Diplomarbeit

Positionsvorhersage von bewegten Objekten in großformatigen Bildsequenzen

Hartmut S. Loos

November 1997

Prof. Dr. Katharina Morik
Fachbereich Informatik
Lehrstuhl Künstliche Intelligenz
Universität Dortmund

Prof. Dr. C. von der Malsburg
Institut für Neuroinformatik
Lehrstuhl Systembiophysik
Ruhr-Universität Bochum

Betreuer: Prof. Dr. Katharina Morik (Dortmund)
Dr.-Ing. Bernd Fritzke (Bochum)

Man kann nur sehen, worauf man seine Aufmerksamkeit richtet, und man richtet seine Aufmerksamkeit nur auf Dinge, die bereits einen Platz im Bewußtsein einnehmen.

Alphonse Bertillon

Danksagung

An dieser Stelle möchte ich mich ganz herzlich bei allen bedanken, die mir, direkt oder indirekt, bei der Durchführung meiner Diplomarbeit geholfen haben. Besonderer Dank gebührt

- Bernd Fritzke für die ständige Bereitschaft nicht nur zu wissenschaftlichen Gesprächen und die damit verbundene ausgezeichnete Betreuung.
- Katharina Morik, die die Betreuung dieser externen Arbeit in Dortmund übernommen hat, für die stimulierenden Diskussionen, die mir viele neue Gesichtspunkte und Einblicke eröffnet haben.
- Christoph von der Malsburg für die Möglichkeit, eine so interessante Arbeit zu schreiben, und für die angenehme und menschliche Atmosphäre, die er zusammen mit seinen Mitarbeitern am Institut für Neuroinformatik an der Ruhr-Universität Bochum geschaffen hat.
- Norbert Krüger für die unermüdliche Verbesserung seines Gesichtsfinders `BananaPeal`.
- Michael Neef für die unauffällige und daher sehr effektive Betreuung des Computernetzes des Instituts.
- Uta Schwalm für die rege Anteilnahme und die Abwicklung vieler organisatorischer Arbeiten im Laufe der Arbeit.

Meinen Eltern gebührt Dank für ihre Geduld und Unterstützung. Ihnen ist die vorliegende Arbeit gewidmet.

Inhaltsverzeichnis

1 Einführung **1**
1.1 Die Problemstellung 2
1.2 Die Methodik zur Vorgehensweise 3
1.3 Die Gliederung der Arbeit 4

2 Die Beschreibung der Aufnahmen **5**
2.1 Die Vorbereitung zu den Aufnahmen 5
2.2 Die untersuchte Szene 9
2.3 Die verwendeten Daten 12

3 Das Problem der Bestimmung der Startpositionen **17**
3.1 Die Lösung mit dem LBG-U Verfahren 18
3.1.1 Die Beschreibung des Verfahrens 18
3.1.2 Die Ergebnisse des Verfahrens 22
3.2 Andere Verfahren 27

4 Das Problem der Vorhersage der nächsten Position **29**
4.1 Die Lösung mit dem k–Nearest-Neighbor Verfahren 29
4.1.1 Die Beschreibung des Verfahrens 30
4.1.2 Die Bestimmung der einstellbaren Parameter 33
4.1.3 Die Ergebnisse 40
4.2 Die Lösung mit Neuronalen Netzen 42
4.2.1 Das Growing Neural Gas Verfahren 43
4.2.2 Das Multilayer Perzeptron 44
4.2.3 Die Ergebnisse der vorgestellten Verfahren 44
4.3 Der Vergleich Neuronale Netze/k–Nearest-Neighbor 48

5 Die Beschreibung des eingesetzten Systems **51**
5.1 Die Komponenten 51
5.1.1 Das Programm `PredictIt` 53
5.1.2 Das Programm `FinderTestIt` 54

5.1.3 Der Gesichtsfinder `BananaPeal` 54
5.1.4 Die Kommunikation der Komponenten 56
5.2 Das Zusammenspiel `PredictIt`/realer Gesichtsfinder 57
5.2.1 Die Schwierigkeiten 57
5.2.2 Die Ergebnisse 59

6 Zusammenfassung und Ausblick 65
6.1 Zusammenfassung 65
6.2 Der Ausblick 66

A Technische Daten 69
A.1 Kamera 69
A.2 Framegrabber 70
A.3 SYNAPSE1 70
A.4 Host 70
A.5 Zusammenspiel der Komponenten 71

B Aufnahmedaten 73
B.1 Aufnahmen 14. November 1996 (CD 1) 73
B.2 Aufnahmen 26. Juni 1997 (CD 2) 74

C Sequenzdaten 77

D Internet 81

Literaturverzeichnis 83

Abbildungsverzeichnis

2.1 Verschiedene Zeitpunkte einer Sequenz. 6
2.2 Weitwinkelaufnahme und Aufnahme mit Zoomobjektiv 8
2.3 Beleuchtung und Kameraposition für die Aufnahme 9
2.4 Die Wege für die Aufnahmen vom 14. November 1996 (CD 1) 10
2.5 Die Wege für die Aufnahmen vom 26. Juni 1997 (CD 2) 11
2.6 Die Markierung der Gesichter in jedem Bild 12
2.7 Datendarstellung: (x, y transformiert) 13
2.8 Datendarstellung: (Breite w, Höhe h) 13
2.9 Datendarstellung: (Zeitpunkt in der Sequenz, x) 14
2.10 Datendarstellung: (Zeitpunkt in der Sequenz, y transformiert) 14
2.11 Die Sequenzdaten umgerechnet auf Vogelperspektive 15

3.1 Der LBG Algorithmus 19
3.2 Der LBG-U Algorithmus 23
3.3 Ein Beispiel für das LBG-U Verfahren 24
3.4 Das Ergebnis des LBG Verfahrens 25
3.5 Das Ergebnis des LBG-U Verfahrens 26

4.1 k-NN: Nachbarn aus disjunkten bzw. beliebigen Sequenzen . . 31
4.2 k-NN: der Einfluß des Durchschnittsbereichs 32
4.3 Die visualisierten Werte mit den Daten von CD 1 37
4.4 Die Ergebnisse zur Bestimmung von k und ϕ 38
4.5 Drei Ergebnisse des 9–Nearest-Neighbor Verfahrens 41
4.6 Grafische Übersicht der Daten von CD 1 und CD 2 42
4.7 Drei Ergebnisse des GNG Verfahrens 46
4.8 Drei Ergebnisse des MLP Verfahrens 47
4.9 Drei Ergebnisse der unterschiedlichen Verfahren im Vergleich . 49

5.1 Der schematische Aufbau und die Funktionsweise des Systems 52
5.2 Gelernte Repräsentation 55
5.3 Gesichtsfindung 55
5.4 Das Kommunikationsprotokoll 56

5.5 Momentaufnahme `PredictIt` und `BananaPeal` 58
5.6 Fünf gelernte Repräsentationen 60
5.7 Eine Bildfolge aus dem Anfang einer Sequenz 61
5.8 Ein Beispiel für die Anwendung der erweiterten Trägheit . . . 62
5.9 Zwei Beispiele zur Vorhersage der Hockbewegung 63

D.1 Das Java-Applet `DemoGNG` 81

Tabellenverzeichnis

4.1 k-NN: die Ergebnisse der ersten Parametersätze 36
4.2 k-NN: die Ergebnisse der eingeschränkten Parametersätze . . . 39
4.3 k-NN: die Summe der Ergebnisse aller fünf Testläufe 40
4.4 Übersicht über die Mengenpaare 42
4.5 Die trainierten Netze für das GNG Verfahren 43
4.6 Die trainierten Netze für das MLP 44
4.7 Die Ergebnisse für die Neuronalen Netze 45
4.8 Der Vergleich aller Verfahren 48

5.1 Eine Übersicht der erzielten Erfolgsquoten 64

B.1 Aufnahmeplan vom 14. November 1996 (CD 1) 74
B.2 Aufnahmeplan vom 26. Juni 1997 (CD 2) 76

C.1 Mengenpaar 1 77
C.2 Mengenpaar 2 78
C.3 Mengenpaar 3 78
C.4 Mengenpaar 4 79
C.5 Mengenpaar 5 79

Kapitel 1

Einführung

Jedes Lebewesen muß sich an seine Umwelt anpassen, um überleben zu können. Neben den angeborenen Fähigkeiten werden viele weitere im Laufe des Lebens erlernt und die angeborenen verbessert. Eine nicht nur für den Menschen wichtige Fähigkeit ist die Aufnahme, Verarbeitung und Interpretation von visuellen Informationen.

Das Sehen bei neugeborenen Kinder ist noch nicht vollständig ausgeprägt. Sie können Objekte in einer Entfernung von ca. 20 - 25 cm bei durchschnittlicher Helligkeit einigermaßen scharf erkennen. Ihre Sehfähigkeit ist deutlich besser, wenn sich das Objekt bewegt. Ein Neugeborenes kann dieser Bewegung sogar in Grenzen folgen und scheint dabei Gesichter und gesichtsähnliche Formen zu bevorzugen — auch schon direkt nach der Geburt ohne vorher je ein Gesicht gesehen zu haben (Oerter und Montada, 1995).

Im Alter von vier bis fünf Monaten ist das Kind in der Lage aktiver zu schauen, es richtet seine Aufmerksamkeit deutlich länger auf interessante Objekte. Die Erfahrung mit ähnlichen Objekten läßt das Kind einem bewegten Objekt nicht nur folgen, sondern dessen Bewegungspfad minimal voraussehen. Das Blickverhalten ist nicht mehr rein passiv — bestimmt durch die physikalischen Merkmale des Objekts —, sondern wird auch durch die bisherigen Erfahrungen gelenkt. Dies führt zu einem sehr viel effektiveren Blickverhalten.

Diese Entwicklung setzt sich fort und mündet in der beim Menschen vorgefundenen Tatsache, daß bereits im seitlichen Kniehocker (*Corpus geniculatum laterale*) 80 % der ankommenden Nervenleitungen aus dem Cortex selbst stammen (Frick et al., 1987). Das Sehen ist also im wesentlichen eine Gehirntätigkeit und weniger eine Fähigkeit der Rezeptoren.

Die Bewegung der Augen ist eng verknüpft mit dem Sehen. Doch warum bewegen wir unsere Augen? Das Zentrum der menschlichen Retina (*Fovea*) ist spezialisiert auf hochauflösendes Sehen. Daher muß ein Objekt im Zentrum des Blickfeldes gehalten werden, um ein scharfes Abbild zu bekommen. Bewegt sich ein Objekt oder richtet sich die Aufmerksamkeit auf ein anderes, müssen die Augen bewegt werden.

Diese schnellen Augenbewegungen heißen Sakkaden. Sie finden ungefähr dreimal in der Sekunde statt und bringen ein stehendes Objekt innerhalb einer viertel Sekunde in die Fovea. Bewegt sich das Objekt, stabilisieren langsame verfolgende Augenbewegungen das Bild auf der Retina und ermöglichen es dadurch der Fovea ein klares Abbild zu liefern.

Theoretisch können die langsamen Augenbewegungen ein sich bewegendes Objekt überhaupt nicht fixieren, da die permanent vorhandene biologische Reaktionszeit es nicht zuläßt. Um dieses Problem zu lösen, werden frühere Erfahrungen mit ähnlichen Objekten verwendet. Das Gehirn benutzt diese Erfahrungen, damit eine Vorhersage der zukünftigen Objektposition gemacht werden kann.

Unbewußt werden die Bewegungen von Objekten vorhergesagt, und sogar weiterverfolgt, wenn diese kurzzeitig verdeckt werden (z. B. ein fahrendes Auto wird durch einen Lastwagen verdeckt, ein Ball verschwindet kurz hinter einem Baum).

Weiterhin kann beim Betrachten einer Szene nicht alles auf einmal aufgenommen werden. Daher richtet sich die Aufmerksamkeit immer nur auf interessante Aspekte einer Szene (Gibson, 1982; Treisman, 1982). Doch auch diese Fokussierung der Aufmerksamkeit wird gelenkt durch die bisherigen Erfahrungen. Nach dem Hinweis „Sieh' mal, ein Flugzeug!“ werden die meisten Menschen in den Himmel schauen. Genauso werden sie nicht nach oben sehen, wenn sie einen Stuhl suchen oder eine Straße überqueren müssen.

1.1 Die Problemstellung

Eine grundlegende Aufgabenstellung im Bereich *Maschinelles Sehen* ist die Lokalisierung von Objekten in Einzelbildern. Dabei geht es um die Bestimmung der Position der Objekte, ihrer scheinbaren Größe und eventuell weiterer Parameter wie der Pose. Hat man ein Verfahren für diese Aufgabenstellung, läßt es sich im Prinzip direkt auch für Bildsequenzen einsetzen, indem das Verfahren auf jedes Einzelbild angewendet wird. Der Aufwand für eine Bildsequenz der Länge n ist dabei n-mal so groß wie der Aufwand für ein Einzelbild. Er wird in bestimmten Anwendungen nicht praktikabel sein, weil z. B. eine Verarbeitung in Echtzeit gefordert ist (um etwa eine Kamera nachzuführen).

Bei vielen Bildsequenzen ist eine Kontinuität der Bewegung einzelner Objekte gegeben und bei ruhender Kamera bewegen sich Objekte oft gar nicht.

Die Verwendung der Lokalisierungsergebnisse vorhergehender Bilder einer Bildsequenz bzw. anderer Bildsequenzen ist naheliegend, da dies zu einer Beschleunigung der Lokalisierung im Folgebild führt.

Eine weitere Beobachtung, die man insbesondere bei fest installierten Kameras machen kann, ist die, daß ein enger Zusammenhang zwischen der scheinbaren Größe von auftretenden Objekten und ihrer Position im Blickfeld besteht. Auch dies weist auf eine Möglichkeit hin, den Aufwand einer naiven Suche (alle Objekte, alle Größen, alle Positionen) zu reduzieren.

Dieses Problem kann als ein Prädiktionsproblem interpretiert werden: gegeben die augenblicklich geschätzte Position, Größe und Art eines Objekts, prädiziere die Werte für das folgende Bild. Gelingt dies, kann eine Einschränkung der Suche auf eine lokale Umgebung im Raum der Positionen, Größen und der Art des Objekts erfolgen.

Auch bei Einzelbildern, die mit einer fest installierten Kamera aufgenommen werden, ist eine charakteristische Antreffwahrscheinlichkeit für ein Objekt bestimmter Größe zu vermuten. Eine Schätzung der entsprechenden Wahrscheinlichkeitsverteilung könnte dazu verwendet werden einen Suchprozeß zu beschleunigen.

Das Ziel der Arbeit ist eine möglichst deutliche Beschleunigung gegenüber der oben genannten naiven Suche ohne wesentliche Beeinträchtigung der Qualität der Ergebnisse.

1.2 Die Methodik zur Vorgehensweise

Die Problematik der Positionsvorhersage von bewegten Objekten in großformatigen Bildsequenzen unterteilt sich in zwei Aufgaben: zum einen die Bestimmung der Startpositionen und zum anderen die Vorhersage der nächsten Position.

Damit die Bewegung eines Objekts überhaupt vorhergesagt werden kann, muß es zunächst gefunden werden. Um nicht den gesamten Parameterraum abzusuchen, sollte ein Mechanismus gefunden werden, der die Aufmerksamkeit auf bestimmte Bereiche konzentriert:

Bestimmung der Startpositionen

gegeben: Bildsequenzen bei fester Kamera,
Objekterkennung,
Position und Größe des Objekts für viele Bilder.
gesucht: wahrscheinliche Startpositionen für das Objekt.

Nach Bestimmung der möglichen Startpositionen werden diese nach dem erwarteten Objekt abgesucht. Wird das Objekt gefunden, kann mit der Vorhersage der nächsten Position begonnen werden:

Vorhersage der nächsten Position

gegeben:	Bildsequenzen bei fester Kamera, Objekterkennung, Position und Größe des Objekts für fast jedes Bild.
gesucht:	wahrscheinliche Position des Objekts im nächsten Bild.

1.3 Die Gliederung der Arbeit

Das **erste Kapitel** enthält eine kurze Einführung zur Motivation der Diplomarbeit. Es werden die biologischen und praktischen Hintergründe angesprochen, das eigentliche Problem erläutert und die Aufteilung in Unteraufgaben beschrieben.

Das **zweite Kapitel** gibt einen Überblick über die Aufnahmen. Im einzelnen werden die Vorbereitungen zu den Aufnahmen und die dabei entstandenen Probleme behandelt. Die untersuchte Szene wird beschrieben und die aus den Aufnahmen extrahierten Daten werden untersucht, mit denen in den nächsten Kapiteln weitergearbeitet wird.

Das **dritte Kapitel** widmet sich der Lösung des Problems der Bestimmung der Startpositionen. In diesem Zusammenhang wird das *LBG-U* Verfahren erläutert und die damit gewonnenen Ergebnisse präsentiert.

Im **vierten Kapitel** werden drei Verfahren zur Lösung des Problems der Vorhersage der nächsten Position vorgestellt. Zuerst wird das statistische Lernverfahren *k–Nearest-Neighbor* beschrieben, anschließend die beiden Neuronalen Netze *Growing Neural Gas* und das *Multilayer Perzeptron*. Die erhaltenen Resultate und die Eigenschaften der Verfahren werden zum Schluß noch miteinander verglichen.

Im **fünften Kapitel** werden die in den Kapiteln drei und vier behandelten Verfahren im Zusammenhang mit einem realen Gesichtsfinder eingesetzt. Die dazu entwickelten Konzepte und Programme werden vorgestellt und die Ergebnisse diskutiert.

Das **sechste Kapitel** faßt die gewonnenen Resultate zusammen und gibt einen Überblick über die Möglichkeiten und Grenzen des in dieser Arbeit behandelten Ansatzes zur Positionsvorhersage von bewegten Objekten in großformatigen Bildsequenzen.

Kapitel 2

Die Beschreibung der Aufnahmen

Die Entwicklung und der Einsatz von Suchstrategien zum Finden und Verfolgen von bewegten Objekten auf Bildsequenzen setzt entsprechende Daten voraus. Diese Daten sind in der Form von Gesichtern in den Bildsequenzen enthalten. Das bedeutet, daß Sequenzen in ausreichender Qualität und Quantität aufgenommen werden müssen. Doch was bedeutet *ausreichende Qualität und Quantität*?

Aus den aufgenommenen Bildsequenzen müssen die Gesichtsdaten wie Position, Größe, etc. extrahiert werden. Welche Möglichkeiten stehen dazu zur Verfügung?

Zum Schluß ist noch zu klären, ob die Daten überhaupt den Einsatz von Verfahren zulassen, die sich mit dem Finden und Verfolgen der bewegten Objekte in den aufgenommenen Bildsequenzen beschäftigen.

2.1 Die Vorbereitung zu den Aufnahmen

Die Vorbereitung zu den Aufnahmen war schwieriger als erwartet. Es galt, viele Probleme zu lösen und offene Fragen zu klären.

Die Auswahl eines geeigneten Aufnahmeortes war durch mehrere Faktoren eingeschränkt. Aus Datenschutzgründen verbot sich die Installation einer Kamera an öffentlich zugänglichen Räumen. Zudem standen nur stationäre Aufnahmesysteme zur Verfügung. Da die Kamera durch ein Kabel an eines dieser Systeme angeschlossen werden muß, kann eine bestimmte Entfernung nicht überschritten werden. Die Wahl fiel auf den Korridor des Instituts für Neuroinformatik in Bochum.

