten der eingesetzten Rechner übereinstimmen. An Compilern kamen wie bei der Gestenerkennung bei der Alpha der DEC C Version 5.9 und bei dem PC der GCC Version 3.3 [GCC03] zum Einsatz.

Zur Ermittlung der Zeiten wurde zuerst eine neue Ansicht eines Objektes gelernt. Anschließend wurde dieses Objekt in allen 57 Bildern gesucht. Die Zeit für das Finden

Abbildung 6.11: Beispielbilder aus der Teststichprobe für die Histogrammsuche. Die Objekte wurden jeweils (a) um 90° um die vertikale Achse rotiert, (b) verschoben und um die vertikale Achse rotiert und (c) verschoben und um beliebige Achsen rotiert.

eines Objektes in einem Bild, gemittelt über alle 57 Bilder, ist in Tabelle 6.2 angegeben. Um den Einfluß verschieden großer Objekte zu untersuchen, wurden die braune Tasse mit einer Breite von $W_O = 73$, einer Höhe von $H_O = 78$ und damit einer Größe von $W_O \cdot H_O = 4300$ und der Zauberwürfel mit einer Breite von $W_O = 37$, einer Höhe von $H_O = 39$ und damit einer Größe von $W_O \cdot H_O = 1017$ als Suchobjekte ausgewählt. Die Histogramme enthielten Farbmerkmale im L*u*v*-Farbraum mit unterschiedlichen Regionenaufteilungen. Die anderen im System eingesetzten Merkmale hätten die Zeiten nicht wesentlich beeinflußt, da für jedes Pixel der Suchbilder die Merkmale nur einmal berechnet werden müssen. Ist die Komplexität der Merkmalsberechnung nicht zu hoch, wie dies bei allen hier verwendeten Merkmalen der Fall ist, ist die Histogrammsuche der dominierende Teil in der Komplexitätsbetrachtung.

Untersucht wurden weiterhin verschiedene in Abschnitt 6.2.4 vorgestellte Berechnungsarten der Histogrammsuche. “Normal” gibt den auf Seite 101 vorgestellten naiven Ansatz der nicht beschleunigten vollständigen Suche an. “Gleitendes Fenster” ist die auf Seite 102 vorgestellte optimierte, aber dennoch vollständige Suche. “Aktive

Rechner	CPU		Speicher	SPEC CPU2000	
	Typ	[MHz]	[MB]	CINT2000	CFP2000
Alpha	Alpha 21264A	667	768	380	452
PC	Pentium® 4	2400	1024	865	872

Tabelle 6.1: Technische Daten der bei den Geschwindigkeitstests eingesetzten Rechner.

Regionen L*u*v*	4x4x4 [ms]		12x16x16 [ms]		20x28x28 [ms]	
	Tasse	Würfel	Tasse	Würfel	Tasse	Würfel
Alpha						
Normal	3956	1343	5569	2903	14.0s	13.9s
Gleitendes Fenster	136	106	146	116	189	162
Aktive Suche	346	376	359	343	446	459
Aktive Suche II	53	53	79	83	216	196
Fuzzyhistogramm	1466	1419	1506	1529	1606	1566
PC						
Normal	894	360	1322	916	3002	3078
Gleitendes Fenster	66	60	74	58	86	80
Aktive Suche	188	180	174	178	204	236
Aktive Suche II	36	36	56	58	102	106
Fuzzyhistogramm	674	670	732	740	872	872

Tabelle 6.2: Geschwindigkeit der Histogrammsuche für verschiedene Histogrammgrößen und verschiedene Such- und Vergleichsmethoden.

Suche" ist auf Seite 104 vorgestellt worden. Es ist die fokussierte Suche an Positionen mit hoher Ähnlichkeit. "Aktive Suche II" ist die aktive Suche mit den auf Seite 104 vorgestellten Optimierungen. "Fuzzyhistogramm" gibt zusätzlich die Zeit für die Berechnung von Fuzzyhistogrammen an. Bei deren Berechnung kam ein Ansatz des optimierten "Gleitenden Fensters" zum Einsatz.

Wie in Tabelle 6.2 zu sehen ist, sind abgesehen von der naiven Suche alle Verfahren weitestgehend unabhängig von der Größe des zu suchenden Objektes. Die nicht optimierte aktive Suche ist unter allen Testbedingungen langsamer als die Suche mit Hilfe eines gleitenden Fensters. Durch die auf Seite 104 vorgestellten Optimierungen läßt sich dies allerdings für nicht zu große Histogramme umkehren. Werden die Histogramme größer, werden die bei der aktiven Suche immer wieder neu durchzuführenden Histogrammvergleiche zu aufwendig im Vergleich zu der Ersparnis durch die Schwellwertberechnung. Die Benutzung von Fuzzyhistogrammen verlangsamt die Suche mit einem gleitenden Fenster um circa einen Faktor von zehn. Nur das naive Suchverfahren

ist in den meisten Fällen langsamer. Das Lernen einer neuen Objektansicht dauerte je nach Rechner und verwendeten Merkmalen circa 30 bis 60 Millisekunden. Da dies pro Objektanfrage maximal einmal durchgeführt werden muß, ist dieser Zeitbedarf zu klein, als daß er im Vergleich zu der Suche nach Objekten relevant wäre.

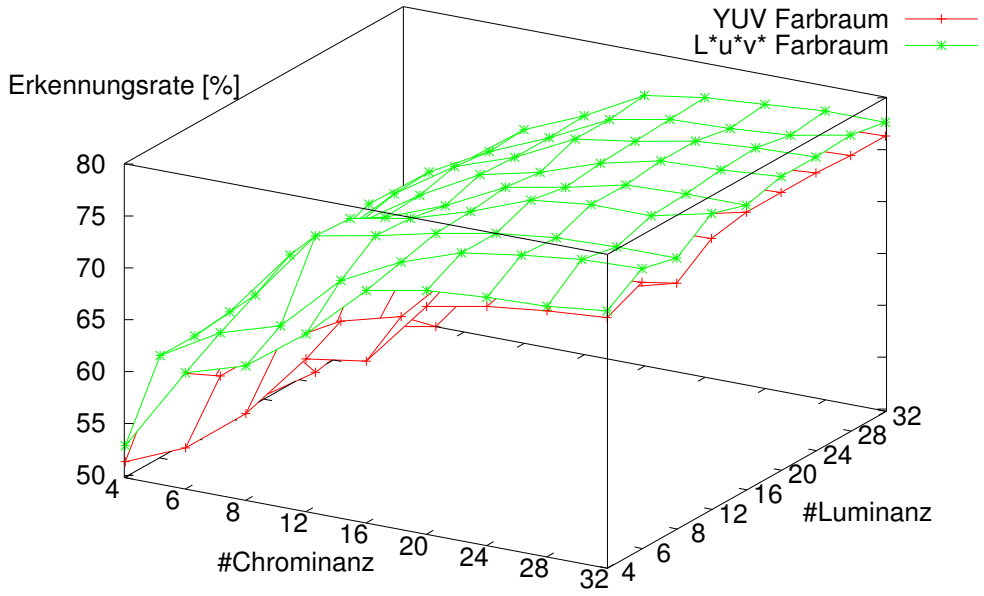
6.3.3 Erkennungsraten bei der Histogrammsuche

Bei allen durchgeführten Experimenten wurde für jede Parametereinstellung zuerst eine neue Ansicht eines Objektes gelernt. Anschließend wurde dieses Objekt in allen 57 Bildern gesucht, wobei die pro Bild am besten bewertete Position in Form einer Bildkoordinate ausgegeben wurde. Die generierte Ansicht des Objektes wurde danach wieder gelöscht. Für die Objektsuche stand damit immer nur eine Ansicht zur Verfügung. Dieser Vorgang wurde für alle acht Objekte wiederholt. Ein Objekt wurde als korrekt erkannt bewertet, falls die ausgegebene Koordinate nicht mehr als 24 Pixel nach der euklidischen Norm vom korrekten Schwerpunkt des Objektes abwich. Einerseits werden damit leichte durch Rauschen verursachte Ungenauigkeiten toleriert. Insbesondere verändert sich die Größe eines Objektes im aufgenommenen Bild aber auch mit der Position auf dem Tisch, womit auch der Vergleich eines kleinen Objektes mit einem beliebigen Teil eines größeren Objektes möglich sein muß. Aus der Anzahl n der korrekt erkannten Objekte wurde die in den Abbildungen angegebene Erkennungsrate E durch $E = n/(57 \cdot 8)$ bestimmt.

Ergebnisse mit Farbhistogrammen

Abbildung 6.12 zeigt die ermittelten Ergebnisse beim Einsatz von normalen Histogrammen mit farbbasierten Merkmalen. Die Grafik vergleicht einerseits die Erkennungsraten beim Einsatz von YUV-Farbmerkmalen und L*u*v*-Farbmerkmalen und andererseits die Änderung der Raten bei unterschiedlicher Regionenaufteilung der Histogramme in den einzelnen Dimensionen. “# Luminanz” gibt die Anzahl der Regionen in der Helligkeitskomponente (Y oder L*) und “# Chrominanz” die Regionenanzahl in den Farbkomponenten (U und V oder u* und v*) an.

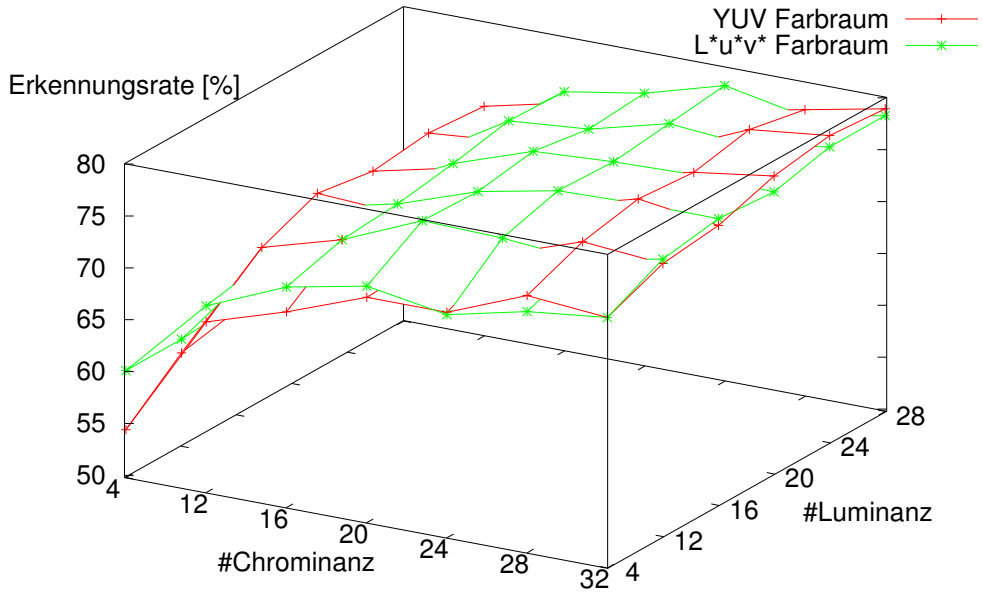
In allen Fällen erzielten die perzeptuell linearen L*u*v*-Merkmale eine bessere Erkennungsrate als die YUV-Merkmale, was auf Grund der immer äquidistanten Einteilung der Histogramme der Erwartung entspricht. Bei der Erhöhung der Anzahl der Histogrammregionen für die Luminanz steigt die Erkennungsrate langsamer als bei der Erhöhung der Anzahl der Chrominanzregionen. Ab circa 20 Farbregionen und 12 Helligkeitsregionen stellt sich eine Sättigung der Erkennungsrate ein. Die maximal erzielte Erkennungsrate war mit 78% für die mit jeweils einer Ansicht sehr geringe Trainingsmenge erstaunlich hoch. Dies zeigt die große Robustheit der Farbhistogramme gegenüber Größen- und Rotationsänderungen und insbesondere auch partiellen Verdeckungen. Partielle Verdeckungen kamen in den Experimenten durch nicht in der Kameraebene liegende Objektrotationen zustande.



	4x4x4	32x4x4	4x12x12	12x20x20	32x32x32	max	∅
E_{YUV}	51.3	57.5	64.5	75.0	76.3	76.5	68.0
E_{L*u*v*}	52.9	62.7	66.9	76.3	77.6	78.3	71.8

Abbildung 6.12: Erkennungsraten bei der Histogrammsuche mit normalen Farbhistogrammen mit variierender Anzahl von Histogrammregionen in den einzelnen Farbkanälen. In der Tabelle sind einige ausgewählte Punkte der Grafik angegeben.

Für die Ergebnisse in Abbildung 6.13 kamen Fuzzyhistogramme mit Farbmerkmalen zum Einsatz. Die benutzten Merkmale und die variierten Parameter waren ansonsten identisch zum im letzten Absatz beschriebenen Experiment. In diesem Fall gibt es keinen klaren Vor- oder Nachteil von L*u*v*-Merkmalen gegenüber YUV-Merkmalen. Durch die kontinuierliche Zugehörigkeitsfunktion von Merkmalen zu Regionen bei Fuzzyhistogrammen treten falsche Gewichtungen einzelner Bereiche des Merkmalsraums durch deren nicht lineare Verteilung nicht verstärkt auf. Die Histogramme werden damit robuster gegenüber den ausgewählten Merkmalen. Andererseits ist allerdings die Erkennungsrate im Vergleich zum normalen Histogramm mit L*u*v*-Merkmalen weder im Durchschnitt noch im Maximum signifikant höher. Dies ist ein Hinweis darauf, daß das auf Seite 94 beschriebene Rauschen der Kamera zur Gewinnung stabiler Merkmale genügt.



	4x4x4	28x4x4	4x12x12	12x20x20	28x32x32	max	∅
E_{YUV}	54.4	61.2	66.2	71.7	78.9	79.4	72.6
E_{L*u*v*}	60.1	60.7	67.8	75.9	78.3	78.3	72.0

Abbildung 6.13: Erkennungsraten bei der Histogrammsuche mit fuzzybasierten Farbhistogrammen mit variierender Anzahl von Histogrammregionen in den einzelnen Farbkanälen. In der Tabelle sind einige ausgewählte Punkte der Grafik angegeben.

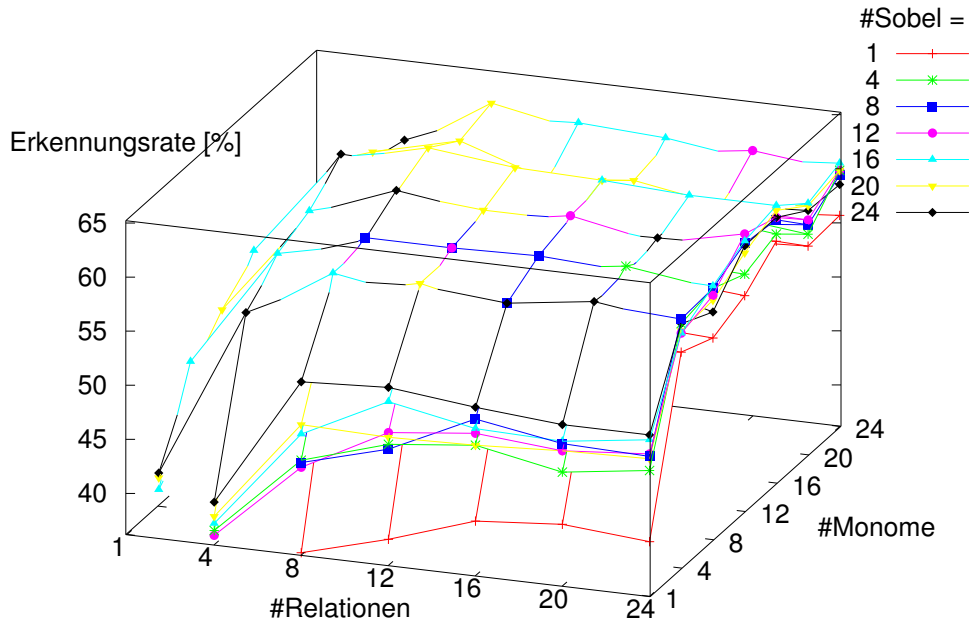
Ergebnisse mit strukturbasierten Histogrammen

Neben farbbasierten Merkmalen wurden auch Erkennungsraten mit strukturbasierten Merkmalen nach den Gleichungen 6.29, 6.30 und 6.33 aus Abschnitt 6.2.5 ermittelt. Mit den in diesem Abschnitt genannten Einschränkungen bleiben für Monome und relationale Merkmale drei freie Parameter M_1, M_2 und R_1:

$$f_{Monom}(I) = \sqrt{I_{M_1,0} \cdot I_{0,M_2}} \qquad f_{Rel}(I) = \mathrm{rel}(I_{R_1,0} - I_{0,0}) \tag{6.34}$$

Zusätzlich kann wie bei den Farbmerkmalen die Regioneneinteilung in den verschiedenen Dimensionen frei gewählt werden. Abbildung 6.14 zeigt die ermittelten Ergebnisse unter Variation der Regionenanzahl. Die restlichen Parameter waren in diesem Experiment auf $M_1 = 4$, $M_2 = 8$ und $R_1 = 4$ gesetzt.

Die Hinzunahme jedes der drei Merkmale verbessert jeweils signifikant die erzielten Erkennungsraten. Sind alle Merkmale mit mindestens vier Histogrammregionen vertreten, wird relativ schnell eine Sättigung der Erkennungsrate erreicht. Ist zusätzlich

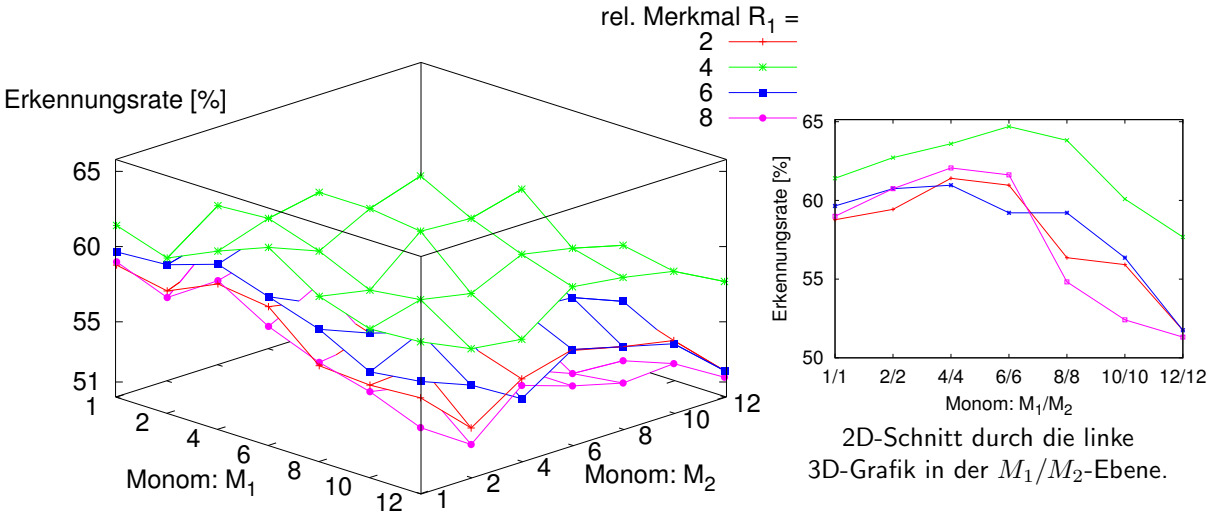


SxMxR	4x4x4	4x4x8	16x4x8	4x16x8	4x4x24	24x24x24	max ≙ 20x16x8	∅
E_{Textur}	52.6	57.2	59.6	62.1	58.8	58.6	63.4	54.4

Abbildung 6.14: Erkennungsraten bei der Histogrammsuche mit normalen Texturhistogrammen mit variierender Anzahl von Histogrammregionen in den einzelnen Dimensionen und festen Parametern $M_1 = 4$, $M_2 = 8$, $R_1 = 4$. Die Tabelle gibt einige ausgewählte Punkte der Grafik an.

das relationale Merkmal in mindestens acht Regionen aufgeteilt, fällt die Erkennungsrate nicht mehr unter 56.6% und steigt maximal bis auf 63.4% bei einer Aufteilung von 20x16x8 an. Dabei gibt 20 die Anzahl Regionen für den Sobel, 16 die Anzahl für die Monome und 8 die Anzahl für das relationale Merkmal an. Die Aufteilung 16x16x8 erzielt mit einem Ergebnis von 63.2% ein nahezu identisches Ergebnis.

Den Einfluß der drei weiteren Parameter M_1, M_2 und R_1 auf die Erkennungsraten zeigt Abbildung 6.15. Dieses Experiment wurde mit einer Regionenaufteilung von 16x16x8 durchgeführt. Zwei weitere hier nicht weiter ausgeführte Experimente mit unterschiedlicher Regionenaufteilung aus dem Sättigungsbereich des letzten Experimentes brachten vergleichbare Ergebnisse. Die 2D-Grafik zeigt zur Verdeutlichung einen Schnitt durch die 3D-Grafik, der durch die Diagonale der M_1/M_2-Ebene verläuft. Die besten Erkennungsraten erzielten in allen Fällen Histogramme berechnet mit $R_1 = 4$. Ein weiteres Kriterium für optimale Ergebnisse war in allen Fällen ein Wert nahe oder auf der M_1/M_2-Diagonalen im Bereich 2/2 bis 6/6. Das Optimum wurde bei



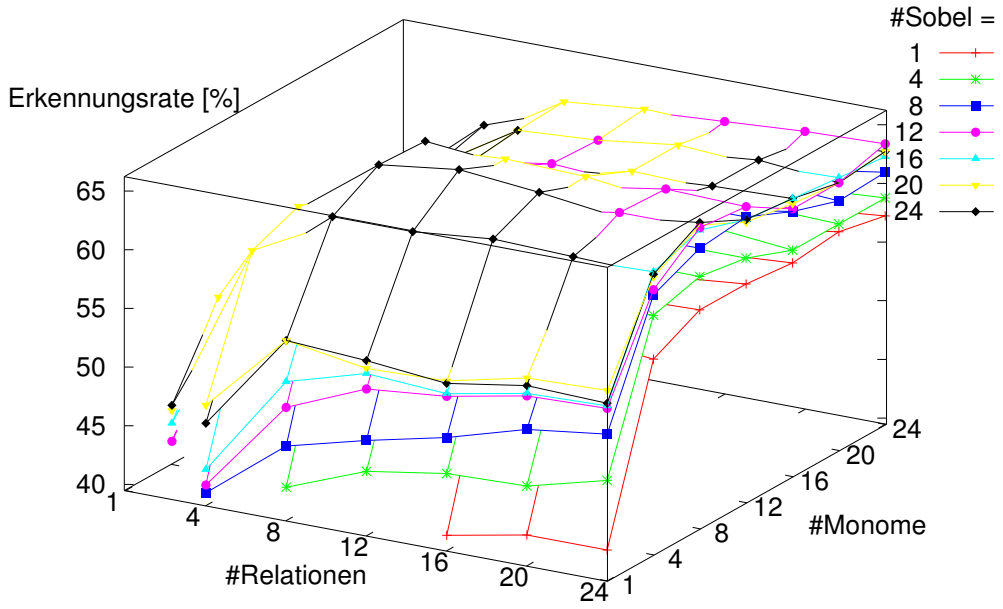
$R_1/M_1/M_2$	8/12/12	4/12/12	2/1/1	4/1/1	8/1/12	4/1/12	max ≙ 4/6/6	∅
E_{Textur}	51.3	57.7	58.8	61.4	54.4	60.1	64.7	58.1

Abbildung 6.15: Erkennungsraten bei der Histogrammsuche mit normalen Texturhistogrammen mit variierenden Parametern M_1, M_2 und R_1 und einer festen Regionenanzahl von 16x16x8. In der Tabelle sind einige ausgewählte Punkte der Grafik angegeben.

einer Parameterwahl von $R_1 = 4$ und $M_1 = M_2 = 6$ mit 64.7% erreicht. Die Ergebnisse der Strukturmerkmale sind insgesamt im Vergleich zu den Farbmerkmalen merklich schlechter. Da die Strukturmerkmale allerdings nur auf dem L*-Farbkanal aufsetzen, haben sie auch weniger Informationen zur Verfügung. Dafür sind sie zum Teil allerdings stabiler gegenüber Beleuchtungsänderungen, was in den Experimenten allerdings nicht weiter untersucht wurde.

Ergebnisse mit strukturbasierten Fuzzyhistogrammen

Die im letzten Abschnitt beschriebenen Versuche mit Histogrammen mit strukturbasierten Merkmalen wurden auch mit Fuzzyhistogrammen durchgeführt. Untersucht wurde der Einfluß der gleichen Parameter auf die Erkennungsrate. Abbildung 6.16 zeigt die Ergebnisse unter Variation der Regionenanzahl. Die restlichen Parameter waren in diesem Experiment auf $M_1 = 4$, $M_2 = 8$ und $R_1 = 4$ gesetzt. Ähnlich wie bei normalen Histogrammen verbessert auch hier jedes zusätzliche Merkmal signifikant die Erkennungsrate. Das Verhalten in Bezug auf die Erkennungsleistung der Monome ist allerdings unterschiedlich. Ab circa 16 Regionen gibt es einen deutlichen Abfall der Erkennungsrate, falls zusätzlich sowohl der Sobel als auch das relationale

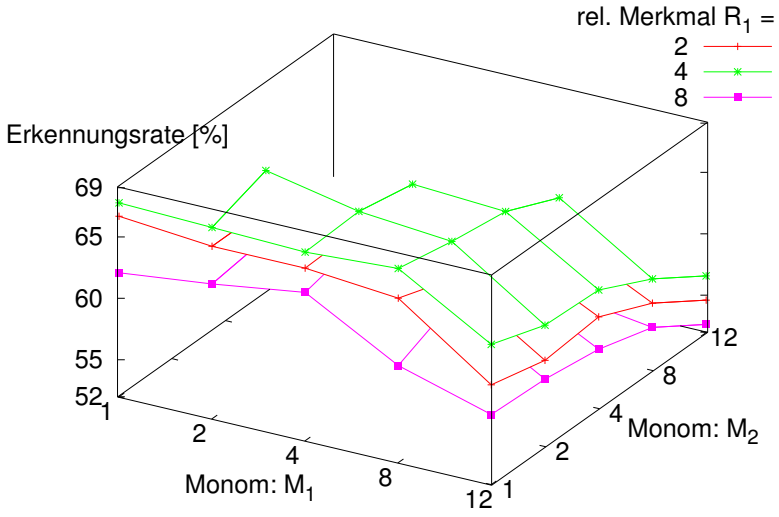


SxMxR	4x4x4	20x4x4	16x4x24	16x24x4	12x24x24	max ≙ 24x8x12	∅
E_{Textur}	51.8	59.0	63.6	58.1	63.4	66.2	56.2

Abbildung 6.16: Erkennungsraten bei der Histogrammsuche mit strukturbasierten Fuzzyhistogrammen mit variierender Anzahl von Histogrammregionen in den einzelnen Dimensionen und festen Parametern $M_1 = 4$, $M_2 = 8$, $R_1 = 4$. Die Tabelle gibt einige ausgewählte Punkte der Grafik an.

Merkmal verwendet wird. Eine Anzahl von 8 oder 12 Regionen bringt konsistent über verschiedene Einteilungen der anderen beiden Merkmale hinaus die besten Ergebnisse.

Den Einfluß der restlichen drei Parameter M_1, M_2 und R_1 auf die Erkennungsergebnisse zeigt Abbildung 6.17. Hier wurde eine Regionenaufteilung von 24x8x12 benutzt, die beste Einteilung des letzten Experimentes. Wie bei den normalen Histogrammen erzielten auch hier Histogramme mit $R_1 = 4$ die besten Ergebnisse. Für gute Ergebnisse ebenfalls wichtig ist mindestens ein eher kleiner Wert für M_1 oder M_2. Im Gegensatz zu den Histogrammen mit Farbmerkmalen sind Fuzzyhistogramme mit Strukturmerkmalen beim Vergleich der jeweils besten Ergebnisse signifikant besser als normale Histogramme. Ein Grund hierfür dürfte die nicht lineare Verteilung der Merkmalswerte über den kompletten Wertebereich sein. Durch die Häufung der Werte in einem kleineren Bereich wird das Histogramm gegenüber Rauschen empfindlicher. Dem entgegen zu wirken hilft das Fuzzyhistogramm. Im direkten Vergleich mit



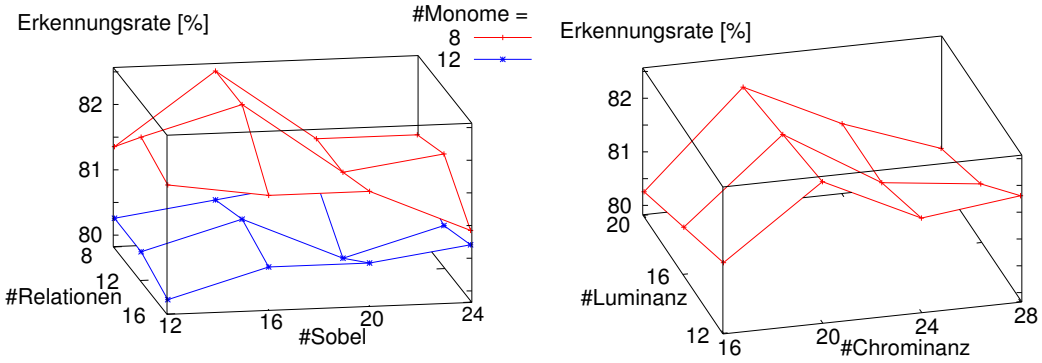
$R_1/M_1/M_2$	8/12/12	4/12/12	8/1/1	4/1/1	8/1/12	4/1/12	max $\widehat{=}$ 4/2/2	∅
E_{Textur}	52.6	56.6	62.1	67.8	57.7	63.4	69.0	62.6

Abbildung 6.17: Erkennungsraten bei der Histogrammsuche mit strukturbasierten Fuzzyhistogrammen mit variierenden Parametern M_1, M_2 und R_1 und einer festen Regionenanzahl von 24x8x12. In der Tabelle sind einige ausgewählte Punkte der Grafik angegeben.

den Farbmerkmalen sind aber auch hier die Strukturmerkmale wie bei den normalen Histogrammen merklich schlechter.

Ergebnisse mit kombinierten Farb- und Texturhistogrammen

Bisher wurden die Erkennungsraten bei der Objektsuche mit Histogrammen unter der Verwendung eines Histogramms vorgestellt. An Merkmalen kamen dabei entweder Farb- oder Strukturmerkmale zum Einsatz. Nun werden die Ergebnisse bei gleichzeitigem Einsatz beider Merkmalstypen betrachtet. Dafür wurden sowohl farb- als auch strukturbasierte Hypothesen wie in den bisherigen Experimenten getrennt bestimmt und anschließend wie in Abschnitt 7.1 vorgestellt kombiniert. Dabei werden die von den Einzelerkennern generierten Wahrscheinlichkeitskarten addiert. Dies entspricht der Kombination zweier getrennter Erkenner. Abbildung 6.18 zeigt die Ergebnisse unter Variation der Anzahl der Histogrammregionen. Da es hierbei fünf freie Parameter gab, die beiden Farbkanäle u* und v* wurden jeweils identisch aufgeteilt, lassen sich die Ergebnisse nicht direkt visualisieren. Die beiden Graphen zeigen daher in jedem Punkt das beste Ergebnis, welches unter Festhaltung der Parameter des Punktes bei gleichzeitiger Variation der restlichen Parameter erzielt werden konnte.

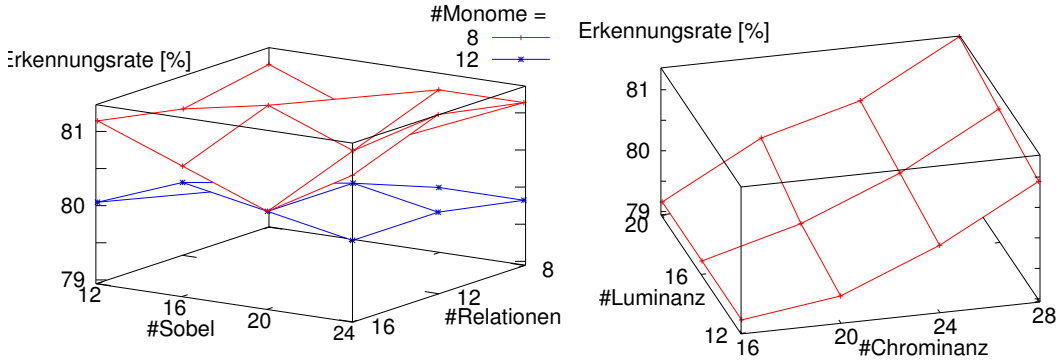


SxMxR LxUxV	12x12x16 20x16x16	12x12x16 16x16x16	12x8x16 16x24x24	12x8x16 12x20x20	16x8x8 20x16x16	max ≙ 16x8x8 12x20x20	∅
E_{Komb}	78.3	80.0	79.6	81.8	79.2	82.5	80.2

Abbildung 6.18: Erkennungsraten bei der Histogrammsuche mit sowohl L*u*v*-Farbmerkmalen als auch Strukturmerkmalen unter Variation der Histogrammregionen. Die restlichen Parameter waren fest auf $M_1 = M_2 = 2$, $R_1 = 4$ gesetzt. In der Tabelle sind einige ausgewählte Einzelergebnisse angegeben.

Bei dem Experiment aus Abbildung 6.18 waren die restlichen Parameter auf $M_1 = M_2 = 2$ und $R_1 = 4$ gesetzt. Die Erkennungsraten waren insgesamt über den getesteten Parameterbereich mit Werten zwischen 78.3% bis 82.5% sehr stabil. Bei allen Parametereinstellungen konnten die Erkennungsraten durch die Kombination im Vergleich zu den einzelnen Farb- und Texturhistogrammen gesteigert werden. Die im Strukturexperiment aus Abbildung 6.15 leicht besseren Parameter $M_1 = M_2 = 6$ und $R_1 = 4$ haben in einem hier nicht weiter dargestellten Experiment ähnliche, aber insgesamt leicht schlechtere Ergebnisse erbracht. Die durchschnittliche Erkennungsrate war um circa 0.4% geringer, was allerdings nicht signifikant ist.