Die Position der Kamera wurde durch einige Probeaufnahmen festgelegt.

Geplant war, die Kamera in einer Höhe von 2,40 m anzubringen. Von dieser Position aus waren die Gesichter allerdings nicht mehr frontal zu erkennen; daher wurde die Kamera in durchschnittlicher Augenhöhe an der Wand befestigt. Ein Stativ kam für die Aufstellung nicht in Frage, da es zu instabil ist und bei weiteren Aufnahmen zu viele Freiheitsgrade erlaubt.

Abbildung 2.1: Verschiedene Zeitpunkte einer Sequenz.

Danach mußte die Frage der Auflösung der Bilder geklärt werden. Die übliche Größe für Bilder zum Zwecke der Gesichtserkennung liegt bei 128 x 128 Pixel (aus dem Englischen abgeleitet von *picture elements*). Diese Auflösung ist jedoch viel zu gering, um eine Gesichtserkennung über eine Strecke von mehreren Metern zu ermöglichen, da das Gesicht meistens nur einen kleinen Teil der Bilder einnimmt. Die technischen Gegebenheiten begrenzen die Auflösung auf maximal 512 x 512 Pixel.

Nach den ersten Testaufnahmen mit verschiedenen Auflösungen fiel die Wahl auf die maximale Auflösung von 512 x 512 Pixel. Bei niedrigeren Auflösungen läßt der Bildausschnitt mit dem Gesicht keine Erkennung mehr zu; alle inneren Gesichtsmerkmale sind zu grob.

Bei den Testaufnahmen offenbarte sich ein weiteres Problem: die Bilder zeigten bei horizontaler Bewegung Querstreifen (Kämme). Diese Artefakte entstehen durch die Aufnahme von zwei Halbbildern zu unterschiedlichen Zeiten. Anschließend werden die beiden Halbbilder im Rechner wieder zu einem Vollbild zusammengesetzt. Dieses Vorgehen erlaubt es, preiswerte Kameras mit hoher Auflösung zu benutzen.

Bewegt sich ein Objekt während der Aufnahme, wird dieses an unterschiedlichen Positionen aufgenommen. Beim Zusammensetzten der Halbbilder entstehen die Kämme. Die Kämme werden immer ausgeprägter, je geringer die Entfernung zur Kamera ist. Aber gerade bei dieser Entfernung sind die Gesichter detailliert und groß und damit besonders zur Gesichtsfindung und -erkennung geeignet. Artefakte dieser Art können je nach verwendeten Verfahren die Gesichtsverarbeitung empfindlich stören. Die Vermeidung der Kämme ist also essentiell und schließt die Verwendung von preiswerten Halbbildkameras aus.

Die zuerst verwendete Weitwinkelkamera konnte nur Halbbilder liefern und wurde daraufhin durch eine andere Kamera ersetzt. Die neue Kamera nahm zwar Vollbilder auf, hatte aber kein Weitwinkel-, sondern nur ein Zoomobjektiv. Dadurch mußte die Kameraposition weiter nach hinten verlegt werden (s. Abbildungen 2.2 und 2.3).

Der schwarze Balken am unteren Bildrand in der Abbildung 2.2 für die Kamera mit dem Zoomobjektiv ist technisch bedingt. Die Kamera hat eine maximale Auflösung in der Vertikalen von 484 Pixel, daher werden die restlichen 28 Zeilen mit schwarzen Pixeln aufgefüllt.

Ein weiterer wichtiger Gesichtspunkt ist die Beleuchtung der Szene. Das Gesicht muß für eine optimale Findung und Erkennung an jeder Stelle der Szene gut ausgeleuchtet sein. Die Verwendung der normalen Deckenbeleuchtung aus Neonlicht ist problematisch, da dieses keine gleichmäßige Helligkeit ausstrahlt, sondern mit 60 Hz flackert. Sequenzen, die bei Neonlicht aufgenommen werden, zeigen ein deutliches An- und Abschwellen der Helligkeit.

Eine Lösung bilden Halogenscheinwerfer, die eine gleichbleibende Helligkeit ausstrahlen. Die gleichmäßige Ausleuchtung einer Szene bedeutet aber einen erheblichen Aufwand, wie in Abbildung 2.3 zu sehen ist.

Die Scheinwerfer befinden sich alle außerhalb des Aufnahmebereichs. Sie

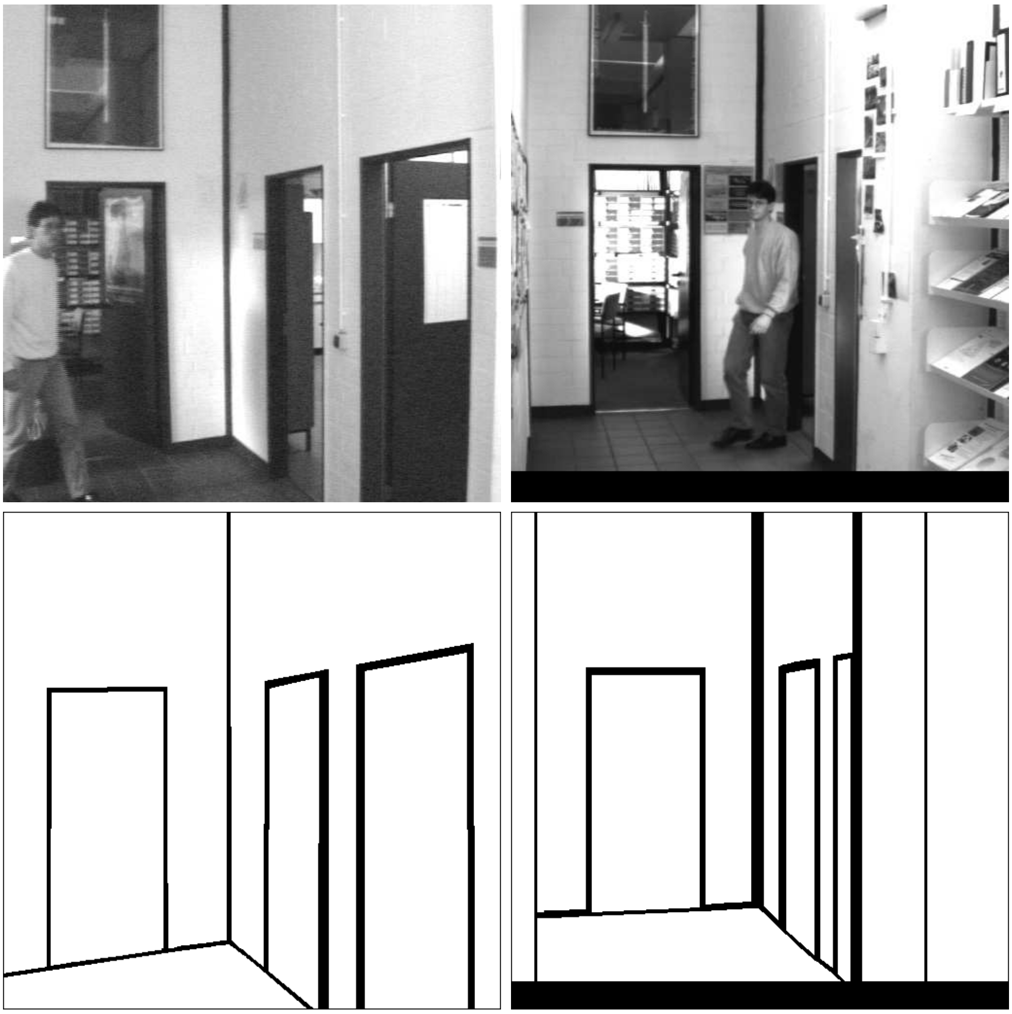


Abbildung 2.2: Weitwinkelaufnahme (links) und Aufnahme mit Zoomobjektiv (rechts) mit den dazugehörigen Piktogrammen.

lagen teilweise auf den Regalen bzw. versperrten als Standscheinwerfer den Gang.

Ein kleineres Problem stellt die große Datenmenge dar, die bei den Aufnahmen anfällt. Ein Pixel (Bildelement) beansprucht 8 Bit (256 Grauwertstufen), ein Bild besteht aus 512^2 Pixel, eine Sequenz aus ca. 100 Bildern. Daraus berechnen sich 25 MByte pro Sequenz und für 30 Sequenzen 750 MByte. Die Verwendung des TIFF-Formats zur Speicherung der einzelnen Bilder gewährt eine verlustfreie Kompression, so daß ein Bild nur noch ca. 220 KByte statt 256 KByte benötigt. Daher passen alle 30 Sequenzen auf

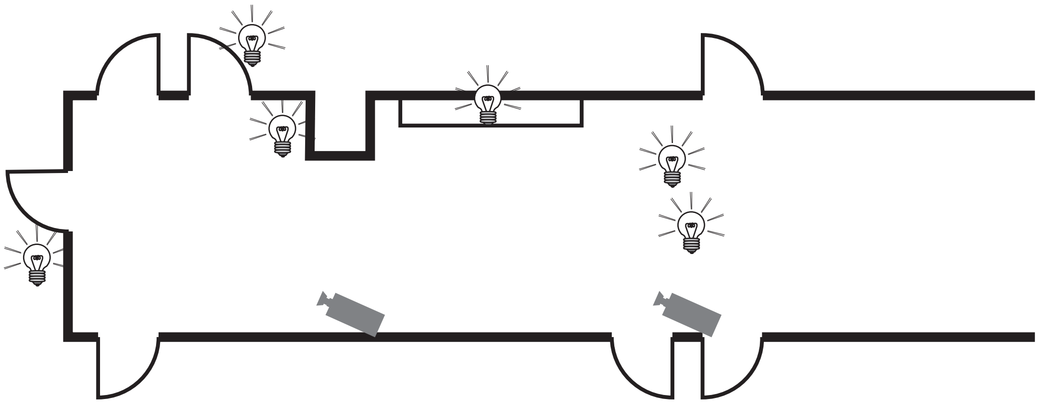


Abbildung 2.3: Beleuchtung und Kameraposition für die Aufnahme: linke Kameraposition Weitwinkelobjektiv (s. Abb. 2.2 links), rechte Kameraposition Zoomobjektiv (s. Abb. 2.2 rechts). Alle Sequenzen wurden mit dem Zoomobjektiv aufgenommen, da die Kamera mit dem Weitwinkelobjektiv Kämme bei bewegten Objekten erzeugt (verschiedene Halbbilder).

eine CD-Rom ($<$ 650 MByte).

Die technischen Einzelheiten zur Kamera und dem eingesetzten Aufnahmesystem sind im Anhang A zu finden.

2.2 Die untersuchte Szene

Die untersuchte Szene stellt den Korridor des Instituts für Neuroinformatik in Bochum dar. In dem betrachteten Ausschnitt des Gangs befinden sich vier Türen: eine links (Sekretariat), eine geradeaus (Prof. von der Malsburg) und zwei rechts (Küche und Seminarraum).

Im Rahmen der Diplomarbeit wurden zwei CDs aufgenommen: die erste CD entstand aus den Aufnahmen vom 14. November 1996 (im folgendem als *CD 1* bezeichnet), die zweite CD wurde mit den Aufnahmen vom 26. Juni 1997 (*CD 2*) bespielt.

Die Aufnahmen von CD 1 sind für erste Untersuchungen einfach gehalten: jede Person mußte alle drei Wege – wie in Abbildung 2.4 beschriebenen – einmal abgehen. Dabei sollte das Gesicht möglichst gut und lange zu erkennen sein. Alle Aufnahmen von CD 1 wurden (bis auf drei Testsequenzen) mit Halogenscheinwerfern ausgeleuchtet. Die drei Testsequenzen entstanden bei normaler Deckenbeleuchtung mit Neonröhren (s. Anhang B.1).

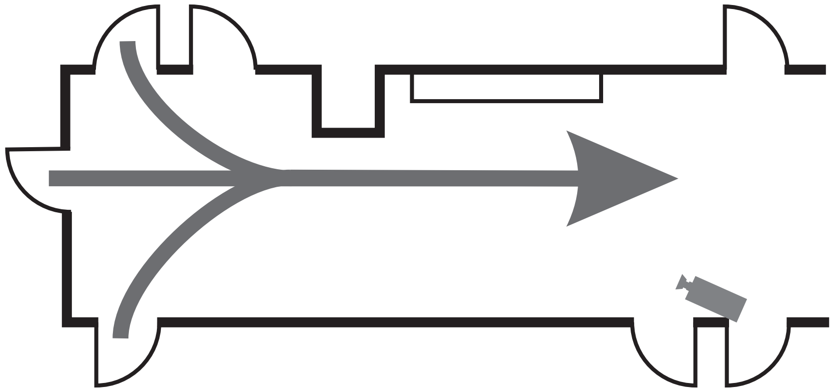


Abbildung 2.4: Die Wege für die Aufnahmen vom 14. November 1996 (CD 1): jede Person mußte einmal alle Wege abgehen; dabei entstanden 30 Sequenzen mit je 100 Bildern (s. Anhang B.1).

Von den 2463 Gesichtern auf CD 1 konnte der Gesichtsfinder auf dem gesamten Bild (512 x 512 Pixel) nur 181 Gesichter bestimmen. Dies entspricht einer Erfolgsquote von 7,35 %[1]. Um ein besseres Ergebnis zu erzielen, wurde über den gesamten Bereich ein Gitter von neun sich überschneidenden 256 x 256 Ausschnitten gelegt. In jedem dieser Ausschnitte sucht der Gesichtsfinder ein Gesicht. Von den neun zurückgelieferten Konfidenzwerten bestimmt der höchste Wert die gefundene Gesichtsposition. Durch dieses Verfahren stieg die Erfolgsrate auf 20,63 % (508 gefundene Gesichter) und die Rechenzeit verneunfachte sich.

Da der eingesetzte Gesichtsfinder mit dem großen Parameterraum (512 x 512 Pixel) keine zufriedenstellenden Ergebnisse lieferte, wurden die Gesichter in jedem Bild per Hand markiert. Diese Daten bilden die Trainingsdaten für alle Auswertungen und Experimente.

Ausgehend von den mit CD 1 gesammelten Erfahrungen entstand CD 2. Diese Aufnahmen sollten die entwickelten Verfahren und Konzepte in schwierigeren Situationen prüfen. Weiterhin wurde auf eine gesonderte Ausleuchtung der Szene mit Halogenscheinwerfern verzichtet.

Die Abbildung 2.5 gibt einen Überblick über die Wege, die jede Person einmal abgehen mußte. Je nach Länge der Wege wurde eine unterschiedliche Anzahl von Bildern aufgenommen (s. Anhang B.2).

Bei den Wegen für CD 2 fällt auf, daß die Personen sich nicht nur in

[1]Der Gesichtsfinder skaliert die 512 x 512 Bilder auf 128 x 128 Pixel und arbeitet mit dieser Auflösung weiter. Daher verwundert es auch nicht, daß die gefundenen Gesichter sich am Ende der Sequenzen befanden, da sie dort am größten sind.

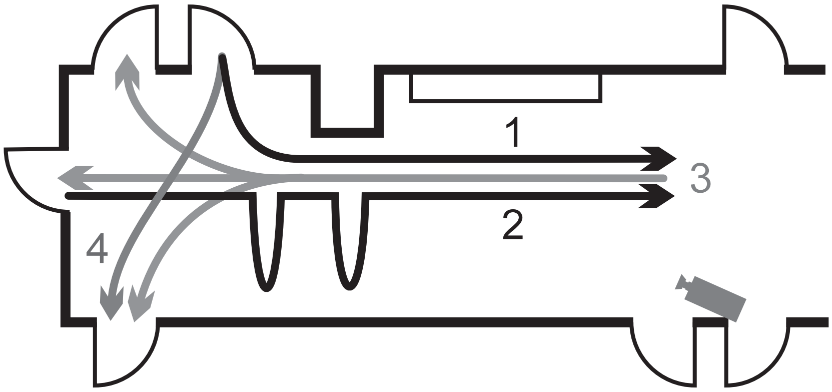


Abbildung 2.5: Die Wege für die Aufnahmen vom 26. Juni 1997 (CD 2): 1 und 2 führen in Richtung der Kamera, 3 von der Kamera weg und 4 parallel zur Kamera (s. Anhang B.2). Auf dem Weg 2 sind die Personen jeweils zweimal in die Hocke gegangen.

Richtung auf die Kamera (1, 2), sondern auch parallel (4) und sogar von der Kamera weg bewegen (3). Bei einem Weg (2) hatten die Versuchspersonen die Vorgabe, zweimal während der Sequenz in die Hocke zu gehen.

Für die Versuchspersonen gab es bei den Aufnahmen von CD 2 nur die Vorgabe, eine gewisse Zeit einzuhalten; der Weg zwischen Start- und Zielpunkt war nur grob vorgegeben und bot genügend Möglichkeiten von der Ideallinie abzuweichen.

Die Gesichter sind auf den Wegen 3 und 4 von CD 2 gar nicht oder nur sehr schwer zu erkennen. Statt der Gesichter wurden auf diesen Bildern die Köpfe markiert, um die verwendeten Verfahren auch unter diesen ausgefallenen Bedingungen zu testen.

Eine andere Möglichkeit, die Gesichter nicht per Hand zu markieren, sondern automatisch zu bestimmen, liegt darin, den Gesichtsfinder ausgehend von den gefundenen Gesichtern rekursiv die vorherigen Gesichter suchen zu lassen.

Der eingesetzte Gesichtsfinder kommt zu guten Ergebnissen, wenn die Gesichter in der Szene nah der Kamera sind. Findet der Gesichtsfinder mit Sicherheit ein Gesicht, wird ein Bild in der Sequenz zurückgegangen und dem Gesichtsfinder nicht das gesamte Bild übergeben, sondern nur ein Ausschnitt. Dieser Ausschnitt wird bestimmt durch die gefundenen Gesichtsposition versehen mit einem Toleranzrahmen. Durch dieses Vorgehen wird der Parameterraum eingeschränkt und der Gesichtsfinder hat somit eine bessere Möglichkeit, das Gesicht zu finden.

Weitere einfache Möglichkeiten, den Parameterraum einzuschränken, sind die Verwendung von Differenzbildern, der Einsatz von Stereokameras und die Nutzung von Farbinformationen.

Stereokameras und Farbinformationen konnten nicht eingesetzt werden, da die entsprechende Hardware nicht vorhanden war. Die Verwendung von Differenzbildern wurde ausprobiert und lieferte unbrauchbare Resultate. Die komplette Szene konnte nämlich nicht so gut ausgeleuchtet werden, daß keine Schatten entstanden. Bei Einsatz von Neonlicht versagen Differenzbilder völlig, da die Helligkeit sich mit jedem Bild ändert.

2.3 Die verwendeten Daten

Zu jedem Bild existiert ein vierdimensionaler Datenvektor $\boldsymbol{\xi} = (x, y, w, h)$. Die Datenvektoren entstanden durch Markierung der Gesichter/Köpfe in jedem Bild per Hand durch ein umschließendes Rechteck $\boldsymbol{r} = (x_0, y_0, w, h)$ (s. Abbildung 2.6).