Abbildung 6.19 zeigt die Ergebnisse bei kombiniertem Einsatz von Farb- und Strukturmerkmalen bei Benutzung von Fuzzyhistogrammen. Die variierten Parameter und die Darstellung der Ergebnisse sind hier identisch zum Experiment mit normalen Histogrammen aus Abbildung 6.18. Die hier nicht variierten Parameter waren auf das Optimum des entsprechenden Einzelexperimentes aus Abbildung 6.17 gesetzt: $M_1 = M_2 = 2$ und $R_1 = 4$. Wie bei den normalen Histogrammen konnten auch bei den Fuzzyhistogrammen die Erkennungsraten durch die Kombination der Merkmale immer gesteigert werden. Die Steigerung ist allerdings im Vergleich zum Einsatz von alleinigen Farbmerkmalen relativ gering. Im Vergleich zu den normalen Histogrammen ist die Erkennungsrate mit den Fuzzyhistogrammen mit durchschnittlich 1.1% geringeren Werten etwas schlechter.



SxMxR LxUxV	12x12x8 12x16x16	12x12x8 20x28x28	12x8x8 12x20x20	12x8x8 20x28x28	24x8x12 12x16x16	max ≙ 24x8x12 20x28x28	∅
E_{Komb}	76.8	79.6	78.7	81.1	78.5	81.4	79.1

Abbildung 6.19: Erkennungsraten bei der Suche mit Fuzzyhistogrammen mit sowohl L*u*v*-Farbmerkmalen als auch Strukturmerkmalen unter Variation der Histogrammregionen. Die restlichen Parameter waren fest auf $M_1 = M_2 = 2$, $R_1 = 4$ gesetzt. In der Tabelle sind einige ausgewählte Einzelergebnisse angegeben.

6.3.4 Zusammenfassung der Ergebnisse

Die Ergebnisse dieses Abschnitts haben die Eignung der Histogrammsuche für die geforderte Aufgabe gezeigt. Trotz minimalen Trainingsmaterials, es wurde nur eine Ansicht pro Objekt benutzt, konnten Erkennungsraten von bis zu 82% erreicht werden. Bei Betrachtung der Erkennungsraten von Fuzzyhistogrammen lohnen sich diese im hier betrachteten Szenario nicht. Nur in den Tests mit strukturbasierten Merkmalen haben sie signifikant bessere Erkennungsraten erzielt. Bei Benutzung von Farbmerkmalen und bei kombiniertem Einsatz beider Merkmalsarten waren keine besseren Erkennungsraten erzielbar. Das den natürlichen Bildern inhärent anhaftende Rauschen erzeugt beispielsweise schon stabile Histogramme. Da Fuzzyhistogramme um zusätzlich circa einen Faktor 10 in der Berechnung langsamer sind als normale Histogramme, werden sie nicht weiter betrachtet.

Im direkten Vergleich konnten Farbhistogramme bessere Ergebnisse erzielen als die Texturhistogramme. Auf Grund der konstanten Beleuchtung war dies allerdings zu erwarten. Der kombinierte Einsatz von Texturhistogrammen und Farbhistogrammen hat in allen durchgeführten Tests nochmals bessere Ergebnisse erzielt als die entsprechenden Einzelhistogramme. Dies belegt noch einmal die in Abschnitt 6.1.4 dargestellte Notwendigkeit des gleichzeitigen Einsatzes unterschiedlicher Ansätze zur Objekterkennung.

Die durchgeführten Optimierungen bei der Histogrammsuche, sowohl bei der Berechnung mit einem gleitenden Fenster als auch bei der aktiven Suche, haben sich als effektiv erwiesen. Die Suche ist mit beiden Verfahren für ein interaktives Arbeiten mit dem System schnell genug. Bei nicht zu großen Histogrammen ist die aktive Suche schneller. Da hierbei allerdings keine vollständige Wahrscheinlichkeitskarte erstellt werden kann, kommt im folgenden im Falle mehrerer parallel eingesetzter Erkenner die Berechnung mit einem gleitenden Fenster zum Einsatz. Durch die dann vorhandenen vollständigen Wahrscheinlichkeitskarten kann eine effektivere Kombination der Objekthypothesen durchgeführt werden.

6.4 Regionenbasierter Merkmalsvergleich

In den letzten beiden Abschnitten wurde die Objektlokalisation mit Hilfe zweier Objekterkenner mit histogrammbasiertem Merkmalsvergleich vorgestellt. Dieser Ansatz hat allerdings, auch wenn er sehr gute Ergebnisse erzielt hat, den Nachteil, daß er weitestgehend Ortsinformationen ignoriert. Objekte, die sich nur durch die Position von Merkmalen innerhalb eines Objektes unterscheiden, können damit mit diesem Ansatz nicht unterschieden werden. In Übereinstimmung mit [Lei03] wurde daher ein anderer kontrastierender Ansatz zur Objektlokalisation und Identifikation untersucht.

[Kum98] stellt einen Ansatz zur Erkennung von acht verschiedenen vorgegebenen Objekten in Farbbildern vor. Die Objekte liegen zum Teil zusätzlich in unterschiedlichen Ausprägungen vor. Zur Merkmalsgenerierung segmentiert der Ansatz das Eingangsfarbbild mit Hilfe eines Polynomklassifikators in zwölf im betrachteten Szenario vorkommende dedizierte Farbklassen. Nach einer Glättung werden Bereiche identischer Farbe zu Regionen zusammengefaßt. Diese Regionen bilden die Grundlage für eine folgende Objekterkennung mit Hilfe eines semantischen Netzes. Das semantische Netz enthält sowohl strukturelles Wissen über die einzelnen Bestandteile der Objekte und deren Anordnung zueinander als auch Informationen über Farbe und Form der Bestandteile. Basierend auf einem Vergleich der im Netz hinterlegten Merkmale und Strukturinformationen mit den segmentierten Regionen werden die Regionen jeweils einem Objektbestandteil zugeordnet. Kann so ein vollständiges Objekt instantiiert werden, ist es im Bild erkannt worden.

Dieser eher konstruktivistisch denn statistisch motivierte Ansatz steht im deutlichen Kontrast zum histogrammbasierten Ansatz der letzten beiden Abschnitte. Allerdings ist der Ansatz auf ein Szenario mit bekannten vorgegebenen Objekten mit flächiger Struktur mit wenigen Farben festgelegt. Wichtig ist insbesondere auch, daß sowohl Gegenstände als auch Hintergründe keine anderen als die dem Polynomklassifikator bekannten Farben haben. Kommen andere Farben vor, werden diese trotzdem auf eine bekannte Farbe abgebildet, was zu einer fehlerhaften Regionensegmentierung führt. Aber insbesondere auf Fehler in der Segmentierung reagiert dieser Ansatz sehr empfindlich, da eine Region komplette Objektteile repräsentiert.

Um diesen Erkennungsansatz im Szenario dieser Arbeit einzusetzen, sind daher deutliche Änderungen des Verfahrens nötig. Einerseits muß der Ansatz deutlich robuster gegenüber unbekannten Farben in Form unbekannter Objekte und bisher unbekannten Hintergründen werden. Andererseits kann auf bekanntes Wissen über Objekte nicht zurückgegriffen werden. Die Objektstruktur und die Objektmerkmale müssen interaktiv gelernt werden. Dies betrifft sowohl die Anordnung einzelner Objektteile zueinander als auch die Eigenschaften der Ansicht der einzelnen Objektteile. Zur Steigerung der Robustheit der Segmentierung kommt ein anderes Segmentierungsverfahren, der Mean-Shift-Algorithmus [Com97], zum Einsatz. Statt eines semantischen Netzes mit integriertem prozeduralem Wissen für die Auswertung der Regionen einzusetzen, werden Nachbarschaftsgraphen aus den Regionen der zu lernenden Objekte und den Regionen einer segmentierten Szene aufgebaut. Durch einen fehlertoleranten Vergleich von Teilgraphen werden Objekte schließlich in der Szene lokalisiert.

6.4.1 Segmentierung von Bildern

Das Ziel der Segmentierung ist die Einteilung der Eingabedaten, hier Farbbilder, in getrennte Regionen, die ein vorgegebenes Homogenitätskriterium erfüllen. Von einer abstrakten Ebene aus betrachtet sollen die Regionen häufig einzelnen Objektteilen oder auch kompletten Objekten entsprechen. Auch im hier präsentierten System ist dies für die nachfolgende Objekterkennung erwünscht. Für die Segmentierung wurden im Laufe der Zeit verschiedenste Verfahren vorgeschlagen, angefangen von histogrammbasierten Verfahren über Clusteransätze bis zu Regionenwachstumsansätzen. [Luc01] gibt einen Überblick über verschiedene Segmentierungsverfahren.

Ein interessanter Ansatz zur Segmentierung von Bildern wird in [Com97] vorgestellt. Im Gegensatz zur Segmentierung mit einem Polynomklassifikator ist dies ein bildweit globaler Ansatz, der das Bild im L*u*v*-Farbraum clustert und im Ortsraum nachbearbeitet. Das Clustering basiert auf der Berechnung des *Mean-Shift-Vektors*, der zuerst von [Fuk75] hergeleitet wurde:

$$M_h(\mathbf{x}) = \frac{h^2}{D+2} \frac{\hat{\nabla} f(\mathbf{x})}{\hat{f}(\mathbf{x})} \tag{6.35}$$

Dabei geben f eine Dichtefunktion von D-dimensionalen Merkmalen $\mathbf{x}$ und h die Größe eines Fensters an, siehe hierzu auch die Gleichung 6.7 des Kernelschätzers auf Seite 93. Der Mean-Shift-Vektor zeigt immer in die Richtung der maximalen Steigung der Dichte und kann damit zur Lokalisation eines lokalen Maximums der Dichte genutzt werden.

Die Segmentierung mit Hilfe des Mean-Shift-Verfahrens unter Zuhilfenahme des Mean-Shift-Vektors erfolgt nun in mehreren Schritten:

1. Wähle mehrere zufällige Punkte im Bild und bestimme die mittlere Farbe in deren lokalen Umgebungen. Wähle den Punkt, dessen Farbe am häufigsten im kompletten Bild vorkommt.

2. Lege ein Fenster um die ausgewählte Farbe im Farbraum, berechne den Mean-Shift-Vektor und verschiebe das Fenster in Richtung des Vektors. Wiederhole dies so lange, bis der Betrag des Vektors unter einer Schwelle liegt und damit ein lokales Maximum erreicht ist.

3. Beseitige die Farben innerhalb des Fensters sowohl im Farbraum als auch im gesamten Ortsraum. Beseitige zusätzlich die im Ortsraum zu den ausgewählten Pixeln benachbarten Pixel, um Farbsäume an Kanten abzufangen.

   Wiederhole die ersten drei Schritte, bis die meisten Merkmale vergeben sind. Damit ist eine initiale Farbpalette bestimmt, die in den folgenden Schritten sowohl mit Hilfe des Farbraums als auch des Ortsraums optimiert wird.

4. Behalte von allen bestimmten Farben diejenigen, die im Ortsraum mindestens eine zusammenhängende Region einer Minimalgröße bilden.

5. Ordne alle Farben, die innerhalb eines Fensters der ausgewählten Farben liegen, ihren Zentren zu. Vergrößere das Fenster und ordne die neuen Pixel dann zu, wenn sie im Ortsraum Kontakt zu einem zugeordneten Pixel haben. Die Mittelwerte der neuen Fenster bilden die finale Farbpalette.

6. Säubere die Segmentierung durch eine Glättung und durch eine Zuordnung von Regionen, die im Ortsraum zu klein sind, zu benachbarten Regionen.

Durch diese verschachtelte Verarbeitung sowohl im Farbraum als auch im Ortsraum wird eine vergleichsweise stabile und saubere Segmentierung erreicht. Bei texturierten Objekten kann dieser Ansatz allerdings keine guten Ergebnisse liefern, da sein Homogenitätskriterium gerade auf den euklidischen Abstand im Farbraum basiert und damit eine vorhandene Textur nicht berücksichtigt. Abbildung 6.20 zeigt Beispiele der Segmentierung für zwei verschiedene Bilder bei jeweils zwei unterschiedlichen Parametereinstellungen für den Radius des Suchfensters und die Regionengrößen. Einige der Objekte, beispielsweise die Federmappe und der Zauberwürfel, werden bei den unterschiedlichen Einstellungen gut segmentiert. Bei komplizierter texturierten Objekten wie der Diskettenschachtel kommt es allerdings zu Fehlern. Da die Regionen als Vorstufe für eine weitere Verarbeitung dienen, ist dies ein Hinweis auf die Objekte, die dieser Erkenneransatz präferiert.

6.4.2 Objektvergleich mit Graphen

Nach der im letzten Abschnitt durchgeführten Segmentierung liegen nun Regionen vor, die einem farbbasierten Homogenitätskriterium genügen. Dies ist allerdings noch keine Segmentierung von Objekten. Viele Objekte, sowohl eher großflächig strukturierte als auch eher texturierte Objekte, bestehen nicht aus einzelnen Regionen. Ein

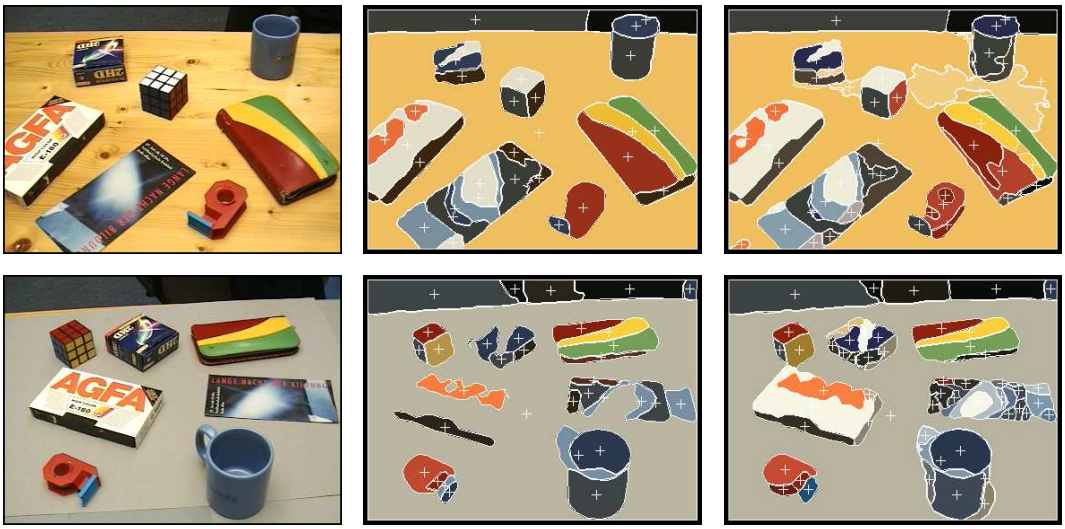


Abbildung 6.20: Beispiele für die Segmentierung eines Bildes mit dem Mean-Shift-Algorithmus bei grober und feiner Einstellung der verschiedenen Parameter.

Beispiel ist die Federmappe aus Abbildung 6.20, die aus mindestens vier Regionen besteht. Diese Regionen haben jeweils sowohl für sich betrachtet als auch in Kombination mit den restlichen Regionen charakteristische Eigenschaften, die sich beispielsweise auch unter Rotation in der Kameraebene nicht ändern und unter anderen Rotationen sehr robust sind. Eine entscheidende Eigenschaft der Regionenmenge ist die Lage der einzelnen Regionen zueinander. Zur Darstellung dieser Abhängigkeiten eignet sich ein Graph, der die Nachbarschaft der Regionen über Kanten ausdrückt. Die komplette Szene läßt sich entsprechend der Objekte ebenfalls durch einen Graphen darstellen. Das Lokalisieren der Objekte wird damit zu einer Suche eines Teilgraphen im Graph der Szene.

Attributierte Graphen sind im Bereich der Mustererkennung ein gängiges Mittel zur Repräsentation von Objektmerkmalen und deren Beziehungen [Bun00]. Um Graphen für die Objekterkennung einsetzen zu können, ist zuerst eine formale Definition von Graphen und deren Vergleich nötig. Die nun folgende Darstellung hält sich an die Notation von [Mes96]. Ein *attributierter Graph* G ist ein 4-Tupel (V, E, μ, ν), wobei

- V eine Menge von Knoten,
- $E \subseteq V \times V$ eine Menge von Kanten,
- $\mu : V \rightarrow A^n$ eine Funktion zum Zuteilen von Attributen zu Knoten und
- $\nu : E \rightarrow A^m$ eine Funktion zum Zuteilen von Attributen zu Kanten ist.

Ein *Teilgraph* T von $G = (V, E, \mu, \nu)$ ist ein 4-Tupel (V_t, E_t, μ_t, ν_t) mit

- $V_t \subseteq V$
- $E_t = E \cap (V_t \times V_t)$
- $\mu_t(v) = \begin{cases} \mu(v) & \text{falls } v \in V_t \\ \text{undefiniert} & \text{sonst} \end{cases}$
- $\nu_t(e) = \begin{cases} \nu(e) & \text{falls } e \in E_t \\ \text{undefiniert} & \text{sonst} \end{cases}$.

Eine bijektive Funktion $f : V \to V'$ ist ein *Graphisomorphismus* von $G = (V, E, \mu, \nu)$ zu $G' = (V', E', \mu', \nu')$, falls

- $\mu(v) = \mu'(f(v))$ für alle $v \in V$ und
- $\forall e = (v_1, v_2) \in E : \exists e' = (f(v_1), f(v_2)) \in E' : \nu(e) = \nu'(e')$ und
  $\forall e' = (v'_1, v'_2) \in E' : \exists e = (f^{-1}(v'_1), f^{-1}(v'_2)) \in E : \nu(e) = \nu'(e')$.

Eine injektive Funktion $f : V \to V'$ von G zu G' ist schließlich ein *Teilgraphisomorphismus*, falls ein Teilgraph T von G' existiert, so daß f ein Graphisomorphismus von G zu T ist.

Im Bereich der Mustererkennung kann man nicht davon ausgehen, Objektmerkmale ohne Fehler aus einem Eingabebild zu bestimmen. Daher müssen beim Vergleich von Graphen auch Abweichungen zwischen den beiden Graphen erlaubt sein. Eine häufig eingesetzte Methode zur Bewertung dieser Abweichungen ist die Definition eines Abstandsmaßes zwischen Graphen basierend auf einer *Edit-Distanz*. Die Edit-Distanz erlaubt verschiedene mit Kosten versehene Operationen auf einem Modellgraphen, um ihn so lange zu modifizieren, bis ein Teilgraphisomorphismus zu einem Eingabegraphen existiert. Die mit den minimalen Kosten versehene Folge von Edit-Operationen, die zu einem Teilgraphisomorphismus führt, spezifiziert die Edit-Distanz zwischen dem Modellgraphen und dem Eingabegraphen. An Edit-Operationen sind hierbei erlaubt:

- Ersetzen der Attribute eines Knotens durch andere Attribute,
- Ersetzen der Attribute einer Kante durch andere Attribute,
- Löschen eines Knotens,
- Löschen einer Kante und
- Einfügen einer Kante.

Das Einfügen eines Knotens muß nicht betrachtet werden, da ein gültiger Teilgraphisomorphismus gesucht wird, wodurch im Modellgraphen nicht vorkommende Knoten schon ignoriert werden.

Im Rahmen dieser Arbeit sollte kein Algorithmus zum Vergleichen von Graphen entwickelt werden. Statt dessen sollte für den hier betrachteten Erkenner ein solcher Algorithmus angewendet werden. Dafür kommt das von Messmer und Bunke entwickelte GUB, das *graph matching toolkit of the University of Bern*, zum Einsatz [Mes96, Mes98]. GUB stellt verschiedene effiziente Algorithmen zum fehlertoleranten Vergleichen von einer Menge von Modellgraphen mit einem Eingabegraphen zur Verfügung. Zur Bestimmung der Kosten der verschiedenen Edit-Operationen können Funktionen angegeben werden, die eine an die momentane Situation angepaßte Spezifikation der Kosten erlauben. Da der vollständige Quelltext verfügbar ist, können spezielle Anpassungen vorgenommen werden.

6.4.3 Lokalisation von Objekten

Wie im letzten Abschnitt angesprochen, werden Objekte und Szenen als aus Regionen bestehenden Nachbarschaftsgraphen dargestellt, wobei Knoten Regionen repräsentieren und Kanten Nachbarschaften zwischen den Regionen. Während des Lernens einer neuen Ansicht eines Objektes wird durch Herausnehmen des Objektes eine Region in einem Bild bestimmt, die ein zu lernendes Objekt enthält (siehe Abschnitt 3.3). Die Farbregionen innerhalb dieser Region bilden die Grundlage für einen Graphen, der die zu erstellende Ansicht repräsentiert und als neue Objektansicht in der Objektdatenbank abgelegt wird. Soll ein Objekt in einer Szene lokalisiert oder identifiziert werden, wird aus der vollständigen segmentierten Szene ein Graph aufgebaut, der mit den Objektgraphen aus der Datenbank verglichen wird.

Da der Vergleich von Teilgraphen eine hohe Komplexität hat – er ist NP-vollständig – wird versucht, die Komplexität der Graphen möglichst klein zu halten. Dazu wird einerseits bei der Mean-Shift-Segmentierung mit einer Untersegmentierung die Anzahl der Regionen klein gehalten. Andererseits wird nur dann im Graph eine Kante zwischen zwei Regionen gezogen, wenn die beiden Regionen im Bild signifikant benachbart sind. Als jeweils signifikant benachbart werden zwei Regionen angesehen, falls sie im Bild eine gemeinsame Kante haben, die zusätzlich im Verhältnis zum Umfang mindestens einer der beiden Regionen eine Länge von mindestens 5% haben muß.

Mit einem weiteren Maß wird versucht, die Stabilität beziehungsweise die Zugehörigkeit einer Farbregion zu einer neu gelernten Ansicht eines Objektes zu bewerten. Durch das Lernverfahren, die Bestimmung einer Region R_L durch Herausnehmen des zu lernenden Objektes, sind insbesondere die Ränder der Region der Gefahr einer Fehlklassifikation unterlegen. Einerseits können es durch das Objekt geworfene Schatten sein, die eigentlich zum Hintergrund gehören, vom Lernverfahren aber als zum Objekt gehörig klassifiziert wurden. Andererseits können es nur ver-

zerrt sichtbare Randbereiche des Objektes sein. Daher versucht dieses Konfidenzmaß, insbesonders aus der Segmentierung hervorgehende Farbklassen schlechter zu bewerten, die ausschließlich am Rand der bestimmten Region liegen. Eine Region, die von der Segmentierung in die Farbklasse C eingeordnet wurde, bekommt einen geringen Konfidenzwert, falls

- alle Pixel der selben Klasse C sich nahe am Rand der Region R_L befinden,
- nur wenige Pixel anderer Farbklassen sich zwischen beliebigen Pixeln der Farbklasse C und dem Rand der Region R_L befinden,
- andere Regionen, die einer Farbklasse ungleich C angehören, entsprechend dieser Bewertungen gut bewertet sind und
- die Region klein ist.

Für die Knoten und Kanten werden verschiedene charakteristische Maße berechnet und im Graphen als Attribute abgelegt. Diese Attribute bilden die Grundlage für den fehlertoleranten Vergleich der Graphen. Die Attribute der Knoten sind:

1. A_L, A_u, A_v: Die mittlere Farbe der dem Knoten zugeordneten Region im weitestgehend perzeptuell linearen L*u*v*-Farbraum.
2. A_S: Die Größe der Region in Pixeln.
3. A_E: Die Exzentrizität der Region, berechnet mit Hilfe der Zentralmomente μ_{pq} [Gon02, Kapitel 11.3.4] als $\frac{(\mu_{20}-\mu_{02})^2+4\mu_{11}^2}{(\mu_{20}+\mu_{02})^2}$.
4. A_C: Die Kompaktheit der Region, berechnet als $\frac{16 \cdot \text{Regionenfläche}}{\text{Regionenumfang}^2}$.
5. A_K: Der im letzten Absatz eingeführte Konfidenzwert.
6. A_A: Der Wert der auf Seite 73 im Abschnitt 5.4.2 berechneten Aufmerksamkeitskarte an der Position des Schwerpunktes der Region. Dieser Wert spezifiziert die Wahrscheinlichkeit dafür, daß der jeweilige Punkt im Fokus einer Zeigegeste lag.

Die Attribute für Kanten ergeben sich aus der Lage und den Attributen der beiden Regionen, die die Kante verbindet:

1. A_{S_1}, A_{S_2}: Die Größe der beiden Regionen in Pixeln.
2. A_D: Falls sich die Regionen berühren, der Mittelwert aus der Kantenlänge zwischen den beiden Regionen gewichtet mit den Umfängen beider Regionen. Falls sich die Regionen nicht berühren, der Abstand der zwei Regionen in Pixeln. Dieser Wert wird nicht nur für durch Kanten verbundene Regionen berechnet, sondern für alle Regionenpaare.

3. A_K: Das Produkt aus den oben eingeführten Konfidenzen beider Regionen.

Für den fehlertoleranten Vergleich eines Szenegraphen mit einer Menge von Objektgraphen ist nun noch die Spezifikation der Kosten für die im letzten Abschnitt eingeführten Edit-Operationen nötig. Dies ist, vergleichbar zu der Auswahl der benutzten Attribute, problemabhängig. Die Kosten für das Löschen einer Kante im Objektgraphen werden auf einen sehr geringen Wert, hier 0.01, gesetzt, falls der Konfidenzwert A_K der Kante sehr gering ist. Ansonsten wird der Abstand der Regionen im Szenegraph betrachtet, auf Grund derer eine Kante im Objektgraphen gelöscht werden soll. Die Kosten werden auf einen hierzu proportionalen Wert gesetzt. Sind die Szeneregionen weit voneinander entfernt, sind die beiden Graphen nicht ähnlich. Die Kosten für das Löschen der Kante müssen dementsprechend hoch sein. Sind die Regionen im Szenegraph dagegen örtlich in nur geringer Entfernung, ist dies vergleichbar zu der Situation im Objektgraphen und die Kosten können dementsprechend gering sein.

Beim Löschen eines Knotens werden nur Attribute des lokalen Knotens betrachtet. Ist der Konfidenzwert A_K des Knotens gering, werden die Kosten wie beim Löschen einer Kante auf 0.01 gesetzt. Ansonsten wird ein zu der Regionengröße A_S und zu dem Wert der Aufmerksamkeitskarte A_A der Gestenerkennung proportionaler Wert genommen. Ist eine Region im Fokus einer Geste, ist sie besonders wichtig und die Kosten für das Löschen müssen dementsprechend hoch sein. Genauso sind größere Regionen wichtiger als kleinere. Um hierbei skalierungsinvariant zu werden, wird die Größe mit dem Median aller Regionengrößen des Graphen normiert.

Die Kosten für das Einfügen einer Kante sind vergleichbar zu den Kosten für das Löschen einer Kante. Bei geringer Konfidenz der Kante sind sie 0.1. Dieser Wert ist geringfügig höher als beim Löschen, um kompaktere Graphen zu bevorzugen. Ansonsten wird der Abstand der Regionen im Objektgraphen betrachtet, zwischen denen die Kante auf Grund einer im Szenegraphen vorhandenen Kante gezogen werden soll. Ist der Abstand gering, sind die Kosten niedrig. Ansonsten sind die Kosten entsprechend höher, um den dann größeren Unterschied zum Szenegraphen widerzuspiegeln.

Beim Ersetzen eines Knotens durch einen anderen Knoten werden die beiden beteiligten Attributmengen für die Berechnung der Kosten K mit jeweils individuellen Gewichtungen verglichen:

$$S = \begin{cases} \frac{2}{3}\frac{A_S^{(1)}}{A_S^{(2)}} & \text{falls } A_S^{(1)} > A_S^{(2)} \\ \frac{2}{3}\frac{A_S^{(2)}}{A_S^{(1)}} & \text{sonst} \end{cases} \tag{6.36}$$

$$\begin{aligned} K = & \Big(3 \cdot \Big(0.2 \left| A_L^{(1)} - A_L^{(2)} \right| + 0.4 \left| A_u^{(1)} - A_u^{(2)} \right| + 0.4 \left| A_v^{(1)} - A_v^{(2)} \right| \Big) + \\ & S + 1.5 \left| A_E^{(1)} - A_E^{(2)} \right| + 1.5 \left| A_C^{(1)} - A_C^{(2)} \right| \Big) \cdot A_A^{(1)} \cdot A_A^{(2)} \end{aligned} \tag{6.37}$$

Die Berechnung der Kosten K für das Ersetzen einer Kante durch eine andere Kante ist vergleichbar. Es werden wiederum die beteiligten Attributmengen für die Kosten-

berechnung mit jeweils individuellen empirisch ermittelten Gewichtungen verglichen:

$$\mathrm{S}(a,b) = \begin{cases} a/b & \text{falls } a > b \\ b/a & \text{sonst} \end{cases} \tag{6.38}$$

$$K = 4\left|A_D^{(1)} - A_D^{(2)}\right| + \left|\mathrm{S}(A_{S_1}^{(1)}, A_{S_2}^{(1)}) - \mathrm{S}(A_{S_1}^{(2)}, A_{S_2}^{(2)})\right| \tag{6.39}$$

Mit der Definition aller benötigten Kostenfunktionen sind nun alle Voraussetzungen für die Durchführung eines fehlertoleranten Vergleiches eines Objektgraphen mit einem Szenegraphen mit Hilfe des Toolkits GUB erfüllt. Kann bei dem Vergleich ein Teilgraphisomorphismus gefunden werden, dessen Kosten normiert auf die Größe des Objektgraphen unterhalb eines Schwellwertes liegen, wird eine neue Objekthypothese generiert. Der Schwerpunkt der Hypothese ergibt sich aus dem Schwerpunkt der für den Isomorphismus benutzten Szeneregionen. Die Bewertung der Hypothese sind die auf die Größe des Objektgraphen normierten Kosten des Graphvergleichs.

6.4.4 Ergebnisse

In den letzten Abschnitten wurde ein Objekterkenner vorgestellt, der Nachbarschaftsgraphen aus segmentierten Farbregionen als Grundlage für die Erkennung nimmt. Diese Graphen werden durch einen fehlertoleranten Vergleich mit einem Szenegraphen zur Objektlokalisation herangezogen. Wie in Abschnitt 6.4.1 an zwei Beispielen dargestellt, gibt es Objekte, die für diesen Ansatz gut geeignet sind und andere, die für eine Erkennung weniger gut geeignet sind. Die Erkennungsleistung wird nun an Hand von verschiedenen Beispielen ausführlicher untersucht. Anschließend wird die Geschwindigkeit des Erkenners getestet.

Erkennungsleistung des regionenbasierten Erkenners

Abbildung 6.21 zeigt das Ergebnis eines Vergleichs eines Objektgraphen mit einem Szenegraphen. In diesem Beispiel wurde die Federmappe aus dem Szenebild durch Herausnehmen neu gelernt. Dabei wurde das Objekt korrekt in vier Regionen segmentiert, aus denen dann der aus vier Knoten bestehende rot eingerahmte Objektgraph erzeugt wurde. Für eine Suchanfrage nach der Federmappe wurde die gleiche Szene wie auch für das Lernen des Objektes genommen. Das Szenebild wurde dafür segmentiert. Die erhaltenen Regionen sind im Bild links unten in Abbildung 6.21 über das Szenebild gelegt. Aus diesen Regionen wurde der große rot eingerahmte Szenegraph erstellt. Der den Hintergrund repräsentierende Knoten in der Mitte des Graphen ist durch seine Nachbarschaft zu fast allen restlichen Regionen gut erkennbar. Da das zu lokalisierende Objekt aus dem selben Bild gelernt wurde, war die Aufgabe des Objekterkenners in diesem Fall die Bestimmung eines Teilgraphisomorphismus ohne Abweichungen und damit mit einer Edit-Distanz von nahezu Null. Dies wurde korrekt erreicht. Dargestellt ist er in der Abbildung durch orange gestrichelte Pfeile. Durch

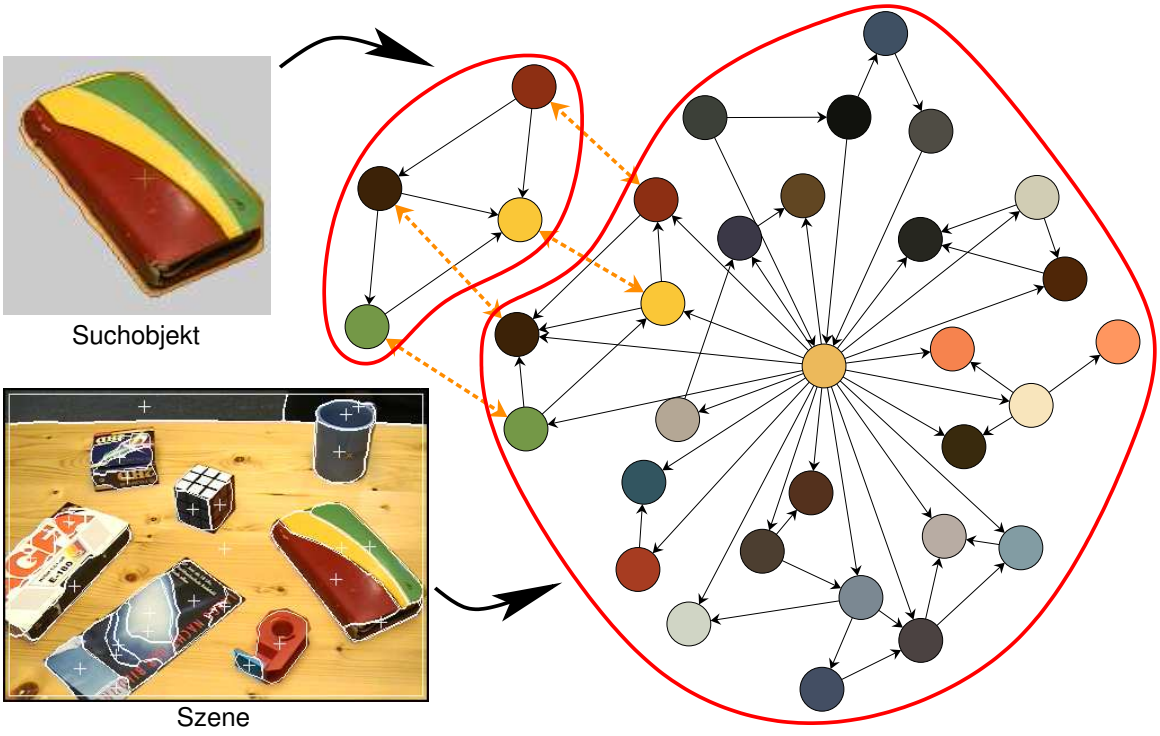


Abbildung 6.21: Die Lokalisation der Federmappe in der Szene durch eine Suche nach einem Teilgraphisomorphismus zwischen dem Graphen der Federmappe und dem Szenegraphen.

den bestimmten Teilgraphisomorphismus ist auch die exakte Position des Objektes in der Szene korrekt bestimmt worden.