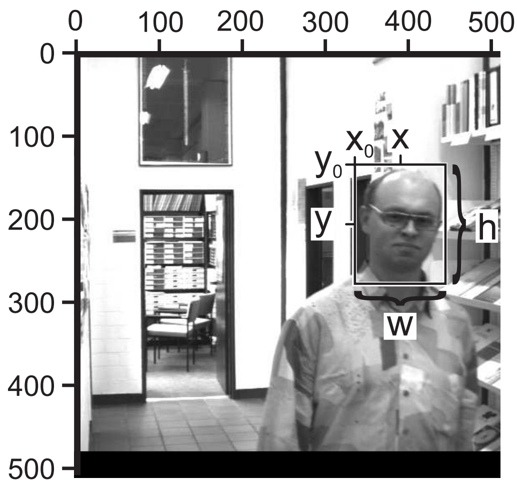


Abbildung 2.6: Ein Beispiel für die Markierung der Gesichter in jedem Bild. Das Paar (x, y) beschreibt den Mittelpunkt des Rechtecks mit der Breite w und der Höhe h.

Die Gesichtsposition und -größe in einem Bild wird bestimmt durch den Vektor $\boldsymbol{\xi}$. Die Komponenten x und y definieren den Mittelpunkt, w und h die Breite und Höhe des umschließenden Rechtecks:

$$x = x_0 + \frac{1}{2}w \tag{2.1}$$

$$y = y_0 + \frac{1}{2}h. \tag{2.2}$$

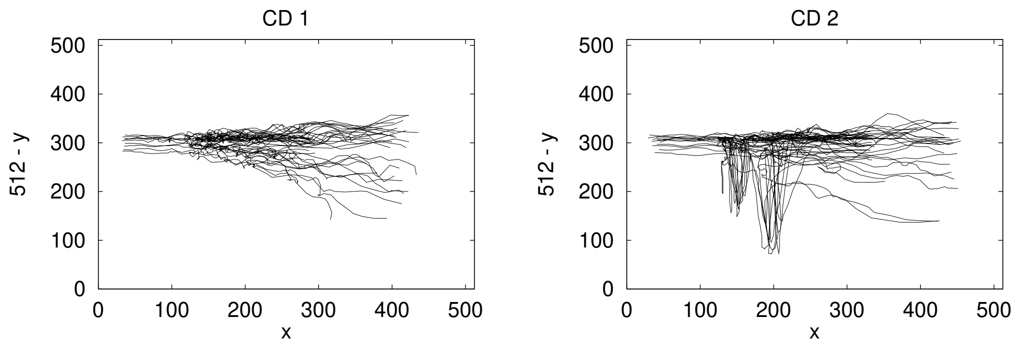


Abbildung 2.7: (x, y transformiert)

Die Abbildungen 2.7 – 2.10 geben eine Übersicht über die Sequenzdaten. Diese Abbildungen zeigen links den vollständigen Datensatz von CD 1 (30 Sequenzen) und rechts den vollständigen Datensatz für CD 2 (36 Sequenzen).

Alle Daten liegen im Format (x, y, w, h) vor. (x, y) beschreibt den Mittelpunkt des gefundenen Gesichts mit der Breite w und Höhe h. `t` bestimmt den Zeitpunkt in der Sequenz.

Durch `512 - y` wird der y Wert so transformiert, daß er besser vergleichbar mit den Bildern wird: der Ursprung bei den Bildern liegt in der linken oberen Ecke (s. Abbildung 2.6).

Ein Datenpunkt in den Abbildungen stellt ein gefundenes Gesicht in einem Bild dar. Bilder ohne Gesicht werden nicht betrachtet. Alle Datenpunkte einer Sequenz sind miteinander in der richtigen zeitlichen Reihenfolge verbunden.

In der Abbildung 2.7 sind die x und y Werte eingezeichnet. Sie zeigen die Verteilung der Gesichtspositionen in der Szene. Wie erwartet, sind keine

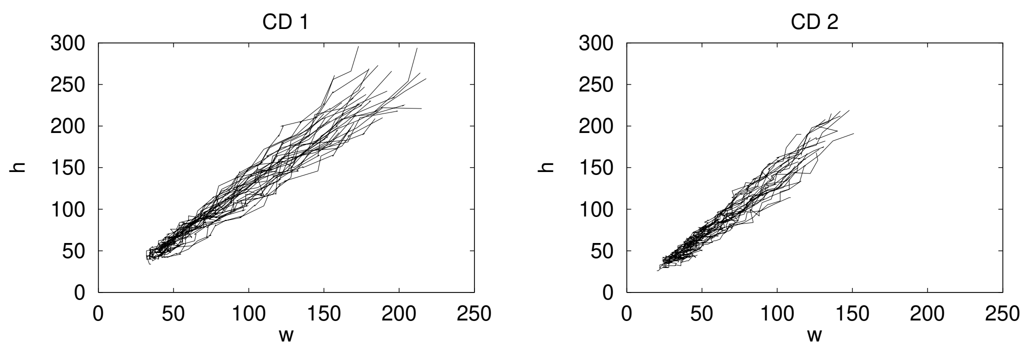


Abbildung 2.8: (Breite w, Höhe h)

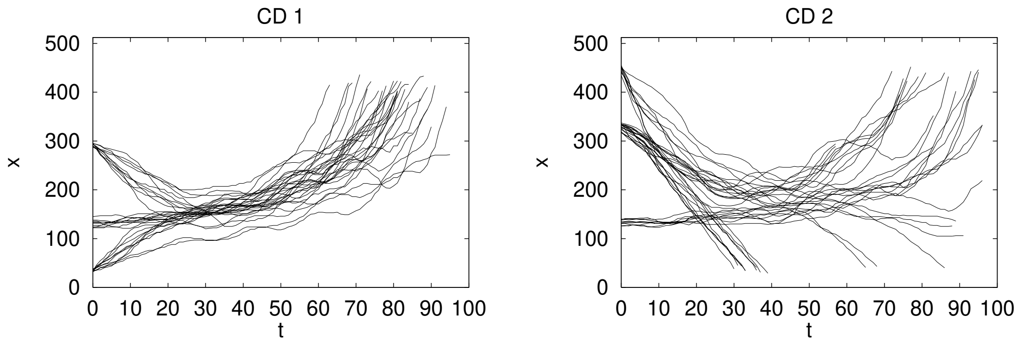


Abbildung 2.9: (Zeitpunkt in der Sequenz, x)

Gesichter an der Decke und am Boden zu finden. Die drei Startpositionen sind in dieser Grafik nicht so deutlich zu erkennen. Sie befinden sich ungefähr in der Umgebung der Positionen (30,300), (130, 300) und (290,300) für CD 1 und an den Positionen (130,300), (320,290) und (450,250) für CD 2. In der Grafik von CD 2 fallen die beiden Zacken auf. An diesen Stellen sind die Versuchspersonen in die Hocke gegangen.

In der Abbildung 2.8 sind die Breite w und die Höhe h eingezeichnet. Es ist deutlich zu sehen, daß die Breite und Höhe der gefundenen Gesichter zueinander proportional sind. Auffallend ist die geringere maximale Größe der Gesichter von CD 2 im Vergleich mit CD 1. Bei den Aufnahmen von CD 1 hatten die Versuchspersonen die Vorgabe, möglichst lange ihr Gesicht der Kamera zu zeigen. Dadurch hielten sie sich länger im Aufnahmebereich der Kamera auf und ihre Gesichter waren dementsprechend größer. Für die Aufnahmen zu CD 2 entfiel diese Vorgabe.

In der Abbildung 2.9 ist der x Wert in Abhängigkeit vom Zeitpunkt in

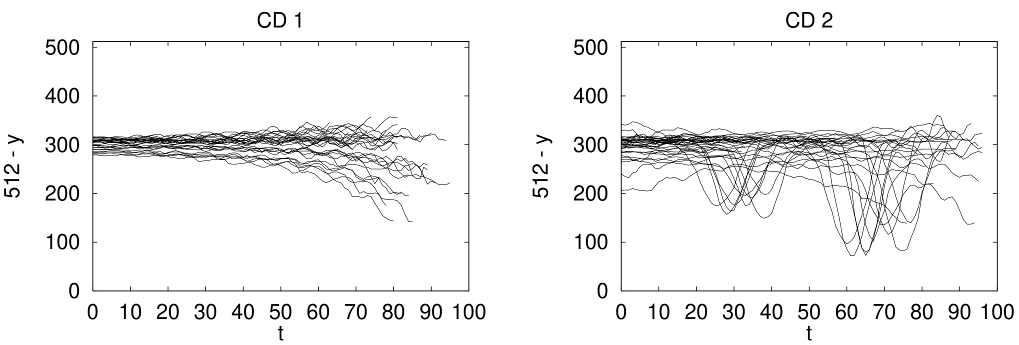


Abbildung 2.10: (Zeitpunkt in der Sequenz, y transformiert)

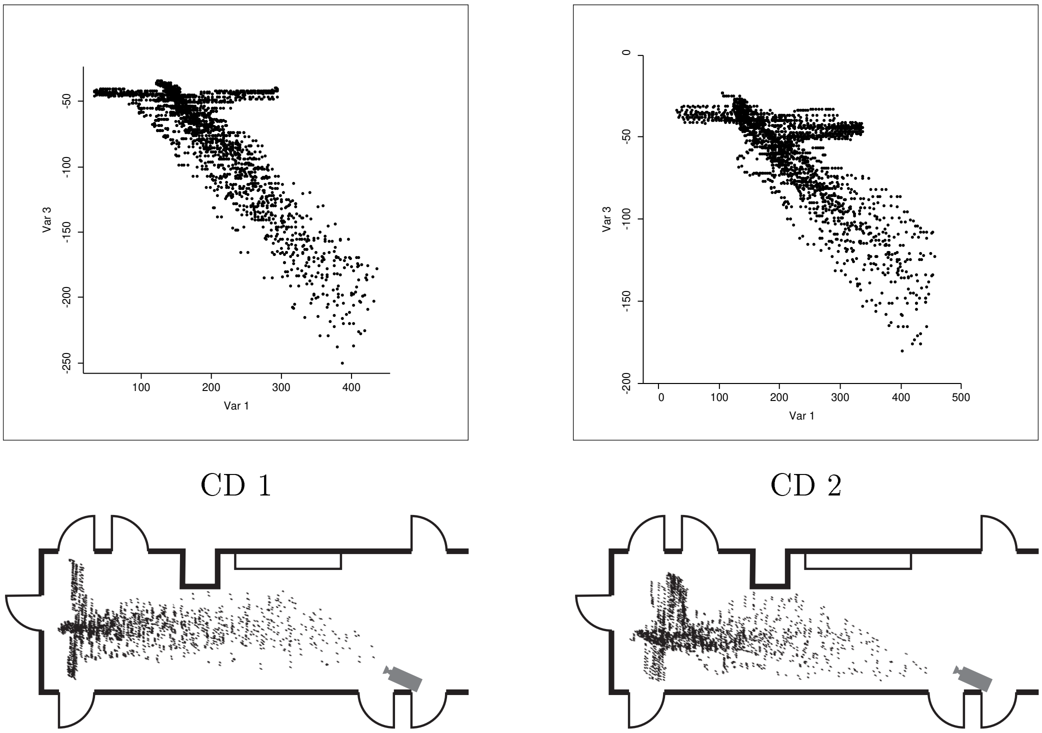


Abbildung 2.11: Die Sequenzdaten umgerechnet auf Vogelperspektive (links CD 1, rechts CD 2) und anschließend eingepaßt in das Gangpiktogramm (unten).

der Sequenz eingezeichnet. Eine Sequenz startet, wenn das erste Gesicht erscheint, Bilder ohne Gesicht werden nicht betrachtet. In beiden Grafiken sind deutlich die drei Startpositionen zu erkennen. Außerdem fällt die unterschiedliche Länge der Sequenzen auf. Die unterschiedliche Länge bei genausoviel aufgenommenen Bildern ensteht durch die verschiedenen Geschwindigkeiten mit der die Versuchspersonen die vorgegebenen Wege abgegangen sind. Bei CD 2 kommt die unterschiedliche Bildanzahl für die verschiedenen Wege hinzu (CD 1: alle 100 Bilder, CD 2: 45, 90 und 100 Bilder).

In der Abbildung 2.10 ist der y Wert in Abhängigkeit vom Zeitpunkt in der Sequenz eingezeichnet. Die Varianz beim Start der Sequenzen in der Grafik für CD 1 rührt von der unterschiedlichen Körpergröße der Versuchspersonen her. Bei CD 2 kommt die unterschiedliche Bewegungsrichtung (auf die Kamera zu und von der Kamera weg) und das zweimalige Hinhocken der Versuchspersonen hinzu. Ein Startpunkt lag hinter der Kamera, daher ist die Varianz zu Beginn der Sequenzen für CD 2 schon größer.

Die Abbildung 2.11 zeigt die Sequenzdaten umgerechnet auf Vogelperspektive; dabei stellt die x-Achse den x-Wert und die y-Achse die negative Quadratwurzel aus dem Produkt von Breite und Höhe ($-\sqrt{wh}$) dar. Für die Darstellung wurde jeweils der vollständige Datensatz von CD 1 und CD 2 verwendet.

Zur besseren Veranschaulichung wurden die umgerechneten Sequenzdaten in das Gangpiktogramm eingepaßt (Abbildung 2.11 unten). Ohne auf die verschiedenen Gesichtsgrößen der einzelnen Versuchspersonen Rücksicht zu nehmen, liefert diese einfache Umformung ein Maß für den Abstand der Versuchspersonen zur Kamera.

Im Anhang C sind weitere Informationen über die erhaltenen Daten zu finden.

Nachdem die Untersuchung der aufgenommenen Daten Gesetzmäßigkeiten erkennen läßt, können Verfahren eingesetzt werden, um die interessierenden Informationen zu extrahieren. In den beiden folgenden Kapiteln werden diese vorgestellt und die Ergebnisse erläutert.

Kapitel 3

Das Problem der Bestimmung der Startpositionen

Damit die Bewegung eines Objekts überhaupt vorhergesagt werden kann, muß es zuerst gefunden werden. Um nicht den gesamten Parameterraum abzusuchen, sollte ein Mechanismus gefunden werden, der die Aufmerksamkeit auf bestimmte Bereiche konzentriert.

Zur Identifizierung solcher Bereiche sind Erfahrungswerte notwendig. Aus diesen Erfahrungswerten können dann die benötigten Informationen extrahiert werden.

Dieses Vorgehen filtert unwahrscheinliche bzw. unmögliche Bereiche (z. B. ein Gesicht hängt nicht in der Luft) heraus und schränkt die Suche so auf wenige interessante Ausschnitte ein. Auch der Mensch stützt sich auf seine Alltagserfahrung und konzentriert seine Aufmerksamkeit auf die Bereiche einer Szene, in der das gesuchte Objekt erfahrungsgemäß häufig anzutreffen ist (Treisman, 1982).

Zur Bestimmung der gesuchten Startpositionen müssen genügend Sequenzen vorhanden sein, in denen das Objekt markiert ist. Die Markierung kann automatisch durch eine Objekterkennung erfolgen oder manuell. Allerdings sollte die automatische Objekterkennung eine gewisse Erfolgsquote erreichen, da ansonsten die Ausschnitte zu groß oder einfach falsch werden können.

Nach Bestimmung der möglichen Startpositionen werden diese nach dem erwarteten Objekt abgesucht. Wird das Objekt gefunden, kann mit der Vorhersage der nächsten Position begonnen werden (s. Kapitel 4).

3.1 Die Lösung mit dem LBG-U Verfahren

Für die Ermittlung der Startpositionen wurde das LBG-U Verfahren (Fritzke, 1997) eingesetzt. Es gehört zu der Klasse der Verfahren mit hartem Wettbewerbslernen (*hard competitive learning* oder auch *winner-take-all learning*) und basiert auf dem LBG Verfahren von Linde, Buzo und Gray (1980). Das LBG-U Verfahren ist eine *Batch*-Methode, die — im Gegensatz zu *Online*-Verfahren — mit einer endlichen Menge von Eingabesignalen arbeitet.

Das LBG-U Verfahren kann u. a. zur Vektorquantifizierung eingesetzt werden, um eine große Datenmenge zu komprimieren. Zu diesem Zweck wird ein Kodebuch berechnet, das stellvertretend für die Daten verwendet wird. Im folgenden wird das Verfahren ausführlich beschrieben.

3.1.1 Die Beschreibung des Verfahrens

Gegeben sei eine Datenmenge $\mathcal{D}$, $|\mathcal{D}| = N$, mit n-dimensionalen Datenvektoren $\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_N}$ und ein Kodebuch $\mathcal{C}$, $|\mathcal{C}| = M$, $M \ll N$, mit n-dimensionalen Kodebuch- bzw. Referenzvektoren $\boldsymbol{w_1}, \dots, \boldsymbol{w_M}$.

Zusätzlich wird noch eine Distanzfunktion d benötigt, die den Abstand zwischen zwei Vektoren bestimmt. Üblicherweise wird hierzu der quadrierte Euklidische Abstand verwendet:

$$d(\boldsymbol{a}, \boldsymbol{b}) := \sum_{i=1}^{n} (a_i - b_i)^2 \qquad \text{für } \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n. \tag{3.1}$$

Zur besseren Lesbarkeit wird eine Indexfunktion $I(\boldsymbol{\xi})$ definiert, die zu einem gegebenen Datenvektor $\boldsymbol{\xi}$ den Index des nächsten Kodebuchvektors liefert. Falls zwei oder mehr Kodebuchvektoren mit dem gleichen Abstand existieren, wird der kleinste Index zurückgegeben, d.h.

$$I(\boldsymbol{\xi}) := \min\{i \mid (d(\boldsymbol{w_i}, \boldsymbol{\xi}) \leq d(\boldsymbol{w_j}, \boldsymbol{\xi}))\ \forall j \in \{1, \dots, M\}\}. \tag{3.2}$$

Die Entscheidung, den kleinsten Index zurückzugeben, ist willkürlich; genauso ist es denkbar, den größten Index zu berechnen.

Die Indexfunktion I gestattet eine einfache Zuordnung eines Datenvektors $\boldsymbol{\xi}$ zu seinem Kodebuchvektor. So lautet beispielsweise der zu $\boldsymbol{\xi}$ gehörige Kodebuchvektor $\boldsymbol{w}_{I(\boldsymbol{\xi})}$.

Um ein Kodebuch $\mathcal{C}$ für eine Datenmenge $\mathcal{D}$ zu bewerten, wird noch ein Fehlermaß benötigt, welches die folgende Fehlerfunktion liefert:

$$E(\mathcal{D}, \mathcal{C}) := \sum_{\boldsymbol{\xi} \in \mathcal{D}} d(\boldsymbol{\xi}, \boldsymbol{w}_{I(\boldsymbol{\xi})}). \tag{3.3}$$

INIT

$\mathcal{C} :=$ (zufällig initialisiert aus der Datenmenge $\mathcal{D}$)

berechne die Menge $\mathcal{R}_i$ für alle $\boldsymbol{w_i} \in \mathcal{C}$

$$\boldsymbol{w_i} := \frac{1}{|\mathcal{R}_i|} \sum_{\xi \in \mathcal{R}_i} \boldsymbol{\xi} \quad \text{für alle } \boldsymbol{w_i} \in \mathcal{C}$$

UNTIL keine Änderungen der $\boldsymbol{w_i}$

RETURN $\mathcal{C}$

Abbildung 3.1: Der LBG Algorithmus

Die Fehlerfunktion summiert den Abstand für jeden Datenvektor $\boldsymbol{\xi}$ zu seinem zugehörigen Kodebuchvektor $\boldsymbol{w}_{I(\boldsymbol{\xi})}$.

Die Aufgabe der Verfahren LBG bzw. LBG-U besteht darin, ein Kodebuch $\mathcal{C}$ zu finden, das den Fehler $E(\mathcal{D}, \mathcal{C})$ für eine gegebene Datenmenge $\mathcal{D}$ minimiert.

Das LBG Verfahren

Das LBG Verfahren, benannt nach seinen Autoren Linde, Buzo und Gray (1980), bildet die Grundlage des LBG-U Verfahrens. In Abbildung 3.1 ist der LBG Algorithmus in Form eines Nassi-Schneiderman Diagramms dargestellt. Dabei enthält die Menge $\mathcal{R}_i$ alle Datenvektoren, die zu dem Kodebuchvektor $\boldsymbol{w_i}$ die geringste Entfernung haben, d.h.

$$\mathcal{R}_i = \{\boldsymbol{\xi} \in \mathcal{D} \mid I(\boldsymbol{\xi}) = i\} \quad \text{für } i = 1, \ldots, M. \tag{3.4}$$

Die Berechnung aller $\mathcal{R}_i$ mit anschließender Anpassung der Kodebuchvektoren $\boldsymbol{w_i}$ wird auch als *Lloyd Iteration* bezeichnet. Somit besteht das LBG Verfahren aus mehreren Lloyd Iterationen, die durch ein Abbruchkriterium überwacht werden.

Die Fehlerfunktion $E(\mathcal{D}, \mathcal{C})$ wird durch das LBG Verfahren in jedem Fall entweder reduziert oder bleibt unverändert. Wenn sich der Fehler nicht mehr ändert, hat das Verfahren ein Minimum der Fehlerfunktion gefunden; in der Regel handelt es sich dabei um ein lokales Minimum.

Das LBG-U Verfahren

Ausgehend von dem durch LBG erzeugten Kodebuch $\mathcal{C}$ berechnet das LBG-U Verfahren für jeden Kodebuchvektor eine Nützlichkeit $\mathcal{U}$ (*Utility*). Der Vektor mit der geringsten Nützlichkeit wird dann in die Nähe des Vektors verschoben, der den höchsten Anteil an dem Fehler hat. Danach berechnet das LBG Verfahren das neue Kodebuch $\mathcal{C}'$. Diese Prozedur wird so lange wiederholt bis das neue Kodebuch $\mathcal{C}'$ keine Verbesserung des Fehlers gegenüber dem alten Kodebuch $\mathcal{C}$ erzielt.

Die Nützlichkeit eines Kodebuchvektors $\boldsymbol{w_i}$ wird gemessen, indem einmal der Fehler für das gesamte Kodebuch $\mathcal{C}$ und anschließend der Fehler für das Kodebuch ohne den Vektor $\boldsymbol{w_i}$ berechnet wird. Die Differenz von beiden ergibt die Nützlichkeit für den Vektor $\boldsymbol{w_i}$:

$$\mathcal{U}(\boldsymbol{w_i}) = E(\mathcal{D}, \mathcal{C} \backslash \boldsymbol{w_i}) - E(\mathcal{D}, \mathcal{C}). \tag{3.5}$$

Das Entfernen von $\boldsymbol{w_i}$ aus $\mathcal{C}$ ändert nur den Fehler der Datenvektoren, die den geringsten Abstand zu $\boldsymbol{w_i}$ haben, d.h. alle Vektoren aus $\mathcal{R}_i$. Jeder Datenvektor $\boldsymbol{\xi} \in \mathcal{R}_i$ wird nun dem zweitnächsten Kodebuchvektor zugeordnet. Die Menge

$$\mathcal{I} = \{1, \ldots, M\} \tag{3.6}$$

enthält alle Indizes der Kodebuchvektoren. Die Indexfunktion $I'(\boldsymbol{\xi})$ liefert zu dem Datenvektor $\boldsymbol{\xi}$ den Index des zweitnächsten Kodebuchvektors:

$$I'(\boldsymbol{\xi}) := \min\{i \,|\, i \in \mathcal{I} \backslash I(\boldsymbol{\xi}) \wedge (d(\boldsymbol{w_i}, \boldsymbol{\xi}) \leq d(\boldsymbol{w_j}, \boldsymbol{\xi}))\ \forall j \in \mathcal{I} \backslash I(\boldsymbol{\xi})\}. \tag{3.7}$$

Alle Datenvektoren, die nicht in $\mathcal{R}_i$ enthalten sind, bleiben weiterhin ihrem Kodebuchvektor zugeordnet. Daher kann die Nützlichkeit aus (3.5) auch folgendermaßen berechnet werden:

$$\mathcal{U}(\boldsymbol{w_i}) = \sum_{\boldsymbol{\xi} \in \mathcal{R}_i} d(\boldsymbol{\xi}, \boldsymbol{w}_{I'(\boldsymbol{\xi})}) - d(\boldsymbol{\xi}, \boldsymbol{w_i}) \quad \text{für } i = 1, \ldots, M. \tag{3.8}$$

Das Maß $\mathcal{U}$ bietet nun die Möglichkeit die Kodebuchvektoren zu bewerten. Kodebuchvektoren ohne Nutzen können entfernt werden, ohne daß sich der Fehler ändert.

Der Kodebuchvektor mit der geringsten Nützlichkeit $\boldsymbol{w_a}$ wird in die Nähe des Vektors mit dem größten Fehler $\boldsymbol{w_b}$ verschoben. Der Fehler für jeden Kodebuchvektor wird folgendermaßen berechnet:

$$E(\boldsymbol{w_i}) = \sum_{\boldsymbol{\xi} \in \mathcal{R}_i} d(\boldsymbol{\xi}, \boldsymbol{w_i}). \tag{3.9}$$

Aus (3.8) ergibt sich für den Vektor mit der geringsten Nützlichkeit

$$\boldsymbol{w_a} = \arg\, \min_{\boldsymbol{w} \in \mathcal{C}} \mathcal{U}(\boldsymbol{w}) \tag{3.10}$$

und aus (3.9) für den Vektor mit dem größten Fehler

$$\boldsymbol{w_b} = \arg\, \max_{\boldsymbol{w} \in \mathcal{C}} E(\boldsymbol{w}). \tag{3.11}$$

Es ist nicht ratsam $\boldsymbol{w_a}$ direkt an dieselbe Position von $\boldsymbol{w_b}$ zu verschieben, da sonst alle Datenvektoren in $\mathcal{R}_b$ dieselbe Entfernung zu $\boldsymbol{w_a}$ und $\boldsymbol{w_b}$ haben. Laut der Definition von $I(\boldsymbol{\xi})$ (3.2) wird der Vektor mit dem kleinsten Index ausgewählt, d.h. immer derselbe Vektor. Dieser befindet sich allerdings schon im Schwerpunkt seiner Datenvektoren und wird bei der Lloyd Iteration seine Position nicht mehr verändern. Der andere Vektor bekommt keinen Datenvektor zugeordnet und liefert somit keinen Beitrag zur Reduktion des Fehlers (wird auch als *dead unit* bezeichnet). Die Verschiebung des Vektors $\boldsymbol{w_a}$ führte also zu einem schlechteren lokalen Minimum.

Um der gerade beschriebenen Situation aus dem Weg zu gehen, wird zu einem der beiden Vektoren ein kleiner Offset-Vektor addiert, der diese unterscheidbar machen soll. Allerdings darf dieser Offset-Vektor nicht zu groß gewählt werden, damit sichergestellt ist, daß $\mathcal{R}_b$ noch in zwei Untermengen unterteilt wird.

Die Länge des Offset-Vektors soll kleiner sein als die standardisierte Abweichung $\sqrt{E(\boldsymbol{w_b})/|\mathcal{R}_b|}$ der Vektoren in $\mathcal{R}_b$. Damit die Kardinalität $|\mathcal{R}_b|$ nicht berechnet werden muß, wird die standardisierte Abweichung durch $\sqrt{E(\boldsymbol{w_b})/N}$ ersetzt. Die Länge des Offset-Vektors wird dadurch nicht größer, da $N \geq |\mathcal{R}_b|$. Anschließend wird noch eine kleine Konstante ϵ gewählt, $0 < \epsilon \ll 1$, so daß die Länge des Offset-Vektors bestimmt wird durch

$$l = \epsilon \sqrt{\frac{E(\boldsymbol{w_b})}{N}}. \tag{3.12}$$

Um die Richtung des Offset-Vektors festzulegen, wird ein zufälliger Vektor $\boldsymbol{u}$ gewählt:

$$\boldsymbol{u} = (\text{zufälliger Vektor aus } \mathbb{R}^n). \tag{3.13}$$

Die neue Position des Kodebuchvektors $\boldsymbol{w_a}$ mit der geringsten Nützlichkeit ergibt sich daher folgendermaßen:

$$\boldsymbol{w_a} := \boldsymbol{w_b} + l\boldsymbol{u}. \tag{3.14}$$

Nach der Verschiebung berechnet LBG das neue Kodebuch. Dieser Vorgang (LBG, Verschiebung eines Kodebuchvektors) wird so lange durchgeführt bis sich der Fehler $E(\mathcal{D}, \mathcal{C})$ nicht mehr verringert. Das bis dahin beste Kodebuch ist das Ergebnis des LBG-U Verfahrens. Der gesamte Algorithmus ist wiederum in Form eines Nassi-Schneiderman Diagramms in Abbildung 3.2 dargestellt.

Die Abbildung 3.3 zeigt die Verbesserung des LBG-U Verfahrens gegenüber des LBG Verfahrens. Der Fehler sinkt um ca. 15 % nach Anwendung des LBG-U Verfahrens.

3.1.2 Die Ergebnisse des Verfahrens

Vor der Ermittlung der möglichen Startpositionen durch das LBG-U Verfahren müssen die in Kapitel 2 besprochenen Daten noch aufbereitet werden. Die Aufbereitung dient der Extraktion der einzelnen Sequenzen. Da der Beginn einer Sequenz nicht markiert ist[1], werden zuerst die einzelnen Sequenzen lokalisiert: wenn in einer gewissen Anzahl von Bildern kein Gesicht vorkommt, dann startet eine Sequenz mit dem nächsten gefundenen Gesicht.

Jetzt stellt sich die Frage, wieviel Datenpunkte vom Anfang jeder Sequenz verwendet werden. Die Entscheidung ist abhängig von mehreren Faktoren:

- Wie schnell bewegen sich die Personen?
- Wie hoch ist die Aufnahmerate?
- Wie liegen die Startpositionen zueinander?

Wird die Anzahl der Datenpunkte zu hoch gewählt, ergibt sich ein zu großer Bereich, d.h. die Startpositionen überschneiden sich oder überdecken die gesamte Szene. Werden zu wenig Datenpunkte betrachtet, sind die gefundenen Bereiche zu klein und das Risiko steigt, daß ein Gesicht nicht in den betrachteten Bereichen erscheint.

In Abhängigkeit von allen erwähnten Aspekten (s. auch Anhang B) wurden die ersten sechs Datenpunkte jeder Trainingssequenz gewählt. Allerdings

[1] Die Information ist durch die Art der Aufnahme der Sequenzen vorhanden. Sie wird aber nicht verwendet, weil die Kamera bei einem Realzeitsystem einen Strom von Bildern liefert und bei den darin enthaltenen Sequenzen sind weder Anfang noch Ende markiert.

INIT

$\mathcal{C}_{\text{aktuell}} :=$ (zufällig initialisiert aus der Datenmenge $\mathcal{D}$)

$\mathcal{C}_{\text{best}} := \mathcal{C}_{\text{aktuell}}$

fertig $:=$ FALSE

$\epsilon := 0,0001$

WHILE $\neg$(fertig)

berechne die Menge $\mathcal{R}_i$ für alle $\boldsymbol{w_i} \in \mathcal{C}$

$$\boldsymbol{w_i} := \frac{1}{|\mathcal{R}_i|} \sum_{\xi \in \mathcal{R}_i} \boldsymbol{\xi} \quad \text{für alle } \boldsymbol{w_i} \in \mathcal{C}$$

UNTIL keine Änderungen der $\boldsymbol{w_i}$

$E(\mathcal{D}, \mathcal{C}_{\text{aktuell}}) < E(\mathcal{D}, \mathcal{C}_{\text{best}})$

T

$\mathcal{C}_{\text{best}} := \mathcal{C}_{\text{aktuell}}$

$\boldsymbol{w_a} = \text{arg min}_{\boldsymbol{w} \in \mathcal{C}_{\text{aktuell}}} \mathcal{U}(\boldsymbol{w})$

$\boldsymbol{w_b} = \text{arg max}_{\boldsymbol{w} \in \mathcal{C}_{\text{aktuell}}} E(\boldsymbol{w})$

$\boldsymbol{u} =$
(zufälliger Vektor aus $\mathbb{R}^n$)

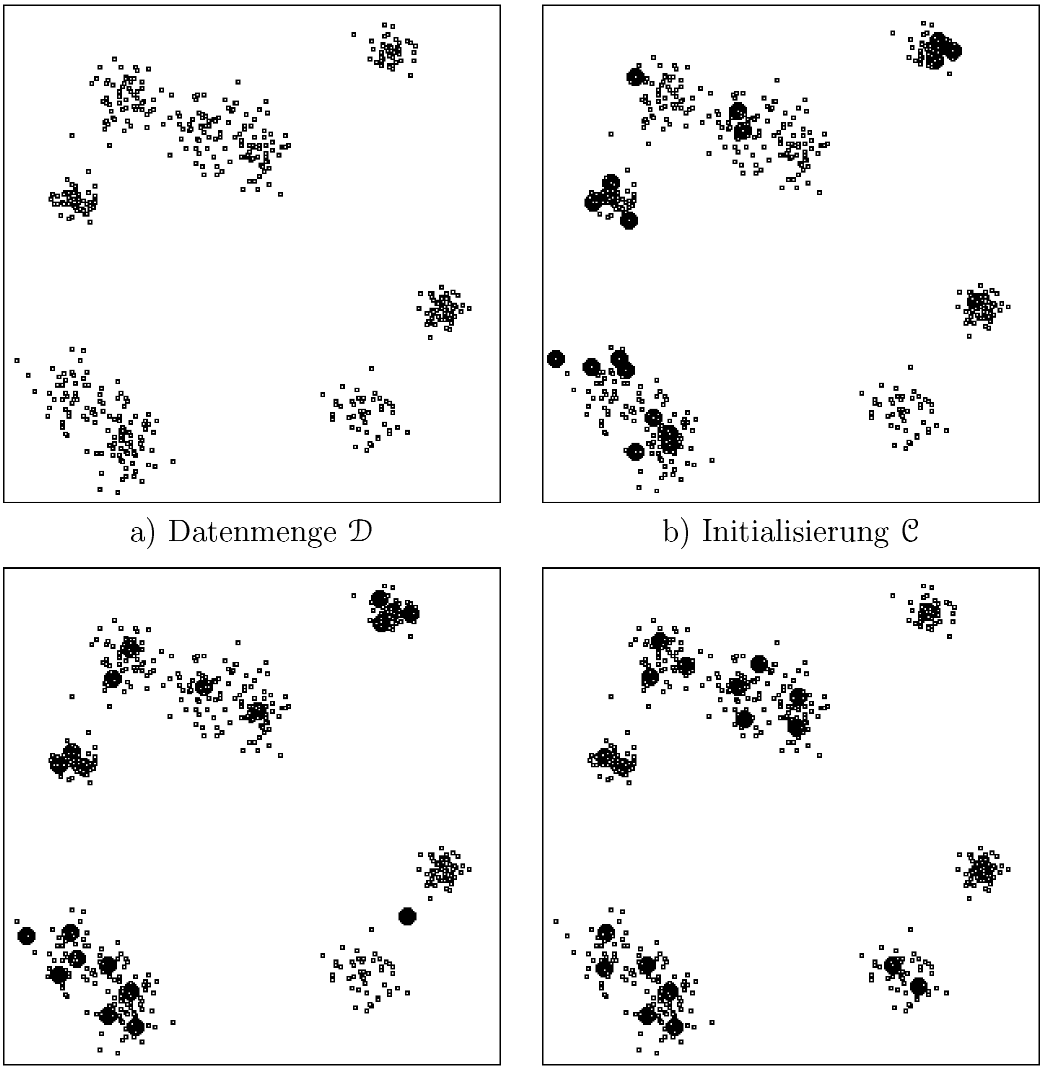
$\boldsymbol{w_a} := \boldsymbol{w_b} + \epsilon\sqrt{\frac{E(\boldsymbol{w_b})}{N}}\,\boldsymbol{u}$

F

fertig $:=$ TRUE

RETURN $\mathcal{C}_{\text{best}}$

Abbildung 3.2: Der LBG-U Algorithmus



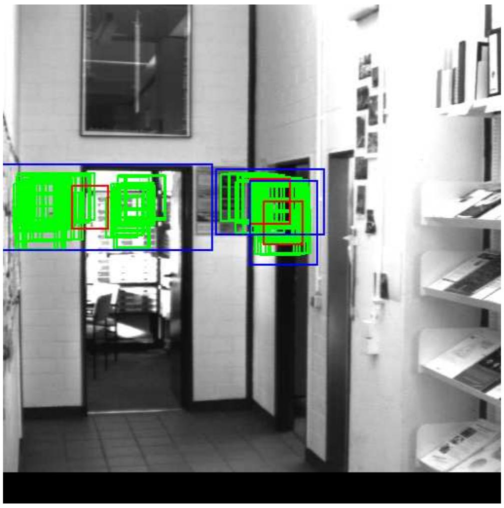
a) Datenmenge $\mathcal{D}$ b) Initialisierung $\mathcal{C}$

c) LBG   d) LBG-U

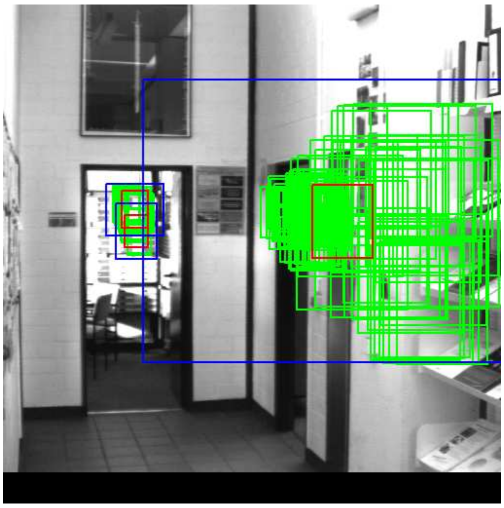
Abbildung 3.3: Ein Beispiel für das LBG-U Verfahren: (a) die Datenmenge $\mathcal{D}$ enthält 500 Vektoren; (b) die zufällige Initialisierung des Kodebuchs $\mathcal{C}$ mit 20 Vektoren; (c) das Kodebuch $\mathcal{C}$ nach dem LBG Verfahren; (d) das Kodebuch $\mathcal{C}$ nach dem LBG-U Verfahren.

führt eine geringfügige Erhöhung oder Verringerung der Datenpunkte zu keinem wesentlich anderem Ergebnis.