Abbildung 6.22 zeigt weitere Ergebnisse von Vergleichen von zwei verschiedenen Objektgraphen mit drei unterschiedlichen Szenegraphen. In diesem Fall waren die Bilder zum Lernen und die Szenebilder jeweils unterschiedlich. In der Abbildung außen links sind die beiden Objekte und ihre Segmentierung dargestellt, wie sie während des Lernprozesses erzeugt wurden. Oben sind die Szenen dargestellt, in denen die Objekte gesucht wurden. Für jede Kombination von Objekten und Szenen ist jeweils das Ergebnis des Vergleichs der entsprechenden Graphen und die lokalisierte Position in der Szene zusammen mit den lokalisierten Szeneregionen dargestellt. Aus Gründen der Übersichtlichkeit der Darstellung sind die Szenegraphen nur ausschnittsweise wiedergegeben. Ansonsten entspricht die Darstellung der Graphen und des Vergleichs der Darstellung in Abbildung 6.21.

Die Federmappe konnte unabhängig von der Rotation, der Position im Bild und dem Hintergrund in allen Fällen sicher lokalisiert werden. Dabei wurde jeweils vom fehlertoleranten Vergleich Gebrauch gemacht, indem das aus einer schwarzen Region bestehende Seitenteil bei allen Vergleichen gelöscht wurde. In den Szenen ist diese

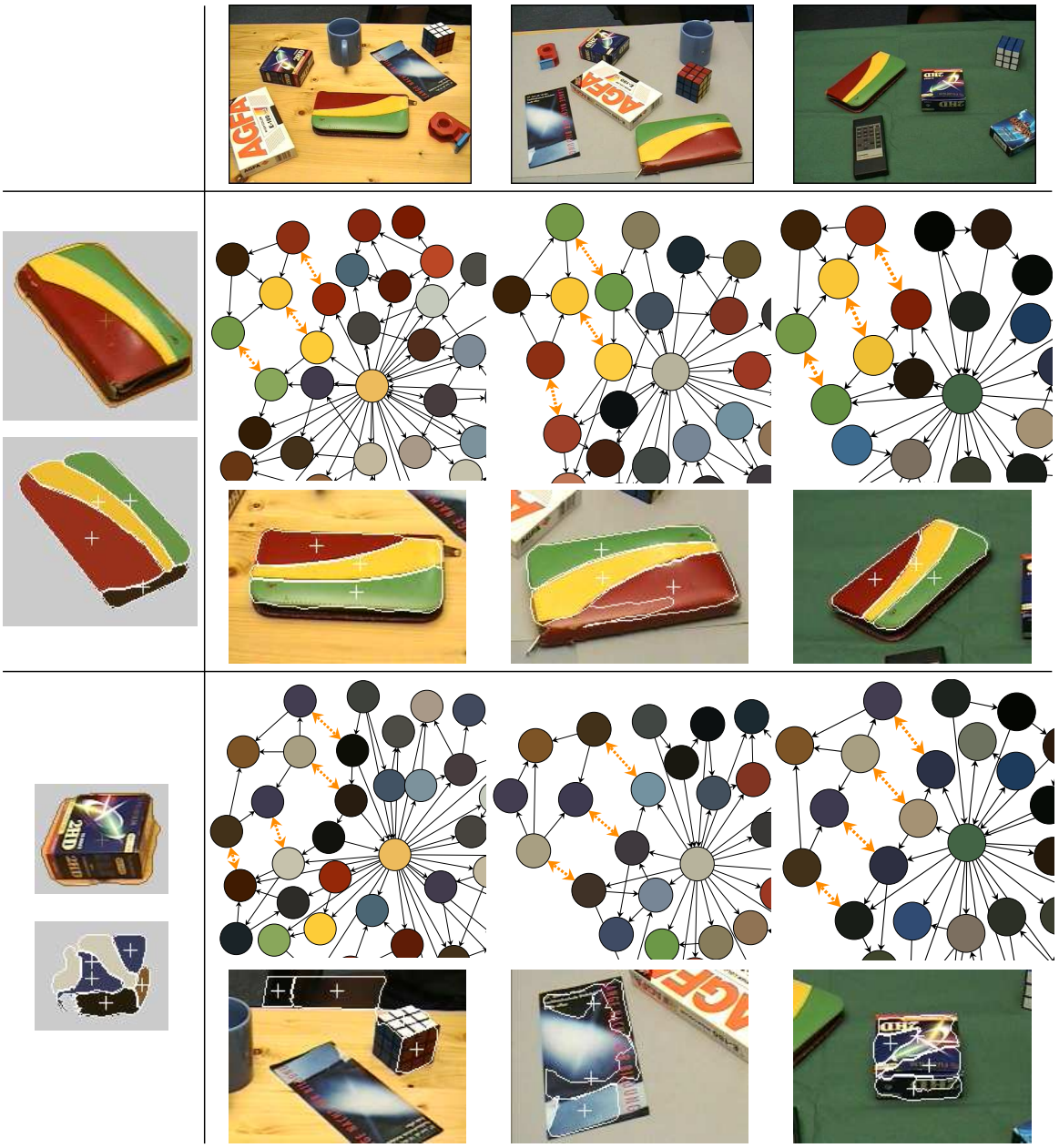


Abbildung 6.22: Ergebnisse des fehlertoleranten Graphvergleichs von zwei Objekten mit unterschiedlichen Szenen. Dargestellt sind jeweils die bestimmten Teilgraphisomorphismen und die gefundene Position des Objektes im Bild.

Region auf Grund perspektivischer Verdeckungen zum Teil gar nicht und zum Teil in merklich anderer Form sichtbar. Auf Grund der Lage am Rand des Lernobjektes hat diese Region allerdings einen schlechteren Konfidenzwert, womit die Kosten für das Löschen der Region und auch der mit der Region verbundenen Kanten vergleichsweise gering ausfallen. Die Bestimmung des korrekten Teilgraphisomorphismus wird damit sicher möglich. Die Erkennungsrate bei der Diskettenschachtel war dagegen schlechter. In einem Fall konnte sie lokalisiert werden, in zwei anderen Fällen wurde allerdings eine falsche Position bestimmt.

Durch genauere Betrachtung der Segmentierung der Diskettenschachtel während des Lernens des Objektes und in den einzelnen Szenebildern wird das Problem dieses Objektes deutlich. Abbildung 6.23 zeigt diese Segmentierungen auf der linken Seite. Die Segmentierung der Schachtel auf dem grünen Hintergrund ist relativ ähnlich zu der initialen Objektsegmentierung. Aber auch hier sind die beiden grauen Regionen aus dem ursprünglichen Objekt schon zu einer Region verschmolzen. Diese Änderung und auch die leichteren Formänderungen aller restlichen Regionen kann der fehlertolerante Graphvergleich korrekt bewältigen. Die Segmentierung der Schachtel in den beiden anderen Szenen ist allerdings drastisch anders, so daß in diesen Fällen der Graphvergleich andere Stellen in der Szene für wahrscheinlicher hält. Durch eine kleine Verringerung der Fenstergröße h aus der Gleichung 6.35 von in diesem Fall 20.53 auf 16.42 während der Mean-Shift-Segmentierung kann das Objekt auch auf dem Holzhintergrund gefunden werden. Abbildung 6.23 rechts zeigt die dann erhaltene Segmentierung und den erfolgreichen Graphvergleich.

An diesen Ergebnissen ist das allgemeine Verhalten des Erkenners zu sehen. Nötig ist eine relativ stabile Segmentierung. Da diese in diesem Ansatz auf einem Homogenitätskriterium im Farbraum aufsetzt, müssen die Objekte für eine stabile Segmentierung einfarbige Flächen haben. Ist das der Fall, kann der fehlertolerante Graphvergleich einzelne Segmentierungsänderungen, die bei real aufgenommenen Bildern nahezu immer auftreten, erfolgreich ausgleichen. Gegenüber Objektrotationen ist der Ansatz invariant, gegenüber Größenänderungen des Objektes ist er robust. Die Invarianz ist in diesem Fall nicht vollständig gegeben, da Relationen zwischen Größen verschiedener Objektregionen in die Kostenberechnung des Graphvergleichs mit eingehen. Diese Ergebnisse sind an den stabilen Erkennungsraten der Federmappe unter verschiedenen Lageänderungen und verschiedenen Hintergründen zu sehen.

Ist das Objekt nicht flächig strukturiert, wie dies bei der Diskettenschachtel der Fall ist, kann mit dem momentanen Ansatz keine sichere Lokalisation durchgeführt werden. Für diesen Fall wären verschiedene Optimierungen denkbar. Einerseits könnten andere Homogenitätskriterien für die Segmentierung gewählt werden, um beispielsweise texturierte Objekte stabiler segmentieren zu können. Andererseits sind die Kostenfunktionen des Graphvergleichs sicherlich noch nicht optimal gewählt. Durch Optimierung der bisher heuristisch gewählten Gewichtungen ließen sich weitere Segmentierungsfehler ausgleichen. Weitere geometrische Einschränkungen der Objekte neben dem Abstand von Regionen, beispielsweise die Anordnung dreier Regionen ent-

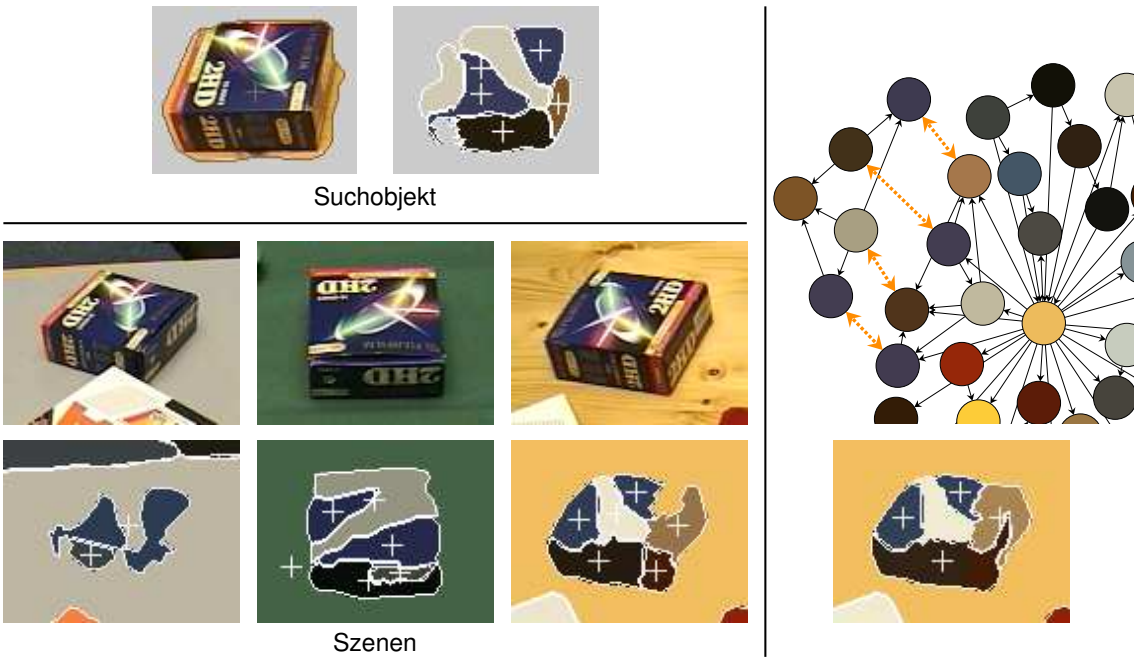


Abbildung 6.23: Ergebnisse der Segmentierung der Diskettenschachtel mit dem Mean-Shift-Verfahren. Rechts ist eine Segmentierung bei kleinerem Suchfenster des Mean-Shift-Algorithmus mit erfolgreicher Objektlokalisation dargestellt.

lang einer Geraden, ließen sich in die Kostenberechnung aufnehmen. Ein interessanter Ansatz, der aber nur mit erheblichem Aufwand realisierbar wäre, wäre eine stärkere Verknüpfung der momentan im wesentlichen getrennten Schritte Segmentierung und Graphvergleich. Die Segmentierung verläuft momentan ohne jegliches Objektwissen. Durch die Möglichkeit des Löschens von Knoten und dem Teilgraphvergleich kann nachträglich nur eine sehr eingeschränkte Art von Regionenverschmelzung und Aufteilung durchgeführt werden. Dies weiter auszubauen und damit eine stärkere Kopplung der beiden Schritte zu erreichen wäre eine erfolgversprechende Optimierung, die dem Erkenner weitere Objektgruppen neben den flächig Strukturierten öffnen würde.

Geschwindigkeit des regionenbasierten Erkenners

Die Geschwindigkeit des regionenbasierten Erkenners wurde für Objekte unterschiedlicher Komplexität untersucht. Dies waren die in den letzten Beispielen schon benutzten Objekte Federmappe und Diskettenschachtel und das Reklameheft. Das Reklameheft war in der Teststichprobe der Histogrammsuche bei einer Darstellung als Graph am komplexesten. Bei den Versuchen kamen die beiden in der Tabelle 6.1 auf Seite 113 vorgestellten Rechner *Alpha* und *PC* zum Einsatz. An Compilern wur-

den wie bei der Histogrammsuche bei der Alpha der DEC C Version 5.9 und bei dem PC der GCC Version 3.3 [GCC03] eingesetzt. Zur Ermittlung der Zeiten wurde zuerst eine neue Ansicht eines Objektes gelernt. Anschließend wurde dieses Objekt im Falle der Diskettenschachtel und des Reklameheftes in allen 57 Bildern der Teststichprobe der Histogrammsuche gesucht. Da die Federmappe in dieser Stichprobe nicht vorkam, wurde für dieses Objekt eine eigene Teststichprobe von 17 Bildern aufgenommen, die vom Aufbau der Stichprobe der Histogrammsuche entsprach. Alle Bilder hatten eine Auflösung von 378x278 Pixeln. Der aus den Bildern erstellte Szenegraph hatte durchschnittlich 32 Knoten und 59 Kanten. Die Komplexität des Objektgraphen und die Zeit für die Segmentierung und das Finden eines Objektes in einem Bild, gemittelt über alle Bilder der Stichprobe, ist in Tabelle 6.2 angegeben.

	Objektgraph		Zeiten	
	# Knoten	# Kanten	Alpha[ms]	PC[ms]
Mean-Shift-Segmentierung	-	-	434	209
Federmappe	4	5	103	63
Diskettenschachtel	5	6	258	145
Reklameheft	8	11	7.8s	12.2s

Tabelle 6.3: Geschwindigkeit des regionenbasierten Erkenners bei Objekten unterschiedlicher Komplexität.

Da die Segmentierung objektunabhängig ist, muß sie unabhängig von der konkreten Anfrage für jede Anfrage nur einmal durchgeführt werden. Die 209 Millisekunden auf dem PC sind daher ein nicht zu hoher Aufwand. Der Graphvergleich sollte dagegen möglichst schnell sein, da er je nach der Anzahl Ansichten eines Objektes und der konkreten Anfrage mehrfach durchgeführt werden muß. Ist die Komplexität der Objektgraphen nicht zu hoch, erreicht der regionenbasierte Erkenner eine hohe, für die Anwendung geeignete Geschwindigkeit. Bei steigender Komplexität wird die Geschwindigkeit allerdings deutlich geringer.

6.5 Zusammenfassung

In diesem Kapitel wurden verschiedene Objekterkenner basierend auf zwei unterschiedlichen Ansätzen vorgestellt. Einerseits wurde ein statistischer Ansatz gewählt, der die Dichtefunktion eines Objektes durch ein Histogramm modelliert. Durch die datenunabhängige Aufteilung von Histogrammen ist deren Bestimmung und auch deren Vergleich sehr schnell möglich. Für den Vergleich der Histogramme kommt der Histogrammschnitt zum Einsatz. Um Objekte in einer Szene lokalisieren zu können, findet der Vergleich jeweils in einem Fenster in der Größe des Objekthistogramms statt, welches über das vollständige Szenebild geschoben wird. Dafür wurden verschiedene

optimierte Berechnungsmethoden vorgestellt und auf einer Teststichprobe evaluiert. Zusätzlich wurden als weitere Berechnungsart von Histogrammen Fuzzyhistogramme evaluiert.

Für die Histogrammmerkmale werden sowohl Farbinformationen als auch rotationsinvariante strukturbasierte Informationen benutzt. Da die Histogramme über Objektkoordinaten berechnet werden, ist der Erkenner damit translations- und rotationsinvariant. Eine Evaluation hat die Leistungsfähigkeit bei geringem Trainingsmaterial und gleichzeitig hoher Geschwindigkeit gezeigt. Fuzzyhistogramme waren dabei im Vergleich zu normalen Histogrammen nicht signifikant besser, aber merklich langsamer, so daß sie nicht weiter eingesetzt werden. Durch kombinierten Einsatz von Farb- und Strukturmerkmalen konnte die Erkennungsrate im Vergleich zum einzelnen Einsatz der Merkmale weiter gesteigert werden.

Um auch globale Strukturinformationen mit zu berücksichtigen, wurde zusätzlich ein konstruktivistischer Ansatz für einen Objekterkenner betrachtet. Dieser Ansatz basiert auf der Farbsegmentierung eines Eingangsbildes. Um möglichst große Stabilität bei hoher Geschwindigkeit zu erreichen, kommt das global arbeitende Mean-Shift-Verfahren zum Einsatz. Aus den bei der Segmentierung entstandenen Regionen wird ein Graph erstellt, der Regionen als Knoten enthält und über Kanten Nachbarschaften zwischen den Regionen abbildet. Über einen fehlertoleranten Vergleich von Objektgraphen mit einem Graphen erstellt aus der kompletten Szene wird eine Lokalisation und Identifikation des Objektes vorgenommen. Der Teilgraphvergleich erlaubt das durch Kosten kontrollierte Löschen und Einfügen von Knoten und Kanten, um auf Variationen in der Segmentierung reagieren zu können. Auch dieser Erkenner ist wie der histogrammbasierte Erkenner translations- und rotationsinvariant.

Kann das Mean-Shift-Verfahren Objekte im wesentlichen stabil segmentieren, erzielt das System gute Erkennungsraten. Dazu sind allerdings Objekte mit großflächig einfarbigen Strukturen nötig. Ist dies nicht gegeben, fällt die Leistung auf Grund zu stark variierender Graphen ab. Ein Ansatz für eine Verbesserung könnte eine stärkere Kopplung der beiden Schritte des Erkenners sein. Momentan ist die Segmentierung problemunabhängig. Würde hier Objektwissen integriert, beispielsweise durch einen auf den Graphen beruhenden Regionenverschmelzungs- und Aufteilungsschritt, könnte die Stabilität der Segmentierung und damit die Erkennungsrate weiter erhöht werden.

7 Modifikation von Hypothesen

Durch die im letzten Kapitel vorgestellten Objekterkennungssysteme können Objekte in einer Szene gefunden und identifiziert werden. Dazu wird durch eine sprachliche Anfrage ein Objekt vorgegeben, welches in einer Datenbank in Form verschiedener merkmalsbasierter Ansichten abgelegt ist. Durch sequentielles Absuchen des Bildes sowohl mit den verschiedenen Objekterkennern als auch mit den verschiedenen Ansichten des Objektes können unterschiedliche Hypothesen über das Vorhandensein der jeweiligen Ansicht an einer Position im Bild erstellt werden. Damit sind allerdings noch nicht alle Informationsquellen für die Lokalisation von Objekten ausgenutzt.

Ausgehend von den durch die Objekterkenner erstellten Hypothesen können verschiedene Modifikationen an den Hypothesen und ihren zugrundeliegenden Ansichten vorgenommen werden, um sowohl die Erkennungsrate in der jetzigen Anfrage als auch die Erkennungsleistung in späteren Anfragen zu steigern. Dieses Kapitel stellt die realisierten Möglichkeiten vor. Im nächsten Abschnitt 7.1 wird die Kombination von einzelnen Hypothesen zu neuen Hypothesen vorgestellt. Für die bisherigen Hypothesen wurden an sprachlicher Information nur die Objektnomen ausgenutzt. Durch Spezifikation von Objekteigenschaften kann der Benutzer die Objektsuche weiter eingrenzen. Dies wird in Abschnitt 7.2 näher beschrieben. Abschnitt 7.3 stellt schließlich die Optimierung von Datenbankeinträgen auf Grund der aktuellen Anfrage vor.

7.1 Kombination von Hypothesen

Die im letzten Kapitel vorgestellten Erkenner generieren Hypothesen eines vorgegebenen Formates. Jede Hypothese enthält eine Referenz auf den Erkenner, mit dem sie erzeugt wurde, auf das ihr zugrundeliegende Objekt und auf dessen Ansicht. Eine Position und ein Radius geben den wahrscheinlichsten Ort des Objektes im Bild und dessen Größe an. In einer Bewertung wird die Wahrscheinlichkeit des Vorhandenseins hinterlegt. Zusätzlich können die Erkenner eine Wahrscheinlichkeitskarte angeben, die für jeden Punkt des Bildes getrennt die Wahrscheinlichkeit des Vorhandenseins der Ansicht spezifiziert. Dies nutzt der in Abschnitt 6.2 vorgestellte histogrammbasierte Erkenner, falls keine aktive Suche durchgeführt wird. Durch diese vom konkreten Erkenner abstrahierte Hypothesendarstellung ist eine Behandlung der Hypothesen unabhängig vom jeweiligen Objekterkenner möglich.

Durch die im letzten Kapitel vorgestellten Erkenner liegen unabhängige Hypothesen von unterschiedlichen Erkennern und unterschiedlichen Ansichten eines Objektes vor. Ein vielversprechender Ansatz zum Erreichen einer besseren und stabileren Erkennungsrate ist die Kombination verschiedener Hypothesen. Für die Kombination verschiedener Klassifikatoren wurden in der Literatur verschiedene Ansätze vorgeschlagen. [Jai00] gibt eine Übersicht über eingesetzte Architekturen und Kombinationsfunktionen. Der hier verfolgte Ansatz hat eine parallele Architektur, in der die Klassifikatoren wie bereits beschrieben unabhängig voneinander aufgerufen werden. Deren Ergebnisse werden anschließend in einem getrennten Schritt kombiniert.

Für die vollständige Hypothesengenerierung steht im System nur eine geringe Menge an Trainingsmaterial zur Verfügung. Ist die Kombination von Hypothesen ohne Trainingsmaterial möglich, steht entsprechend pro freiem Parameter mehr Material für die initiale Objekterkennung zur Verfügung. Daher werden parameterfreie Kombinationsarten präferiert. Eine große Robustheit gegenüber schlechten Klassifikationsraten ist aus dem selben Grund ebenfalls wünschenswert. Ein Verlaß auf gute Klassifikationsraten für die einzelnen Erkenner ist bei dem geringen Trainingsmaterial nicht gegeben. Um möglichst flexibel zu sein, sollte die Kombination beliebig vieler Hypothesen möglich sein. Ein Ansatz, der diese Voraussetzungen gut erfüllt und gleichzeitig sehr einfach und schnell ist, ist die Summenregel. Die Summenregel ist insbesondere auch stabiler gegenüber Fehlern der einzelnen Klassifikatoren als beispielsweise die Produktregel [Kit98].

Damit zwei Hypothesen kombiniert werden können, gibt es verschiedene Voraussetzungen. Sie müssen das gleiche Objekt beschreiben, von unterschiedlichen Erkennern erzeugt worden sein und bevorzugt die gleiche Ansicht als Grundlage haben. Im vorgestellten System müssen allerdings nicht nur Konfidenzen für einzelne Klassen kombiniert werden. Die Hypothesen tragen zusätzlich Ortsinformationen über die Lokalität der Hypothese, die ebenfalls konsistent bleiben muß. Dies wird als erfüllt angesehen, falls entweder beide Hypothesen Wahrscheinlichkeitskarten haben oder ein genügend großer Überlapp zwischen den beiden ansonsten kreisförmigen Hypothesen besteht. Als genügend groß wird der Überlapp angesehen, falls das größere Objekt das kleinere mit 40% der Gesamtgröße überlappt und das kleinere das größere mit 60%. Damit werden gewisse Positionsabweichungen und Größenunterschiede zwischen den Hypothesen toleriert.

Können zwei Hypothesen kombiniert werden, werden ihre Wahrscheinlichkeitskarten pixelweise addiert. Die neue Bewertung ist das Maximum der neuen Wahrscheinlichkeitskarte. Das Maximum gibt auch die neue Hypothesenposition an. Hat eine der beiden Hypothesen keine Wahrscheinlichkeitskarte, ist die neue Bewertung die Summe der beiden einzelnen Bewertungen. Die neue Hypothesenposition ist in diesem Fall der mittlere Punkt zwischen den beiden Einzelhypothesen. Als neuer Radius wird der größere der beiden alten Radien übernommen. Das Kombinieren von Hypothesen wird solange fortgesetzt, bis sich keine weiteren kombinierbaren Hypothesen finden. Als erstes werden dabei Hypothesen genommen, die von der gleichen Objektansicht

generiert wurden und die eine möglichst gute Bewertung haben. Diese sind einerseits vermutlich besonders ähnlich zueinander und andererseits besonders abgesichert, was sie zu guten Kombinationskandidaten macht.

Abbildung 7.1 zeigt zwei Beispiele für die Kombination jeweils zweier Wahrscheinlichkeitskarten, die von den im letzten Kapitel vorgestellten Erkennern mit farbbasierten beziehungsweise texturbasierten Histogrammen erzeugt wurden. In beiden Fällen hat jeweils einer der Erkenner eine fehlerhafte wahrscheinlichste Position zurückgeliefert. Bei der Fernbedienung war dies der farbbasierte Erkenner und bei den Karten der texturbasierte Erkenner. Durch die Kombination mit dem jeweils anderen Erkenner konnten diese Fehler korrigiert werden. Auf Seite 120 in Abschnitt 6.3.3 wurden Ergebnisse vorgestellt, die durch diese Art der Kombination verschiedener Hypothesen erzeugt wurden. In allen Experimenten konnte eine Steigerung der Erkennungsrate im Vergleich zu den jeweiligen Einzelergebnissen erzielt werden.

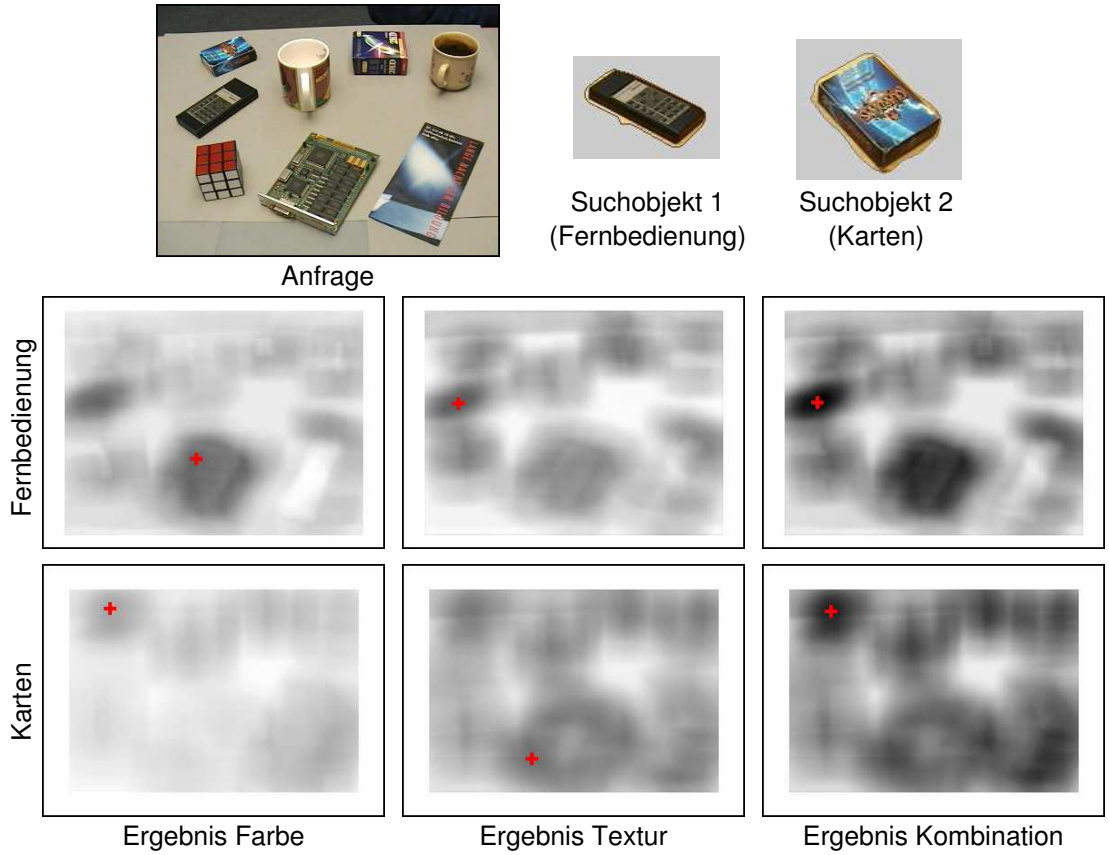


Abbildung 7.1: Zwei Beispiele für die Kombination von Hypothesen. In der Anfrage wurde mit zwei getrennten Erkennern nach der Fernbedienung bzw. den Karten gesucht. In den Ergebnissen ist die wahrscheinlichste Position mit einem roten Kreuz markiert.

7.2 Modifikationen durch sprachliche Attribute

Bei den bisher generierten Hypothesen wurde an sprachlicher Information nur ein Objektnomen ausgewertet. Wie in Kapitel 4 vorgestellt, sind bei der Anfrage zusätzlich Farb- und Größenadjektive für eine genauere Objektspezifikation erlaubt. Dies ist in Einklang mit den in Abschnitt 2.1.2 vorgestellten linguistischen Erkenntnissen, die einen häufig vorkommenden Einsatz dieser Merkmale bei Objektbenennungen aufzeigen. Daher fließen auch diese sprachlichen Informationen in die Objekterkennung mit ein.

Für eine vollständige Verarbeitung dieser Objektmerkmale ist die Integration von Unsicherheiten auf verschiedenen Ebenen ein entscheidender Punkt. Die Bedeutung der sprachlichen Merkmale ist nur unscharf definiert. Das ein Objekt groß ist, bedeutet beispielsweise nicht, daß es eine Fläche von exakt 5.7 Quadratmetern einnimmt. Ein als Rot bezeichnetes Objekt kann auch Orange sein. Weitere Unsicherheiten kommen auch durch Fehler in der Spracherkennung und der Objekterkennung zu Stande. Auch durch Kontextobjekte wird die Benennung signifikant beeinflußt. Dabei ist ein Kontext nicht gegeben, er kann für jeden Benutzer und jede Anfrage individuell variieren. Für die Integration all dieser Unsicherheiten und die Schlußfolgerung mit ihnen gibt es verschiedene Paradigmen. In der Dempster-Shafer Theorie wird sowohl Unsicherheit als auch Unwissenheit modelliert [Yag94]. Bayes-Netzwerke oder auch Belief Netzwerke sind azyklische gerichtete Graphen, die Abhängigkeiten zwischen verschiedenen Zufallsvariablen modellieren [Jen96]. Die Modellierung von Unsicherheit gehorcht hier der Wahrscheinlichkeitstheorie.

Eine weitere Möglichkeit ergibt sich durch die Fuzzy Logik [Kli95], die die Definition von Mengen und die Operationen mit ihnen modifiziert. In der Fuzzy Logik ist die Zugehörigkeit eines Elementes aus einer Grundmenge G zu einer Menge nicht hart, sondern graduell durch eine Zugehörigkeitsfunktion $f : G \rightarrow [0, 1]$ definiert. Ein Beispiel hierfür sind auch die Zugehörigkeitsfunktionen der Fuzzyhistogramme (siehe Seite 96). Im hier vorgestellten System werden solche Funktionen sowohl für Größen- als auch für Farbadjektive eingesetzt. Im folgenden wird dies näher vorgestellt.

7.2.1 Größenadjektive

Die Auswahl von Größenadjektiven bei der Benennung von Objekten hat zum Teil objektinhärente Einflußfaktoren und zu einem großen Teil auch kontextbezogene Faktoren. In experimentellen Untersuchungen in einem Konstruktionsszenario wurden beide Faktoren festgestellt [Vor01a, Vor01b]. Die Auswahl der Attribute ist dabei insbesondere abhängig von Kontextobjekten der gleichen Kategorie. Gleichzeitig sichtbare Objekte anderer Kategorien beeinflussen die Wahl nur geringfügig. Beispielsweise wurden in dem in [Vor01a, Vor01b] beschriebenen Experiment lange Schrauben auch dann mit lang bezeichnet, wenn an sich merklich längere Leisten gleichzeitig zu sehen waren.

Für die Objektlokalisation besonders wichtig ist die Auswertung der kontextbezogenen Benennung. Diese differenziert häufig zwischen den in der Szene vorhandenen Objekten und wird in diesen Fällen für eine korrekte Auswahl des benannten Objektes benötigt. Die Auswertung bei Größenadjektiven beschränkt sich momentan auf diesen wichtigen Fall. Die Auswertung der Attribute geschieht durch Adaption der Bewertungen der auf Grund von Objektnomen bestimmten Objekthypothesen. Die Bewertung von Hypothesen, die besonders gut zu einem sprachlichen Merkmal passen, werden erhöht. Bewertungen von schlecht zu einem sprachlichen Merkmal passenden Hypothesen werden reduziert. Dazu werden sowohl Hypothesen als auch sprachliche Attribute betrachtet.

Durch die Semantikanalyse wurde eine Abbildung der vom Benutzer verwendeten Größenadjektive auf eine Liste von Adjektiven aus einer Basismenge $S^{(G)}$ von Adjektiven durchgeführt:

$$S^{(G)} = \{\text{winzig, klein, mittelgroß, groß, riesengroß, riesig}\} \tag{7.1}$$

Den sechs möglichen Adjektiven sind kontinuierliche Gauß'sche Zugehörigkeitsfunktionen zugeordnet, die die Zugehörigkeit der Hypothesen zu den sprachlichen Attributen bestimmen. Das Intervall ihrer Funktionswerte wurde von $[0, 1]$ auf $[0.2, 1.7]$ erweitert. Damit wird einerseits eine vollständige Rückweisung einer Hypothese auf Grund eines möglicherweise falsch interpretierten Attributes vermieden. Andererseits werden Hypothesen, die gut zu Attributen passen, stärker bevorzugt. Abbildung 7.2 zeigt diese Funktionen. Für ihre Anwendung auf die Objekthypothesen müssen die Hypothesen in einen für die Zugehörigkeitsfunktionen geeigneten Definitionsbereich konvertiert werden. Dazu werden die Hypothesen entsprechend ihrer Größe sortiert und linear im Intervall $[0, 1]$ angeordnet. Die zugeordnete Position einer Hypothese i sei $P_i^{(G)}$. Die Bewertung der Hypothese wird nun mit dem Wert aller über die Sprache zugeordneten Zugehörigkeitsfunktionen an der Position $P_i^{(G)}$ multipliziert.

Ein Beispiel verdeutlicht die Auswirkungen. Sind zwei alternative Hypothesen bestimmt worden, sind die zugeordneten Werte $P_i^{(G)}$ 0 und 1. Liegt das Attribut "winzig" vor, wird die kleinere Hypothese damit mit 1.7 und die größere mit 0.26 multipliziert. Mit großer Wahrscheinlichkeit war in diesem Fall die kleinere Hypothese gemeint. Liegen drei Hypothesen vor, ergeben sich die $P_i^{(G)}$ durch lineare Verteilung im Intervall $[0, 1]$ zu 0, 0.5 und 1. Die mittlere Hypothese wird damit durch einen Faktor von 0.88 nicht völlig ausgeschlossen. Die anderen beiden Hypothesen werden wie im ersten Beispiel mit 1.7 und 0.26 multipliziert. Angepaßt an den Kontext und an die sprachlichen Attribute wird damit die Bewertung variiert.

7.2.2 Farb- und Helligkeitsadjektive

Farbe wie auch Helligkeit ist ein im wesentlichen absolutes Merkmal. Damit ist der Einfluß des Kontextes auf die Benennung im Vergleich zu den Größenadjektiven

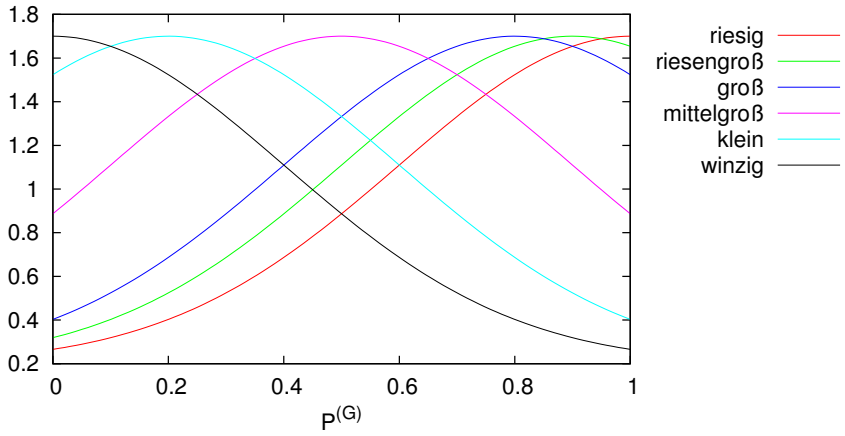


Abbildung 7.2: Zugehörigkeitsfunktionen bei der Auswertung von Größenadjektiven.

gering. Bei entsprechenden psycholinguistischen Experimenten hat sich allerdings gezeigt, daß bei Objekten mit Farben wie Violett oder Orange häufig Grundfarben wie Blau beziehungsweise Rot spezifiziert werden [Wac01]. Diese beiden Ergebnisse werden im System beachtet. Wie bei den Größenadjektiven wird die Bewertung von Hypothesen angepaßt an die sprachlich spezifizierten Farben modifiziert. Im Gegensatz zu den Größenadjektiven wird der Kontext allerdings nicht beachtet. Die Modifikationen sind spezifisch für eine Hypothese und auch spezifisch für einen Objekterkenner. Hypothesen enthalten, wiederum im Gegensatz zu Größeninformation, keine Farbinformation, so daß eine erkennerunabhängige Behandlung von Farbe nicht möglich und auch nicht sinnvoll ist.

Wie bei den Größenadjektiven wird durch die Semantikanalyse eine Abbildung der vom Benutzer verwendeten Farb- und Helligkeitsadjektive auf eine Liste von Adjektiven aus einer Basismenge $S^{(F)}$ von Adjektiven durchgeführt:

$$S^{(F)} = \{\text{weiß, hell, grau, dunkel, schwarz, rot, orange, gelb, grün, blau, violett}\} \tag{7.2}$$

Die Adjektive werden im HSV-Farbraum ausgewertet. Weitere Informationen zum HSV-Farbraum finden sich in Anhang A. Den einzelnen Adjektiven werden kontinuierliche Zugehörigkeitsfunktionen zugeordnet, die aus Differenzen sigmoider Funktionen bestehen. Damit wird ein zentraler voll zugehöriger Kern modelliert, der auf beiden Seiten von einem unscharfen Übergangsbereich flankiert wird.

Die Funktionen der Helligkeitsadjektive zeigt Abbildung 7.3 in Abhängigkeit von der Helligkeit eines Farbwertes. Bei der Anwendung der Funktionen auf einen HSV-Wert wird der Farbton H ignoriert. Die Sättigung S muß bei den Adjektiven weiß und grau gering sein. Ansonsten wird auch die Sättigung ignoriert.

Die Zugehörigkeitsfunktionen der Farbadjektive zeigt Abbildung 7.4. Bei deren Anwendung auf einen HSV-Farbwert werden alle drei Farbanteile durch Multiplikation

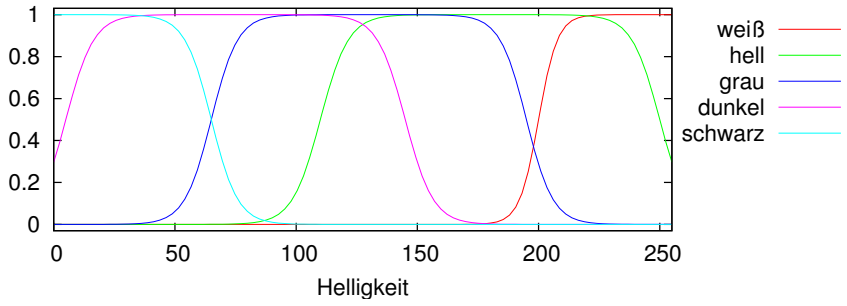


Abbildung 7.3: Zugehörigkeitsfunktionen bei der Auswertung von Helligkeitsadjektiven.

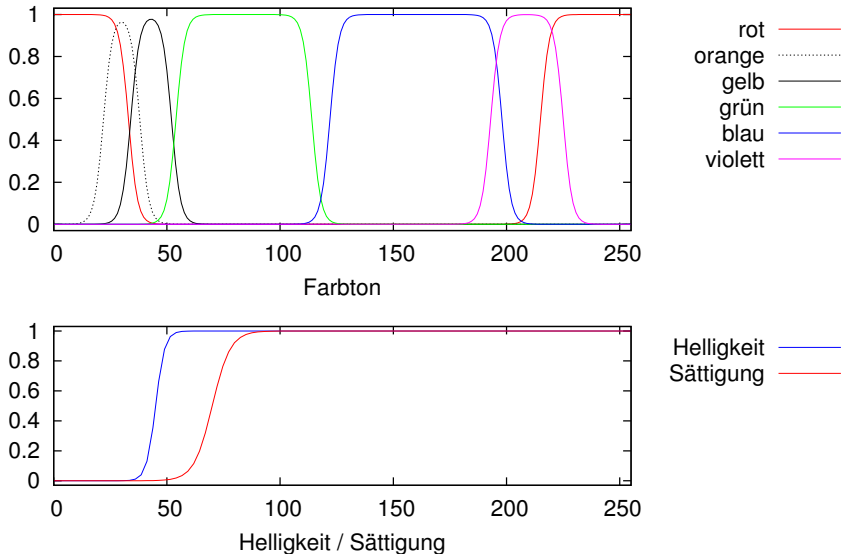


Abbildung 7.4: Zugehörigkeitsfunktionen bei der Auswertung von Farbadjektiven. Sowohl der Farbton als auch die Helligkeit und die Sttigung wird beachtet.

der entsprechenden Werte der Zugehörigkeitsfunktionen beachtet. Den Grundfarben wie Rot oder Grün ist bei den Farbtönen ein breiterer Bereich zugeordnet als Farben wie Orange oder Violett. Helligkeit und Sättigung sind unabhängig von der Farbe definiert.

Die weitere Auswertung der sprachlich benannten Adjektive wird von allen Objekterkennern durchgeführt, die zu der zu bewertenden Hypothese beigetragen haben und Farbe verarbeiten. Der Erkenner basierend auf Texturhistogrammen verarbeitet keine Farbinformationen und bewertet die Adäquatheit der Adjektive daher nicht. Die

anderen beiden Erkenner, der regionenbasierte und der auf Farbhistogrammen basierende, verarbeiten die Sprachinformation vergleichbar. Der regionenbasierte Erkenner betrachtet die Werte der Zugehörigkeitsfunktion f an den Positionen der Farbwerte $\text{Farbe}_{\text{HSV}}(R_i)$ aller Regionen R_i der Objekthypothese und normiert und skaliert diese passend:

$$J' = 2 + 2 \cdot \left(\frac{\sum_{R_i} f(\text{Farbe}_{\text{HSV}}(R_i)) \cdot \text{Größe}(R_i)}{\sum_{R_i} \text{Größe}(R_i)} - 1 \right)^3 \tag{7.3}$$

Abbildung 7.5 zeigt einen Plot dieser Funktion. Die bisherige Bewertung J der Hypothesen wird nun mit J' multipliziert. Gehören beispielsweise alle Pixel des Objektes entsprechend der Zugehörigkeitsfunktion vollständig zu der sprachlich beschriebenen Farbklasse, wird die Bewertung damit verdoppelt. Gehört ein Fünftel der Pixel dazu, wird sie beibehalten. Ist die Menge noch geringer, wird sie verringert. Sind mehrere sprachliche Attribute angegeben, wird diese Berechnung entsprechend mehrfach iteriert durchgeführt.

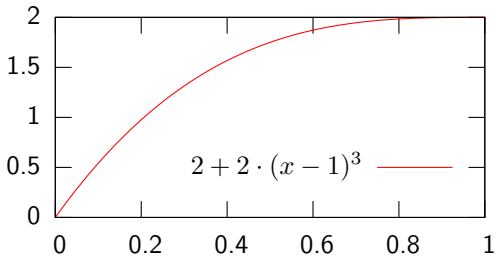


Abbildung 7.5: Bewertungsmodifikation einer Objekthypothese in Abhängigkeit des Wertes einer Zugehörigkeitsfunktion für ein sprachliches Attribut.

Der auf Farbhistogrammen basierende Erkenner modifiziert die Bewertung vergleichbar. Der einzige Unterschied ist hier, daß der Erkenner anstelle von Bildregionen mit den Regionen der Farbhistogramme arbeitet. Er bestimmt den Wert der Zugehörigkeitsfunktionen an der Position der Farbe aller Regionen H_i des Farbhistogramms H und normiert mit dem Wert der Region des Histogramms. Damit ergibt sich

$$J' = 2 + 2 \cdot \left(\frac{\sum_i f(\text{Farbe}_{\text{HSV}}(H_i)) \cdot H_i}{\sum_i H_i} - 1 \right)^3 \tag{7.4}$$

Die weitere Verarbeitung ist identisch zum regionenbasierten Erkenner.

7.3 Ausnutzung von Benutzerrückmeldungen

In Abschnitt 2.2 wurden zwei verschiedene Formen des Lernens vorgestellt, der initiale Wissenserwerb und die inkrementelle Verbesserung und Verfeinerung von Fähigkeiten. Sowohl das bisher beschriebene Vorgehen beim Lernen eines neuen Objektes

als auch beim Lernen einer neuen Ansicht eines Objektes entspricht im wesentlichen einem initialen und damit bewußten Wissenserwerb. Beim Menschen ist aber auch die fortwährend stattfindende Verfeinerung von Fähigkeiten ein wichtiger Faktor, um Wissen aktuell und akkurat zu halten. Dieses Verfahren wird beim Lernen neuer Ansichten nur in einem sehr begrenzten Maße realisiert – im wesentlichen ist auch dies initialer Wissenserwerb.

Der Mensch bezieht zur fortwährenden Optimierung seines Wissens das Ergebnis seiner Handlungen mit ein. Dieser Ansatz wird auch erfolgreich im Information Retrieval eingesetzt. In der inhaltsbasierten Suche nach Bildern ist das Konzept des *relevance feedback* äußerst erfolgreich [Zho03]. Dieses ursprünglich in der Dokumentensuche entwickelte Verfahren [Roc66] präsentiert dem Benutzer nach einer Anfrage mehrere Ergebnisse, die er in Bezug auf die Relevanz in Hinblick auf seine ursprüngliche Anfrage bewerten kann. Das System kann daraufhin seine internen Suchparameter optimieren und ein verbessertes Ergebnis präsentieren.

Zum Beantworten der direkten Systemanfragen – die Lokalisation und Identifikation eines Objektes – ist dieses Verfahren in der in dieser Arbeit untersuchten Fragestellung nicht geeignet. Müßte der Benutzer erst mehrmals Rückmeldungen über Zwischenergebnisse geben, könnte er das Objekt häufig auch selber lokalisieren. Dagegen bietet sich die Einbeziehung von Rückmeldungen für eine fortwährende Optimierung der Datenbankeinträge an. Dabei wird jede Anfrage im Sinne des *relevance feedback* als eine Runde zur Optimierung der vollständigen Datenbank angesehen. Wiederum ist es hierbei aber wichtig, den Benutzer mit möglichst wenigen Aufgaben zu belasten. Er möchte nur ein Objekt haben und will keine Datenbank optimieren. Direkte Nachfragen zu einer Relevanzbewertung werden daher vermieden. Statt dessen werden nur die Rückmeldungen berücksichtigt, die er sowieso gibt.

Wie in Abschnitt 4.4 vorgestellt, hat der Benutzer die Möglichkeit, Fehler des Systems sprachlich zu korrigieren. Mit der Korrektur gibt er gleichzeitig auch eine negative Bewertung für die ausgewählte Hypothese ab. Führt er keine Korrektur aus, gibt er implizit eine positive Bewertung ab. Mit dieser Bewertung lassen sich die in der Objektdatenbank befindlichen Merkmale, die für die bewertete Hypothese eingesetzt wurden, optimieren. Alle Merkmale in der Datenbank sind mit einer Bewertung über ihre Güte für die Erkennung des zugehörigen Objektes versehen. Diese Werte werden bei dem Eintrag eines neuen Merkmals auf eins gesetzt. Durch eine negative Bewertung wird die Merkmalsbewertung aller Merkmale, die zu der Hypothese beigetragen haben, um einen konstanten Faktor reduziert. Bei einer positiven Bewertung des Benutzers werden die betroffenen Merkmalsbewertungen um einen konstanten Faktor erhöht.

Eingesetzt werden die Merkmalsbewertungen durch eine Modifikation der Bewertungen der Hypothesen. Die Bewertungen der generierten Hypothesen werden mit allen Bewertungen aller an der Hypothese beteiligten Merkmale multipliziert. Damit werden Hypothesen bevorzugt, die aus häufig erfolgreichen Merkmalen generiert wurden. Andere Hypothesen bekommen eine schlechtere Bewertung, da sich ihre zugrun-

deliegenden Objektmodelle in Form von Merkmalen als nicht zuverlässig herausgestellt haben.

7.4 Zusammenfassung

Nach der Vorstellung der initialen Hypothesengenerierung durch einzelne Objekterkenner im letzten Kapitel wurde in diesem Kapitel die Ausnutzung weiterer Informationen zur Optimierung der Hypothesen beschrieben. Im einzelnen waren dies die Kombination von einzelnen Hypothesen zu komplexeren Hypothesen, die Ausnutzung von sprachlich spezifizierten Größen- und Farbadjektiven und die Auswertung von Benutzerrückmeldungen. Für einen kompakten Überblick über die einzelnen in diesem Kapitel vorgestellten Optimierungsansätze folgt nun eine formalere Übersicht der einzelnen nacheinander durchgeführten Schritte von der im letzten Kapitel vorgestellten initialen Hypothesengenerierung bis zur Auswahl einer Hypothese:

- Sei $M_{ijk} = M_i(V_j(O_k))$ das in der Objektdatenbank abgelegte i'te Merkmal der j'ten Ansicht des k'ten Objektes. Dies kann beispielsweise ein Histogramm oder ein Objektgraph einer Ansicht eines gelernten Objektes sein.

- Die Objekterkenner generieren nun eine Menge von N Hypothesen, die jeweils von einem Merkmal abhängig sind: $H^\nu = \{H_1^\nu(M_{ijk}), \ldots, H_N^\nu(M_{lmn})\}$.

- Durch die Kombination von Hypothesen wird eine neue Menge von $M \leq N$ Hypothesen generiert, die nun von mehreren Merkmalen abhängig sind: $H^\kappa = \{H_1^\kappa(M_{i'j'k'}, \ldots, M_{l'm'n'}), \ldots, H_M^\kappa(\ldots)\}$.

- Die Bewertung J der Hypothesen H^κ wird abhängig von den Bewertungen der in den Hypothesen kombinierten Einzelhypothesen H_j^ν und der Bewertung der benutzten Merkmale M_{xyz} gesetzt: $J(H_i^\kappa(M_{ijk}, ..., M_{lmn})) = \left(\prod_{m\in\mathcal{M}} J(m)\right) \sum_j J(H_j^\nu) \qquad \mathcal{M} = \{M_{ijk}, ..., M_{lmn}\}$

- Die Bewertung J der Hypothesen H^κ wird abhängig von den sprachlich geäußerten Größenadjektiven $S_j^{(G)}$ und den Hypothesen aus der Menge H^κ modifiziert. Dabei wird eine Bewertungsfunktion J_G für Größenadjektive eingesetzt: $J(H_i^\kappa) \leftarrow J(H_i^\kappa) \cdot J_G(H^\kappa, S_j^{(G)})$

- Die Bewertung J der Hypothesen H^κ wird abhängig von den sprachlich geäußerten Farbadjektiven $S_j^{(F)}$ modifiziert. Dabei wird eine Bewertungsfunktion J_F für Farbadjektive eingesetzt, die abhängig von den für die Hypothese eingesetzten Objekterkennern definiert ist: $J(H_i^\kappa) \leftarrow J(H_i^\kappa) \cdot J_F(H_i^\kappa, S_j^{(F)})$

- Ist die Bewertung der am besten bewerteten Hypothese über einem Schwellwert, wird diese Hypothese als das intendierte Objekt angesehen und dem Benutzer mit Hilfe des Synthesemoduls mitgeteilt.

- Sprachliche Rückmeldungen des Benutzers verändern die Bewertungen J der in der ausgewählten Hypothese vertretenen Merkmale:

$$J(M_{ijk}) \leftarrow J(M_{ijk}) \cdot \alpha \qquad \begin{cases} \alpha < 1 & \text{falls negative Rückmeldung} \\ \alpha > 1 & \text{falls positive Rückmeldung} \end{cases}$$

Durch diese Art der Hypothesenauswahl werden verschiedenste Informationen auf unterschiedlichen Zeitskalen ausgenutzt. Damit wird einerseits die Auswahl einer Hypothese für die aktuelle Anfrage optimiert. Andererseits wird auch die Objektdatenbank durch inkrementelle Bewertungsoptimierungen für zukünftige Anfragen verbessert.

8 Systemevaluierung

In den bisherigen Kapiteln wurde ein interaktives System zum Lernen von dem System bisher unbekannten Objekten vorgestellt. Ein Benutzer kann sprachliche Anfragen nach der Lokalisation und Identifikation von Objekten stellen. Durch deiktische Gesten kann er die Anfrage weiter einschränken und präzisieren. Das System versucht die Anfragen korrekt zu beantworten. Gelingt dies nicht, wird eine neue Ansicht des Objektes gelernt. Das Mißlingen kann dabei sowohl das System nach dem Finden keiner passenden Objekthypothese als auch der Benutzer explizit durch eine nachträgliche sprachliche Korrektur des Systems bekanntgeben.

Verschiedene Teile des Gesamtsystems wurden in den bisherigen Kapiteln sowohl qualitativ als auch quantitativ ausgewertet. Im einzelnen waren dies die Gestenerkennung, die verschiedenen Objekterkenner und die Hypothesenkombination. Die Ergebnisse waren im Hinblick auf das Gesamtsystem sehr vielversprechend. Diese Einzelergebnisse sagen allerdings noch nichts über die Gesamtperformanz des Systems aus. Erst durch die Interaktion der Einzelkomponenten kann das Gesamtsystem die vom Benutzer gestellten Aufgaben lösen. Erst eine gemeinsame Betrachtung des vollständigen Systems im normalen Einsatz mit realen Eingabedaten gibt daher Auskunft über die Adäquatheit des gewählten Ansatzes.

In diesem Kapitel werden vier verschiedene Experimente und die dabei erzielten Ergebnisse vorgestellt. Bei allen Experimenten kam jeweils das vollständige in Kapitel 3 beschriebene System zum Einsatz. Mit dem System unterschiedlich erfahrene Benutzer haben die Experimente durchgeführt. Im Rahmen eines Pretestes hat die ersten beiden Experimente ein mit dem System erfahrener Benutzer durchgeführt. Zwei weitere Experimente wurden von insgesamt elf Versuchspersonen absolviert, die größtenteils das System vorher nicht kannten. Die Ergebnisse eines weiteren mit dem System durchgeführten Testes finden sich in [Löm02].

8.1 Ergebnisse eines Pretestes

In einem ersten Pretest geringeren Umfangs sollte die Eignung des Systems zum Lernen, Lokalisieren und Identifizieren von Objekten getestet werden. Dazu wurden zwei unterschiedliche Experimente mit je einer Versuchsperson, die sich gut mit dem System auskannte, durchgeführt. Als Szenario wurde ein an der Wand stehendes Re-

gal in einem Büro gewählt. Das Büro war durch die normale Bürobeleuchtung bestehend aus vier Neonröhren beleuchtet. Beobachtet wurde das Regal von schräg oben durch eine Kamera vom Typ Sony EVI-D31. Sowohl die Gestenerkennung als auch die Objekterkennung hat diese Kamera benutzt. Die Gestenerkennung hat dabei mit Bildern der Größe 189x139 Pixel gearbeitet. Für die Objekterkennung wurden Bilder der Größe 378x278 Pixel aufgenommen. Abbildung 8.1 zeigt einige Bilder, die die Kamera für die Objekterkennung während der beiden Experimente aufgenommen hat. Sprache wurde während der Experimente über ein Nahbesprechungsmikrophon, ein drahtloses Sennheiser 1083-U, in Mono mit 16 Bit und einer Abtastfrequenz von 16 Kiloherz aufgenommen.

Abbildung 8.1: Beispielbilder des Szenarios, die die Kamera während des Pretestes für die Objekterkennung aufgenommen hat.

Bei den beiden Experimenten kamen insgesamt drei Rechner zum Einsatz. Für die Gestenerkennung wurde der OSF Alpha Rechner mit 433 MHz benutzt, der in Tabelle 5.1 auf Seite 81 beschrieben ist. Das Kontrollmodul lief zusammen mit der Objekterkennung auf einem PC mit 2.4 GHz, beschrieben in der selben Tabelle 5.1. Auf Grund der technischen Gegebenheiten lief die Spracherkennung auf einem eigenen Rechner, einer OSF Alpha mit 667 MHz (siehe Tabelle 6.1, Seite 113).

Experiment 1: Lernen und Lokalisieren

Das Ziel des ersten Experimentes war das Lernen und Lokalisieren von sechs unterschiedlichen Objekten, die jeweils sprachlich durch eine Anfrage benannt wurden. Die Objektdatenbank war in diesem Experiment initial leer. Gelernt werden sollten die in Abbildung 8.2 dargestellten sechs Zeitschriften. Während des Experimentes wurden Anfragen wie “Nimm den Computer.” an das System gestellt. Dabei wurden nur Wörter benutzt, die im Lexikon des Spracherkenners vorhanden waren. Da das zu suchende Objekt angegeben war, mußte eine Objektlokalisation durchgeführt werden. Falls das System ein Objekt nicht finden konnte oder ein falsches Objekt gefunden hatte, wurde eine neue Objektansicht gelernt. Bei einem Fehler des Spracherkenners wurde die Anfrage wiederholt. Als Spracherkennungsfehler wurden dabei nur solche gezählt, bei denen die semantische Interpretation der Äußerung eine korrekte Aus-

wertung nicht mehr zugelassen hat. Beispiele hierfür waren das Erkennen keines oder des falschen Objektes oder das Erkennen eines nicht zutreffenden Objektattributes. In die weitere Auswertung gingen Spracherkennungsfehler nicht mit ein. Nach dem Stellen von Anfragen für alle Objekte wurden die Objekte jeweils innerhalb der Szene neu angeordnet.

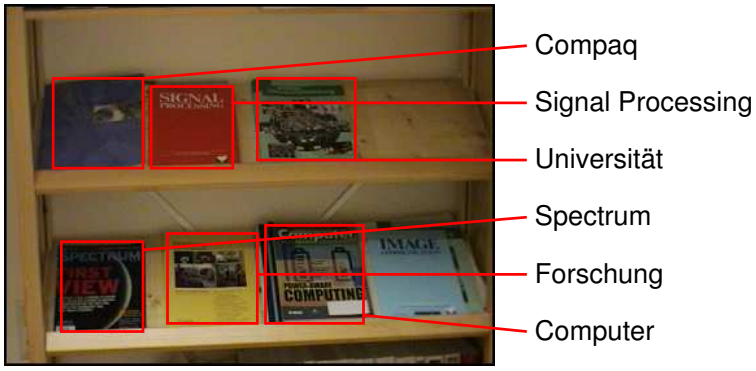


Abbildung 8.2: Alle Objekte – sechs unterschiedliche Zeitschriften – die im Pretest gelernt, lokalisiert und identifiziert werden sollten.

Tabelle 8.1 zeigt die erzielten Ergebnisse des ersten Experimentes. In der Tabelle sind nur absolute Werte und damit keine Erkennungsraten angegeben. Eine zentrale Eigenschaft des Systems ist das fortwährend inkrementelle Lernen. Eine Erkennungsrate, die sich durch Normierung auf die durchgeführten Experimente ergibt, sollte sich daher mit der Anzahl der durchgeführten Experimente verbessern. Gleichzeitig gibt sie aber die Anzahl der Experimente nicht mehr wieder. Daher hat dieser Wert in den hier durchgeführten Experimenten eine nur begrenzte Aussagekraft. Besonders stark tritt dieser Sachverhalt bei den Experimenten auf, die mit einer initial leeren Objektdatenbank starten. Daher sind in diesen Fällen nur absolute Werte angegeben. Auf Grund der initial leeren Objektdatenbank mußte auch zumindest die erste Suchanfrage für jedes Objekt fehlschlagen. Ohne eine gelernte Ansicht kann keine Anfrage nach einem Objekt korrekt beantwortet werden. Für vier der sechs Objekte trat nur dieser initiale erste Fehler auf, alle weiteren Anfragen konnten korrekt beantwortet werden. Bei den restlichen zwei Objekten war eine weitere Ansicht nötig. Die Bestimmung der korrekten Region während des Lernens einer neuen Ansicht und die Berechnung der Objektmerkmale waren in allen acht Fällen korrekt. Die Gestenerkennung hat korrekterweise keine deiktischen Gesten erkannt. Spracherkennungsfehler kamen in diesem Experiment ebenfalls nicht vor.

Experiment 2: Identifizieren

Im zweiten Experiment waren die Auswertung der Gestenerkennung und die Identifikation von Objekten das Ziel. Das Experiment startete mit der Objektdatenbank

	Objekte		Ansichten	
	Erkannt	Fehler	Anzahl gelernt	Fehler Lernen
Compaq	6	1	1	0
Signal Proc.	6	1	1	0
Universität	6	1	1	0
Spectrum	6	1	1	0
Forschung	6	2	2	0
Computer	6	2	2	0
Summe	36	8	8	0
Gestenerkennung	0 falsch positiv erkannte Gesten			

Tabelle 8.1: Ergebnisse des Lernens und Lokalisierens bisher unbekannter Objekte währende des Pretestes. Die Tabelle gibt absolute Werte an. Da mit einer leeren Objektdatenbank gestartet wurde, mußte für jedes Objekt minimal ein Fehler auftreten, da minimal eine Objektansicht gelernt werden mußte.

aus dem ersten Experiment. Bei allen Anfragen dieses Experimentes wurden unspezifische Objektbezeichner wie in “Zeige mir das Teil.” benutzt. Gleichzeitig wurde eine Zeigegeste ausgeführt. Bei der Hautfarbsuche kam in diesem Experiment der gaußbasierte Klassifikator zum Einsatz. Trainiert wurde er durch eine interaktive Modellgenerierung mittels Winken vor der Kamera. Abbildung 8.3 zeigt als ein Beispiel eine der durchgeführten Gesten, die auf Grund der Geste erstellte Aufmerksamkeitskarte und das auf Grund der Geste ausgewählte Objekt. Auf Grund des unspezifizierten Objektes hatte das System in diesem Fall nur die Gestikinformation zur Verfügung. Es mußte in diesem Experiment zur Beantwortung der Anfrage eine Suche über die vollständige Objektdatenbank unter Berücksichtigung der aus der Zeigegeste generierten Aufmerksamkeitskarte durchgeführt werden. Nach dem Stellen von Anfragen für alle Objekte wurden die Objekte wie im ersten Experiment jeweils innerhalb der Szene neu angeordnet.

Tabelle 8.2 zeigt die erzielten Ergebnisse des Experimentes, wobei sowohl absolute als auch auf die Anzahl der Versuche normierte relative Werte angegeben sind. Waren bei einer Anfrage sowohl die Gestenerkennung als auch die Objekterkennung fehlerfrei, wurde die Anfrage als “Korrekt erkannt” in die Tabelle aufgenommen. Trat ein Fehler der Gestenerkennung auf, war es unabhängig von einer gegebenenfalls korrekten Objekterkennung ein Gestikfehler. Falls nur ein falsches Objekt erkannt wurde, die Geste aber korrekt interpretiert wurde, wurde die Anfrage als ein Fehler der Objekterkennung gekennzeichnet.

Wurde im Experiment für eine Anfrage ein falsches Objekt gefunden, wurde das System wie im ersten Experiment korrigiert. In den drei aufgetretenen Fällen wurde

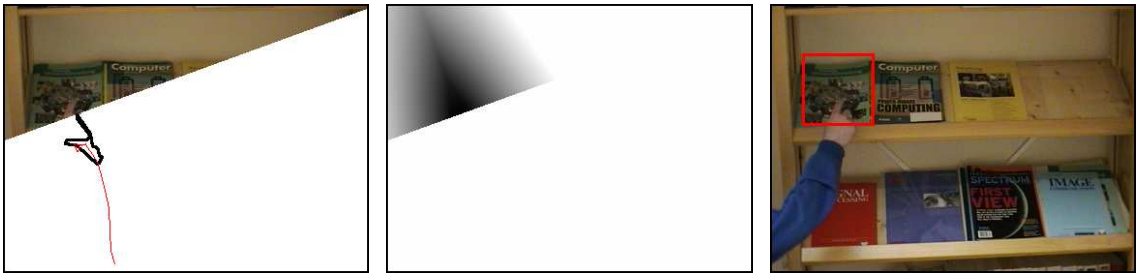


Abbildung 8.3: Eine im Pretest durchgeführte Geste, die auf Grund der Geste erstellte Aufmerksamkeitskarte und das ausgewählte Objekt.