Zuerst wurde versucht, die Startpositionen mit dem LBG Verfahren zu ermitteln. Dieses Verfahren ist sehr schnell und konvergiert schon nach wenigen Durchläufen. Leider endet das Verfahren manchmal in einem lokalen



CD 1



CD 2

Abbildung 3.4: Das Ergebnis des LBG Verfahrens bei unglücklicher Initialisierung. $|\mathcal{C}| = 3$, CD 1: $|\mathcal{D}| = 126$, CD 2: $|\mathcal{D}| = 144$.

Minimum (s. Abbildung 3.4), da es auf eine gute Initialisierung angewiesen ist.

Ein Rechteck in den Abbildungen 3.4 und 3.5 ist definiert durch einen vierdimensionalen Vektor $\boldsymbol{r} = (x, y, w, h)$. Die Komponenten x und y definieren den Mittelpunkt, w und h die Breite und Höhe des Rechtecks.

Die vielen hellen Rechtecke stellen die Datenmenge $\mathcal{D}$ (CD 1: $|\mathcal{D}| = 126$, CD 2: $|\mathcal{D}| = 144$) dar, die kleinen dunklen Rechtecke repräsentieren die Kodebuchvektoren (CD 1 und 2: $|\mathcal{C}| = 3$).

Die umschließenden dunklen Rechtecke werden aus den Kodebuchvektoren berechnet und bestimmen die Startpositionen. Für jeden Kodebuchvektor $\boldsymbol{w_i} = (w_{i_x}, w_{i_y}, w_{i_w}, w_{i_h})$ wird für jede Komponente seiner zugeordneten Datenvektoren $\boldsymbol{\xi} \in \mathcal{R}_i$ die Standardabweichung $\boldsymbol{s_i} = (s_{i_x}, s_{i_y}, s_{i_w}, s_{i_h})$ bestimmt. Die Startpositionen $\boldsymbol{p_i} = (p_{i_x}, p_{i_y}, p_{i_w}, p_{i_h})$ berechnen sich dann wie folgt:

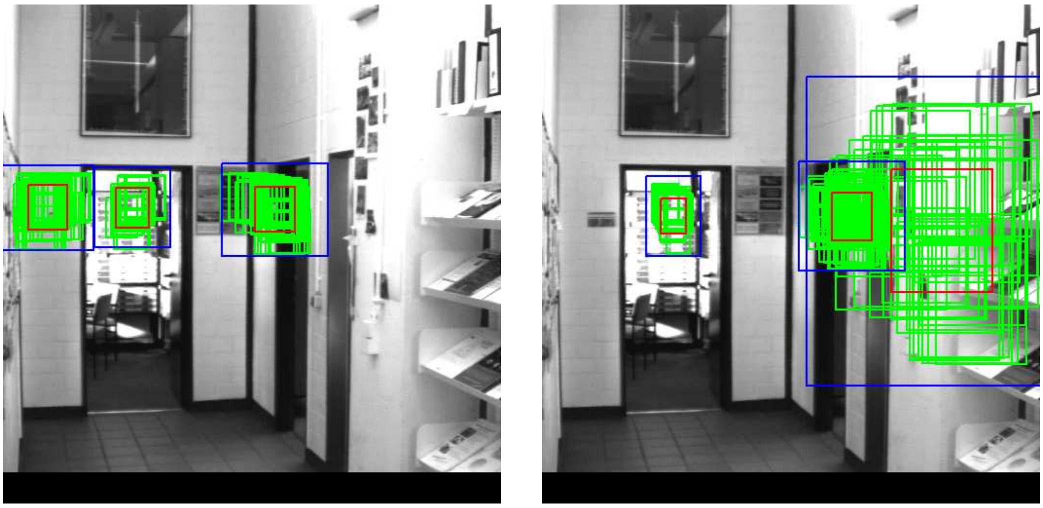
$$p_{i_x} = w_{i_x}, \tag{3.15}$$

$$p_{i_y} = w_{i_y}, \tag{3.16}$$

$$p_{i_w} = w_{i_w} + 2f_w(s_{i_w} + s_{i_x}), \tag{3.17}$$

$$p_{i_h} = w_{i_h} + 2f_h(s_{i_h} + s_{i_y}) \qquad \text{für } f_w,\ f_h \in \mathbb{R} \text{ und } \boldsymbol{s_i},\ \boldsymbol{p_i} \in \mathbb{R}^4. \tag{3.18}$$

Die Position des Rechtecks wird also nicht verändert, sondern nur die Größe. Die Faktoren f_w und f_h dienen zur Skalierung, damit die Fläche der Startpositionen den Erfordernissen des eingesetzten Gesichtsfinders angepaßt werden kann. Sie waren für alle Testläufe konstant ($f_w = 2{,}2$ und $f_h = 1{,}6$).



CD 1 CD 2

Abbildung 3.5: Das Ergebnis des LBG-U Verfahrens: es konvertiert immer zu den abgebildeten guten Startpositionen. $|\mathcal{C}| = 3$, CD 1: $|\mathcal{D}| = 126$, CD 2: $|\mathcal{D}| = 144$.

Nachdem das LBG Verfahren teilweise unbefriedigende Ergebnisse lieferte, wurde das LBG-U Verfahren eingesetzt. LBG-U ist unabhängig von einer guten Initialisierung des Kodebuchs und ermittelt immer gute Startpositionen (s. Abbildung 3.5).

Die Abbildung 3.5 zeigt das Ergebnis für die Bestimmung der Startpositionen der Daten CD 1 und CD 2 mit dem LBG-U Verfahren. Das Verfahren konvertiert immer zu demselben Minimum. Alle Daten werden richtig klassifiziert, bei CD 1 liegen 120 der 126 Datenvektoren vollständig in den ermittelten Startpositionen, bei CD 2 alle 144.

Die Art der verwendeten Daten erlaubt eine vereinfachte Berechnung des Offset-Vektors für das LBG-U Verfahren (s. (3.12) - (3.14)).

Zur Festlegung eines Offset-Vektors reicht ein beliebiger Einheitsvektor, da die kleinste Einheit ein Bildpunkt (Pixel) ist. Dieser Offset-Vektor erfüllt alle in Abschnitt 3.1.1 gestellten Bedingungen.

Die Anzahl der Kodebuchvektoren wird vom Benutzer vorgegeben und ist identisch mit den real existierenden Startpositionen in der betrachteten Szene. Eng beieinander liegende Startpositionen können auch verschmolzen werden, indem entsprechend die Anzahl von Kodebuchvektoren verringert wird.

Zur Bestimmung der Anfangsposition in der aktuellen Sequenz werden alle möglichen Startpositionen gewichtet nach ihrer Wahrscheinlichkeit zyklisch durchlaufen (pro Bild wird nur eine Position getestet). Damit erspart man sich unnötigen Rechenaufwand, weil die Gesichter mehrfach in den berechneten Startpositionen auftauchen.

Mit einem zyklischem Durchlauf durch alle berechneten Ergebnisvektoren kann auf jeden Fall der Startpunkt einer Sequenz bestimmt werden. Die Betrachtung eines einzelnen kleinen Ausschnitts aus der ursprünglichen Szene schränkt den Parameterraum stark ein, so daß eine Lösung des Problems der Bestimmung der Startpositionen gefunden wurde.

Eine neue Sequenz startet, wenn in einer gewissen Anzahl von Bildern kein Gesicht gefunden wurde. Ein zyklischer Durchlauf durch alle Startpositionen endet, wenn ein Gesicht gefunden wird. Anschließend beginnt die Vorhersage der nächsten Gesichtsposition (s. Kapitel 4).

3.2 Andere Verfahren

Es existieren mehrere andere Verfahren, die das Problem der Bestimmung der Startpositionen lösen könnten. Warum wurde also das LBG-U Verfahren ausgewählt?

In Betracht kamen verschiedene *Online*-Lernverfahren, wie zum Beispiel k-means (MacQueen, 1967) oder Growing Neural Gas (Fritzke, 1994, 1995). Aber nachdem das LBG-U Verfahren implementiert und getestet war, gab es keinen Grund mehr, noch andere Verfahren zu erproben. Die Ergebnisse des LBG-U Verfahrens waren so überzeugend (s. Abschnitt 3.1.2), daß ein anderes Verfahren keine sichtbaren Verbesserungen hätte erzielen können.

Die Vorteile des LBG-U Verfahrens sind:

- es konvergiert in einer endlichen Anzahl von Schritten,
- es benötigt sehr wenige Iterationen bis zur Konvergenz,
- außer der Anzahl der Kodebuchvektoren brauchen keine Parameter eingestellt zu werden,
- eine einzelne Iteration nimmt sehr wenig Rechenzeit in Anspruch,
- es konvergiert bei den vorhandenen Daten immer zum globalen Minimum.

Um das Kodebuch der Abbildung 3.5 zu berechnen, benötigt LBG-U für CD 1 durchschnittlich 18 Lloyd Iterationen und für CD 2 durchschnittlich 15 Lloyd Iterationen. Der gesamte Algorithmus konvergiert nach weniger als einer halben Sekunde[2].

[2]Rechner: Sun SS 20/612 mit Solaris 2.5 und Intel P 100 mit Linux 2.0.29

Kapitel 4

Das Problem der Vorhersage der nächsten Position

Bereits im Alter von vier Monaten entwickelt sich beim Kind die Fähigkeit einem bewegten Objekt mit den Augen nicht nur zu folgen, sondern dessen Bewegungspfad in Grenzen vorherzusagen (Oerter und Montada, 1995). Das Kind hat also aus den bisherigen Erfahrungen mit ähnlichen Objekten gelernt. Diese Verhaltensweise erlaubt ein sehr viel effektiveres Blickverhalten.

Dieses Kapitel beschäftigt sich mit dem Problem der Vorhersage der nächsten Position eines sich bewegenden Objekts. Um eine Vorhersage machen zu können, wird die aktuelle Position des Objekts als bekannt vorausgesetzt. Die Bestimmung der Startpositionen in einer Szene für ein bewegtes Objekt wurde bereits im letzten Kapitel besprochen.

Für die Lösung dieses Problems werden Verfahren eingesetzt, die sehr unterschiedliche Methoden repräsentieren. Zuerst wird ein statistisches Lernverfahren untersucht (*k–Nearest-Neighbor*), anschließend zwei Neuronale Netze. Das eine Netz besitzt keine feste Netzwerkgröße (*Growing Neural Gas*), das andere ist das *Multilayer Perzeptron*.

4.1 Die Lösung mit dem k–Nearest-Neighbor Verfahren

Das k–Nearest-Neighbor (k-NN) Verfahren ist eines der am gründlichsten untersuchten Verfahren auf dem Gebiet des Maschinellen Lernens. Es ist einfach und erste theoretische Ergebnisse wurden schon früh von Cover und Hart (1967) publiziert. Duda und Hart (1973) geben einen guten Überblick über das Verfahren. Eine kurze Zusammenfassung ist in Mitchell (1997) zu finden.

Das k–Nearest-Neighbor Verfahren gehört zu der Klasse der beispielbasierten Schätzverfahren. Diese versuchen direkt die (diskrete oder kontinuierliche) Zielfunktion zu schätzen. Das Lernen der Trainingsmuster erfolgt durch einfaches Abspeichern. Neue Muster werden klassifiziert durch Berechnung der ähnlichsten Trainingsbeispiele und Übernahme des entsprechenden Klassenlabels.

4.1.1 Die Beschreibung des Verfahrens

Gegeben sei eine Datenmenge $\mathcal{D}$, $|\mathcal{D}| = N$, mit n-dimensionalen Datenvektoren $\boldsymbol{\xi_1}, \dots, \boldsymbol{\xi_N} \in \mathbb{R}^n$. Zur Bestimmung der nächsten Nachbarn eines Datenvektors $\boldsymbol{\tau}$ wird eine Distanzfunktion benötigt. Als Distanzfunktion wird üblicherweise der Euklidische Abstand verwendet.

Der Euklidische Abstand für zwei Vektoren $\boldsymbol{a} = (a_1, \dots, a_n)$ und $\boldsymbol{b} = (b_1, \dots, b_n)$ ist definiert als $d(\boldsymbol{a}, \boldsymbol{b})$ mit

$$d(\boldsymbol{a}, \boldsymbol{b}) := \sqrt{\sum_{i=1}^{n} (a_i - b_i)^2} \qquad \text{für } \boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^n. \tag{4.1}$$

Der Lernalgorithmus für das k–Nearest-Neighbor Verfahren ist trivial und lautet wie folgt:

1. Füge jeden Trainingsvektor $\xi_i \in \mathcal{D}$ in die Menge der Trainingsbeispiele $\mathcal{T}$ ein.

Für kontinuierliche Räume wird der k–Nearest-Neighbor Algorithmus beschrieben durch:

1. Gegeben sei der Datenvektor $\boldsymbol{g} \in \mathbb{R}^n$.
2. Bestimme die zu $\boldsymbol{g}$ k nächsten Nachbarn $\boldsymbol{g_1}, \dots, \boldsymbol{g_k}$ aus der Menge der Trainingsbeispiele $\mathcal{T}$ bezüglich der Distanzfunktion $d()$.
3. Der Rückgabewert $\boldsymbol{r}$ berechnet sich dann aus dem Mittelwert der k nächsten Nachbarn:

$$\boldsymbol{r} = \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{g_i}. \tag{4.2}$$

Für diskrete Räume kann der oben angegebene Algorithmus einfach abgeändert werden: statt des Mittelwerts der k nächsten Nachbarn (Gleichung (4.2)) wird die am häufigsten vorkommende Klasse zurückgeliefert.

Die Erweiterung des k–Nearest-Neighbor Verfahrens

Das vorgesehene Einsatzgebiet und die Art der Daten machen eine Erweiterung des k-NN Algorithmus' notwendig. Folgende Aspekte sind dabei zu beachten:

- Die Daten liegen als vierdimensionale Vektoren angeordnet in Sequenzen vor. Jeder Vektor hat einen genau definierten Nachfolger bis auf den Endvektor jeder Sequenz.
- Das k–Nearest-Neighbor Verfahren wird eingesetzt um für ein gefundenes Gesicht die nächste Position und Größe vorherzusagen.

Zuerst muß geklärt werden, ob die nächsten Nachbarn nur aus verschiedenen Sequenzen stammen dürfen oder ob die Wahl der Nachbarn beliebig ist. In Abbildung 4.1 ist der Unterschied zwischen nächsten Nachbarn aus beliebigen und disjunkten Sequenzen deutlich zu erkennen. Die Sequenzen $S_0, \ldots, S_4$ sind in der Trainingsmenge $\mathcal{T}$ enthalten. Zum Vektor $\boldsymbol{g}$ sind die 5 nächsten Nachbarn $\boldsymbol{g_1}, \ldots, \boldsymbol{g_5}$ eingezeichnet. Je nach Wahl der Nachbarn kann es zu ganz unterschiedlichen Ergebnissen kommen.

Als nächstes wird der neue Parameter Durchschnittsbereich ϕ eingeführt. Für $\phi = 0$ ändert sich an dem k-NN Algorithmus nichts. Ist ϕ größer als Null,

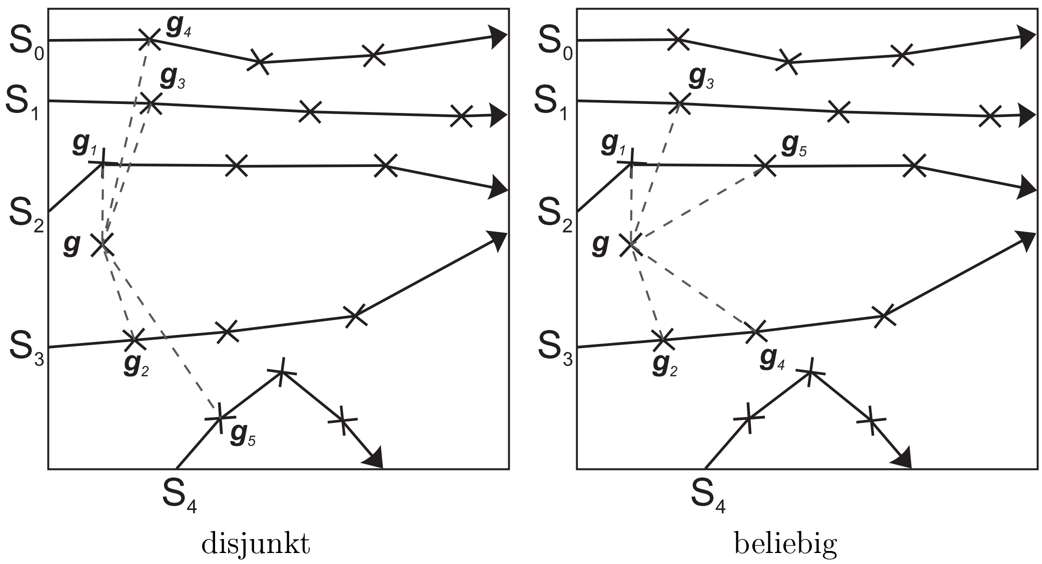


Abbildung 4.1: Das eingesetzte k-Nearest-Neighbor Verfahren (hier $k = 5$) zur Bestimmung der nächsten Gesichtsposition: der Unterschied zwischen nächsten Nachbarn aus disjunkten bzw. beliebigen Sequenzen.

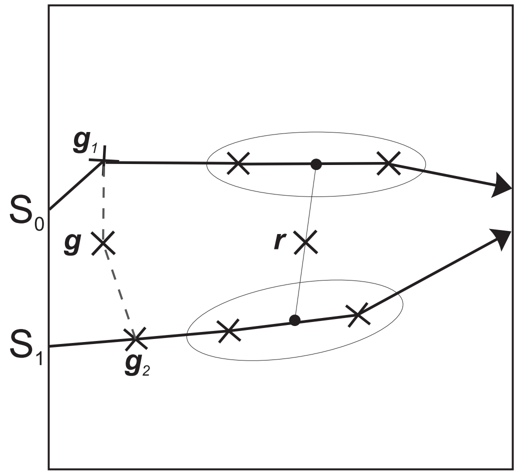


Abbildung 4.2: Der Einfluß des Durchschnittsbereichs ϕ (average range) auf den Rückgabewert $\boldsymbol{r}$ des k–Nearest-Neighbor Verfahrens. In diesem Beispiel ist $k = 2$ und $\phi = 2$.

wird statt des Mittelwerts der k nächsten Nachbarn $\boldsymbol{g_1}, \ldots, \boldsymbol{g_k}$ der Mittelwert ihrer $k * \phi$ Nachfolger gebildet (s. Abbildung 4.2).