	#			%		
	Korrekt erkannt	Fehler		Korrekt erkannt	Fehler	
		Gestik	Objekt		Gestik	Objekt
Compaq	5	2	0	71.4	28.5	0
Signal Proc.	5	1	0	83.3	16.6	0
Universität	5	1	1	71.4	14.2	14.2
Spectrum	5	0	1	83.3	0	16.6
Forschung	5	0	0	100	0	0
Computer	5	0	1	83.3	0	16.6
Summe	30	4	3	81	10.8	8.1
Gestenerkennung	0 falsch positiv erkannte Gesten					

Tabelle 8.2: Ergebnisse des Identifizierens von Objekten mit Hilfe einer Geste. Die Tabelle gibt absolute Werte und auf die Anzahl der Versuche gemittelte relative Werte an. Es wurde die Datenbank des ersten Experimentes des Pretestes benutzt.

dabei allerdings im nachfolgenden Lernschritt keine neue Ansicht in der Datenbank abgelegt. Bevor eine neu gelernte Ansicht in der Datenbank abgelegt wird, werden zuerst alle für die ursprüngliche Anfrage generierten Hypothesen betrachtet. Das System versucht unter diesen eine Hypothese zu finden, die sowohl den gleichen Objekttyp wie die neue Ansicht hat als auch an der gleichen Position im Bild liegt. Gelingt dies, werden nur die Bewertungen der für diese Hypothese eingesetzten Merkmale verbessert. Die neue Ansicht wird nicht in der Datenbank abgelegt. Dieses Vorgehen vermeidet das Generieren doppelter und damit redundanter Ansichten innerhalb der Datenbank. In den drei im Experiment aufgetretenen Fällen konnte jeweils eine passende Hypothese gefunden werden. Damit wurden neben den von Anfang an vorhandenen acht Objektansichten keine weiteren Ansichten generiert.

Auch dieses Experiment hat vielversprechende Ergebnisse gezeigt. In insgesamt

sechs Fällen mußte auf Grund von Spracherkennungsfehlern die Anfrage wiederholt werden. Nur in vier Fällen konnte die Geste nicht erkannt werden, in drei weiteren wurde das falsche Objekt identifiziert. In 30 Fällen konnte die Anfrage vollständig beantwortet werden, obwohl keine sprachlichen Objekteinschränkungen gegeben waren. In beiden Experimenten hat die Gestenerkennung dabei keinerlei falsch positive Fehler verursacht, es wurde bei keiner der Anfragen eine nicht durchgeführte Geste erkannt. Damit waren die Ergebnisse vielversprechend genug, um das System in einem Szenario größerer Flexibilität mit mehreren mit dem System nicht vertrauten Versuchspersonen zu testen.

8.2 Ergebnisse des Haupttestes

Nachdem das System seine Adäquatheit im Rahmen eines Pretestes gezeigt hat, wurde es im Rahmen eines Haupttestes bei größerer Flexibilität des Versuchsaufbaus mit mehreren Versuchspersonen evaluiert. Dazu wurden wie im Pretest zwei unterschiedliche Experimente durchgeführt. Zu Vergleichszwecken wurde zusätzlich jedes der Experimente sowohl von einer Gruppe mit dem System nicht vertrauter, aber kooperativer Benutzer als auch mit einem mit dem System vertrauten Benutzer durchgeführt. Als Szenario wurde für diesen Test ein Tisch mit darauf liegenden Objekten gewählt. Abbildung 8.4 zeigt ein Bild des Szenarios.

In diesem Test kamen getrennte Kameras für die Gestenerkennung und für die Objekterkennung zum Einsatz. Damit konnte der sichtbare Bereich für die Gestenerkennung im Vergleich zum Bereich der Objekterkennung vergrößert werden. Zur Erkennung von Zeigegesten benötigt die Gestenerkennung Trajektorien der Hände, die beim Zeigen auf Objekte am Bildrand von außerhalb des Bildes aber im Bild nicht sichtbar werden. Durch den vergrößerten Bereich werden auch in diesem Fall zumindest teilweise Trajektorien im Bild der zweiten Kamera sichtbar. Beide Kameras waren vom Typ Sony EVI-D31. Die Gestenerkennung hat mit Bildern der Größe 189x139 Pixel gearbeitet. Für die Objekterkennung wurden Bilder der Größe 378x278 Pixel aufgenommen. Einige Bilder, die diese Kameras geliefert haben, zeigt Abbildung 8.4. Zwei verschiedene Hintergründe wurden während des Testes eingesetzt, ein aus Holzplatten bestehender Hintergrund und ein weißes Tuch mit einem blauen Gittermuster. Sprache wurde wie im Pretest über ein Nahbesprechungsmikrophon, ein drahtloses Sennheiser 1083-U, in Mono mit 16 Bit und einer Abtastfrequenz von 16 Kiloherz aufgenommen.

Für die Experimente in diesem Test wurden jeweils vier Rechner eingesetzt. Für die Gestenerkennung kam der in Tabelle 5.1 auf Seite 81 beschrieben OSF Alpha Rechner mit 433 MHz zum Einsatz. Da sich der Grabber der zweiten Kamera in einem weiteren OSF Alpha Rechner mit 433 MHz befand, lief hier ein Bildserver, der Bilder für die Objekterkennung zur Verfügung gestellt hat. Das Kontrollmodul lief zusammen mit der Objekterkennung auf einem Pentium IV PC mit 2.4 GHz, beschrieben in der

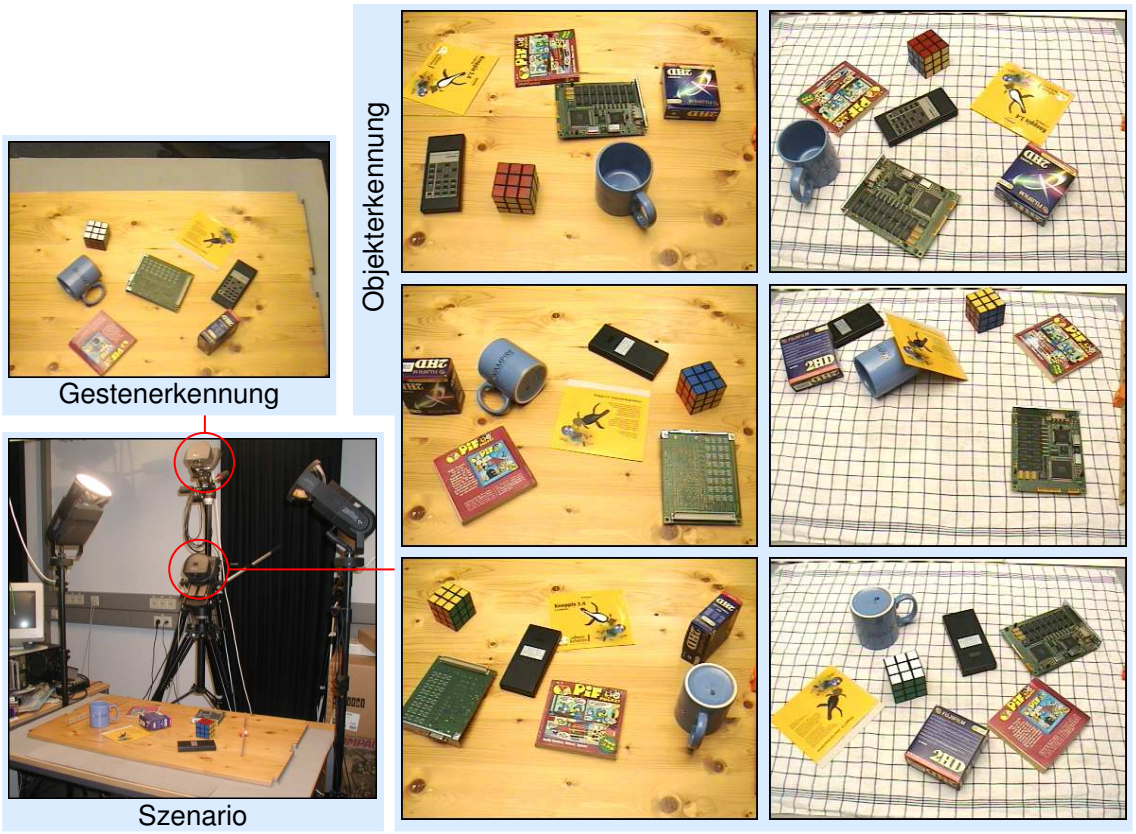


Abbildung 8.4: Das Szenario des Haupttestes im Überblick und aus der Sicht der Gestenerkennung und der Objekterkennung. Im Gegensatz zum Pretest kamen in diesem Test zwei getrennte Kameras zum Einsatz.

selben Tabelle 5.1. Die Spracherkennung lief auf Grund der technischen Gegebenheiten ebenfalls auf einem eigenen Rechner vom Typ OSF Alpha mit 667 MHz (siehe Tabelle 6.1, Seite 113).

Experiment 1: Lernen und Lokalisieren

Die Aufteilung der beiden Experimente des Haupttestes entsprach der Aufteilung des Pretestes. Auch hier sollte das erste Experiment das Lernen und Lokalisieren von sprachlich benannten Objekten untersuchen. Die Objektdatenbank war in diesem Experiment initial leer. Gelernt werden sollten die in Abbildung 8.5 dargestellten sieben Objekte. Dazu wurde den Versuchspersonen, kooperative mit dem System nicht vertraute Informatiker, die Funktionsweise des Systems und der Ablauf des Experimentes erklärt. Das Experiment wurde anschließend ohne Testlauf für die Versuchspersonen durchgeführt. Während des Experimentes wurden Anfragen wie “Nimm den Würfel.” an das System gestellt, um damit das Lernen und die Lokalisation von Objekten zu

evaluieren. Dabei wurden nur Wörter aus dem Lexikon des Spracherkenners benutzt. Nach dem Stellen von Anfragen für alle Objekte wurden die Objekte von den Versuchspersonen innerhalb der Szene verschoben und um beliebige Achsen rotiert. Das weitere Vorgehen war identisch zum ersten Experiment des Pretestes.

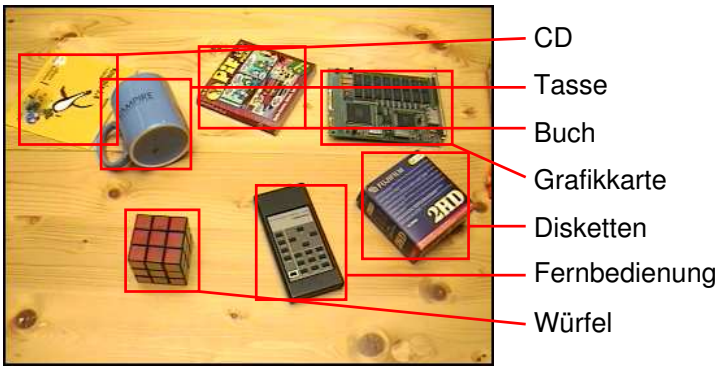


Abbildung 8.5: Alle sieben Objekte, die im Haupttest gelernt, lokalisiert und identifiziert werden sollten.

Dieses erste Experiment haben zehn nicht mit dem System vertraute Versuchspersonen durchgeführt und dabei insgesamt 421 Anfragen an das System gestellt. Während des Experimentes wurden beide Hintergründe verwendet. Zu Vergleichszwecken wurde das Experiment je einmal auf beiden Hintergründen von einer mit dem System vertrauten Versuchsperson durchgeführt. Hierbei wurden insgesamt 171 Anfragen gestellt. Die auf eine Versuchsperson beziehungsweise ein Experiment gemittelten Ergebnisse beider Gruppen zeigt Tabelle 8.3. Auch hier sind wieder wie beim ersten Experiment des Pretestes absolute Werte und keine normierten Erkennungsraten angegeben, da mit der Versuchsanzahl normierte Erkennungsraten das inkrementelle Lernen des Systems verdecken würden. Da die Ergebnisse nicht signifikant vom Hintergrund abhängen, sind keine nach Hintergrund aufgeteilten Ergebnisse angegeben.

Wie auch im ersten Experiment des Pretestes mußten auch in diesem Experiment initial Fehler für jedes Objekt auftreten, da minimal eine Ansicht gelernt werden mußte, bevor überhaupt irgend etwas erkannt werden konnte. Der Würfel hat von jeder seiner sechs Seiten auf Grund seiner Farbe ein unterschiedliches Aussehen. Daher entspricht es den Erwartungen, daß dieses Objekt vergleichsweise viele Ansichten benötigt. Viele der weiteren Objekte kamen mit nur wenig mehr als einer Ansicht aus. Die Ergebnisse der beiden Benutzergruppen decken sich im wesentlichen. Die etwas größere Menge von Ansichten bei dem erfahrenen Benutzer war auf Grund der größeren Variabilität durch die größere Menge von Anfragen und damit Objektlagen zu erwarten.

Das Auftreten von geringfügig mehr Fehlern bei der Objekterkennung im Vergleich zu der Anzahl gelernter Ansichten hat zwei Gründe. Einerseits haben Spracherken-

	Unerfahrene Versuchspersonen				Erfahrene Versuchsperson			
	Objekte		Ansichten		Objekte		Ansichten	
	Erkannt	Fehler	Anzahl gelernt	Fehler Lernen	Erkannt	Fehler	Anzahl gelernt	Fehler Lernen
Würfel	4.9	2.4	2.1	0	10.5	4	3.5	0
Fernbedienung	4.8	2	1.8	0	10	2	2	0
Buch	4.4	1.1	1.1	0	10	1	1	0
CD	4	1.1	1	0	10	1	1	0
Tasse	5	1.2	1.1	0	10	1.5	1.5	0
Disketten	4.5	1.3	1.3	0	10	2.5	2	0
Grafikkarte	4.3	1.1	1.1	0	10	3	3	0
Summe	31.9	10.2	9.5	0	70.5	15	14	0
Gestenerkennung	0 falsch positiv erkannte Gesten							

Tabelle 8.3: Ergebnisse des Lernens und Lokalisierens bisher unbekannter Objekte während des Haupttestes. Die Tabelle gibt auf eine Person gemittelte absolute Werte an. Da mit einer leeren Objektdatenbank gestartet wurde, mußte für jedes Objekt minimal ein Fehler auftreten, da minimal eine Objektansicht gelernt werden mußte.

nungsfehler, die ansonsten in der Tabelle nicht weiter aufgeführt sind, in wenigen Fällen das Lernen verhindert. Andererseits hat das System zum Teil wie auf Seite 155 beschrieben nicht eine neu gelernte Ansicht übernommen, sondern Bewertungen von schon in der Objektdatenbank befindlichen Merkmalen einer alternativen Hypothese erhöht. Die Bestimmung der korrekten Region während des Lernens einer neuen Ansicht und die Berechnung der Objektmerkmale war in allen Fällen korrekt. Die Gestenerkennung hat korrekterweise keine deiktischen Gesten erkannt. Spracherkennungsfehler traten durchschnittlich in 3.4 Fällen bei den unerfahrenen Benutzern und in fünf Fällen bei dem erfahrenen Benutzer auf.

Experiment 2: Gesten zur Identifikation und Lokalisation

Das zweite Experiment des Haupttestes hatte die Benutzung von Gesten bei der Lokalisation und Identifikation von Objekten zum Ziel. An Objekten kamen die sieben schon im ersten Experiment verwendeten Objekte zum Einsatz. Drei der Objekte, der Würfel, die Disketten und die Tasse, waren doppelt vorhanden. Das Experiment startete mit der Objektdatenbank aus dem ersten Experiment. Jeweils die Versuchspersonen, die auch diese Datenbank interaktiv erstellt hatten, führten auch dieses zweite Experiment durch. Vor dem Experiment wurde ihnen die trajektorienbasierte Funktionsweise der Gestenerkennung erläutert. Sie wurden aufgefordert, bei jeder Anfrage eine Geste zu benutzen. Drei mögliche Anfragearten wurden ihnen vorgestellt. Dies waren im einzelnen, sortiert nach aufsteigender Schwierigkeit:

1. Die Benennung eines nur einmal vorhandenen Objektes mit einer objektspezifischen Bezeichnung: “Zeige mir das Buch.”. Auch ohne die Geste hätte die Anfrage beantwortet werden können.

2. Die Benennung eines doppelt vorhandenen Objektes mit einer objektspezifischen Bezeichnung: “Zeige mir den Würfel.”. Die Geste wird für die Disambiguierung der zwei Objekte benötigt.

3. Die Benennung eines Objektes mit einer allgemeinen, nicht objektspezifischen Bezeichnung: “Zeige mir das Teil.”. Durch die Sprache wird die Suche in diesem Fall nur initialisiert. Die Geste stellt die wesentliche Information zur Verfügung.

Die Versuchspersonen konnten zwischen diesen Anfragearten frei wählen. Als Hautfarbklassifikator für die Gestenerkennung wurde in diesem Experiment der Polynomklassifikator eingesetzt. Abbildung 8.6 zeigt als ein Beispiel drei der durchgeführten Gesten und die auf Grund der Geste ausgewählten Objekte. Nach dem Stellen von Anfragen für alle Objekte wurden die Objekte wie im ersten Experiment von den Versuchspersonen innerhalb der Szene verschoben und um beliebige Achsen rotiert.

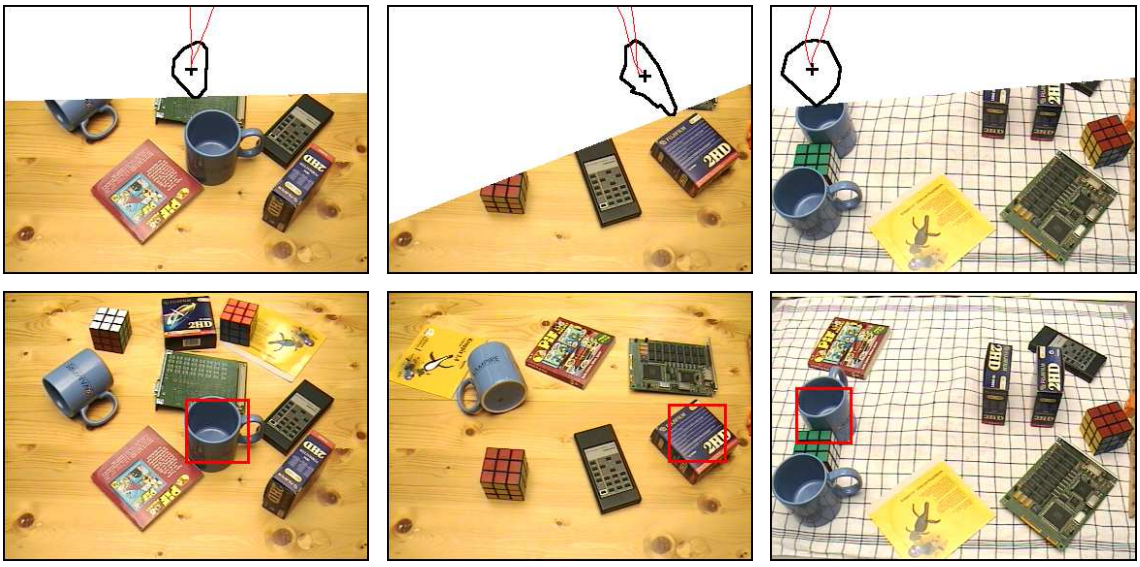


Abbildung 8.6: Drei Beispiele für im Haupttest durchgeführte Gesten und die auf Grund der Gesten ausgewählten Objekte.

Neun nur durch das erste Experiment mit dem System vertraute Versuchspersonen haben dieses Experiment durchgeführt und dabei insgesamt 383 Anfragen an das System gestellt. Zu Vergleichszwecken wurde das Experiment je einmal auf beiden Hintergründen von einer mit dem System vertrauten Versuchsperson durchgeführt.

Hierbei wurden insgesamt 161 Anfragen gestellt. Tabelle 8.4 zeigt die auf eine Versuchsperson beziehungsweise ein Experiment gemittelten Ergebnisse beider Gruppen. Zusätzlich sind die Ergebnisse auch auf die Anzahl der Versuche gemittelt angegeben. Waren bei einer Anfrage sowohl die Gestenerkennung als auch die Objekterkennung fehlerfrei, wurde die Anfrage als "Korrekt erkannt" in die Tabelle aufgenommen. Trat ein Fehler der Gestenerkennung auf, war es unabhängig von einer gegebenenfalls korrekten Objekterkennung ein Gestikfehler. Falls nur ein falsches Objekt erkannt wurde, die Geste aber korrekt interpretiert wurde, wurde die Anfrage als ein Fehler der Objekterkennung gekennzeichnet.

Unerfahrene Versuchspersonen									
	#				%				
	Korrekt erkannt	Fehler			Korrekt erkannt	Fehler			# Ansichten
		Gestik	G.FP	Objekt		Gestik	G.FP	Objekt	
Würfel	7.8	3.7	0	1.6	59.8	28.2	0	11.9	2.3
Fernbedienung	3.3	0.7	0	0.2	78.9	15.7	0	5.2	1.9
Buch	2.6	0.7	0	0	79.3	20.6	0	0	1
CD	3.3	0.7	0	0	83.3	16.6	0	0	1
Tasse	7.2	1	0	0.1	86.6	12.0	0	1.3	1.2
Disketten	5.1	1	0	0	83.6	16.3	0	0	1.3
Grafikkarte	3.2	0.4	0	0	87.8	12.1	0	0	1.1
Summe	32.6	8.1	0	1.9	76.5	19.0	0	4.4	9.9
Erfahrene Versuchsperson									
Würfel	14.5	0	0	2.5	85.2	0	0	14.7	3.5
Fernbedienung	8	0.5	0	0	94.1	5.8	0	0	2
Buch	8	0.5	0	0	94.1	5.8	0	0	1
CD	8	0	0	0	100	0	0	0	1
Tasse	14	0	0	0	100	0	0	0	1.5
Disketten	14	2	0	0.5	84.8	12.1	0	3	2
Grafikkarte	8	0	0	0	100	0	0	0	3
Summe	74.5	3	0	3	92.5	3.7	0	3.7	14

Tabelle 8.4: Ergebnisse des Identifizierens von Objekten mit Hilfe einer Geste. Die Tabelle gibt auf eine Person gemittelte absolute Werte und auf die Anzahl der Versuche gemittelte relative Werte an. Es wurde die Datenbank des ersten Experimentes des Haupttestes benutzt.
G.FP: Anzahl/Rate falsch positiv erkannter Gesten.

Während des Experimentes wurden von der unerfahrenen Benutzergruppe insgesamt zwei neue Ansichten des Würfels und eine neue Ansicht der Tasse fehlerfrei gelernt. Die restlichen Fehler wurden von den Versuchspersonen entweder zu einem

kleineren Teil nicht korrigiert oder das System hat Bewertungen von schon in der Datenbank befindlichen Merkmalen einer alternativen Hypothese erhöht. Spracherkennungsfehler traten durchschnittlich in 1.7 Fällen bei den unerfahrenen Benutzern und in 1.5 Fällen bei dem erfahrenen Benutzer auf.

Insgesamt zeigen die Ergebnisse, daß die Gesten und die Objekte relativ sicher erkannt werden konnten. Wie in Abschnitt 5.4.2 angegeben, sollte insbesondere auch die Rate der falsch positiv erkannten Gesten möglichst gering sein. Dies wurde mit null fälschlicherweise erkannten Gesten voll erreicht. Bei genauerer Betrachtung zeigen die Ergebnisse einige interessante Besonderheiten. Ein großer Teil der Objekterkennungsfehler der unerfahrenen Benutzergruppe sind durch den Würfel verursacht. Die Hälfte aller aufgetretenen Fehler trat dabei bei einer einzigen Versuchsperson auf. Bei dieser Person wurde bei neun Anfragen der Würfel korrekt erkannt, bei weiteren sieben Anfragen wurde er nicht erkannt. Diese Person hatte nur eine einzige Ansicht für den sehr variabel erscheinenden Würfel gelernt, wodurch die hohe Fehlerzahl in diesem Fall zu erwarten war. Betrachtet man nur die Ergebnisse der acht anderen Personen, ergeben sich gemittelte Werte von 7.6 korrekt erkannten Würfeln und 0.9 falsch erkannten Würfeln, was einer deutlich besseren Erkennungsleistung entspricht. Der relative Fehler bei den Würfeln sinkt in diesem Fall von 11.9% auf 7%. Gleichzeitig steigt die Anzahl der für den Würfel gelernten Ansichten auf 2.5.

Auffällig ist auch die im Vergleich zum erfahrenen Benutzer größere Menge von Fehlern bei der Gestikerkennung innerhalb der unerfahrenen Benutzergruppe. Zwei Fehlerarten traten hierbei in den Vordergrund. Einerseits müssen innerhalb des Systems nicht einer Zeigegeste entsprechende Handbewegungen zurückgewiesen werden. Dies war innerhalb aller während der Evaluierung durchgeführten Experimente sehr erfolgreich. Es wurde bei keiner der Anfragen eine nicht durchgeführte Geste erkannt. Andererseits hat es allerdings auch zur Rückweisung von Gesten geführt. Bei zwei der Versuchspersonen wurden überproportional viele Gesten auf Grund des zeitlichen Abstandes zwischen dem definiten Artikel und der Trajektorie zurückgewiesen. Eine der beiden Personen hat zwar normal gezeigt, zeitlich entsprach dies allerdings zum Teil nicht dem aus dem Experiment aus Abschnitt 5.4.1, Seite 68 erstellten Modell. Die andere Versuchsperson hat sich häufiger während des Zeigens mit dem kompletten Oberkörper vorgebeugt, wodurch das Zeigen verlangsamt wurde und eine Integration mit der Sprache zum Teil verhindert wurde.

Eine weitere Quelle für Fehler war das Zeigen auf Objekte, die dicht am vorderen oder seitlichen Bildrand lagen. Gleichzeitig kam auch die Bewegung der Hände aus dieser Richtung. Für eine trajektorienbasierte Erkennung von Gesten ist dies auf Grund der nur sehr kurz sichtbaren Hände eine an sich nicht lösbare Aufgabe. Durch die Benutzung einer eigenen Kamera mit einem erweiterten Sichtfeld für die Gestenerkennung sind zwar auch in diesem Fall kurze Handtrajektorien im Bild der zweiten Kamera sichtbar. Insbesondere bei schneller Handbewegung umfaßt die Trajektorie allerdings trotzdem maximal nur wenige Frames. Nur bei langsamem und genauem Zeigen kann in diesem Fall die korrekte Analyse der Zeigegeste gelingen.

Dieses Problem des Szenarienaufbaus hat der erfahrene Benutzer im Gegensatz zu den unerfahrenen Benutzern entsprechend beachtet.

8.3 Zusammenfassung

In diesem Kapitel wurden zwei Tests mit insgesamt vier Experimenten vorgestellt, die die Evaluierung des in dieser Arbeit vorgestellten Systems zum Lernen von unbekannten Objekten zum Ziel hatten. Im ersten Test wurde das Lernen, Lokalisieren und Identifizieren von in einem Regal liegenden Zeitschriften betrachtet. In einem zweiten Test sollten unterschiedliche, auf einem Tisch liegende Objekte gelernt und auch lokalisiert und identifiziert werden. Der erste Test wurde von einer mit dem System vertrauten Versuchsperson durchgeführt. Den zweiten Test absolvierte sowohl eine erfahrene Versuchsperson als auch eine Gruppe kooperativer, mit dem System aber nicht vertrauter Versuchspersonen.

Nach kurzer Einweisung konnte auch die mit dem System nicht vertraute Gruppe mit dem System erfolgreich interagieren. Das Lernen neuer Ansichten hat sich während der Tests dabei als sehr stabil herausgestellt. Unterscheidet sich das Objekt vom Hintergrund, wie dies bei den unterschiedlichen in den Tests benutzten Objekten der Fall war, kann es auch sicher segmentiert werden. Die Analyse der Greifaktion hilft dabei, die korrekte Region bei unterschiedlichen Szenenmodifikationen auszuwählen. Nach dem Lernen weniger Ansichten konnten die Objekte korrekt erkannt werden. Wird das System dagegen zu wenig instruiert, indem auftretende Fehler nicht korrigiert werden, kann sich das System auch nicht an das zu lernende Objekt ausreichend anpassen. Bei nachfolgenden Anfragen nach dem Objekt kommt es dann zu weiteren Fehlern. Diese Fehler zeigen auch die Effektivität des Lernverfahrens. Wird es durch die Korrektur von Fehlern durchgeführt, bringt es auch Vorteile.

Durch die Benutzung von deiktischen Gesten können zusätzliche Informationen bei der Lokalisation und Identifikation von Objekten gegeben werden. Bei der Gestenerkennung kam es während der Tests zu keinerlei falsch positiven Fehlern, es wurde bei keiner der Anfragen eine nicht durchgeführte Geste erkannt. Die durchgeführten Gesten konnten recht stabil erkannt werden. Hier erzielt allerdings die mit dem System vertraute Versuchsperson im Vergleich mit der unerfahrenen Gruppe im Gegensatz zu der Objekterkennungsleistung eine signifikant bessere Erkennungsrate. Ein wichtiger Grund hierfür liegt am Szenario. Bei Gesten, die von außerhalb des Bildes auf Objekte am Bildrand durchgeführt werden, kann nur eine sehr kurze Trajektorie beobachtet werden. Damit ist eine trajektorienbasierte Erkennung von Gesten in diesem Fall nur schwer durchführbar. Sie kann nur bei langsamer und genauer Ausführung der Geste gelingen. Eine zusätzlich integrierte Erkennung von Gesten auf Grund der Postur der Hand könnte hier die Rückweisungsrate bei der Gestenerkennung weiter senken.

9 Zusammenfassung und Ausblick

In den letzten Jahrzehnten gab es große Anstrengungen und auch signifikante Fortschritte in Bereichen wie der Objekterkennung, der Spracherkennung oder auch der Sprachverarbeitung [Chr03a]. Sowohl durch den Anstieg der verfügbaren Rechenleistung als auch durch verbesserte und neue Verfahren konnte dies erreicht werden. Im Bereich der integrierten kognitiven Systeme sind die Fortschritte dagegen merklich geringer. Erst in den letzten Jahren wurden die Aktivitäten in diesem Bereich verstärkt.

Diese Arbeit hat versucht, einen Beitrag in diesem Bereich der integrierten kognitiven Systeme zu liefern. Im folgenden Abschnitt wird eine Zusammenfassung der vorgestellten Arbeit gegeben. Anschließend werden sinnvoll erscheinende Anschlußarbeiten diskutiert.

9.1 Zusammenfassung

Während der zwischenmenschlichen Kommunikation werden häufig physikalische Objekte sprachlich referenziert. Für die Lokalisation und Identifikation der Objekte haben die Kommunikationspartner dabei mentale Modelle, die während der Kommunikation ständig aktualisiert und verbessert werden. Sowohl die Auswertung visueller als auch sprachlicher Informationen wird dabei berücksichtigt. Werden für einen Kommunikationspartner unbekannte Objekte referenziert, müssen die entsprechenden Modelle neu aufgebaut werden. Dies geschieht häufig interaktiv durch das Zeigen und Vorführen des Objektes. Auch für einen Serviceroboter sind dies erstrebenswerte Ziele, wenn er beispielsweise im Haushalt oder in anderen Bereichen erfolgreich unterstützend tätig sein soll. Idealerweise kann man dem Roboter einfach neue Objekte, die während des alltäglichen Einsatzes als Arbeitshilfe immer wieder auftreten, in einer natürlichen und komfortablen Weise zeigen und sprachlich beschreiben.

Im Rahmen dieser Arbeit wurde ein vollständiges System vorgestellt, welches einen ersten Schritt in die Richtung interaktivem, multimodalem, visuell basiertem Lernen unbekannter Objekte geht. Als zugrundeliegendes Paradigma läßt sich für das System das soziale Lernen in Verbindung mit einem Lernen durch Beispiele nennen. Der Benutzer tritt mit dem System in Interaktion, indem er das System nach Objekten befragt, es durch die Korrektur von Fehlern instruiert oder auch Fragen des Systems

nach Objekten, die dem System bisher unbekannt waren, beantwortet. Objekte legt das System in Form beliebig vieler aus unterschiedlichen Merkmalen bestehenden Ansichten in einer Datenbank ab. Das System ist domänenunabhängig. Es ist weder auf spezielle Objekte noch einen speziellen Hintergrund angewiesen.

Das Szenario des Systems wurde im Hinblick auf eine einfache und intuitive Beherrschbarkeit durch den Benutzer gewählt. Durch eine oder wahlweise zwei Farbkameras wird eine Szene beobachtet. Der Benutzer kann gesprochene Sprache benutzen, um dialoggestützt auf Objekte innerhalb der Szene zu referenzieren. Das System versucht, die referenzierten Objekte zu lokalisieren oder auch zu identifizieren. Deiktische Gesten und durchgeführte Greifaktionen des Benutzers werden als zusätzliche Informationsquellen in die Analyse integriert.

Kann das System eine Anfrage nach einem Objekt nicht beantworten, lernt es in Interaktion mit dem Benutzer eine neue Ansicht des Objektes. Dabei entfernt der Benutzer nach Aufforderung durch das System das Objekt aus der Szene. Zur Lokalisation des unbekannten Objektes wird je ein Bild von vor und nach dem Herausnehmen des Objektes betrachtet. Die Differenz der beiden Bilder erlaubt die Segmentierung des Objektes von einem unbekannten Hintergrund. Globale Helligkeitsänderungen werden durch eine Schwellwertanpassung bei der Differenzbildberechnung beachtet. Sekundäre Szenenänderungen können durch die Analyse der Greifaktion des Benutzers ignoriert werden. Es werden bevorzugt Szenenänderungen am Ziel der Greifaktion als neue Objektansichten übernommen.

Das komplette System ist sprachgesteuert. Die Dialogkomponente des Systems initiiert damit alle weiteren Aktionen. Zwei Arten von Anfragen verarbeitet der Dialog. Einerseits kann der Benutzer das interaktive Trainieren des im Modul Handverfolgung benutzten Hautfarbklassifikators anstoßen. Andererseits kann er Anfragen nach der Lokalisation und Identifikation von Objekten stellen. Neben Farb- und Größenadjektiven wird bei der Interpretation einer Anfrage auch eine Zeigegeste mit ausgewertet.

Auf Grund der Integration von Gestikinformationen und Objekterkennungsergebnissen in das Ablaufmodell des Dialogs läßt sich diese Komponente als ein integriertes System klassifizieren. Zur Erreichung größerer Robustheit gegenüber Erkennungsfehlern arbeiten die einzelnen Komponenten des Dialogsystems eng verzahnt. Syntaktisches und semantisches Wissen wird gemeinsam von der Spracherkennung bis zu der Semantikanalyse eingesetzt. Auch durch die Beschränkung des Parsings auf Schlüsselphrasen wird die Robustheit gesteigert. Die komplette Sprachkomponente ist leicht anpaßbar an andere Sprachen angelegt. Ein deutsches und ein englisches Demonstrationssystem wurde realisiert.