Die Gleichung (4.2) wird ersetzt durch

$$\boldsymbol{r} = \begin{cases} \frac{1}{k} \sum_{i=1}^{k} \boldsymbol{g_i} & \text{für } \phi = 0, \\ \frac{1}{k\phi} \sum_{i=1}^{k} \sum_{j=1}^{\phi} p(\boldsymbol{g_i}, j) & \text{sonst.} \end{cases} \tag{4.3}$$

Die Funktion $p(\boldsymbol{g^{S,m}}, l)$ liefert für den Vektor $\boldsymbol{g^{S,m}}$ in der Sequenz S an der Stelle m den l-ten Nachfolger $\boldsymbol{g^{S,m+l}}$ oder den letzten Vektor der Sequenz $\boldsymbol{g^{S,|S|}}$.

$$p(\boldsymbol{g^{S,m}}, l) = \begin{cases} \boldsymbol{g^{S,m+l}} & \text{für } m + l \leq |S|, \\ \boldsymbol{g^{S,|S|}} & \text{sonst.} \end{cases} \tag{4.4}$$

Durch die Anordnung der Trainingsdaten in Sequenzen und die Einführung des Durchschnittbereichs ϕ ist es jetzt möglich das k–Nearest-Neighbor Verfahren zur Vorhersage einzusetzen.

4.1.2 Die Bestimmung der einstellbaren Parameter

Durch die Erweiterung des k–Nearest-Neighbor Algorithmus' sind vor dem Einsatz des Verfahrens drei Parameter zu ermitteln: zu der Anzahl der zu betrachtenden nächsten Nachbarn k kommt der Durchschnittsbereich ϕ und die Wahl der Nachbarn aus beliebigen oder disjunkten Sequenzen.

Um die unterschiedlichen Parametersätze zu testen, werden die vorhandenen Daten von CD 1 und CD 2 (s. Kapitel 2) aufgeteilt in verschiedene Trainings- und Testmengen. Die Trainingsmenge ist doppelt so groß wie die Testmenge und enthält keine Sequenzen aus der Testmenge, d. h. die beiden Mengen sind disjunkt (s. Anhang C).

Für jeden Parametersatz werden alle Vektoren der Testmenge dem mit der Trainingsmenge gelernten k-NN Verfahren präsentiert und das erhaltene Resultat verglichen mit dem tatsächlichen Wert. Diese Werte werden gemittelt und ergeben ein Maß für die Güte der Vorhersage mit dem verwendeten Parametersatz.

Ausgehend von dem ersten Gesichtsvektor $\boldsymbol{g}^{\boldsymbol{S_i,0}}$ einer Sequenz S_i wird durch das k-NN Verfahren der nächste Gesichtsvektor $\boldsymbol{v}^{\boldsymbol{S_i,1}}$ bestimmt. Der Vektor $\boldsymbol{v}^{\boldsymbol{S_i,1}}$ wird mit der tatsächlichen Gesichtsposition $\boldsymbol{g}^{\boldsymbol{S_i,1}}$ verglichen und der mittlere quadratische Fehler MSE_{i1} berechnet (*mean square error, MSE*).

Für jeden Gesichtsvektor $\boldsymbol{g}^{\boldsymbol{S_i,j}}$ einer Sequenz S_i, $|S_i| = n_i$, errechnen sich der MSE_{ij} folgendermaßen:

$$MSE_{ij} = \frac{1}{2}(\boldsymbol{g}^{\boldsymbol{S_i,j}} - \boldsymbol{v}^{\boldsymbol{S_i,j}})^2 \tag{4.5}$$

Für jede Sequenz S_i wird dann der kleinste ($MSE_{\min,i}$), der größte ($MSE_{\max,i}$) und der durchschnittliche mittlere quadratische Fehler (MSE_i) berechnet.

$$MSE_i = \frac{1}{n_i - 1} \sum_{j=1}^{n_i} MSE_{ij} \tag{4.6}$$

$$MSE_{\min,i} = \min\{MSE_{ij}\} \tag{4.7}$$

$$MSE_{\max,i} = \max\{MSE_{ij}\} \tag{4.8}$$

Aus allen m Testsequenzen wird anschließend der kleinste ($\varnothing MSE_{\min}$), der größte ($\varnothing MSE_{\max}$) und der globale durchschnittliche mittlere quadratische Fehler $\varnothing MSE$ für diesen Testlauf berechnet.

$$\varnothing MSE = \frac{1}{m}\sum_{i=1}^{m} MSE_i \tag{4.9}$$

$$\varnothing MSE_{\min} = \frac{1}{m}\sum_{i=1}^{m} MSE_{\min,i} \tag{4.10}$$

$$\varnothing MSE_{\max} = \frac{1}{m}\sum_{i=1}^{m} MSE_{\max,i} \tag{4.11}$$

In allen Grafiken dieses Abschnitts wird der globale durchschnittliche mittlere quadratische Fehler $\varnothing MSE$ in Abhängigkeit von k und dem Durchschnittsbereich angezeigt. Die Einträge in den Tabellen sind für $\varnothing MSE$ der Größe nach sortiert.

Die Wahl der nächsten Nachbarn

Es bietet sich an zuerst zu klären, ob die nächsten Nachbarn nur aus verschiedenen Sequenzen stammen dürfen oder ob die Wahl der Nachbarn beliebig ist. Dazu wurden die Daten von CD 1 und CD 2 in unterschiedliche Trainings- und Testmengenpaare unterteilt:

1. Trainingsmenge von CD 1, Testmenge von CD 1
   (Die Trainings- und Testmenge wurde durch Aufteilung der Daten von CD 1 erzeugt.)

2. Trainingsmenge von CD 2, Testmenge von CD 2
   (Die Trainings- und Testmenge wurde durch Aufteilung der Daten von CD 2 erzeugt.)

3. Trainingsmenge von CD 1 und CD 2, Testmenge von CD 1 und CD 2
   (Die Trainingsmenge setzt sich aus der Vereinigung der Trainingsmengen von CD 1 und CD 2 zusammen, die Testmenge ebenso aus den Testmengen von CD 1 und CD 2.)

4. Trainingsmenge von CD 1 und CD 2 frontal, Testmenge von CD 1 und CD 2 frontal
   (Die Daten entsprechen denen aus dem vorherigen Mengenpaar mit der Ausnahme, daß alle Sequenzen, die von der Kamera wegführen oder zur Kamera parallel laufen, aus der Trainings- und Testmenge entfernt worden sind (s. Abbildung 2.5, Wege 3 und 4). Diese Daten sind praktisch interessant, da der eingesetzte Gesichtsfinder nur frontale Gesichter, aber nicht Seitenansichten oder Hinterköpfe finden kann.)

Ausführliche Informationen zu den Daten sind im Kapitel 2 und im Anhang C zu finden.

Für jedes dieser Paare wurden Testläufe mit variierendem k und ϕ durchgeführt, und zwar einmal mit der Wahl der Nachbarn aus disjunkten und einmal aus beliebigen Sequenzen. Um einen ersten Überblick zu bekommen, wurden die Werte für k und dem Durchschnittsbereich ϕ aus dem Intervall $[1, 15]$ gewählt.

Die Ergebnisse sind in der Tabelle 4.1 dargestellt. Eine repräsentative Visualisierung der Daten ist in Abbildung 4.3 zu sehen. Die Grafiken in dieser Abbildung zeigen die Ergebnisse der Testläufe für CD 1. Die Daten für die anderen Mengenpaare weisen dieselben Charakteristika auf.

Es zeigt sich, daß die Ergebnisse besser sind, wenn die Wahl der nächsten Nachbarn nicht eingeschränkt wird. Über alle Tests gesehen sind die Ergebnisse für eine beliebige Wahl der Nachbarn für $\varnothing MSE$ um ca. 13 %, für $\varnothing MSE_{\text{min}}$ um ca. 35 % und für $\varnothing MSE_{\text{max}}$ um ca. 8 % besser als die eingeschränkte Wahl der Nachbarn (s. Tabelle 4.1). Der beste Parametersatz für disjunkte Nachbarn bzgl. $\varnothing MSE$ erscheint erst an 21.–35. Stelle nach den Parametersätzen für beliebige Nachbarn.

Bei allen Experimenten zur Wahl der Nachbarn ist der globale durchschnittliche mittlere quadratische Fehler geringer, wenn die k nächsten Nachbarn aus beliebigen Sequenzen ausgewählt werden. Daher ist es nicht nötig, auf die Wahl der nächsten Nachbarn zu achten. Dieses Vorgehen spart auch Rechenzeit bei der Berechnung der nächsten Nachbarn.

Die Anzahl der nächsten Nachbarn k und der Durchschnittsbereich ϕ

Die Datensätze für diese Experimente entsprechen denen für die Wahl der Nachbarn. Es wurde nur ein Datensatz hinzugenommen, um die Auswirkungen neuer, d.h. nicht gelernter Sequenzen auf die Parameter zu untersuchen:

5. Trainingsmenge von CD 1, Testmenge von CD 1 und CD 2 frontal
   (Als Trainingsmenge wurde die Trainingsmenge von CD 1 verwendet, die Testmenge setzt sich aus der Testmenge von CD 1 und der Testmenge von CD 2 zusammen. Aus der Testmenge von CD 2 sind alle Sequenzen entfernt worden, die von der Kamera wegführen oder zur Kamera parallel laufen (s. Abbildung 2.5, Wege 3 und 4).)

In den folgenden Experimenten kommt es im wesentlichen auf die Bestimmung der Werte von k und dem Durchschnittsbereich ϕ an. Aus den Experimenten zur Wahl der Nachbarn ist zu ersehen, daß das Intervall für k

CD 1

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	6	1	nein	134,43	2,28	1190,79
2.	8	1	nein	134,74	2,73	1221,60
3.	7	1	nein	136,07	2,86	1211,07
4.	8	2	nein	136,24	3,07	1141,41
⋮	⋮	⋮	⋮	⋮	⋮	⋮
35.	3	1	ja	153,37	6,19	1138,30

CD 2

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	76,27	2,40	600,77
2.	9	1	nein	76,45	2,59	542,73
3.	7	1	nein	76,88	2,62	608,80
4.	10	1	nein	77,41	3,08	552,77
⋮	⋮	⋮	⋮	⋮	⋮	⋮
25.	5	1	ja	95,07	2,31	882,00

CD 1 und CD 2

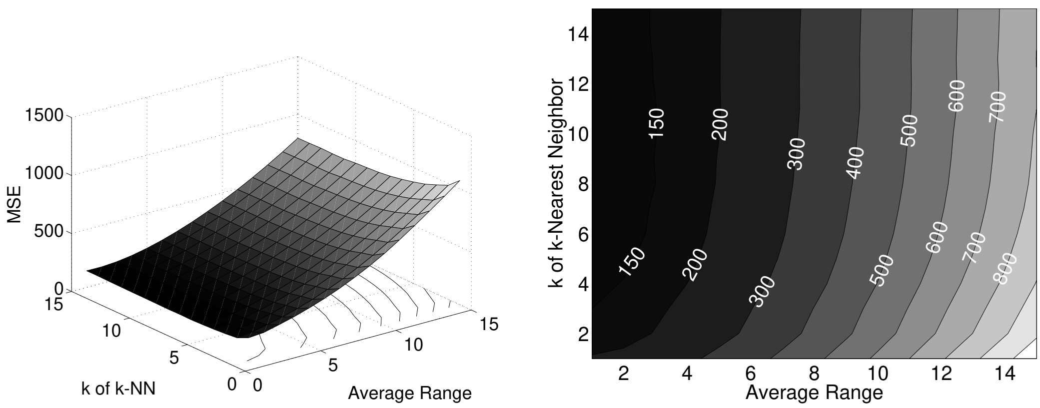
Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	83,79	2,13	814,02
2.	9	1	nein	83,87	2,08	793,21
3.	14	1	nein	84,34	2,36	782,98
4.	13	1	nein	84,36	2,25	770,60
⋮	⋮	⋮	⋮	⋮	⋮	⋮
21.	5	1	ja	90,88	2,75	845,28

CD 1 und CD 2 frontal

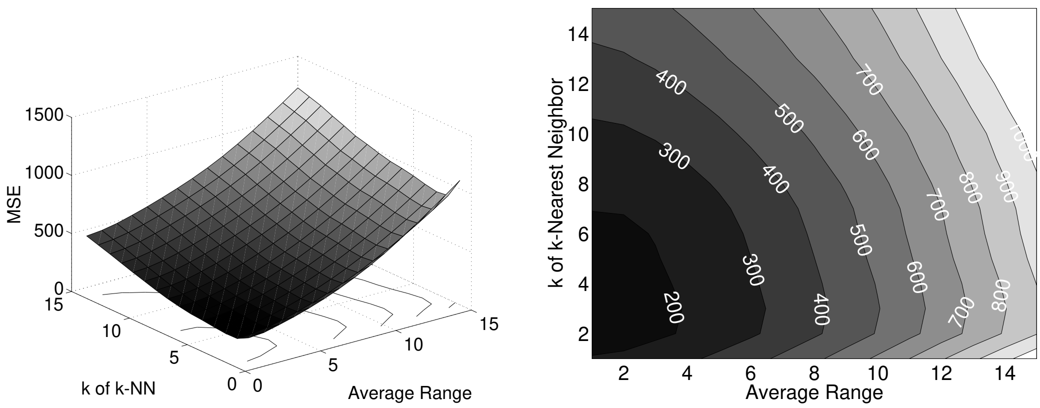
Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	98,26	2,15	993,85
2.	9	1	nein	98,46	1,84	946,14
3.	7	1	nein	99,48	2,43	1018,08
4.	13	1	nein	99,77	2,72	934,46
⋮	⋮	⋮	⋮	⋮	⋮	⋮
21.	4	1	ja	107,76	4,32	1002,79

Tabelle 4.1: Die Ergebnisse der Parametersätze sortiert nach $\varnothing MSE$ für die einzelnen Mengenpaare. Die Anzahl der nächsten Nachbarn k ist aus dem Intervall $[1, 15]$ gewählt, genauso wie der Durchschnittsbereich ϕ. Die letzte Zeile gibt den besten Parametersatz für die Wahl der nächsten Nachbarn nur aus verschiedenen (disjunkten) Sequenzen an.

CD 1



(a)



(b)

Abbildung 4.3: Die visualisierten Werte für die Testläufe mit den Daten von CD 1: Grafik (a) zeigt die Ergebnisse für nächste Nachbarn aus beliebigen Sequenzen und Grafik (b) nächste Nachbarn nur aus verschiedenen Sequenzen.

zu klein und das Intervall für den Durchschnittsbereich zu groß gewählt war (beides aus $[1, 15]$).

Um auszuschließen, daß es noch bessere Parametersätze für $k > 15$ gibt, wird das Intervall auf $[1, 30]$ vergrößert. Gleichzeitig wird das Intervall für den Durchschnittsbereich ϕ auf $[1, 5]$ verkleinert, da die Parametersätze mit $\phi > 5$ keine guten Ergebnisse lieferten.

Die Ergebnisse sind in der Tabelle 4.2 dargestellt. Eine repräsentative Visualisierung der Daten ist in Abbildung 4.4 zu sehen. Die Grafiken in dieser

CD 1

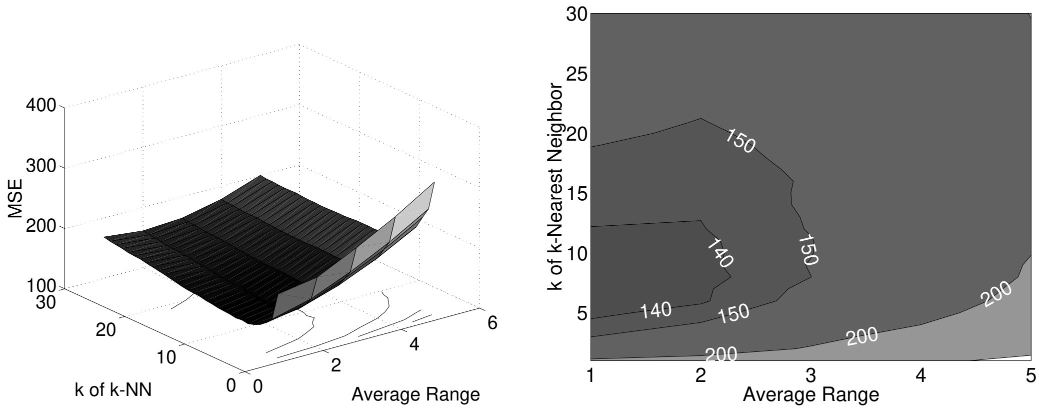


Abbildung 4.4: Die visualisierten Werte für die Testläufe mit den Daten von CD 1. Die Grafiken zeigen die Ergebnisse zur Bestimmung von k und ϕ.

Abbildung zeigen die Ergebnisse der Testläufe für CD 1. Die Daten für die anderen Mengenpaare weisen dieselben Charakteristika auf.

Alle Ergebnisse zeigen eindeutig, daß für den Durchschnittsbereich ϕ der Wert 1 eine gute Wahl ist: die besten Ergebnisse weisen alle diesen Wert auf. Dieses Ergebnis ist auch plausibel im Hinblick auf die zu lösende Aufgabe: das Verfahren soll die Gesichtpositionen im nächsten Bild vorhersagen.

Für die Bestimmung der Anzahl der nächsten Nachbarn k werden die Ergebnisse aller fünf Testläufe der verschiedenen Mengenpaare aufsummiert und wiederum nach $\varnothing MSE$ sortiert. Das Resultat ist in Tabelle 4.3 zu sehen.

Die Wahl für die Anzahl der nächsten Nachbarn fiel auf $k = 9$. Die Auswahl basiert auf der Beobachtung, daß für $k = 9$ der $\varnothing MSE$ unwesentlich schlechter ist als die besten Parametersätze und der $\varnothing MSE_{\max}$ durchschnittlich deutlich geringer ist als bei ähnlich guten Parametern.

Die Experimente haben einen sehr robusten Parametersatz ergeben. Dieser liefert auch dann sehr gute Ergebnisse, wenn Sequenzen der Testmenge hinzugefügt werden, für die es in der Trainingsmenge keine Äquivalente gibt. Diese Situation ergab sich durch die Hinzunahme von Sequenzen mit sich hinhockenden Personen im Experiment für das fünfte Mengenpaar.

Außerdem sind im Datensatz von CD 2 auch Sequenzen, die von der Kamera wegführen, also der allgemeinen Bewegungsrichtung entgegengesetzt sind. Auch dafür brachte der gewählte Parametersatz gute Ergebnisse.

Hier noch einmal eine Zusammenfassung über den in diesem Abschnitt ermittelten Parametersatz für das erweiterte k–Nearest-Neighbor Verfahren:

- nächste Nachbarn aus beliebigen Sequenzen,

CD 1

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	6	1	nein	134,43	2,28	1190,79
2.	8	1	nein	134,74	2,73	1221,60
3.	7	1	nein	136,07	2,86	1211,07
4.	8	2	nein	136,24	3,07	1141,41
5.	9	1	nein	136,54	3,14	1275,21

CD 2

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	76,27	2,40	600,77
2.	9	1	nein	76,45	2,59	542,73
3.	7	1	nein	76,88	2,62	608,80
4.	10	1	nein	77,41	3,08	552,77
5.	11	1	nein	78,86	2,65	551,22

CD 1 und CD 2

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	83,79	2,13	814,02
2.	9	1	nein	83,87	2,08	793,21
3.	14	1	nein	84,34	2,36	782,98
4.	13	1	nein	84,36	2,25	770,60
5.	10	1	nein	84,58	1,99	816,65

CD 1 und CD 2 frontal

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	98,26	2,15	993,85
2.	9	1	nein	98,46	1,84	946,14
3.	7	1	nein	99,48	2,43	1018,08
4.	13	1	nein	99,77	2,72	934,46
5.	10	1	nein	99,79	2,08	983,75

Training CD 1, Test CD 1 und CD 2 frontal

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	6	1	nein	373,26	3,39	3637,32
2.	8	1	nein	374,59	4,22	3644,05
3.	7	1	nein	374,69	4,05	3639,67
4.	5	1	nein	375,97	4,23	3645,07
5.	9	1	nein	376,54	4,37	3677,44

Tabelle 4.2: Die Ergebnisse der Parametersätze sortiert nach $\varnothing MSE$ für die einzelnen Mengenpaare. Die Anzahl der nächsten Nachbarn k ist aus dem Intervall $[1, 30]$ gewählt, der Durchschnittsbereich ϕ aus $[1, 5]$.