An Gesten werden das Zeigen auf Objekte und das Greifen und Herausnehmen von Objekten unterstützt. Beide Arten von Gesten werden durch die Analyse von Handtrajektorien ausgewertet, die wiederum durch das Verfolgen von segmentierten Händen mit Hilfe eines Kalmanfilters gewonnen werden. Hypothesen für die zu segmentierenden Hände werden durch eine Hautfarbklassifikation in allen mit Bewegung versehenen Bereichen des Kamerabildes bestimmt. Für die Farbsegmentierung kann

sowohl ein Polynomklassifikator als auch eine Klassifikation mit Hilfe einer Gaußverteilung zum Einsatz kommen. Eine Bewertung der unterschiedlichen Hautfarbregionen auf Basis der Farbe und der Bewegung in der unmittelbaren Umgebung der Region erlaubt die Auswahl der Regionen, die am wahrscheinlichsten Hände repräsentieren.

Eine Voraussetzung für eine Zeigegeste ist die Benutzung eines definiten Artikels in der sprachlichen Anfrage. Eine Zeigegeste wird in diesem Fall in der zeitlichen Umgebung des Artikels hypothetisiert und durch eine Analyse der Trajektorie verifiziert oder verworfen. Durch ein Experiment wurde der zeitliche Rahmen einer Zeigegeste zum definiten Artikel bestimmt. Eine Greifaktion wird in Abhängigkeit vom Dialogkontext hypothetisiert. Hat der Dialog nach dem Herausnehmen eines Objektes gefragt, erwartet das System auch die entsprechende Aktion. An Hand der aufgetretenen Trajektorien wird die Aktion verifiziert oder verworfen.

Um auch bei sich ändernder Beleuchtung arbeiten zu können, kann die Farbsegmentierung interaktiv durch einfaches Winken vor der Kamera neu trainiert werden. Keine weitere Hilfestellung des Benutzers ist dabei nötig.

Für die Generierung von Objekthypothesen für die Lokalisation und Identifikation von Objekten werden verschiedene visuelle und sprachliche Informationen ausgenutzt. Verschiedene auf Bildern arbeitende parallel eingesetzte Objekterkenner generieren initiale Hypothesen, indem sie das komplette Bild mit in der Datenbank gespeicherten Merkmalen von Objekten absuchen. Zeigegesten werden dabei mit in die Suche einbezogen, indem sie die bei der Suche entstehenden Wahrscheinlichkeitskarten modifizieren. Die Wahrscheinlichkeitskarten werden in einem weiteren Schritt soweit möglich kombiniert. Sprachlich geäußerte Größen- und Farbadjektive modifizieren die ermittelten Wahrscheinlichkeiten der Hypothesen durch den Einsatz von aus der Fuzzy Logik kommenden Zugehörigkeitsfunktionen. Größenadjektive werden dabei in Abhängigkeit zu allen Hypothesen ausgewertet, Farbadjektive in Abhängigkeit zu den für eine Hypothese eingesetzten Objekterkennern.

An Objekterkennern sind im System drei unterschiedliche realisiert. Zwei der Erkenner modellieren die Dichtefunktion eines Objektes durch Histogramme und benutzen den Histogrammschnitt für den Vergleich der Histogramme. Durch unterschiedliche Optimierungen des Histogrammschnitts kann eine Suche in einem kompletten Bild sehr schnell durchgeführt werden. An Merkmalen werden L*a*b*-Farbmerkmale im ersten Erkenner und verschiedene rotationsinvariante strukturbasierte Merkmale im zweiten Erkenner eingesetzt. Um zusätzlich globalere Strukturinformationen mit zu berücksichtigen, wurde zusätzlich ein auf farbsegmentierten Bildern basierender Erkenner betrachtet. Aus den segmentierten Bildern erstellt dieser Nachbarschaftsgraphen. Die Graphen repräsentieren die Regionen als Knoten, Kanten bilden die Nachbarschaft der Regionen ab. Ein fehlertoleranter Teilgraphvergleich zwischen einem Szenegraphen und einem Objektgraphen erlaubt das Lokalisieren und Identifizieren von Objekten in Szenen. Alle diese Erkenner sind sowohl translations- als auch rotationsinvariant. Dies ist eine wichtige Eigenschaft, um die benötigte Trainingsmenge klein zu halten.

Durch die Ausnutzung von sprachlichen Rückmeldungen des Benutzers werden die in der Datenbank befindlichen Merkmale nach jeder Anfrage optimiert. Damit ist neben dem Lernen von Objektansichten auch eine ständige inkrementelle Optimierung des Gelernten realisiert.

Unterschiedliche Teile des Systems wurden jeweils qualitativ und quantitativ evaluiert. Im einzelnen waren dies die Gestenerkennung, die Objekterkennung und das Gesamtsystem. Die Gestenerkennung ist in der Lage, auch auf mittelschnellen Rechnern in Echtzeit zu arbeiten. Dabei können die Hände auch unter schwierigen Bedingungen wie beispielsweise einem Holzhintergrund verfolgt werden. Die Objekterkennung konnte bei sehr geringem Trainingsmaterial eine gute Erkennungsrate erzielen. Dabei war die Geschwindigkeit auch für eine interaktive Nutzung äußerst adäquat.

Bei Tests des Gesamtsystems konnten auch mit dem System nicht vertraute Versuchspersonen sehr schnell erfolgreich mit dem System arbeiten. Neue Ansichten von Objekten konnten fehlerfrei gelernt werden. Eine erfolgreiche Lokalisation der Objekte war schon nach dem Lernen weniger Ansichten möglich, auch wenn die Objekte beliebig verschoben und rotiert wurden. Haben Versuchspersonen häufiger Fehler des Systems nicht korrigiert, sank die Erkennungsrate entsprechend. Dies zeigt auch die Effektivität des Lernvorgangs. Wird er durch die Korrektur von Fehlern durchgeführt, bringt er auch Vorteile. Durch die Integration von Zeigegesten konnten Objekte auch dann identifiziert werden, wenn dies auf Grund fehlender Disambiguierungsmöglichkeiten durch sprachliche Informationen allein nicht möglich gewesen wäre. Während aller durchgeführten Experimente traten dabei keinerlei falsch positiven Erkennungen von Gesten auf.

9.2 Ausblick

Das Lernen von Objekten innerhalb eines kognitiven Systems, wie es in dieser Arbeit vorgestellt wurde, ist ein weites Feld. Daher konnten viele Bereiche und Probleme im bisherigen System nur angeschnitten werden. Ein wichtiges zu lösendes Problem ist beispielsweise die Skalierbarkeit in Hinblick auf mehr Objekte in mehr Ansichtsvarianten. Auch die Skalierbarkeit im Hinblick auf die Szene und die Aufnahmebedingungen sollte verbessert werden.

Wenn das System momentan eine neue Ansicht lernt, wird diese direkt in der Datenbank abgelegt. Eine Suche nach Objekten findet dann mit Hilfe des Nächsten-Nachbar-Klassifikators durch Vergleich mit allen in Frage kommenden Ansichten statt. Dadurch ergeben sich bei vielen Ansichten sowohl Geschwindigkeits- als auch Stabilitätsprobleme. Eine Clusterung der gelernten Ansichten im Rahmen einer hin und wieder stattfindenden globalen Optimierungsphase könnte bei beiden Problemen helfen.

Auch Änderungen bei den Aufnahmebedingungen führen zu Skalierungsproblemen. Die verwendeten Merkmale für die Objekterkennung sind momentan zum Teil nicht beleuchtungsinvariant. Dies führt dazu, daß unter neuen Beleuchtungsverhältnissen

neue Ansichten benötigt werden. Die beste Lösung für dieses Problem wäre eine stabile Möglichkeit der Bestimmung der vorliegenden Beleuchtung. Ein interessanter Ansatz wäre die Ausnutzung von Top-Down-Informationen. Können einzelne Objekte in der Szene erkannt werden, kann aus diesen gegebenenfalls die Beleuchtung bestimmt werden. Durch diese Information ließen sich auch die restlichen Objekte leichter erkennen. Steht eine Tiefenkarte der Szene zur Verfügung, könnten auch im Bild skalierte Objekte leichter erkannt werden. Ein Laserscanner könnte diese Karte bestimmen.

Zur Steigerung der Stabilität der Gestenerkennung wäre eine Posturerkennung sehr hilfreich. Die momentane rein trajektorienbasierte Erkennung hat das Problem, daß Trajektorien im Bereich der Kamera sichtbar sein müssen. Dies ist beim Zeigen auf Objekte im Randbereich des Sichtfeldes der Kamera aber nicht gegeben. Außerdem kann ein rein posturbasiertes Zeigen nicht erkannt werden. Die Postur der Hände könnte in beiden Fällen helfen und als ein weiterer Indiz neben der Trajektorie für eine Zeigegeste dienen.

Weitere sprachliche Informationen könnten beim Lernen und Finden von Objekten helfen. Beispiele hierfür sind das Auswerten von räumlichen Relationen ("das Objekt links neben ..."), die sprachliche Beschreibung von Objekten ("Erdbeeren sind rot und größer als Kirschen") oder auch das explizite Benennen von Objekten ("das [zeigt auf ein Objekt] ist eine Erdbeere"). Auch weitere Informationen aus der linguistischen Forschung könnten ausgenutzt werden. Wird beispielsweise eine Farbe bei der Benennung eines Objektes benutzt, ist die Farbe mit größerer Wahrscheinlichkeit nicht charakteristisch für das Objekt.

Das Lernen von Objekten geschieht momentan rein ansichtenbasiert. Damit werden funktionale Aspekte des Objektes momentan in keinster Weise berücksichtigt. Einen Stuhl kann man beispielsweise auch dadurch definieren, daß es ein Gegenstand ist, auf den man sich setzen kann. Das Aussehen wird hierdurch in nur geringer Weise berücksichtigt, die Größe muß geeignet sein und die Oberfläche des Objektes sollte einigermaßen gerade sein. Auch das Lernen von Kategorien ist bei einer rein ansichtenbasierten Vorgehensweise schwierig. Um diese Probleme in voller Allgemeinheit wirklich zu lösen, müßte das System allerdings über umfangreiches Weltwissen verfügen und dieses insbesondere auch anwenden können. Hiervon ist die momentane Forschung allerdings noch weit entfernt – es bleibt noch viel Raum für weitere Arbeiten im Bereich der kognitiven Systeme.

A Farbräume

Farbe ist der perzeptuelle Eindruck, den Menschen durch den Einfall von Licht auf die Retina des Auges wahrnehmen. Die menschliche Retina hat vier verschiedene Rezeptoren für Licht, wobei einer dieser vier, die sogenannten Stäbchen, nur auf die Intensität des Lichtes reagiert. Die anderen drei Rezeptoren, die Zapfen, reagieren jeweils mit unterschiedlichen Absorptionsraten auf einfallendes Licht verschiedener Wellenlänge. Da Farbe durch diese drei unterschiedlichen Rezeptoren wahrgenommen wird, kann sie durch drei numerische Werte dargestellt werden. Ein Farbraum definiert nun ein Koordinatensystem in einem Unterraum innerhalb der möglichen Farben. Eine Farbe ist durch einen Punkt innerhalb dieses Raumes gegeben.

Im Laufe der Zeit wurden verschiedenste Farbräume vorgestellt, die jeweils in bestimmten Anwendungen von Vorteil sind oder aus einer bestimmten Sicht motivierbar sind. In diesem Kapitel werden diejenigen Farbräume kurz vorgestellt, die im Rahmen diese Arbeit oder bei hier vorgestellten Arbeiten zur Anwendung kamen. Weitere Details zum Thema Farbe und Farbräume finden sich beispielsweise in [Gon02, Kapitel 6], [Poy97] und [For98].

Der RGB Farbraum

Der RGB-Farbraum ist ein additives Farbsystem, in dem Farben durch die Kombination der Grundfarben Rot, Grün und Blau entstehen. Dies ist auch das System, in dem beispielsweise Monitore Bilder darstellen. Abbildung A.1 zeigt den RGB-Farbraum in Form eines normierten Würfels. Dieser Farbraum ist nicht perzeptuell linear. Dies bedeutet, daß eine Änderung der Farbe gleicher Stärke nicht immer gleich wahrgenommen wird. Die wahrgenommene Änderung ist abhängig von der Position, an der die Änderung stattfand.

Der rg Farbraum

In der Bildverarbeitung ist häufig eine Unabhängigkeit von der Helligkeit einer Farbe gewünscht. Ein Farbraum, der dies durch eine möglichst einfache Umrechnung aus dem RGB-Farbraum zu ermöglichen versucht, ist der rg-Farbraum. Der rg-Farbraum wird aus dem RGB-Farbraum durch Helligkeitsnormierung gewonnen:

$$r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B} \qquad b = \frac{B}{R+G+B} \tag{A.1}$$

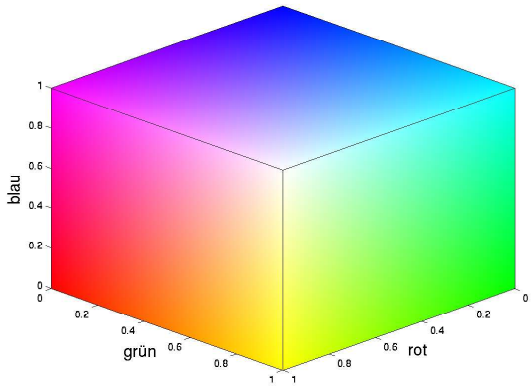


Abbildung A.1: Der RGB-Farbraum dargestellt als normierter Würfel. Schwarz ist an der Koordinate (0,0,0) zu finden, Weiß an der Koordinate (1,1,1).

Da die Summe $r + g + b$ immer 1 ist, ist dies ein Farbraum mit zwei Freiheitsgraden. Damit kann beispielsweise b entfallen.

Der YUV Farbraum

Der YUV-Farbraum teilt die Darstellung in eine Helligkeitskomponente und zwei Farbkomponenten auf. Die Helligkeit oder Luminanz wird durch die Y-Komponente dargestellt, die Farbe oder Chrominanz durch die rot-cyan Balance U und die gelb-blau Balance V. Dieser Farbraum wird häufig für die Übertragung und Kompression von Videosignalen eingesetzt. Das menschliche Auge ist für Helligkeitsinformationen empfindlicher als für Farbinformationen, so daß eine Reduktion der zu übertragenden Information durch eine geringere Auflösung der Farbanteile erreicht werden kann. Die Transformation zwischen dem RGB-Farbraum und dem YUV-Farbraum ist durch folgende Formeln definiert:

$$Y = 0,2990R + 0,5870G + 0,1140B \tag{A.2}$$
$$U = -0,1687R - 0,3313G + 0,5000B \tag{A.3}$$
$$V = 0,5000R - 0,4187G - 0,0813B \tag{A.4}$$

Damit ist dieser Farbraum ebenfalls nicht perzeptuell linear.

Die Farbräume HSV und HSI

In den Farbräumen HSV und HSI wird die Farbe durch den Farbton H (hue), die Sättigung S und eine Helligkeitskomponente V (value) oder I (intensity) repräsentiert. Darstellen lassen sich diese Farbräume durch einen Kegel (HSV), der in Abbildung A.2 schematisch zu sehen ist, beziehungsweise einen Doppelkegel (HSI). Der Farbton ist der Winkel in dem Kegel, die Sättigung der Abstand zur Kegelachse und die Helligkeit

der Abstand zur Kegelspitze. Die Definition einer Farbe ist für den Menschen durch die separate Behandlung des Farbtons in diesem Farbraum intuitiver möglich als in den bisher vorgestellten Farbräumen.

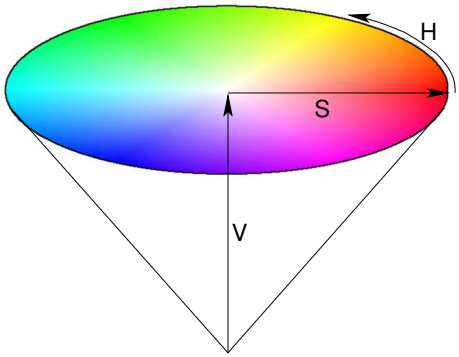


Abbildung A.2: Der HSV-Farbraum dargestellt als Kegel.

Die Umrechnung von RGB nach HSV läßt sich beispielsweise durch folgende Formeln durchführen:

$$V = \max\{R, G, B\} \tag{A.5}$$

$$S = \begin{cases} 0 & , \quad V = 0 \\ \frac{V - \min\{R,G,B\}}{V} & , \quad V \neq 0 \end{cases} \tag{A.6}$$

$$H = \begin{cases} 0 & , \quad S = 0 \\ \frac{60(G-B)}{SV} & , \quad V = R \\ \frac{60(2+B-R)}{SV} & , \quad V = G \\ \frac{60(4+R-G)}{SV} & , \quad V = B \end{cases} \tag{A.7}$$

$$H \rightarrow H + 360 \quad , \text{falls } H < 0 \tag{A.8}$$

Neben dieser Definition gibt es eine große Anzahl weiterer Definitionen ähnlicher Farbräume, die sich häufig in nur wenigen Details leicht voneinander unterscheiden.

Der CIE L*u*v* Farbraum

Alle bisher vorgestellten Farbräume sind nicht perzeptuell linear. Ein Farbraum, der dies nahezu erfüllt, ist der CIE L*u*v* Farbraum [CIE86]. Er ist basierend auf dem CIE XYZ Farbraum definiert und hat damit seine Grundlagen in Untersuchungen des menschlichen Sehsystems. Der XYZ Farbraum ist so gewählt, daß er jede sichtbare Farbe durch drei positive Werte darstellen kann. Dies trifft auf Farbräume wie RGB nicht zu. Die L*-Komponente des L*u*v* Farbraums entspricht der Helligkeit, u*

und v* geben die Farbe wieder. Die Farben ergeben sich aus den in Abbildung A.3 dargestellten (u',v') Farbkoordinaten durch

$$u^* = 13L^*(u' - u'_n) \tag{A.9}$$
$$v^* = 13L^*(v' - v'_n) \tag{A.10}$$

wobei u'_n und v'_n den gewählten Weißpunkt angeben.

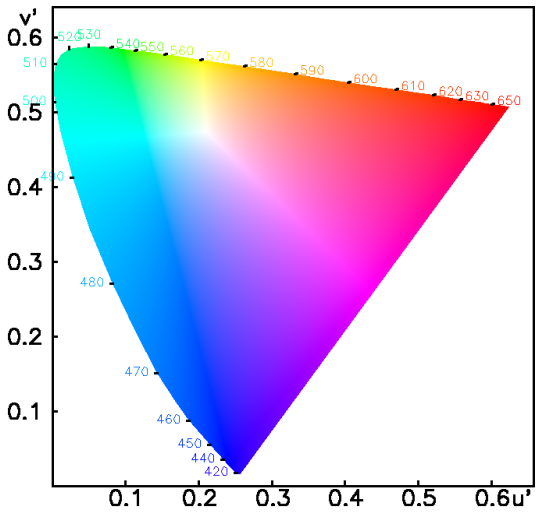


Abbildung A.3: Das CIE Farbdiagramm dargestellt in den CIE Koordinaten u' und v'.

Der CIE L*a*b* Farbraum

Der CIE L*a*b* Farbraum [CIE86] ist ähnlich dem CIE L*u*v* Farbraum perzeptuell nahezu linear. Im Gegensatz zum L*u*v* Farbraum ist er direkt basierend auf dem CIE XYZ Farbraum definiert. Die L*-Komponente des Farbraums entspricht wiederum der Helligkeit, a* und b* geben die Farbe an. a* ist eine rot-grün Koordinate, b* eine gelb-blau Koordinate.

B iceWing – Ein CASE Tool

Bei der Entwicklung von Software im wissenschaftlichen Bereich gibt es einige sehr aufwendige Aufgaben, die jedes mal wieder anfallen, aber nicht direkt zu der eigentlichen Aufgabe gehören. Sehr häufig benötigt man eine Möglichkeit, möglichst einfach und schnell Parameter eines Algorithmus zu beeinflussen. Ähnlich wichtig ist auch die Möglichkeit der einfachen visuellen Darstellung von beliebigen Daten. Die Daten müssen jederzeit leicht inspizierbar und abspeicherbar sein. Bei größeren integrierten Systemen ist es von Vorteil, wenn Einzelkomponenten getrennt voneinander entwickelt werden können und anschließend mit möglichst wenig Geschwindigkeitsverlust untereinander flexibel kommunizieren können.

Eine ergonomisch ausgefeilte Oberfläche, die auch für mit einem Programm nicht vertraute Person leicht bedienbar ist, wird im wissenschaftlichen Bereich dagegen meist nicht benötigt. Wichtig ist meist, daß ein kleines mit der zu lösenden Aufgabe gut vertrautes Team einfach ihren speziellen Algorithmus entwickeln und optimieren kann. Dieses Team von Spezialisten muß die Oberfläche leicht bedienen können. Dabei soll meist kein fertiges Programm für einen Endanwender entstehen. Verschiedene Systeme bieten auf diesen Gebieten schon Funktionalität an. Das kommerzielle MATLAB hat beispielsweise umfangreiche Möglichkeiten zur Visualisierung und zur Oberflächengenerierung, die durch die MATLAB eigene Scriptsprache einfach benutzt werden können [The03]. Die freie Scriptsprache Tcl mit ihrem graphischen Toolkit Tk bietet ebenfalls sehr einfache Möglichkeiten, eine Oberfläche zu generieren [Ous94].

Bestehende Systeme sind allerdings meist nicht auf die speziellen Belange bei der Entwicklung von wissenschaftlicher Software optimiert. Daher wurde iceWing, *the Integrated Communication Environment Which Is Not Gesten*[1], neu entwickelt. iceWing ist eine graphische Shell, die für zur Laufzeit nachladbare Plugins die oben angeführten Funktionalitäten auf einfache Weise zur Verfügung stellt. In den nächsten Abschnitten wird iceWing nun näher vorgestellt. Dazu wird zuerst ein Überblick über die Erstellung von Plugins gegeben. Die nachfolgenden Kapitel gehen dann auf verschiedene Details exemplarisch näher ein. Somit ist die folgende Beschreibung kein komplettes Referenzhandbuch mit einer Darstellung aller Funktionen und Typen von iceWing. Weitere hier nicht erwähnte Punkte finden sich in den Headern und den Beispielplugins von iceWing.

[1] Dies ist ein Hinweis auf ein älteres Programm, den Vorgänger von iceWing.

B.1 Überblick

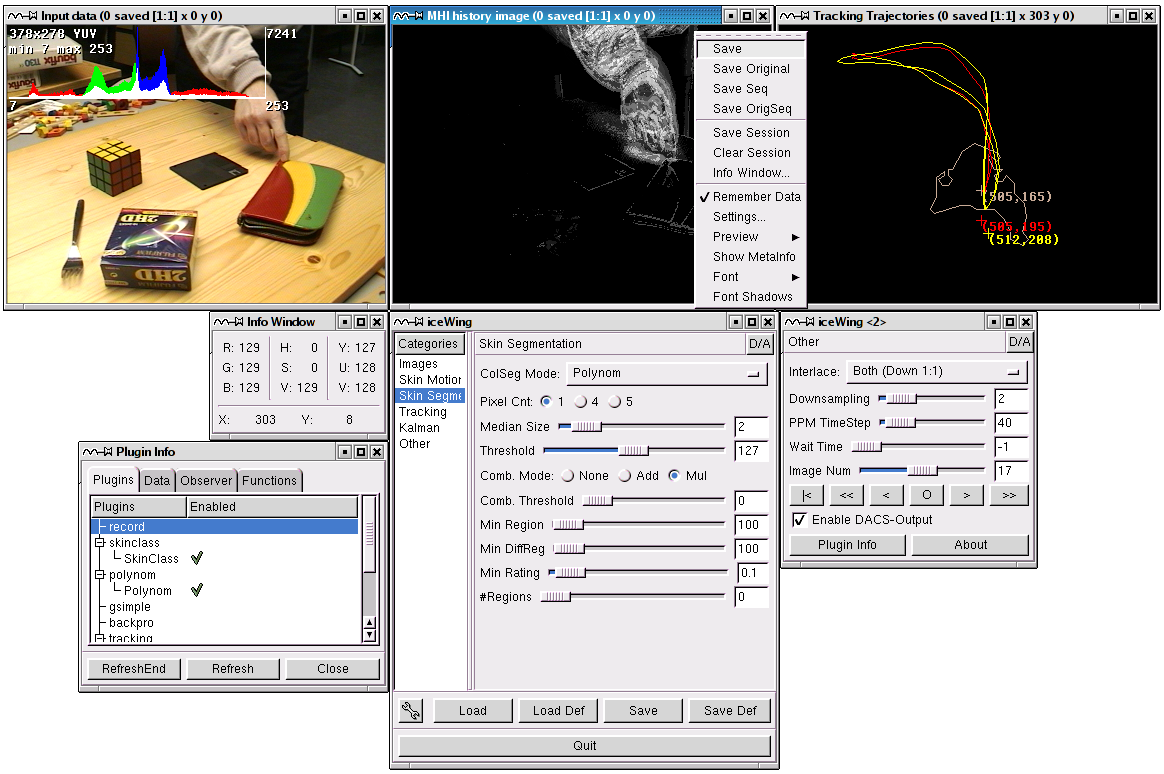


Abbildung B.1: Eine typische Sitzung mit iceWing. Verschiedene Zwischenergebnisse sind visualisiert und können begutachtet werden. Auf verschiedene Parameter kann interaktiv Einfluß genommen werden.

iceWing ist ein in C geschriebenes Programm, das als shared Libraries realisierte Plugins zur Laufzeit nachladen kann. Getestet wurde es unter i386 Linux mit dem GCC Version 2.95 bis Version 3.3 [GCC03] und auf Alphas unter OSF 4.0f mit den Compilern DEC C Version 5.9 und GCC Version 3.1. Für graphische Ausgaben und die Oberflächengenerierung wird zwingend das GTK Toolkit[2] in der Version 1.2 benötigt. Durch andere externe Libraries kann der Funktionsumfang beispielsweise bezüglich unterstützter Graphikformate gesteigert werden. Abbildung B.1 zeigt eine typische Sitzung mit iceWing, bei der Hände in einer Bildsequenz segmentiert und verfolgt werden.

Plugins in iceWing

Das Hauptprogramm iceWing stellt nur eine initiale Oberfläche und verschiede-

[2]siehe http://www.gtk.org/

ne Hilfsroutinen zur Verfügung. Die eigentliche Funktionalität wird durch Plugins realisiert. Plugins für iceWing müssen das in Abbildung B.2 dargestellte Interface implementieren. An Hand eines minimalen Plugins ist dies in Abbildung B.3 dargestellt. Wird ein Plugin in Form einer shared Library geladen, wird zuerst eine Funktion mit Namen `plug_get_info()` aufgerufen. Dies ist zugleich der einzig fest vorgegebene Einsprungpunkt in die Library. Die Funktion dient der Instantiierung einer neuen Instanz eines Plugins, sie hat damit die Aufgabe einer Factory-Funktion. Die Funktion gibt einen Zeiger auf eine gefüllte Struktur vom Typ `plugDefinition` zurück.

```
typedef struct plugDefinition {
    char *name;
    plugType type;
    void (*init) (struct plugDefinition *plug,
                  grabParameter *para, int argc, char **argv);
    int  (*init_options) (struct plugDefinition *plug);
    void (*cleanup) (struct plugDefinition *plug);
    BOOL (*process) (struct plugDefinition *plug,
                     char *id, struct plugData *data);
} plugDefinition;
```

Abbildung B.2: Die Struktur plugDefinition, die Plugins implementieren müssen.

Die Struktur `plugDefinition` enthält alle Informationen, die iceWing über ein neues Plugin haben muß. Die Funktionszeiger `init()`, `init_options()`, `cleanup()` und `process()` definieren Einsprungpunkte für die Instanz des Plugins. `init()` dient der allgemeinen Initialisierung einer Instanz. Hier werden beispielsweise auch Kommandozeilenargumente verarbeitet. In `init_options()` wird das graphische Benutzerinterface initialisiert. Genaueres hierzu findet sich in Abschnitt B.3. `cleanup()` wird am Programmende aufgerufen, um die Freigabe von Ressourcen zu ermöglichen. `process()` wird schließlich aufgerufen, wenn die eigentliche Funktionalität des Plugins ausgeführt werden soll. Wann dies der Fall ist, kann mit Hilfe der Funktionen zur Kommunikation zwischen Plugins definiert werden. Näheres hierzu findet sich in Abschnitt B.2. Die Variable `name` gibt den Namen der Instanz an. Da der Name zur Identifikation der Instanz dient, muß er eindeutig sein. `type` sollte aus Kompatibilitätsgründen immer auf `PLUG_IMAGE` gesetzt werden.

Da das Plugin aus Abbildung B.3 immer einen festen Namen zurückgibt, läßt es sich nur einmal instantiieren. Um dies zu ändern, muß die Struktur vom Typ `plugDefinition` dynamisch alloziert werden und der dort enthaltene Name muß eindeutig gemacht werden. Dies kann durch die Integration der Instanznummer `cnt` in den Namen geschehen. `cnt` gibt an, wie häufig das Plugin schon instantiiert werden sollte. `plug_get_info()` ändert sich damit zu:

```
*append = TRUE;
plugDefinition *def = calloc (1, sizeof(plugDefinition));
```

```
#include "main/plugin.h"

static void min_init (plugDefinition *plug,
                      grabParameter *para, int argc, char **argv)
{
    ...
}

...

static plugDefinition plug_min = {
    "Min",
    PLUG_IMAGE,
    min_init,
    min_init_options,
    min_cleanup,
    min_process
};

plugDefinition *plug_get_info (int cnt, BOOL *append)
{
    *append = TRUE;
    return &plug_min;
}
```

Abbildung B.3: Das Grundgerüst eines minimalen Plugins.

```
*def = plug_min;
def->name = g_strdup_printf ("Min%d", cnt);
return def;
```

Damit kann das Plugin nun beliebig häufig instantiiert werden.

Sollen Plugins in C++ geschrieben werden, kann dies entsprechend dem bisher erläuterten Vorgehen geschehen. Alternativ steht aber auch die in Abbildung B.4 dargestellte C++-Klasse zur Verfügung. Durch Ableiten von dieser Klasse ist ebenfalls die Erzeugung einer Plugin-Instanz möglich. Die Factory-Funktion `plug_get_info()` wird in diesem Fall zu

```
*append = TRUE;
ICEWING::Plugin* newPlugin =
        new ICEWING::MinPlugin (g_strdup_printf("C++Min%d", cnt));
return newPlugin;
```

wobei `ICEWING::MinPlugin` eine von `ICEWING::Plugin` abgeleitete Klasse ist.

```
namespace ICEWING {
    class Plugin : public plugDefinition {
    public:
        Plugin (char *name);
        virtual ~Plugin() {};

        virtual void Init (grabParameter *para, int argc, char **argv) = 0;
        virtual int  InitOptions () = 0;
        virtual bool Process (char *ident, plugData *data) = 0;
    };
}
```

Abbildung B.4: Die Klasse Plugin, die eine der Sprache C++ gerechte Erstellung von Plugins in C++ erlaubt.

B.2 Kommunikation zwischen Plugins

Innerhalb einer Hauptschleife ruft iceWing wiederholt die verschiedenen Plugins auf. Die Reihenfolge des Aufrufs können dabei die Plugins bestimmen. Hierfür und für die weitergehende Kommunikation zwischen Plugins stehen verschiedene Möglichkeiten zur Verfügung. Plugins können Daten untereinander austauschen. Sie können das Ablegen von Daten beobachten und sie können Funktionen zur Verfügung stellen und zur Verfügung gestellte Funktionen aufrufen. Details zu diesen Kommunikationsmöglichkeiten werden nun vorgestellt.

Austausch von Daten

Daten in iceWing bestehen immer aus einem String, der als Identifier dient, einem Referenzzähler und den Daten in Form eines Zeigers. Abgelegt und für andere Plugins zur Verfügung gestellt werden sie mit der Funktion

```
typedef void (*plugDataDestroyFunc) (void *data);

void plug_data_set (plugDefinition *plug, char *ident, void *data,
                    plugDataDestroyFunc destroy);
```

Die Funktion `destroy()` wird aufgerufen, wenn der Referenzzähler der Daten am Ende eines Hauptschleifendurchlaufs auf Null gesunken ist. Für den Zugriff auf abgelegte Daten gibt es die Funktion

```
typedef struct plugData {
    plugDefinition *plug;   /* Plugin, welches die Daten abgelegt hat */
    char *ident;            /* ident, mit dem die Daten abgelegt wurden */
    void *data;             /* Die abgelegten Daten */
} plugData;

plugData* plug_data_get (char *ident, plugData *data);
```

und Varianten für den vereinfachten Aufruf. Mit `plug_data_set()` lassen sich unter einem Identifier verschiedene Daten ablegen. Mit Hilfe des Parameters `data` bei `plug_data_get()` ist es möglich, auf diese nacheinander zuzugreifen. Ist dieser `NULL`, werden die ersten unter dem spezifizierten Identifier abgelegten Daten zurückgegeben. Bei nachfolgenden Aufrufen mit dem jeweils zurückgegebenen Datenzeiger werden die jeweils nächsten Daten zurückgegeben. Bei jedem Aufruf von `plug_data_get()` erhöht sich jeweils der Referenzzähler der Daten. Diese Referenzen können durch die Funktion

```
void plug_data_unget (plugData *data);
```

wieder freigegeben werden. Für jeden erfolgreichen Aufruf von `plug_data_get()` ist damit ein Aufruf von `plug_data_unget()` nötig, damit einmal abgelegte Daten durch den Aufruf der Funktion `destroy()` wieder freigegeben werden können.

Beobachten von Daten

Mit der bisher beschriebenen Funktionalität von iceWing ist noch nicht klar, wann die Funktion `process()` der Plugins aufgerufen wird. Dies wird durch das Beobachten von Daten festgelegt. Mit der Funktion

```
void plug_observ_data (plugDefinition *plug, char *ident);
```

kann ein Plugin das Ablegen von Daten mit dem Identifier `ident` beobachten. Werden *neue* Daten unter diesem Identifier abgelegt, wird die Funktion `process()` des Plugins aufgerufen. Neu sind diejenigen Daten, die seit dem Start des aktuellen Hauptschleifendurchlaufs abgelegt wurden. Am Anfang eines Hauptschleifendurchlaufs legt iceWing Pseudodaten unter dem Identifier `"start"` ab. Andere Plugins können diese Daten beobachten und werden damit jeweils am Anfang der Hauptschleife aufgerufen. Diese Plugins können nun ihrerseits Daten mit anderen Identifiern ablegen und so den Aufruf von weiteren Plugins initiieren. Sind keine Plugins mehr vorhanden, die sich für neu abgelegte Daten angemeldet haben, beginnt die Hautschleife durch Ablegen von Pseudodaten unter dem Identifier `"start"` von neuem. Die Plugins werden dabei sequentiell aufgerufen. Erst nachdem die Funktion `process()` eines Plugins beendet wurde, wird die `process()`-Funktion des nächsten Plugins aufgerufen. Wurden mehrere Daten unter dem gleichen Identifier abgelegt, werden die Plugins trotzdem nur einmal aufgerufen. Sollen die Plugins alle Daten verarbeiten, müssen sie diese sequentiell mit Hilfe von `plug_data_get()` abholen. Durch den Aufruf von

```
void plug_observ_data_remove (plugDefinition *plug, char *ident);
```

kann ein Plugin das Beobachten von Daten schließlich wieder beenden.

Austausch von Funktionen

Das im letzten Abschnitt beschriebene Kommunikationsverfahren ist rein Datengetrieben. Zusätzlich gibt es aber auch noch eine Möglichkeit, Funktionen eines Plugins für andere Plugins zur Verfügung zu stellen. Dies geschieht mit Hilfe der Funktion

```
typedef void (*plugFunc) ();

void plug_function_register (plugDefinition *plug,
                             char *ident, plugFunc func);
```

Durch den Identifier `ident` kann auf die registrierte Funktion wieder zugegriffen werden. Dies geschieht mit Hilfe der Funktion

```
typedef struct plugDataFunc {
    plugDefinition *plug;   /* Plugin, welches die Funktion abgelegt hat */
    char *ident;            /* ident, mit dem die Funktion abgelegt wurde */
    plugFunc func;          /* Die registrierte Funktion */
} plugDataFunc;

plugDataFunc* plug_function_get (char *ident, plugDataFunc *func);
```

Ähnlich wie bei dem Ablegen von Daten können auch mehrere Funktionen unter einem Identifier abgelegt werden. Durch Setzen von `func` auf `NULL` kann auf die erste unter einem Identifier abgelegte Funktion zugegriffen werden. Bei nachfolgenden Aufrufen mit dem jeweils zurückgegebenen Funktionszeiger werden die jeweils nächsten Funktionen zurückgegeben. Durch die Funktion

```
void plug_function_unregister (plugDefinition *plug,
                               char *ident, plugFunc func);
```

kann eine zur Verfügung gestellte Funktion wieder zurückgezogen werden.

B.3 Graphische Funktionalitäten

Die graphischen Funktionalitäten von iceWing lassen sich in drei Gruppen einteilen. Zu der ersten Gruppe zählen Funktionen zur Erstellung eines Benutzerinterfaces bestehend aus Widgets. Die zweite Gruppe von Funktionen beschäftigt sich mit der Darstellung von verschiedenen Daten. Zusätzlich gibt es noch verschiedene Funktionen, die sich diesen Gruppen nicht direkt zuordnen lassen. Die Erstellung des graphischen Interfaces findet hauptsächlich in der Funktion `init_options()` der verschiedenen Plugins statt. Alle in diesem Kapitel aufgeführten Funktionen können aber auch zu jedem späteren Zeitpunkt aufgerufen werden. In den folgenden Abschnitten werden die verschiedenen Funktionalitäten nun näher erläutert.

B.3.1 Erstellung eines Benutzerinterfaces

Alle Widgets, die sich mit Funktionen von iceWing erstellen lassen, folgen der gleichen Philosophie. Bei der Erstellung eines Widgets wird jeweils die Adresse einer Variablen angegeben. Diese Variable wird im Hintergrund geändert, ohne daß das Plugin eingreifen muß, dies anderseits aber auch nicht kann. Daneben ist das Layout, in dem die Widgets erstellt werden, weitestgehend vorgegeben. Durch diese Einschränkung der Funktionalität ergeben sich zwei Vorteile:

- Die Erstellung und Verwaltung von Widgets wird sehr einfach.
- Es gibt die Möglichkeit des automatischen Ladens, Speichern und extern über DACS gesteuerten Setzens der Werte von Widgets. Diese Funktionalität ist vollständig unabhängig von jedweder Unterstützung seitens der Plugins.

Abbildung B.5 zeigt eine Übersicht aller Widgets, die sich mit iceWing erstellen lassen.

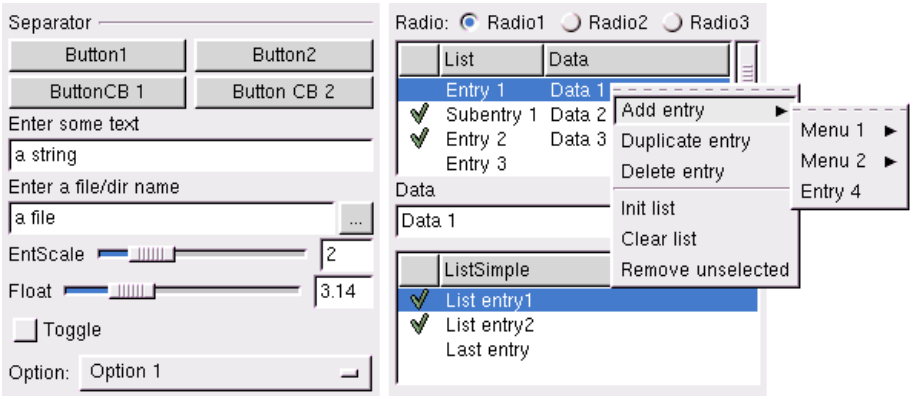


Abbildung B.5: Alle Widgets, die iceWing für die Oberflächengenerierung zur Verfügung stellt.

Widgets lassen sich in iceWing auf beliebig vielen Seiten im iceWing-Hauptfenster, im Kontextmenü von Anzeigefenstern und im "Settings"-Fenster von Anzeigefenstern erstellen. In der Abbildung B.1 sind diese verschiedenen Stellen zu sehen. Eine neue Seite im iceWing-Hauptfenster kann mittels der Funktion

```
int opts_page_append (char *title);
```

erstellt werden. `title` gibt den Namen an, der in der Liste "Categories" angezeigt wird. Zurück gegeben wird ein Index, der bei der Erstellung von Widgets angegeben werden muß. Den Index der "Settings"-Fenster von Anzeigefenstern, der das Erstellen von Widgets in diesen Fenstern erlaubt, erhält man durch die Funktion

```
int prev_get_page (prevBuffer *b);
```

Widgets lassen sich mit Funktionen erstellen, die jeweils nach dem Schema `opts_<widgetname>_create()` benannt sind. Für vier verschiedene Widgettypen sind dies beispielsweise

```
void opts_separator_create (long page, char *title);
void opts_button_create (long page, char *title, char *ttip, gint *value);
void opts_entscale_create (long page, char *title, char *ttip,
                           gint *value, gint left, gint right);
void opts_float_create (long page, char *title, char *ttip,
                        gfloat *value, gfloat left, gfloat right);
```

Die Parameterliste der Funktionen ist jeweils einheitlich aufgebaut. `page` gibt an, auf welcher Seite das Widget erscheinen soll. Neue Widgets werden jeweils unten auf der angegebenen Seite angefügt. `title` gibt den Namen des Widgets an. In Abbildung B.5 war er beispielsweise `"Enter some text"` für das String Widget und `"EntScale"` für den Integer-Slider. Dieser Name muß in Verbindung mit dem Seitennamen programmweit eindeutig sein, da er auch als Identifier für das Widget eingesetzt wird. Der Identifier ist dabei `"pagetitle.widgettitle"`. Durch diesen Identifier läßt sich das Widget auch nachträglich ansprechen. `ttip` spezifiziert den Tooltip des Widgets. Durch `value` wird die Adresse einer Variablen angegeben. Die Variable wird von iceWing in einem eigenen Thread im Hintergrund modifiziert, ohne daß das Plugin eingreifen muß. Die restlichen Parameter spezifizieren jeweils die erlaubten Werte der Variablen.

Zusätzlich ist es möglich, den Wert eines Widgets nachträglich zu setzen und ein Widget wieder zu löschen. Dafür dienen folgende zwei Funktionen:

```
long opts_value_set (char *title, void *value);
gboolean opts_widget_remove (char *title);
```

`title` ist dabei der Identifier eines Widgets. In `value` wird, im Falle eines Integers, der zu setzende Wert übergeben. Ansonsten ist es ein Zeiger auf den zu setzenden Wert. Soll beispielsweise das Widget `"EntScale"` aus Abbildung B.5 auf der Seite `"demo"` auf den Wert 3 gesetzt werden, kann dies durch

```
opts_value_set ("demo.EntScale", GINT_TO_POINTER(3));
```

geschehen. Beim Setzen des Widget `"Float"` geschieht dies hingegen durch

```
float newval = 3.0;
opts_value_set ("demo.Float", &newval);
```

B.3.2 Graphische Anzeige von Daten

iceWing enthält verschiedene Möglichkeiten, Daten zu visualisieren. Plugins können beliebig viele Fenster zur Darstellung von Daten anlegen. Der Benutzer kann diese jederzeit öffnen oder wieder schließen, in ihnen zoomen und scrollen, ihren Inhalt abspeichern oder sich Metainformationen über den Inhalt anzeigen lassen. Bei all diesen Aktionen sind die Plugins nicht involviert. Um die nötigen Zoom- und Redrawaktionen kümmert sich vollständig iceWing. Vektorobjekte wie Linien und Ellipsen werden dabei je nach Zoomstufe neu gezeichnet.

Verwaltung von Fenstern

Neue Fenster können mit der Funktion

```
typedef struct prevBuffer {
    ...
    guchar *buffer;          /* Buffer fuer das zu erzeugende Bild */
    int width, height;       /* Breite, Hoehe des Buffers */
```

```
    GtkWidget *window;      /* Fenster fuer den Buffer */
    ...
} prevBuffer;

prevBuffer *prev_new_window (char *title, int width, int height,
                             gboolean gray, gboolean show);
```

angelegt werden. `title` ist einerseits der Name des Fensters und andererseits ein Identifier, der zusammen mit den Namen der mit `opts_page_append()` angelegten Seiten im iceWing-Hauptfenster programmweit eindeutig sein muß. Enthält der Name einen Punkt (`"."`), werden die Fenster in der "Images"-Liste im Hauptfenster in einer Baumstruktur angezeigt. `width` und `height` geben die initiale Größe des Fensters an. Der Benutzer kann die Fenstergröße nachträglich jederzeit ändern. Durch Angabe von `-1` wird die Standardbreite beziehungsweise Höhe genommen. `gray` gibt an, ob alles in Graustufen oder in Farbe dargestellt werden soll. Durch `show = TRUE` wird das Fenster immer nach dem Anlegen sofort geöffnet. Ansonsten passiert dies erst, wenn der Benutzer das Fenster per Doppelklick in der "Images"-Liste öffnet.

Nennenswert Speicher wird durch ein Fenster erst nach dem ersten Öffnen verbraucht. Alle iceWing-Funktionen zum Zeichnen in einem Fenster testen zuerst, ob es offen ist und kehren sofort zurück, falls dies nicht der Fall ist. Damit ist es aus der Sicht des Ressourcenverbrauchs unproblematisch, viele Fenster anzulegen und Ausgaben in diesen ohne weitere Tests vorzunehmen. Sind Berechnungen für die Ausgabe von Daten nötig, kann ein Test, ob ein Fenster offen ist, aber sinnvoll sein. Dies kann durch `(buffer->window != NULL)` geschehen. Durch

```
void prev_free_window (prevBuffer *b);
```

kann ein per `prev_new_window()` angelegtes Fenster schließlich wieder entfernt werden.

Zum Teil ist es nötig, daß ein Plugin Informationen über Mausaktionen in einem Fenster bekommt. Dies wird durch die Funktion

```
typedef enum {
    PREV_BUTTON_PRESS               = 1 << 0,
    PREV_BUTTON_RELEASE             = 1 << 1,
    PREV_BUTTON_MOTION              = 1 << 2
} prevEvent;
typedef void (*prevButtonFunc) (prevBuffer *b, prevEvent signal,
                                int x, int y, void *data);

void prev_signal_connect (prevBuffer *b, prevEvent sigset,
                          prevButtonFunc cback, void *data);
```

ermöglicht. Tritt eins der mittels `sigset` angegebenen Ereignisse ausgelöst mit der linken Maustaste im Fenster `b` auf, wird die Funktion `cback()` mit `data` als zusätzliches Argument aufgerufen. `cback()` wird zusätzlich mit dem aufgetretenem Signal

und den Mauskoordinaten aufgerufen. Bei den Koordinaten sind Modifikationen durch Zoomen und Scrollen herausgerechnet.

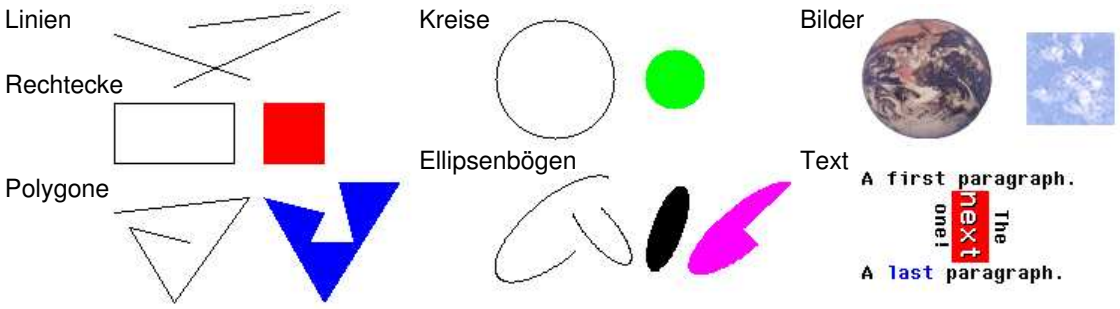


Abbildung B.6: Alle graphischen Elemente, die iceWing darstellen kann.

Darstellung von graphischen Objekten

iceWing bietet die Möglichkeit, verschiedene Graphikprimitiva in den Fenstern darzustellen. Abbildung B.6 zeigt eine Übersicht aller in iceWing enthaltener Objekte. Die Darstellung der Objekte ist mehrstufig. In einem ersten optionalen Schritt wird eine Kopie aller Originaldaten angelegt, die für die Darstellung eines Objektes benötigt werden. Damit erhält iceWing die Möglichkeit, das Bild komplett ohne Interaktion mit dem Plugin neu aufzubauen – Scrollen und Zoomen ohne Pluginhilfe wird möglich. Anschließend werden die Objekte entsprechend den aktuellen Scroll- und Zoomwerten in einem Offscreenbuffer dargestellt. Dieser Buffer zeigt exakt den Bildausschnitt, der auch im Fenster zu sehen sein wird, und kann damit für ein Redraw des Fensters eingesetzt werden. In einem finalen Schritt wird der Buffer schließlich in das Fenster übertragen. Die verschiedenen Objekte können über ein einheitliches Interface verschiedener Funktionen dargestellt werden. Zwei von ihnen sind

```
typedef enum {
    PREV_IMAGE,
    PREV_TEXT,
    PREV_LINE,
    ...
    PREV_NEW = 30
} prevType;

#define RENDER_THICK    (1<<30)  /* beachte die thickness-Einstellung */
#define RENDER_CLEAR    (1<<31)  /* loesche den Buffer */

void prev_render_data (prevBuffer *b, prevType type, void *data,
                       int disp_mode, int width, int height);
void prev_render_list (prevBuffer *b, prevType type, void *data,
                       int size, int cnt,
                       int disp_mode, int width, int height);
```

Bei allen Renderfunktionen gibt es verschiedene Standardargumente. In dem Fenster `b` wird das Objekt dargestellt. `width` und `height` geben die Gesamtgröße aller Objekte an, die in dem Fenster noch dargestellt werden. Diese beiden Werte werden benötigt, um gegebenenfalls den Zoomfaktor für die Darstellung des Objektes korrekt zu berechnen, falls ein "Fit to window"-Darstellungsmodus für das Fenster gewählt ist. Über `disp_mode` kann die Darstellungsart modifiziert werden. Das Flag `RENDER_CLEAR` löscht das komplette Fenster, `RENDER_THICK` erlaubt die dickere Darstellung von Linien entsprechend einer gesondert gewählten Einstellung. Die restlichen Argumente spezifizieren jeweils das darzustellende Objekt. Bei `prev_render_data()` wird ein beliebiges Objekt dargestellt, bei `prev_render_list()` ein Feld gleicher Objekte. `type` gibt den Typ der Objekte an. `data` ist entweder ein Zeiger auf die Daten eines Objektes oder ein Zeiger auf ein Feld von Objektdaten. Im zweiten Fall wird über `cnt` die Länge des Feldes und über `size` die Größe eines Feldelementes angegeben.

Werte für darzustellende Objekte werden über Strukturen angegeben. Für Linien ist dies beispielsweise

```
typedef struct {
    imgColtab ctab;
    int r, g, b;
    int x1, y1, x2, y2;
} prevDataLine;
```

Die Strukturen der restlichen vorhandenen Objekte sind vergleichbar aufgebaut. Über `ctab` wird angegeben, wie `r`, `g` und `b` interpretiert werden sollen. Durch das Setzen von `ctab` auf `IMG_RGB` werden sie beispielsweise als Punkt im RGB-Farbraum interpretiert, durch `IMG_YUV` als Punkt im YUV-Farbraum und durch einen Zeiger auf eine Farbtabelle wird `r` als Index in diese Farbtabelle interpretiert. Die Angabe von `-1` für `r`, `g` oder `b` hat eine Sonderbedeutung. In diesem Fall wird der entsprechende Farbkanal beim Darstellen des Objektes nicht geändert. `x1`, `y1`, `x2` und `y2` sind schließlich die Koordinaten der Endpunkte der Linie.

Um den Aufruf zu vereinfachen gibt es für `prev_render_list()` je einen Wrapper pro Objekttyp. Felder von Linien oder Bildern lassen sich beispielsweise auch durch

```
void prev_render_lines (prevBuffer *b, prevDataLine *lines, int cnt,
                        int disp_mode, int width, int height);
void prev_render_imgs (prevBuffer *b, prevDataImage *imgs, int cnt,
                       int disp_mode, int width, int height);
```

darstellen. Bei der Bildverarbeitung muß sehr häufig ein Bild dargestellt werden. Für den häufigen Fall von 8 Bit Bildern gibt es dafür

```
void prev_render (prevBuffer *b, guchar **planes,
                  int width, int height, imgColtab ctab);
```

womit in diesem Fall keine Struktur gefüllt werden muß. Diese Funktion löscht zusätzlich durch Setzen von `RENDER_CLEAR` vor der Darstellung des Bildes das komplette

Fenster. Sollen andere Bildtypen dargestellt werden, iceWing unterstützt auch Bildtiefen von 16 Bit, 32 Bit, Float und Double, oder wird anderweitig mehr Flexibilität benötigt, müssen entweder `prev_render_imgs()` oder die allgemeinen Renderfunktionen benutzt werden.

Für die vereinfachte Textausgabe gibt es zusätzlich die Funktion

```
void prev_render_text (prevBuffer *b, int disp_mode, int width, int height,
                       int x, int y, char *format, ...);
```

Diese Funktion führt intern ein `sprintf()` aus und gibt den erhaltenen String dann aus. Im String können zusätzlich Formatierungsanweisungen enthalten sein, um Farbe, Art und Ausrichtung zu modifizieren. Durch ein eingebettetes `<fg="255 0 0" bg="0 0 0" font=big>` kann beispielsweise auf einen großen roten Font auf schwarzem Grund umgestellt werden. Weitere Details zu den möglichen Formatierungsanweisungen finden sich im Header "Grender.h".

Die Darstellung von Objekten läßt sich durch zwei weitere Funktionen beeinflussen:

```
void prev_set_bg_color (prevBuffer *buf, uchar r, uchar g, uchar b);
void prev_set_thickness (prevBuffer *buf, int thickness);
```

`prev_set_bg_color()` gibt an, mit welcher Farbe das Fenster bei Auftreten von `RENDER_CLEAR` gelöscht werden soll. Durch `prev_set_thickness()` werden Linien in der angegebenen Dicke gezeichnet, falls `RENDER_THICK` bei der Ausgabe von Objekten mit angegeben war.

Alle bisher beschriebenen Funktionen zum Darstellen von Objekten zeichnen die Objekte inkrementell in einen einem Fenster zugeordneten Offscreenbuffer. Dieser läßt sich in einem finalen Schritt mittels

```
void prev_draw_buffer (prevBuffer *b);
```

in dem Fenster darstellen. Zusammenfassend sind somit folgende Schritte nötig, um mittels iceWing etwas in einem Fenster darzustellen:

Anlegen eines neuen Fensters `b` mittels `prev_new_window()`. Dies ist initial einmal nötig. Eine gute Stelle hierfür ist die Funktion `init_options()` des Plugins.
LOOP
Darstellen beliebiger Objekte in dem Offscreenbuffer des Fensters `b` durch mehrfaches Aufrufen von `prev_render_xxx()`-Funktionen. Bei dem ersten Aufruf sollte `RENDER_CLEAR` gesetzt sein, um das Fenster zu löschen.
Darstellen des Offscreenbuffers im Fenster `b` durch Aufruf der Funktion `prev_draw_buffer()`.

Neben diesem hier beschriebenen Interface mit den `prev_render_xxx()`-Funktionen zur Darstellung von Objekten gibt es auch noch ein Interface bestehend aus `prev_drawXxx()`-Funktionen. Dies ist ein Interface niedriger Abstraktionsrate, welches beispielsweise die Scrollposition oder die Zoomeinstellung ignoriert. Weitere Details hierzu finden sich in den entsprechenden Headern.

B.3.3 Weitere graphische Funktionalitäten

Muß die Programmausführung beendet werden, sollte dies niemals mit der normalen ANSI-Funktion `exit()` geschehen. Für das sofortige Beenden des Programms sollte immer die Funktion

```
void gui_exit (int status);
```

benutzt werden. Der direkte Aufruf von `exit()` kann sowohl zu einer "Segmentation violation" als auch zum Stehenbleiben des Programms führen. Da verschiedene graphische Funktionalitäten mit Hilfe eines eigenen Threads realisiert sind, muß beim Beenden eine korrekte Synchronisierung mit diesem Thread stattfinden. `gui_exit()` stellt dies sicher.

Bilder werden in iceWing durch die Struktur

```
typedef enum {
    IMG_8U, IMG_16U, IMG_32S, IMG_FLOAT, IMG_DOUBLE
} imgType;

typedef struct imgImage {
    guchar *data[3];    /* Die eigentlichen Bilddaten */
    int planes;         /* Anzahl der Farbebenen von data */
    imgType type;       /* Typ von data */
    int width, height;  /* Breite, Hoehe des Bildes in Pixel */
    int rowstride;      /* Abstand zweier Linien in Bytes */
                        /*   >0: Farbbilder sind interleaved in data[0] */
    imgColtab ctab;     /* Farbraum von data */
} imgImage;
```

verwaltet. Die Bilddaten können hier in verschiedenen Typen und in verschiedenen Anordnungen abgelegt sein. Durch `imgType` wird der Typ von `data` spezifiziert – von 8 Bit unsigned bis double. Die Anordnung der Daten kann sowohl *planed* als auch *interleaved* sein. Bei einer planed Anordnung sind die einzelnen Farbebenen des Bildes getrennt voneinander in `data[0]`, `data[1]` und `data[2]` angegeben. Bei interleaved enthält `data[0]` die einzelnen Farbwerte für jedes Pixel direkt hintereinander. Für `planes=3` im RGB-Farbraum kommt beispielsweise zuerst der Rotwert von Pixel 1, dann der Grünwert und der Blauwert. Anschließend kommen diese drei Werte für Pixel 2 bis schließlich die drei Werte für Pixel `width*heigth` in `data[0]` abgelegt sind. Bilder aller dieser Typen und Anordnungen können sowohl angelegt, freigegeben, geladen, gespeichert als auch angezeigt werden.

Für die Verwaltung von Bildern gibt es verschiedene Funktionen. Einige wichtige sind

```
imgImage* img_new (void);
imgImage* img_new_alloc (int width, int height, int planes, imgType type);
void      img_free (imgImage *img, imgFree what);
imgImage* img_load (char *fname, imgStatus *status);
imgStatus img_save_format (imgImage *img, imgFormat format,
                           char *fname, imgFileData *data);
```

Weitere Details zu diesen Funktionen sowie verschiedene andere Funktionen zur Verwaltung von Bildern finden sich im Header “Gimage.h”.

B.4 Weitere Funktionalitäten

Neben den bisher beschriebenen Funktionalitäten stehen weitere sowohl im graphischen Bereich als in anderen Bereichen zur Verfügung. In diesem Abschnitt wird eine Auswahl von ihnen näher vorgestellt. Daneben gibt es andere, die nicht genauer vorgestellt werden. Im Header “output.h” finden sich beispielsweise Funktionen zur Vereinfachung der Ausgabe von Daten über DACS. Es stehen Funktionen zur Ausgabe von Bildern, zur Ausgabe von Statusmeldungen und zur Ausgabe von allgemeinen Daten über Streams zur Verfügung. Weiterhin können Funktionen über DACS zur Verfügung gestellt werden. Die Anmeldung bei DACS, die Fehlerbehandlung und die Generierung eindeutiger Stream- und Funktionsnamen werden jeweils von diesen Funktionen übernommen.

Weitere Beispiele für weitere Funktionen sind das “session management” oder Funktionen für das Registrieren eines neuen Graphikprimitives, um diesen mit den allgemeinen Renderfunktionen darzustellen. Weitere Details zu diesen wie auch zu den bisher vorgestellten Funktionen finden sich in den verschiedenen Headern von iceWing.

Das Plugin “grab”

Ein zentrales Plugin ist das in iceWing eingebaute Plugin “grab”. Es ermöglicht das Einlesen und Weiterreichen einer Folge von Bildern. Die Bilder können von einem Grabber (angesteuert durch Video4Linux 2 oder FireWire unter Linux und MME auf Alphas), von DACS im `Bild_t`-Format oder aus Dateien verschiedener Pixelformate eingelesen werden. Sobald das Plugin von iceWing aufgerufen wird, liest es das nächste Bild ein und stellt es mit Hilfe von `plug_data_set()` unter dem Ident `"image"` für andere Plugins zur Verfügung. Als Format wird dabei das folgende `grabImageData` benutzt:

```
typedef struct grabImageData {
    imgImage *img;        /* Das eigentliche Bild */
    struct timeval time;  /* Zeitpunkt des Grabbens */
    int img_number;       /* Fortlaufende Nummer */
    char *fname;          /* Bild aus einer Datei? -> Name der Datei */
    int downw, downh;     /* Faktor, um den das Bild verkleinert wurde */
} grabImageData;
```

Zusätzlich stellt es die eingelesenen Bilder wahlweise auch über einen DACS-Stream im `Bild_t`-Format zur Verfügung. Über eine DACS-Funktion können das aktuelle und wahlweise auch ältere Bilder ebenfalls im `Bild_t`-Format von anderen Programmen abgeholt werden.

Hilfsfunktionen

"tools.h" stellt verschiedene kleinere Hilfsroutinen zur Verfügung, die immer wieder benötigt werden. Dazu gehören Funktionen zur Ausgabe von Fehlern, von Warnungen, von Debugmeldungen und Funktionen zum Testen von Assertions:

```
void debug_1 (int level, char *str);
void debug_2 (int level, char *str, ARG1);
...
void warning_1 (char *str);
void warning_2 (char *str, ARG1);
...
void error_1 (char *str);
void error_2 (char *str, ARG1);
...
void assert_1 (scalar expression, char *str);
void assert_2 (scalar expression, char *str, ARG1);
...
```

Alle diese Funktionen führen intern ein `sprintf()` aus. `debug_x()` und `assert_x()` erzeugen nur dann Code, wenn das Macro `DEBUG` beim Kompilieren gesetzt ist. `debug_x()` gibt nur dann etwas aus, wenn `level` kleiner als der über die iceWing-Kommandozeile gewählte `talklevel` ist. `warning_x()` gibt die Meldung immer aus, `error_x()` bricht zusätzlich die Programmausführung ab.

Auch für die Messung der Ausführungszeit von Programmteilen stehen verschiedene Funktionen zur Verfügung. Unter anderem sind dies

```
int  time_add (char *name);
void time_start (int nr);
long time_stop (int nr, BOOL show);

#define time_add_static(number,name) ...
#define time_add_static2(number,name,number2,name2) ...
```

`time_add()` legt einen neuen Zeitmesser an und gibt einen Index zum Ansprechen des Messers zurück. Per `time_start()` kann er gestartet und per `time_stop()` wieder gestoppt werden. Durch `show = TRUE` wird die vergangene Zeit direkt nach `stdout` ausgegeben. Ansonsten geschieht dies nach einer gewissen Anzahl Durchläufe der Hauptschleife. Über `time_add_static()` kann die Initialisierung vereinfacht werden. Hier wird eine neue static Variable mit Namen `number` definiert und per einmaligem Aufruf von `time_add()` initialisiert. Die Benutzung kann wie folgt aussehen:

```
/* Definition anderer Variablen */
...
time_add_static (time_demo, "Demo Messung");

time_start (time_demo);
/* Ausfuehrung des zu messenden Programmteils */
...
time_stop (time_demo, FALSE);
```

Die Auswertung von Kommandozeilenargumenten wird durch die Funktion

```
char parse_args (int argc, char **argv, int *nr, void **arg, char *pattern);
```

erleichtert. Die Funktion testet, ob `argv[*nr]` in `pattern` vorhanden ist, wobei die Groß-/Kleinschreibung von `argv[*nr]` ignoriert wird. Anschließend wird `*nr` passend erhöht, so daß beim nächsten Aufruf von `parse_args()` das nächste Argument untersucht werden kann. Das Format von `pattern` in EBNF ist `{ "-" token ":" ch ["r"|"ro"|"i"|"io"|"c"] " " }`, wobei `token` ein beliebiger String ohne `" "` und `":"` ist und `ch` ein beliebiges Zeichen ist. `ch` wird zurückgegeben, falls `-token` gefunden wurde. Die restlichen optionalen Zeichen in `pattern` sind Modifikatoren. `"r"` gibt an, daß zusätzlich ein Argument benötigt wird, welches in der Variablen `arg` zurückgegeben wird. Bei `"i"` wird ein zusätzliches Integer-Argument benötigt. `"c"` bedeutet, daß `token` beliebig fortgesetzt werden kann. Diese Fortsetzung wird in der Variablen `arg` zurückgegeben. Durch `"o"` kann schließlich angegeben werden, daß das String- oder Int-Argument optional ist. Die Anwendung der Funktion `parse_args()` zeigt am einfachsten ein Beispiel:

```
void *arg;
char ch, *str_arg;
int nr = 0, int_arg;

str_arg = NULL;
int_arg = 0;
while (nr < argc) {
    ch = parse_args (argc, argv, &nr, &arg,
                     "-I:Ii -S:Sr -H:H -HELP:H --HELP:H");
    switch (ch) {
        case 'I':
            int_arg = (int)(long)arg;
            break;
        case 'S':
            str_arg = (char*)arg;
            break;
        case 'H':
        case '\0':
            help();
        default:
            fprintf (stderr, "Unknown character %c!\n", ch);
            help();
    }
}
```

Literaturverzeichnis

[Aho86] A. V. Aho, R. Sethi, J. D. Ullman: *Compilers: Principles, Techniques, and Tools*, Addison-Wesley, Reading, Massachusetts, 1986.

[All95] J. F. Allen, G. Ferguson, B. W. Miller, E. K. Ringger: *Spoken Dialogue and Interactive Planning*, in *Proceedings of the 1995 ARPA Spoken Language Systems Technology (SLST) Workshop*, Austin, Texas, Januar 1995.

[All01] J. F. Allen, D. K. Byron, M. Dzikovska, G. Ferguson, L. Galescu, A. Stent: *Toward Conversational Human-Computer Interaction*, *AI Magazine*, Bd. 22, Nr. 4, 2001, S. 27–38.

[Arg75] M. Argyle, J. A. Graham: *A Cross-Cultural Study of the Communication of Extra-verbal Meaning by Gestures*, *International Journal of Psychology*, Bd. 10, 1975.

[Ark98] R. Arkin: *Behaviour based robotics*, MIT Press, Cambridge, MA, 1998.

[Ass98] M. Assan, K. Grobel: *Video-Based Sign Language Recognition Using Hidden Markov Models*, in I. Wachsmuth, M. Fröhlich (Hrsg.): *Gesture and Sign Language in Human-Computer Interaction*, Lecture Notes in Artificial Intelligence, Springer, Bielefeld, 1998, S. 97–109.

[Bar02a] K. Barnard, L. Martin, A. Coath, B. Funt: *A Comparison of Computational Color Constancy Algorithms; Part One: Methodology and Experiments with Synthesized Data*, *IEEE Transactions in Image Processing*, Bd. 11, Nr. 9, 2002, S. 972–984.

[Bar02b] K. Barnard, L. Martin, A. Coath, B. Funt: *A Comparison of Computational Color Constancy Algorithms; Part Two: Experiments with Image Data*, *IEEE Transactions in Image Processing*, Bd. 11, Nr. 9, 2002, S. 985–996.