Pos.	k	ϕ	disjunkt	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1.	8	1	nein	767,65	13,64	7274,30
2.	9	1	nein	771,87	14,02	7234,75
3.	7	1	nein	772,61	14,17	7317,50
4.	10	1	nein	776,80	13,92	7430,90
5.	6	1	nein	777,62	12,13	7414,60

Tabelle 4.3: Die Summe der Ergebnisse aller fünf Testläufe sortiert nach $\varnothing MSE$.

- Durchschnittsbereich (average range) $\phi = 1$,
- Anzahl nächste Nachbarn $k = 9$.

4.1.3 Die Ergebnisse

Für alle in den Testläufen ermittelten Ergebnissen wurde der im letzten Abschnitt erhaltene Parametersatz verwendet. Im folgenden wird also nur noch allgemein von dem k-NN Verfahren gesprochen; damit ist dann das 9–Nearest-Neighbor Verfahren mit Nachbarn aus beliebigen Sequenzen und dem Durchschnittsbereich $\phi = 1$ gemeint.

Die Abbildung 4.5 zeigt links drei reale Sequenzen und rechts die dazugehörigen Voraussagen. Die Gesichter sind auf jedem Bild durch ein Rechteck markiert. Zur Übersichtlichkeit wurde nur für die Start- und Endposition das komplette Rechteck eingezeichnet. Von den anderen wurden jeweils immer die zueinander gehörigen Ecken verbunden. Diese Darstellung soll einen Eindruck der zeitlichen Reihenfolge vermitteln.

Die Voraussage oben basiert auf der Trainings- und Testmenge von CD 1 (Testmengenpaar 1), die mittlere auf der von CD 2 (Testmengenpaar 2) und die untere auf der Trainingsmenge von CD 1 und der Testmenge von CD 1 und CD 2 frontal (Testmengenpaar 5).

Die untere Vorhersage ist mit Abstand die ungenauste in allen Tests. Das liegt daran, daß keine Trainingsbeispiele für die Testsequenz gelernt wurden. Daher bildet das Verfahren die unbekannten Sequenzen auf gelernte ab. Dieser hohe Fehler bietet aber ein Indiz dafür, daß die entsprechende Testsequenz in die Menge der Trainingsbeispiele aufgenommen werden sollte. Für die obere und mittlere Vorhersage wurden entsprechende Sequenzen gelernt, so daß die Abweichungen sehr gering ausfallen.

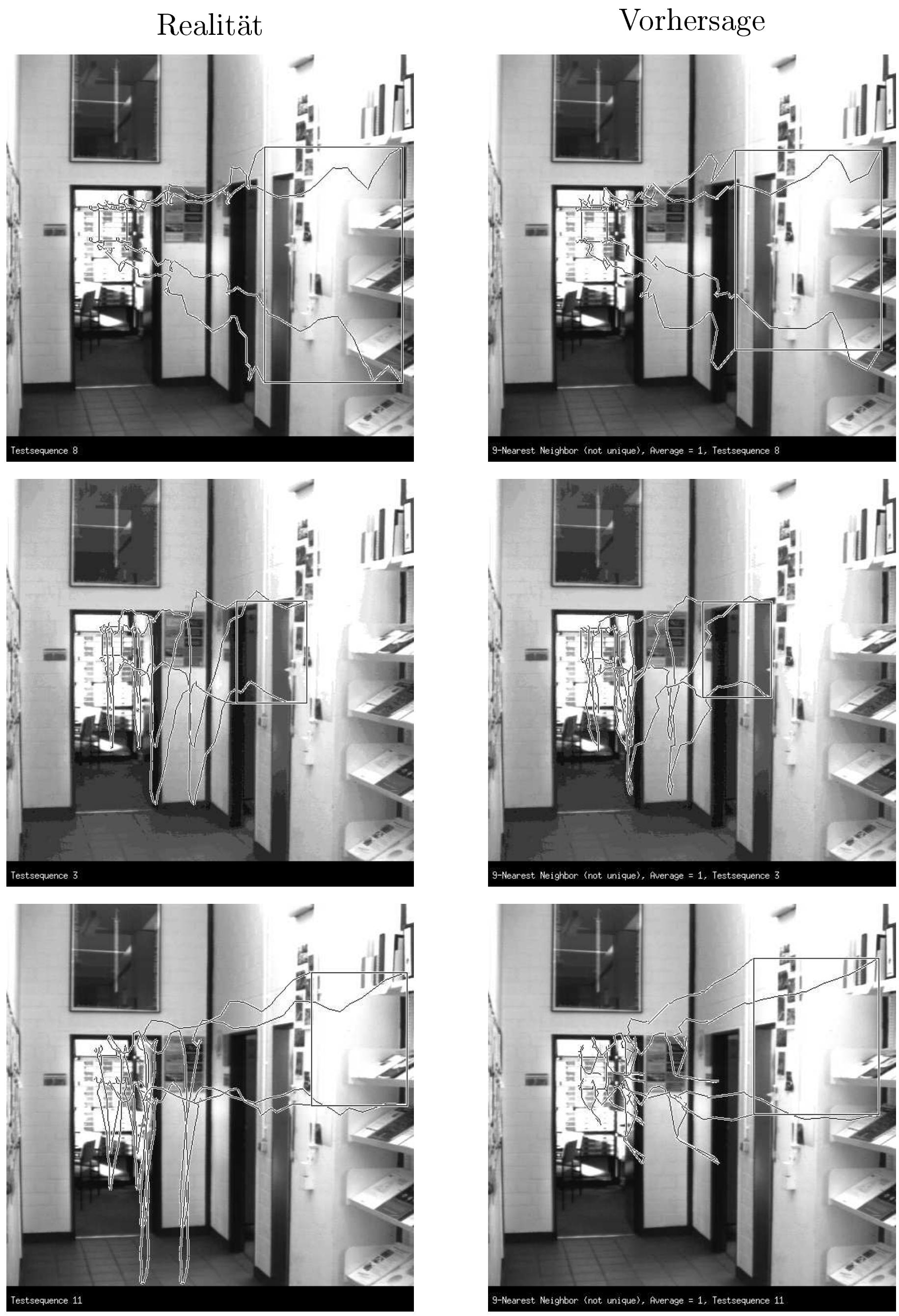


Abbildung 4.5: Drei Ergebnisse des 9–Nearest-Neighbor Verfahrens mit Nachbarn aus beliebigen Sequenzen und Durchschnittsbereich $\phi = 1$. Links zu sehen ist die reale Kopfbewegung der Person, rechts daneben die vorhergesagte. Von oben nach unten vergrößert sich der Fehler: $MSE_{\text{oben}} = 72,32$, $MSE_{\text{Mitte}} = 148,42$ und $MSE_{\text{unten}} = 3222,64$.

4.2 Die Lösung mit Neuronalen Netzen

Zum Vergleich der Ergebnisse des k–Nearest-Neighbor Verfahren werden noch zwei andere Verfahren eingesetzt: zum einen das Growing Neural Gas und zum anderen das Multilayer Perzeptron. Beide bekommen als Eingabe einen vierdimensionalen Vektor und liefern als Ausgabe auch einen vierdimensionalen Vektor. Diese Vektoren entsprechen den in Abschnitt 2.3 beschriebenen Gesichtsvektoren.

Trainiert werden die beiden Netze mit den schon im vorherigen Abschnitt verwendeten Trainingsmengen. Das fertig trainierte Netzwerk wird anschließend mit den Testmengen durchlaufen und dasselbe Fehlermaß berechnet, wie für das k–Nearest-Neighbor Verfahren. Zur besseren Übersicht ist die Aufteilung und Zusammenstellung der Mengenpaare kurz in Abbildung 4.6 und Tabelle 4.4 erläutert.

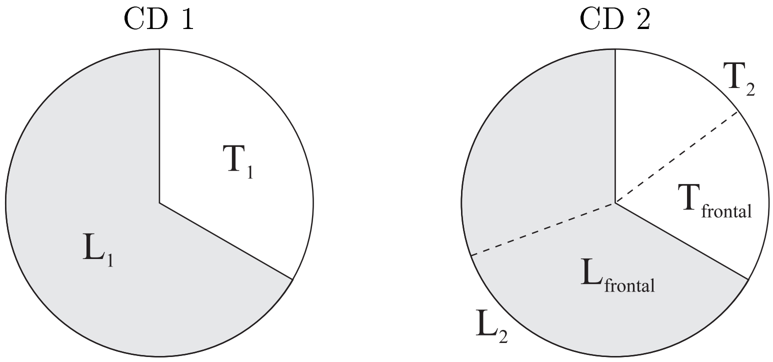


Abbildung 4.6: Die Aufteilung der Daten von CD 1 und CD 2 in der grafischen Übersicht ($\mathbf{L_i}$ sind die Trainingsmengen, $\mathbf{T_i}$ die Testmengen).

Mengenpaar	Trainingsmenge	Testmenge
1	$\mathbf{L_1}$	$\mathbf{T_1}$
2	$\mathbf{L_2}$	$\mathbf{T_2}$
3	$\mathbf{L_1} \cup \mathbf{L_2}$	$\mathbf{T_1} \cup \mathbf{T_2}$
4	$\mathbf{L_1} \cup \mathbf{L}_{\text{frontal}}$	$\mathbf{T_1} \cup \mathbf{T}_{\text{frontal}}$
5	$\mathbf{L_1}$	$\mathbf{T_1} \cup \mathbf{T}_{\text{frontal}}$

Tabelle 4.4: Die im letzten Abschnitt eingeführten Mengenpaare in der Übersicht.

Für das Training der Neuronalen Netze müssen die Trainingsmengen noch aufbereitet werden. Die Vektoren müssen zu Ein-/Ausgabepaaren (Mustern) der Form (Gesichtsvektor, Gesichtsvektor im nächsten Bild) angeordnet werden. Von jedem Gesichtsvektor $\boldsymbol{g}^{\boldsymbol{S},\boldsymbol{m}}$ der Trainingsmenge aus der Sequenz S an der Stelle m wird folgendermaßen ein Muster zusammengestellt:

$$(\boldsymbol{g}^{\boldsymbol{S},\boldsymbol{m}},\, p(\boldsymbol{g}^{\boldsymbol{S},\boldsymbol{m}}, 1)\,) \tag{4.12}$$

Die Funktion $p()$ liefert für einen gegebenen Vektor einen Nachfolger (hier den nächsten) aus derselben Sequenz oder den letzten Vektor der Sequenz (s. Gleichung (4.4)).

Mit diesen Mustern wird für jede Trainingsmenge ein Netzwerk trainiert und anschließend mit Vektoren aus der Testmenge verifiziert.

4.2.1 Das Growing Neural Gas Verfahren

Das Growing Neural Gas (GNG) Verfahren gehört zu der Klasse der Verfahren mit weichem Wettbewerbslernen (*soft competitive learning* oder auch *winner-take-most learning*) ohne feste Netzwerkgröße (Fritzke, 1994, 1995). Dieses Modell unterscheidet sich von anderen Neuronalen Netzen dadurch, daß sich die Anzahl der Neuronen während der Ausführung erhöht. Es startet mit zwei Neuronen zu denen nach und nach weitere hinzugefügt werden. Diese neuen Neuronen werden in die Nähe von anderen eingefügt, die den größten aufsummierten Fehler besitzen (vgl. auch LBG-U, Kapitel 3).

Das GNG Verfahren wird eingesetzt, um für ein Radial Basis Function (RBF) Netzwerk (Moody und Darken, 1988) die Position, Parameter und Anzahl der Neuronen im Eingaberaum zu bestimmen. Diese Neuronen sind über gewichtete Verbindungen mit jedem Neuron der Ausgangsschicht verbunden. Da nur eine Schicht mit gewichteten Verbindungen existiert, kann die Delta-Regel (Widrow und Hoff, 1960) eingesetzt werden, um die Gewichte der Verbindungen zu berechnen.

Es wurden vier Netzwerke trainiert, da die Mengenpaare 1 und 5 dieselbe Trainingsmenge besitzen. Die für das GNG Verfahren erforderlichen Parame-

Menge	Neuronen	Iterationen
1, 5	54	24700
2	100	47700
3	98	45500
4	100	46300

Tabelle 4.5: Die Charakteristika für die fünf mit dem GNG Verfahren trainierten Netze. Die Trainingsmengen 1 und 5 sind gleich, daher wurde dasselbe Netzwerk verwendet.

ter wurden mit folgenden Werten belegt: $\lambda = 200$, $\epsilon_b = 0,05$, $\epsilon_n = 0,0006$, $\alpha = 0,5$, $\beta = 0,0005$ und $a_{\max} = 88$. Das Sigma der RBF Einheiten wurde lokal[1] berechnet. Die trainierten Netzwerke haben die in der Tabelle 4.5 aufgeführten Charakteristika.

4.2.2 Das Multilayer Perzeptron

Ein in der Literatur häufig anzutreffendes Neuronales Netzwerk ist das Multilayer Perzeptron (MLP). Es ist ein vorwärtsgekoppeltes (*feed-forward*) Netzwerk mit einer Eingabe-, einer Ausgabe- und mindestens einer versteckten Schicht. Die Ausgänge einer Schicht sind mit den Eingängen der darauf folgenden Schicht vollständig vernetzt. Das Lernen dieses Netzwerks erfolgt durch den bekannten Backpropagation Algorithmus (Rummelhart und McClelland, 1986). Für alle Lernaufgaben wurden MLPs mit einer versteckten Schicht verwendet, die Lernrate war $\eta = 0,01$ und das Momentum $\alpha = 0,9$. Als Aktivierungsfunktion wurde die Sigmoidfunktion $1/(1 + \exp(-x))$ verwendet.

Die Eingabedaten mußten vor dem Lernen des Multilayer Perzeptrons noch auf den Bereich $[0,1]$ skaliert werden. Dies geschah durch Division aller Werte durch 512. Die Charakteristika der trainierten Netze sind in Tabelle 4.6 aufgelistet. Für jede Trainingsmenge wurden mehrere MLPs trainiert und anschließend das beste Netz ausgewählt.

Menge	MLP	Iterationen
1, 5	4-50-4	454800
2	4-96-4	1036400
3	4-94-4	1196000
4	4-96-4	872400

Tabelle 4.6: Die Charakteristika für die fünf mit dem MLP trainierten Netze. Die Trainingsmengen 1 und 5 sind gleich, daher wurde dasselbe Netzwerk verwendet.

4.2.3 Die Ergebnisse der vorgestellten Verfahren

Das Training der Netze für das Growing Neural Gas erfolgte problemlos, die Daten wurden schnell gelernt. Im Gegensatz dazu war das Training der

[1] Das Sigma berechnet sich aus dem durchschnittlichen Abstand zu den nächsten Nachbarn bezüglich der induzierten Delaunay-Triangulation (Martinetz, 1993) geteilt durch 2.

Growing Neural Gas

Mengenpaar	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1	166,03	7,26	1197,78
2	82,12	3,22	680,50
3	84,26	3,51	596,60
4	98,45	3,58	819,45
5	481,09	7,56	4810,84
Summe:	911,94	25,12	8105,15

Multilayer Perzeptron

Mengenpaar	$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
1	2221,09	236,27	5642,25
2	3167,39	1003,40	5901,15
3	2213,51	487,49	5341,50
4	1829,83	305,22	5697,15
5	2349,44	307,71	5875,35
Summe:	11781,30	2340,09	28457,40

Tabelle 4.7: Die Ergebnisse aller fünf Testläufe für die Neuronalen Netze.

MLPs sehr aufwendig. Bevor die Daten dem MLP präsentiert werden konnten, mußten sie noch skaliert werden. Während des Trainings oszillierte der Ausgabefehler und näherte sich nur langsam seinem Minimalwert. Dadurch benötigte das MLP auch ungefähr 20 mal so viele Iterationen wie das GNG.

Trotz dieser erheblich längeren Trainingsphase sind die Ergebnisse des MLPs um ein Vielfaches schlechter als die des GNG. Das mag einerseits an den Daten liegen, andererseits auch an den nicht optimal eingestellten Parametern (Lernrate η und Momentum α).

Allerdings wurden verschiedene Parameter ausprobiert mit ähnlichen Ergebnissen. Es ist unwahrscheinlich, daß ein anderer Parametersatz die Resultate des MLPs so stark verbessert, daß damit so gute Ergebnisse wie mit dem GNG erzielt werden.

Die Ergebnisse der Testläufe für das Growing Neural Gas bzw. das Multilayer Perzeptron sind in der Tabelle 4.7 dargestellt.

Die Abbildungen 4.7 und 4.8 zeigen, wie die Abbildung 4.5, drei reale Sequenzen mit den dazugehörigen Voraussagen für jedes Verfahren. In allen drei Abbildungen wurden zu Vergleichszwecken dieselben Sequenzen verwendet.

Growing Neural Gas

Realität Vorhersage

Abbildung 4.7: Drei Ergebnisse des GNG Verfahrens. Links zu sehen ist die reale Kopfbewegung der Person, rechts daneben die vorhergesagte. Von oben nach unten vergrößert sich der Fehler: $MSE_{\text{oben}} = 77,09$, $MSE_{\text{Mitte}} = 163,06$ und $MSE_{\text{unten}} = 4190,1$.