[Bat94] A. Batliner, S. Burger, A. Kießling: *Außergrammatische Phänomene in der Spontansprache: Gegenstandsbereich, Beschreibung, Merkmalinventar*, Technischer Report Nr. 57, Verbmobil, München, Erlangen, Februar 1994.

[Bis98] H. Bischof, A. Leonardis: *Robust recognition of scaled eigenimages through a hierarchical approach*, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, IEEE Computer Society Press, 1998, S. 664–670.

[Bla99] A. W. Black, P. Taylor, R. Caley: *The Festival Speech Synthesis System, System documentation, Edition 1.4, for Festival Version 1.4.0*, Edinburgh University, UK, Juni 1999.

[Bob97] A. Bobick, J. Davis: *Action Recognition Using Temporal Templates*, in M. Shah, R. Jain (Hrsg.): *Motion-Based Recognition*, Kluwer Academic Publishers, 1997, S. 125–146.

[Bol80] R. A. Bolt: *"Put-That-There": Voice and Gesture at the Graphics Interface*, in *Proceedings of the 7th Annual Conference on Computer Graphics and Interactive Techniques*, Bd. 14, 1980, S. 262–270.

[Bra99a] H. Brandt-Pook, G. A. Fink, S. Wachsmuth, G. Sagerer: *Integrated Recognition and Interpretation of Speech for a Construction Task Domain*, in H.-J. Bullinger, J. Ziegler (Hrsg.): *Proc. 8th Int. Conf. on Human-Computer Interaction*, Bd. 1, München, 1999, S. 550–554.

[Bra99b] H. Brandt-Pook: *Eine Sprachverstehenskomponente in einem Konstruktionsszenario*, Dissertation, Universität Bielefeld, Technische Fakultät, 1999.

[Bre03a] N. Bredeche, Y. Chevaleyrea, J.-D. Zucker, A. Drogoula, G. Sabah: *A Meta-Learning Approach to Ground Symbols from Visual Percepts*, *Robotics and Autonomous Systems, Special issue on Perceptual Anchoring: Anchoring Symbols to Sensor Data in Single and Multiple Robot Systems*, Bd. 43, Nr. 2-3, 2003, S. 149–162.

[Bre03b] N. Bredeche, J.-D. Zucker, S. Zhongzhi: *Online learning for object identification by a mobile robot*, in *Proceedings of the 5th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2003)*, Bd. 2, IEEE Press, Kobe, Japan, 2003, S. 630–635.

[Bri95] C. Brindöpke, M. Johanntokrax, A. Pahde, B. Wrede: *Darf ich dich Marvin nennen? Instruktionsdialoge in einem Wizard-of-Oz-Szenario: Materialband*, Report 7, SFB 360, Universität Bielefeld, 1995.

[Bro91] R. A. Brooks: *Intelligence without representation*, *Artificial Intelligence*, Bd. 47, 1991, S. 139–159.

[Bro00] I. N. Bronstein, K. A. Semendjajew, G. Musiol, H. Mühlig: *Taschenbuch der Mathematik*, Verlag Harri Deutsch, 5. Ausg., 2000.

[Bun00] H. Bunke: *Graph Matching for Visual Object Recognition*, *Spatial Vision*, Bd. 13, Nr. 2-3, November 2000, S. 335–340.

[Bur98] W. Burgard, A. B. Cremers, D. Fox, D. Hähnel, G. Lakemeyer, D. Schulz, W. Steiner, S. Thrun: *The Interactive Museum Tour-Guide Robot*, in *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98)*, Madison, Wisconsin, 1998, S. 11–18.

[Cam01] R. J. Campbell, P. J. Flynn: *A Survey Of Free-Form Object Representation and Recognition Techniques*, *Computer Vision and Image Understanding*, Bd. 81, Nr. 2, Februar 2001, S. 166–210.

[Can86] J. F. Canny: *A computational approach to edge detection*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 8, Nr. 6, November 1986, S. 679–698.

[Cha85] E. Charniak, D. McDermott: *Introduction to Artificial Intelligence*, Addison-Wesley, Reading, Massachusetts, 1985.

[Chr03a] H. I. Christensen: *Cognitive (Vision) Systems*, *ERCIM News*, Bd. 53, April 2003, S. 17–18.

[Chr03b] H. I. Christensen, H.-H. Nagel: *Report on Dagstuhl Seminar 03441: Cognitive Vision Systems*, Schloss Dagstuhl, Oktober 2003, http://www.dagstuhl.de/03441/Report/.

[CIE86] CIE: *CIE Colorimetry Specifications*, No. 15.2, Central Bureau of the CIE, Wien, Österreich, 1986.

[Com97] D. Comaniciu, P. Meer: *Robust Analysis of Feature Space: Color Image Segmentation*, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Puerto Rico, 1997, S. 750–755.

[Cro03] J. L. Crowley: *Cognitive Vision Research Roadmap*, IST-2001-35454, European Research Network for Cognitive AI-enabled Computer Vision Systems, März 2003, http://www.ecvision.org/research_planning/ECVisionRoadMapv2.5.pdf.

[Dav52] K. H. Davis, R. Biddulph, S. Balashek: *Automatic Recognition of Spoken Digits*, *The Journal of the Acoustical Society of America*, Bd. 24, Nr. 6, November 1952, S. 637–642.

[Dem77] A. P. Dempster, N. M. Laird, D. B. Rubin: *Maximum Likelihood from Incomplete Data via the EM Algorithm*, *Journal of the Royal Statistical Society Series B*, Bd. 39, Nr. 1, 1977, S. 1–38.

[Dic99] S. Dickinson: *Object Representation and Recognition*, in E. Lepore, Z. Pylyshyn (Hrsg.): *Rudgers University Lectures on Cognitive Science*, Basil Blackwell publishers, 1999, S. 172–207.

[Dut96] T. Dutoit, V. Pagel, N. Pierret, F. Bataille, O. van der Vrecken: *The MBROLA Project: Towards a Set of High Quality Speech Synthesizers Free for Use for Non Commercial Purposes*, in *International Conference on Speech and Language Processing*, Philadelphia, 1996, S. 1393–1396.

[Dut97] T. Dutoit: *High-quality text-to-speech synthesis : an overview*, *Journal of Electrical & Electronics Engineering, Australia: Special Issue on Speech Recognition and Synthesis*, Bd. 17, Nr. 1, 1997, S. 25–37.

[Efr83] B. Efron, G. Gong: *A Leisurely Look at the Bootstrap, the Jackknife, and Cross-Validation.*, *The American Statistican*, Bd. 37, Nr. 1, Februar 1983, S. 36–48.

[Eng88] R. Engelmore, T. Morgan (Hrsg.): *Blackboard Systems*, Addison-Wesley, Reading, Massachusetts, 1988.

[Fin99] G. A. Fink: *Developing HMM-based Recognizers with ESMERALDA*, in V. Matoušek, P. Mautner, J. Ocelíková, P. Sojka (Hrsg.): *Lecture Notes in Artificial Intelligence*, Bd. 1692, Springer, Berlin Heidelberg, 1999, S. 229–234.

[Fod75] J. A. Fodor: *The Language of Thought*, Thomas Y. Crowell, New York, 1975.

[For98] A. Ford, A. Roberts: *Colour Space Conversions*, 1998, http://www.poynton.com/ColorFAQ.html.

[Fre95] Y. Freund, R. E. Schapire: *A decision-theoretic generalization of on-line learning and an application to boosting*, in *Computational Learning Theory: Second European Conference, EuroCOLT 95*, Springer-Verlag, 1995, S. 23–37.

[Fri00a] J. Fritsch, F. Lömker, M. Wienecke, G. Sagerer: *Detecting Assembly Actions by Scene Observation*, in *Proceedings International Conference on Image Processing*, Bd. I, IEEE, Vancouver, September 2000, S. 212–215.

[Fri00b] J. Fritsch, F. Lömker, M. Wienecke, G. Sagerer: *Erkennung von Konstruktionshandlungen aus Bildfolgen*, in *Mustererkennung 2000, 22. DAGM-Symposium Kiel*, Informatik aktuell, Springer, 2000, S. 389–396.

[Fri03] J. Fritsch: *Vision-based Recognition of Gestures with Context*, Dissertation, Universität Bielefeld, Technische Fakultät, 2003.

[Fuk75] K. Fukunaga, L. D. Hostetler: *The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition*, *IEEE Transactions on Information Theory*, Bd. IT-21, Januar 1975, S. 32–40.

[Fun95] B. V. Funt, G. D. Finlayson: *Color Constant Color Indexing*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 17, Nr. 5, Mai 1995, S. 522–529.

[Fun98] B. Funt, K. Barnard, L. Martin: *Is Machine Colour Constancy Good Enough?*, in *Computer Vision - ECCV'98: 5th European Conference on Computer Vision*, Bd. 1406 von *Lecture Notes in Computer Science*, Springer, Juni 1998, S. 445–459.

[GCC03] GCC: *The GNU Compiler Collection*, 2003, http://www.gcc.org.

[Gol90] G. H. Golub, C. F. V. Loan: *Matrix computations*, Johns Hopkins University Press, 2. Ausg., 1990.

[Gon02] R. C. Gonzalez, R. E. Woods: *Digital Image Processing*, Prentice-Hall, New Jersey, 2. Ausg., 2002.

[Gri58] R. L. Grimsdale, F. H. Sumner, C. J. Tunis, T. Kilburn: *A System for the Automatic Recognition of Patterns*, *IEE Proceedings*, Bd. B: 106, 1958, S. 210–221.

[Gri75] H. P. Grice: *Logic and Conversation*, in P. Cole, J. L. Morgan (Hrsg.): *Syntax and Semantics: Vol. 3: Speech Acts*, Academic Press, New York, 1975, S. 41–58.

[Har90] S. Harnad: *The Symbol Grounding Problem*, *Physica D*, Bd. 42, 1990, S. 335–346.

[Her94] T. Herrmann, J. Grabowski: *Sprechen: Psychologie der Sprachproduktion*, Spektrum, Heidelberg, 1994.

[Her03] T. Hermann, T. Henning, H. Ritter: *Gesture Desk – An Integrated Multimodal Gestural Workplace for Sonification*, in *Gesture-Based Communication in Human-Computer Interaction: 5th International Gesture Workshop*, Lecture Notes in Computer Science, Springer, Genua, Italien, 2003, S. 369–379.

[Hua79] T. Huang, G. Yang, G. Tang: *A fast two-dimensional median filtering algorithm*, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Bd. 27, Nr. 1, Februar 1979, S. 13–18.

[Hua01] X. Huang, A. Acero, H.-W. Hon: *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*, Prentice Hall, Englewood Cliffs, New Jersey, 2001.

[Hun94] M. Hunke, A. Waibel: *Face Locating and Tracking for Human-Computer Interaction*, in *Twenty-Eight Asilomar Conference on Signals, Systems & Computers*, Monterey, California, November 1994.

[Hut90] D. P. Huttenlocher, S. Ullman: *Recognizing solid objects by alignment with an image*, *International Journal of Computer Vision*, Bd. 5, Nr. 2, November 1990, S. 195–212.

[Isa98] M. Isard, A. Blake: *Condensation – conditional density propagation for visual tracking*, *International Journal of Computer Vision*, Bd. 29, Nr. 1, 1998, S. 5–28.

[Jai00] A. K. Jain, R. P. W. Duin, J. Mao: *Statistical Pattern Recognition: A Review*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 22, Nr. 1, Januar 2000, S. 4–37.

[Jen96] F. V. Jensen: *An introduction to Bayesian networks*, UCL Press, London, 1996.

[Joh73] G. Johansson: *Visual Perception of Biological Motion and a Model for its Analysis*, *Perception & Psychophysics*, Bd. 14, Nr. 2, 1973, S. 201–211.

[Jun98] N. Jungclaus: *Integration verteilter Systeme zur Mensch-Maschine-Kommunikation*, Dissertation, Universität Bielefeld, Technische Fakultät, 1998.

[Kal60] R. E. Kalman: *A New Approach to Linear Filtering and Prediction Problems*, *Transactions of the ASME–Journal of Basic Engineering*, Bd. 82, Nr. D, 1960, S. 35–45.

[Kaw98] T. Kawahara, C. H. Lee, B. H. Juang: *Flexible Speech Understanding Based on Combined Key-Phrase Detection and Verification*, *IEEE Transactions on Speech and Audio Processing*, Bd. 6, Nr. 6, 1998, S. 558–568.

[Kel97] A. Kellner, B. Rueber, F. Seide, B.-H. Tran: *PADIS – an automatic telephone switchboard and directory information system*, *Speech Communication*, Bd. 23, Nr. 1, Oktober 1997, S. 95–111.

[Ken86] A. Kendon: *Current Issues in the Study of Gestures*, in J. L. Nespoulous, P. Peron, A. R. Lecours (Hrsg.): *The Biological Foundations of Gestures: Motor and Semiotic Aspects*, Lawrence Erlbaum Associates, 1986, S. 23–47.

[Kit98] J. Kittler, M. Hatef, R. P. W. Duin, J. Matas: *On Combining Classifiers*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 20, Nr. 3, 1998, S. 226–239.

[Kli95] G. J. Klir, B. Yuan: *Fuzzy sets and fuzzy logic: theory and applications*, Prentice Hall, New York, 1995.

[Kum98] F. Kummert, G. A. Fink, G. Sagerer, E. Braun: *Hybrid object recognition in image sequences*, in *14th International Conference on Pattern Recognition*, Bd. II, Brisbane, 1998, S. 1165–1170.

[Kur90] G. Kurtenbach, E. A. Hulteen: *Gestures in Human-Computer Communication*, in B. Laurel (Hrsg.): *The Art of Human-Computer Interface Design*, Addison-Wesley, 1990, S. 309–317.

[Lan03] S. Lang, M. Kleinehagenbrock, S. Hohenner, J. Fritsch, G. A. Fink, G. Sagerer: *Providing the Basis for Human-Robot-Interaction: A Multi-Modal Attention System for a Mobile Robot*, in *Proc. Int. Conf. on Multimodal Interfaces*, ACM, Vancouver, Kanada, November 2003.

[LaV99] J. J. LaViola: *A Multimodal Interface Framework for Using Hand Gestures and Speech in Virtual Environment Applications*, in A. Braffort, R. Gherbi, S. Gibet, J. Richardson, D. Teil (Hrsg.): *Gesture-Based Communication in Human-Computer Interaction: International Gesture Workshop, GW'99*, Lecture Notes in Computer Science, Springer, Frankreich, 1999, S. 303–314.

[Lei03] B. Leibe, B. Schiele: *Analyzing Appearance and Contour Based Methods for Object Categorization*, in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Madison, USA, Juni 2003.

[Lev83] S. Levinson, L. Rabiner, M. Sondhi: *An Introduction to the Application of the Theory of Probabilistic Functions of a Markov Process to Atomatic Speech Recognition*, *Bell Systems Technical Journal*, Bd. 62, Nr. 4, 1983, S. 1035–1074.

[Lev85] M. D. Levin: *Vision in man and machine*, McGraw-Hill, New York, 1985.

[Lew95] J. P. Lewis: *Fast template matching*, in *Proceedings of the Vision Interface Conference*, 1995, S. 120–123, Neue Version "Fast Normalized Cross-Correlation" unter http://www.idiom.com/~zilla/Work/nvisionInterface/.

[Lie02] R. Lienhart, J. Maydt: *An Extended Set of Haar-like Features for Rapid Object Detection*, in *IEEE International Conference on Image Processing 2002*, Bd. 1, September 2002, S. 900–903.

[Löm02] F. Lömker, G. Sagerer: *A Multimodal System for Object Learning*, in L. V. Gool (Hrsg.): *Pattern Recognition, 24th DAGM Symposium, Zürich, Schweiz*, Lecture Notes in Computer Science 2449, Springer, Berlin, September 2002, S. 490–497.

[Luc98] M. Lucente, G.-J. Zwart, A. D. George: *Visualization Space: A Testbed for Deviceless Multimodal User Interface*, in M. Coen (Hrsg.): *Intelligent Environments: Papers from the AAAI Spring Symposium*, The AAAI Press, März 1998.

[Luc01] L. Lucchese, S. K. Mitra: *Color Image Segmentation: A State-of-the-Art Survey, Image Processing, Vision, and Pattern Recognition, Proc. of the Indian National Science Academy*, Bd. 67, A, Nr. 2, März 2001, S. 207–221.

[Lun67] A. Lunts, V. Brailovskiy: *Evaluation of Attributes Obtained in Statistical Decision Rules*, *Engineering Cybernetics*, Bd. 3, 1967, S. 98–109.

[Man87] R. Mangold: *Schweigen kann Gold sein - Über förderliche, aber auch nachteilige Effekte von Überspezifikation*, *Sprache & Kognition*, Bd. 4, 1987, S. 165–176.

[Man03] R. Mangold: *Sprechen über Objekte*, in G. Rickheit, T. Herrmann, W. Deutsch (Hrsg.): *Psycholinguistik: Ein internationales Handbuch*, de Gruyter, Berlin, 2003, S. 368–376.

[May79] P. S. Maybeck: *Stochastic models, estimation, and control*, Bd. 141 von *Mathematics in Science and Engineering*, Academic Press, Inc., 1979.

[McN92] D. McNeill: *Hand and Mind: What Gestures Reveal about Thought*, The University of Chicago Press, 1992.

[Mes96] B. T. Messmer: *Efficient Graph Matching algorithms for preprocessed model graphs*, Dissertation, University of Bern, Institute of Computer Science and Applied Mathematics, 1996.

[Mes98] B. T. Messmer, H. Bunke: *A New Algorithm for Error-Tolerant Subgraph Isomorphism Detection*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 20, Nr. 5, Mai 1998, S. 493–504.

[Mic83] R. S. Michalski, T. W. Mitchel, T. M. Mitchell: *Machine Learning: An Artificial Intelligence Approach*, Morgan Kaufmann, Los Altos, 1983.

[Min86] M. Minsky: *The society of mind*, Simon & Schuster, New York, 1986.

[Mon97] G. Montgomery, J. Goldberg, M. Pines: *Seeing, hearing, and smelling the world*, Howard Hughes Medical Institute, 1997, http://www.hhmi.org/senses/.

[Mun92] J. L. Mundy, A. Zisserman: *Geometric invariance in computer vision*, MIT Press, 1992.

[Mun96] J. L. Mundy, A. Liu, N. Pillow, A. Zisserman, S. Abdallah, S. Utcke, S. K. Nayar, C. Rothwell: *An Experimental Comparison of Appearance and Geometric Model Based Recognition*, in *Object Representation in Computer Vision II*, Springer Verlag, April 1996, S. 247–269.

[Mur95] H. Murase, S. K. Nayar: *Visual Learning and Recognition of 3-D Objects from Appearance*, *International Journal of Computer Vision*, Bd. 14, 1995, S. 5–24.

[Nag68] G. Nagy: *State of the Art in Pattern Recognition*, *Proceedings of IEEE*, Bd. 56, Nr. 5, Mai 1968, S. 836–862.

[Nak02] K. Nakadai, K. Hidai, H. Mizoguchi, H. G. Okuno, H. Kitano: *Real-Time Auditory and Visual Multiple-Speaker Tracking for Human-Robot Interaction*, *Journal of Robotics and Mechatronics*, Oktober 2002, S. 479–489.

[Nen96] S. A. Nene, S. K. Nayar, H. Murase: *Columbia object image library (COIL-100)*, CUCS-006-96, Department of Computer Science, Columbia University, New York, 1996.

[Nöl98] C. Nölker, H. Ritter: *Detection of Fingertips in Human Hand Movement Sequences*, in I. Wachsmuth, M. Fröhlich (Hrsg.): *Gesture and Sign Language in Human-Computer Interaction*, Lecture Notes in Artificial Intelligence, Springer, Bielefeld, 1998, S. 209–218.

[Ohb97] K. Ohba, K. Ikeuchi: *Detectability, Uniqueness, and Reliability of Eigen Windows for Stable Verification of Partially Occluded Objects*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 19, Nr. 9, September 1997, S. 1043–1048.

[Ols70] D. R. Olson: *Language and Thought: Aspects of a Cognitive Theory of Semantics*, *Psychological Review*, Bd. 77, Nr. 4, 1970, S. 257–273.

[Ots79] N. Otsu: *A threshold selection method from gray-level histograms*, *IEEE Transactions on Systems, Man, and Cybernetics*, Bd. 9, Nr. 1, 1979, S. 62–66.

[Ous94] J. K. Ousterhout: *Tcl and the Tk Toolkit*, Addison-Wesley, März 1994.

[Ovi03] S. Oviatt (Hrsg.): *Proceedings of the 5th International Conference on Multimodal Interfaces*, ACM Press, New York, 2003.

[Pau92] D. B. Paul, J. M. Baker: *The Design for the Wall Street Journal-based CSR Corpus*, in *Speech and Natural Language Workshop*, Morgan Kaufmann, 1992, S. 357–362.

[Pau00] D. Paulus, K. Horecki, K. Wojciechowski: *Localization of Colored Objects*, in *Proceedings International Conference on Image Processing*, Bd. 3, IEEE, Vancouver, September 2000, S. 492–495.

[Pav97] V. Pavlovic, R. Sharma, T. S. Huang: *Visual Interpretation of Hand Gestures for Human-Computer Interaction: A Review*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 19, Nr. 7, 1997, S. 677–695.

[Por97] T. Portele, B. Heuft: *Towards a prominence-based speech synthesis system*, *Speech Communication*, Bd. 21, Februar 1997, S. 61–72.

[Poy97] C. A. Poynton: *Frequently Asked Questions about Color*, 1997, http://www.poynton.com/ColorFAQ.html.

[Que95] F. K. H. Quek: *Eyes In The Interface*, *Image and Vision Computing*, Bd. 13, Nr. 6, August 1995, S. 511–525.

[Rae00] R. Rae: *Gestikbasierte Mensch-Maschine-Kommunikation auf der Grundlage visueller Aufmerksamkeit und Adaptivität*, Dissertation, Universität Bielefeld, Technische Fakultät, 2000.

[Rei03] N. Reithinger, J. Alexandersson, T. Becker, A. Blocher, R. Engel, M. Löckelt, J. Müller, N. Pfleger, P. Poller, M. Streit, V. Tschernomas: *SmartKom - Adaptive and Flexible Multimodal Access to Multiple Applications*, in *Fifth International Conference on Multimodal Interfaces (ICMI 2003)*, ACM Press, Vancouver, Kanada, November 2003, S. 101–108.

[Roc66] J. J. Rocchio: *Document retrieval systems - Optimization and evaluation*, Dissertation, Harvard Computational Laboratory, Harvard University, Cambridge, MA, 1966.

[Ros75] E. Rosch, C. B. Mervis: *Family Resemblances: Studies in the Internal Structure of Categories*, *Cognitive Psychology*, Bd. 4, 1975, S. 573–605.

[Roy99] D. Roy: *Learning Words from Sights and Sounds: A Computational Model*, Dissertation, Massachusetts Institute of Technology, 1999.

[Roy02] D. Roy, A. Pentland: *Learning Words from Sights and Sounds: A Computational Model*, *Cognitive Science*, Bd. 26, Nr. 1, Januar-Februar 2002, S. 113–146.

[Roy03] D. K. Roy: *Grounded Spoken Language Acquisition: Experiments in Word Learning*, *IEEE Transactions on Multimedia*, 2003.

[Rub01] Y. Rubner, C. Tomasi: *Perceptual Metrics for Image Database Navigation*, Kluwer Academic Publishers, Boston, Massachusetts, 2001.

[Sag97] G. Sagerer, H. Niemann: *Semantic Networks for Understanding Scenes*, Plenum Publishing Corporation, New York, 1997.

[Sak78] H. Sakoe, S. Chiba: *Dynamic Programming Optimization for Spoken Word Recognition*, *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Bd. 26, 1978, S. 43–49.

[Sch96] J. Schürmann: *Pattern Classification: A Unified View of Statistical and Neural Approaches*, Wiley, New York, 1996.

[Sch97] B. Schiele: *Object Recognition using Multidimensional Receptive Field Histograms*, Dissertation, Institut National Polytechnique de Grenoble, 1997, English translation.

[Sch00] B. Schiele, J. L. Crowley: *Recognition without correspondence using multidimensional receptive field histograms*, *International Journal of Computer Vision*, Bd. 36, Nr. 1, 2000, S. 31–52.

[Sch01] M. Schael: *Texture Defect Detection Using Invariant Textural Features*, in *Pattern Recognition, 23rd DAGM-Symposium, Proceedings*, Bd. 2191 von *Lecture Notes in Computer Science*, Springer, München, September 2001, S. 17–24.

[Sea80] J. R. Searle: *Minds, Brains and Programs*, *Behavioral and Brain Sciences*, Bd. 3, 1980, S. 417–424.

[Sha00] N. Sharkey, T. Ziemke: *Life, Mind and Robots. The Ins and Outs of Embodied Cognition*, in S. Wermter, R. Sun (Hrsg.): *Hybrid Neural Systems*, Lecture Notes in Computer Science Vol. 1778, Springer, Heidelberg, 2000, S. 313–332.

[Sig02] S. Siggelkow: *Feature Histograms for Content-Based Image Retrieval*, Dissertation, Albert-Ludwigs-Universität Freiburg im Breisgau, Fakultät für Angewandte Wissenschaften, 2002.

[Sil86] B. W. Silverman: *Density Estimation for Statistics and Data Analysis*, Monographs on Statistics and Applied Probability, Chapman and Hall, London, 1986.

[Sim83] H. A. Simon: *Why Should Machines Learn?*, in R. S. Michalski, T. W. Mitchel, T. M. Mitchell (Hrsg.): *Machine Learning: An Artificial Intelligence Approach*, Kap. 2, Morgan Kaufmann, Los Altos, 1983, S. 25–37.

[SM95] H. Schulz-Mirbach: *Invariant features for gray scale images*, in G. Sagerer, S. Posch, F. Kummert (Hrsg.): *17. DAGM - Symposium "Mustererkennung"*, Informatik aktuell, Springer, Bielefeld, 1995, S. 1–14.

[Sor00] M. Soriano, B. Martinkauppi, S. Huovinen, M. Laaksonen: *Skin Detection in Video Under Changing Illumination Conditions*, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Bd. 1, 2000, S. 839–842.

[Sow02] T. Sowa, I. Wachsmuth: *Interpretation of Shape-Related Iconic Gestures in Virtual Environments*, in I. Wachsmuth, T. Sowa (Hrsg.): *Gesture and Sign Language in Human-Computer Interaction*, LNAI 2298, Springer, Berlin, 2002, S. 21–33.

[ST95] E. G. Schukat-Talamazzini: *Automatische Spracherkennung*, Vieweg, Wiesbaden, 1995.

[Sta95] T. Starner, A. Pentland: *Visual Recognition of American Sign Language Using Hidden Markov Models*, in *International Workshop on Automatic Face and Gesture Recognition*, Zürich, Schweiz, 1995.

[Ste01] L. Steels, F. Kaplan: *AIBO's first words. The social learning of language and meaning*, *Evolution of Communication*, Bd. 4, Nr. 1, 2001.

[Stö99] M. Störring, H. J. Andersen, E. Granum: *Skin colour detection under changing lighting conditions*, in H. Araújo, J. Dias (Hrsg.): *SIRS'99 Proceedings 7th Int. Symposium on Intelligent Robotic Systems*, Juli 1999, S. 187–195.

[Swa91] M. J. Swain, D. H. Ballard: *Color Indexing*, *International Journal of Computer Vision*, Bd. 7, Nr. 1, 1991, S. 11–32.

[The03] The Mathworks: *MatLab*, 2003, http://www.mathworks.com.

[Thr00] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hähnel, C. Rosenberg, N. Roy, J. Schulte, D. Schulz: *Probabilistic Algorithms and the Interactive Museum Tour-Guide Robot Minerva*, *International Journal of Robotics Research*, Bd. 19, Nr. 11, November 2000, S. 11–18.

[Tov96] M. J. Tovée: *An introduction to the visual system*, Cambridge Univ. Press, Cambridge, 1996.

[Tri98] J. Triesch, C. von der Malsburg: *A Gesture Interface for Human-Robot-Interaction*, in *Proceedings of the Third IEEE International Conference on Automatic Face and Gesture Recognition (FG'98)*, IEEE, Nara, Japan, April 1998, S. 546–551.

[Ver02] J. Vermaak, P. Pérez, M. Gangnet, A. Blake: *Towards Improved Observation Models for Visual Tracking: Selective Adaptation*, in A. Heyden, G. Sparr, M. Nielsen, P. Johansen (Hrsg.): *Computer Vision - ECCV 2002, 7th European Conference on Computer Vision, Copenhagen, Denmark, Mai 28-31, 2002, Proceedings, Part I*, Bd. 2350 von *Lecture Notes in Computer Science*, Springer, Mai 2002, S. 645–660.

[vH01] C. von Hardenberg, F. Bérard: *Bare-Hand Human-Computer Interaction*, in *Workshop on Perceptive User Interfaces*, ACM Digital Library, November 2001, ISBN 1-58113-448-7.

[Vin97] V. V. Vinod, H. Murase: *Focused color intersection with efficient searching for object extraction*, *Pattern Recognition*, Bd. 30, Nr. 10, 1997, S. 1787–1797.

[Vin98] V. V. Vinod, H. Murase: *Image retrieval using efficient local-area matching*, *Machine Vision and Applications*, Bd. 11, Nr. 1, 1998, S. 7–15.

[Vio01] P. Viola, M. Jones: *Rapid Object Detection using a Boosted Cascade of Simple Features*, in *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, Bd. I, Kauai, Hawaii, Dezember 2001, S. 511–518.

[Vor01a] C. Vorwerg: *Kategorisierung von Größen- und Formattributen*, in A. Zimmer, K. Lange, K. Bäuml, R. Loose, R. Scheuchenpflug, O. Tucha, H. Schnell, R. Findl (Hrsg.): *Abstracts der 43. Tagung experimentell arbeitender Psychologen, Experimentelle Psychologie*, Pabst Science Publishers, Lengerich, 2001.

[Vor01b] C. Vorwerg: *Kategorisierung von Größen- und Formattributen*, in *Posterbeitrag auf der 43. Tagung experimentell arbeitender Psychologen*, Regensburg, April 9–11 2001.

[vT68] H. L. van Trees: *Detection, Estimation, and Modulation Theory, Part I*, Wiley, New York, 1968.

[Wac98] S. Wachsmuth, G. A. Fink, G. Sagerer: *Integration of Parsing and Incremental Speech Recognition*, in *Proc. of the European Signal Processing Conf.*, Bd. 1, Rhodes, September 1998, S. 371–375.

[Wac01] S. Wachsmuth: *Multi-modal Scene Understanding Using Probabilistic Models*, Dissertation, Universität Bielefeld, Technische Fakultät, 2001.

[Wex98] A. Wexelblat: *Research Challenges in Gesture: Open Issues and Unsolved Problems*, in I. Wachsmuth, M. Fröhlich (Hrsg.): *Gesture and Sign Language in Human-Computer Interaction*, Lecture Notes in Artificial Intelligence, Springer, Bielefeld, 1998, S. 1–11.

[Wic02] G. Wichert, C. Klimowicz, W. Neubauer, T. Wösch, G. Lawitzky, R. Caspari, H.-J. Heger, P. Witschel: *The Robotic Bar - An Integrated Demonstration of Man-Robot Interaction in a Service Scenario*, in *Proc. of the 11th IEEE Int. Workshop on Robot and Human interactive Communication, ROMAN2002*, Berlin, September 2002, S. 374–379.

[Wie00] M. Wienecke: *Erkennung und Klassifikation von Konstruktionshandlungen aus Bildfolgen*, Diplomarbeit, Technische Fakultät, Universität Bielefeld, 2000.

[Wre97a] C. Wren, F. Sparacino, A. J. Azarbayejani, T. J. Darrell, J. W. Davis, T. E. Starner, A. Kotani, C. M. Chao, M. Hlavac, K. B. Russell, A. Bobick, A. P. Pentland: *Perceptive Spaces for Performance and Entertainment: Untethered Interaction using Computer Vision and Audition*, *Applied Artificial Intelligence*, Bd. 4, Nr. 11, Juni 1997, S. 267–284.

[Wre97b] C. R. Wren, A. Azarbayejani, T. Darrell, A. Pentland: *Pfinder: Real-Time Tracking of the Human Body*, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Bd. 19, Nr. 7, 1997, S. 780–785.

[Wro00] S. Wrobel, K. Morik, T. Joachims: *Maschinelles Lernen und Data Mining*, in G. Görz, C.-R. Rollinger, J. Schneeberger (Hrsg.): *Handbuch der künstlichen Intelligenz*, Kap. 14, Oldenbourg, 2000, S. 517–597.

[Yag94] R. R. Yager, J. Kacprzyk, M. Fedrizzi (Hrsg.): *Advances in the Dempster-Shafer Theory of Evidence*, John Wiley & Sons, Februar 1994.

[Yu03a] C. Yu, D. H. Ballard: *A Multimodal Learning Interface for Grounding Spoken Language in Sensory Perceptions*, in *Fifth International Conference on Multimodal Interfaces (ICMI 2003)*, ACM Press, Vancouver, Kanada, 2003.

[Yu03b] C. Yu, D. H. Ballard, R. N. Aslin: *The Role of Embodied Intention in Early Lexical Acquisition*, in *25th Annual Meeting of Cognitive Science Society (CogSci 2003)*, Boston, 2003.

[Zho03] X. S. Zhou, T. S. Huang: *Relevance feedback in image retrieval: A comprehensive review*, *Multimedia Systems*, Bd. 8, Nr. 6, April 2003, S. 536–544.

[Zue00] V. Zue, S. Seneff, J. Glass, J. Polifroni, C. Pao, T. J. Hazen, L. Hetherington: *JUPITER: A Telephone-Based Conversational Interface for Weather Information*, *IEEE Transactions on Speech and Audio Processing*, Bd. 8, Nr. 1, Januar 2000.

Index