Multilayer Perzeptron

Realität Vorhersage

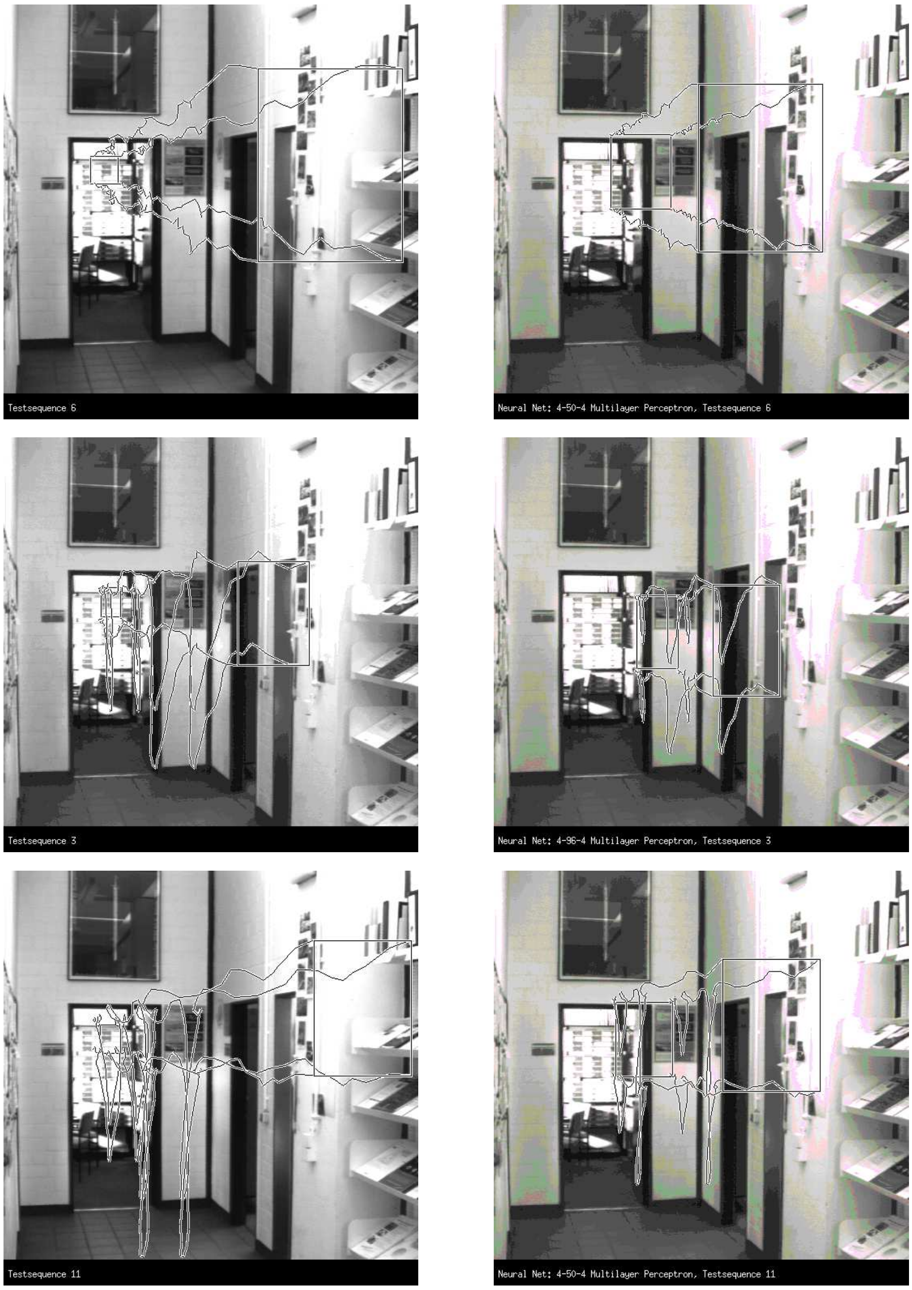


Abbildung 4.8: Drei Ergebnisse des MLP. Links zu sehen ist die reale Kopfbewegung der Person, rechts daneben die vorhergesagte. Von oben nach unten vergrößert sich der Fehler: $MSE_{\text{oben}} = 4391,39$, $MSE_{\text{Mitte}} = 6433,88$ und $MSE_{\text{unten}} = 6516,71$.

4.3 Der Vergleich der Neuronalen Netze mit dem k–Nearest-Neighbor Verfahren

Zum besseren Vergleich sind alle Ergebnisse der drei verschiedenen Verfahren für alle Mengenpaare in der Tabelle 4.8 noch einmal aufgeführt.

Mengenpaar		$\varnothing MSE$	$\varnothing MSE_{min}$	$\varnothing MSE_{max}$
	k–NN	136,54	3,14	1275,21
1	GNG	166,03	7,26	1197,78
	MLP	2221,09	236,27	5642,25
	k–NN	76,35	2,59	542,73
2	GNG	82,12	3,22	680,50
	MLP	3167,39	1003,40	5901,15
	k–NN	83,87	2,13	814,02
3	GNG	84,26	3,51	596,60
	MLP	2213,51	487,49	5341,50
	k–NN	98,46	1,84	946,14
4	GNG	98,45	3,58	819,45
	MLP	1829,83	305,22	5697,15
	k–NN	376,54	4,37	3677,44
5	GNG	481,09	7,56	4810,84
	MLP	2349,44	307,71	5875,35
	k–NN	771,87	14,02	7234,75
Summe:	GNG	911,94	25,12	8105,15
	MLP	11781,30	2340,09	28457,40

Tabelle 4.8: Der Vergleich aller Verfahren miteinander für jedes einzelne Mengenpaar.

Das Multilayer Perzeptron liefert die schlechtesten Resultate aller getesteten Verfahren. Allerdings ist aus dem grafischen Vergleich der Verfahren in Abbildung 4.9 zu erkennen, daß die Daten im Prinzip richtig gelernt worden sind. Die Vorhersagen sind aber viel zu ungenau, so daß das MLP nicht weiter betrachtet wird. Immerhin ist das MLP das einzige Verfahren, das annähernd die richtigen Vorhersagen für den Test mit der ungelernten Sequenz trifft (Abbildung 4.8 und 4.9 unten).

Im Vergleich mit dem k–Nearest-Neighbor Verfahren sind die Ergebnisse des Growing Neural Gas Verfahrens ähnlich gut. Tests mit einem realen Gesichtsfinder haben ergeben, daß das GNG eine Alternative zum k-NN ist. Nur beim Mengenpaar fünf gibt es eine deutliche Abweichung. Ein Blick auf

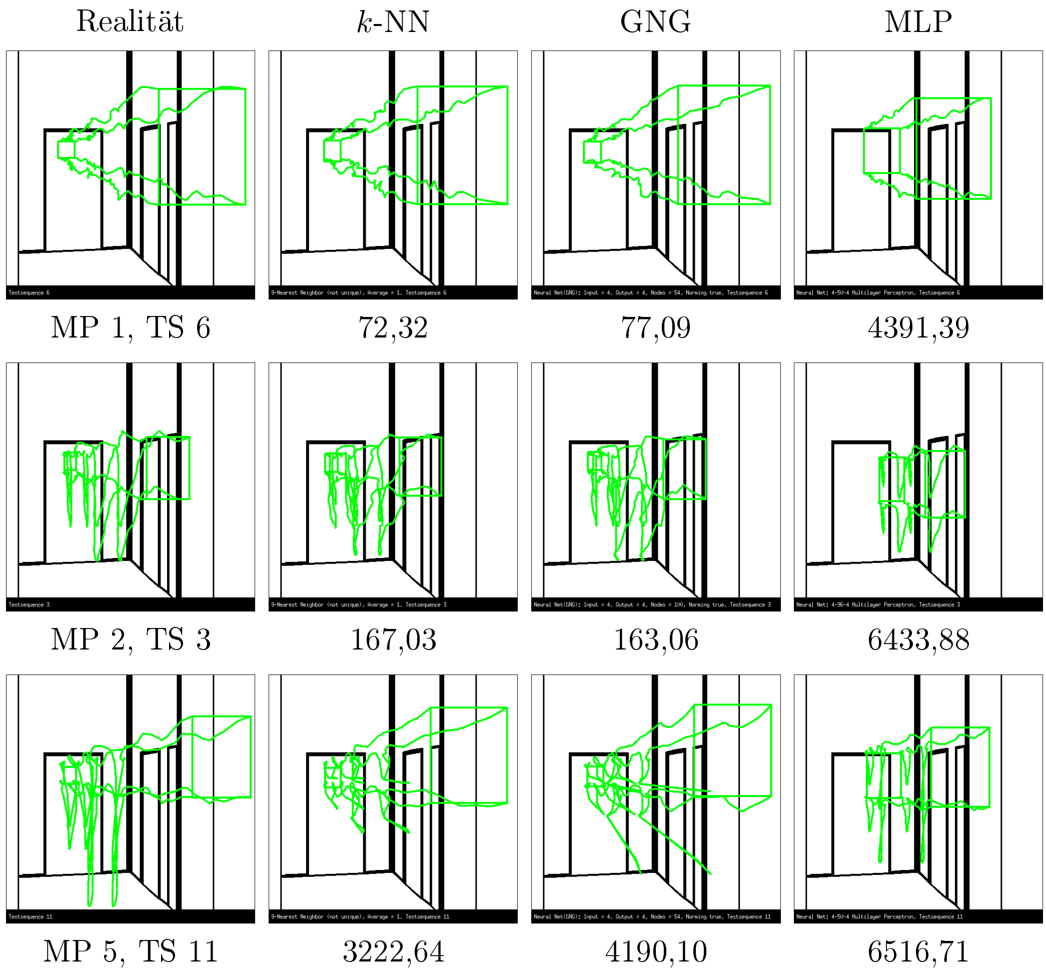


Abbildung 4.9: Drei Ergebnisse der unterschiedlichen Verfahren im grafischen Vergleich für jeweils eine Sequenz: 1. Realität, 2. k-NN, 3. GNG, 4. MLP. Von oben nach unten nimmt der Fehler zu, dabei ist oben eine der besten, in der Mitte eine interessante durchschnittliche und unten die schlechteste Vorhersage ausgewählt (bzgl. k-NN und GNG; für das MLP lagen die Fehler im Bereich $[1868, 8839]$). Unter den Vorhersagen ist der durchschnittliche Fehler für die abgebildete Sequenz aufgeführt, unter der realen Sequenz das Mengenpaar (MP) und die verwendete Testsequenz (TS).

---

die Abbildung 4.9 (unten) offenbart das Problem: der Test mit den ungelernten Sequenzen verursacht das schlechtere Resultat. Offenbar ist das Verhalten des GNG bei neuen, nicht gelernten Daten abhängig von dem gelernten Netz. Wird ein anderes Netz mit denselben Daten trainiert, können die Ergebnisse bei den ungelernten Sequenzen wieder ganz anders aussehen.

Beim k-NN ist das Verhalten bei neuen Daten genau festgelegt: sie werden immer auf ihre nächsten gelernten Daten abgebildet. Das Lernen neuer Sequenzen gestaltet sich auch denkbar einfach, da diese nur zu den vorhandenen hinzugefügt werden. Das GNG muß ein neues Netz mit den ungelernten und den vorhandenen Sequenzen ganz neu trainieren.

Aus diesen Gründen, und da das k–Nearest-Neighbor Verfahren in der Summe die etwas besseren Resultate liefert, wurde das Growing Neural Gas Verfahren nicht weiter untersucht. Stattdessen wurde das k–NN verwendet und zusammen mit einem realen Gesichtsfinder eingesetzt.

Der Vorteil des schnellen Lernens beim k-NN Verfahren wird durch die aufwendigere Berechnung der Ausgabewerte relativiert. Für jeden Eingabewert muß der Abstand zu jedem Wert in der Trainingsmenge berechnet werden. Anschließend müssen die k nächsten Nachbarn gefunden werden. Dieser Prozeß kann je nach Umfang der Trainingsmenge und Dimensionalität der verwendeten Daten erhebliche Rechenzeit benötigen.

Bently (1975) und Friedman et al. (1975) beschreiben eine Methode zur Markierung der einzelnen Trainingsdaten, so daß die nächsten Nachbarn schneller gefunden werden. Diese kd–Baum genannte Markierungsmethode speichert die Trainingsdaten in den Blättern eines Baums. Ähnliche Trainingsdaten werden dabei so nebeneinander plaziert, daß die nächsten Nachbarn bei einer Anfrage schnell gefunden werden können.

Kapitel 5

Die Beschreibung des eingesetzten Systems

Die in den letzten Kapiteln vorgestellten Ergebnisse wurden ermittelt durch das Zusammenspiel verschiedener Programme. Ein Programm wurde entwickelt, um die vorgestellten Verfahren zu realisieren, ein anderes zum Testen dieser Verfahren.

Zum ersten Mal werden in diesem Kapitel die Verfahren im Zusammenhang mit einem realen Gesichtsfinder eingesetzt und müssen ihre Tauglichkeit mit einer konkreten Applikation beweisen. Dabei tauchten viele kleinere Probleme und Schwierigkeiten auf, die gelöst werden mußten.

Zum Abschluß entstand ein System, das nicht nur die Suche in großformatigen Bildsequenzen beschleunigt, sondern auch die Erkennungsrate um ein Vielfaches erhöht.

5.1 Die Komponenten

Im Rahmen der Diplomarbeit entstanden die Programme `PredictIt` und `FinderTestIt`. `PredictIt` realisiert die in den Kapiteln 3 und 4 behandelten Verfahren. Zum Testen der Verfahren wurde das Programm `FinderTestIt` entwickelt; dieses Programm dient als Ersatz für einen optimalen Gesichtsfinder. Als realer Gesichtsfinder kam `BananaPeal` zum Einsatz.

Alle hier vorgestellten Programme sind in `C++` geschrieben. Sie können mit Hilfe einfacher Befehle im Batch-Betrieb gesteuert werden, oder komfortabel mit einer in Tcl/Tk geschriebenen Oberfläche.

Die Abbildung 5.1 gibt einen Überblick über das System `PredictIt`/Gesichtsfinder. Durch die sehr allgemein gehaltene Schnittstelle kann `PredictIt` einfach mit einem beliebigen Gesichtsfinder kommunizieren. Damit das Sys-

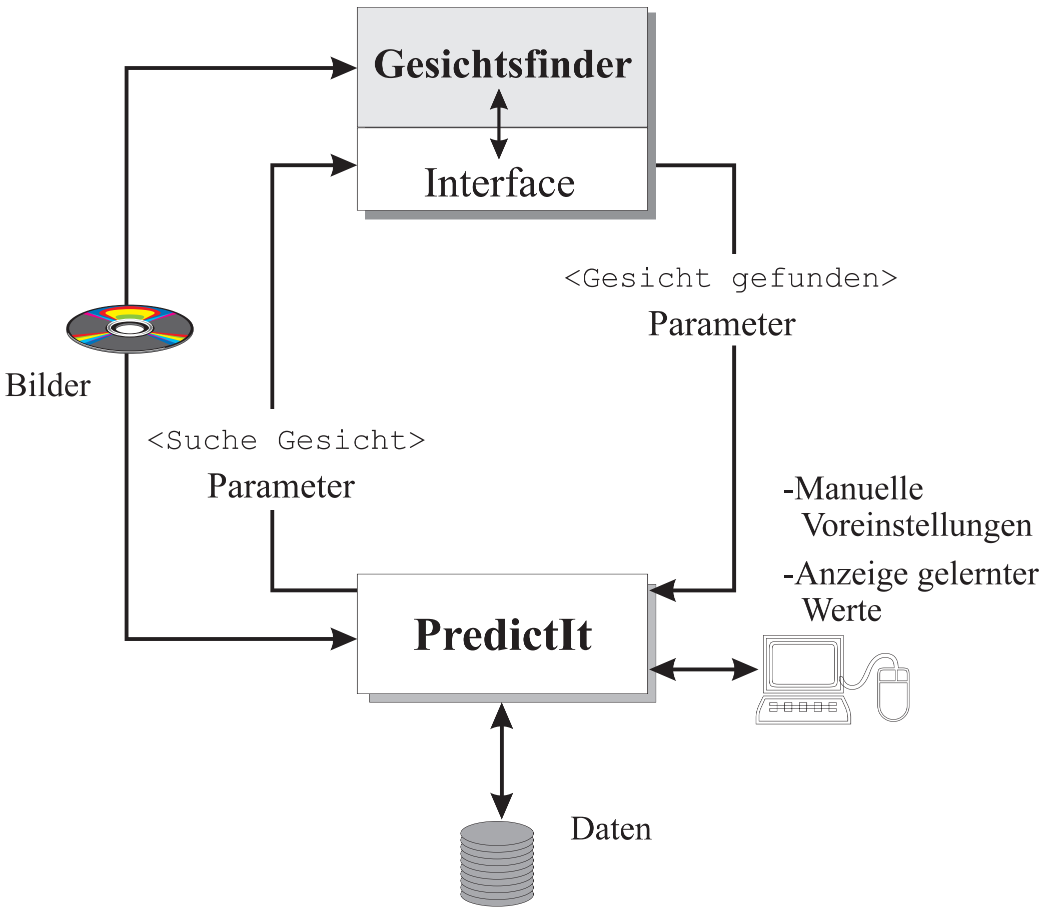


Abbildung 5.1: Der schematische Aufbau und die Funktionsweise des Systems `PredictIt`/Gesichtsfinder.

---

tem funktioniert, muß der Gesichtsfinder einige notwendige Anforderungen erfüllen:

- ereignisgesteuert: der Gesichtsfinder sollte auf äußere Signal reagieren können.
- Funktionen für
  - Bildausschnitte verarbeiten: der Gesichtsfinder soll nur in einem bestimmten Bereich des gesamten Bildes suchen.
  - nächstes Bild einlesen: je nach Strategie bestimmt `PredictIt` den Zeitpunkt für das nächste Bild.
- eine genügend hohe Erfolgsquote besitzen und

- eine zuverlässige Konfidenz aufweisen für
  - *Gesicht gefunden* und
  - *kein Gesicht gefunden.*

Anhand der Konfidenz beurteilt der Gesichtsfinder selber, ob der ermittelte Bereich ein Gesicht darstellt oder nicht. Es wäre zudem vorteilhaft, wenn der eingesetzte Gesichtsfinder auch auf großen Bildern Gesichter zuverlässig finden könnte, da die in `PredictIt` implementierten Verfahren Daten zum Lernen benötigen. Ansonsten gibt es aber auch die Möglichkeit, die Gesichter mit der Hand zu markieren.

Erfüllt der Gesichtsfinder die an ihn gestellten notwendigen Anforderungen, muß nur noch die Schnittstelle eingebettet werden.

5.1.1 Das Programm `PredictIt`

Das Programm `PredictIt` kombiniert verschiedene Funktionen unter einer Oberfläche. Diese Funktionen lassen sich in verschiedene Gruppen einteilen:

- Interaktion mit dem Gesichtsfinder,
- Einstellung der Parameter für die implementierten Verfahren,
- Berechnung von Ergebnisse im *stand-alone* Betrieb,
- Eingabe und Verarbeitung von Daten.

Zu der Interaktion mit dem Gesichtsfinder gehört die manuelle Rückgabe von Bildausschnitten. Neben den berechneten Werten der verschiedenen Verfahren kann der Benutzer auch selbst in die Kommunikation eingreifen. Außerdem hat der Benutzer die Möglichkeit, ein Standardverfahren für die Bestimmung der Startposition und zur Vorhersage der nächsten Position zu definieren. Während der Laufzeit kann er trotzdem noch andere Verfahren auswählen.

Die Parameter für die implementierten Verfahren sind nicht fest im Programm verankert, sondern können vor und nach dem Start geändert werden.

Im *stand-alone* Betrieb werden die Ergebnisse der einzelnen Verfahren ohne einen Gesichtsfinder ermittelt. Zur Auswahl stehen k–Nearest-Neighbor, Growing Neural Gas und das Multilayer Perzeptron. Für jedes Verfahren können die Test- und Trainingsmengen und die zu untersuchenden Parameterintervalle eingestellt werden. Die Ergebnisse dieser Versuche können grafisch und textuell angezeigt und/oder abgespeichert werden. Der Benutzer hat bei der grafischen Ausgabe die Wahl zwischen verschiedenen Anzeigearten.

Weiterhin ist die Anzeige der Ergebnisse vergangener Versuche möglich. Generell ist die Generierung statistischer Informationen in verschiedenen Formaten implementiert, so daß die weitere Auswertung mit anderen Programmen vorgenommen werden kann. Die Eingabe und Bearbeitung von Gesichtspositionen für Bilder ist komfortabel per Maus durchführbar und erlaubt dem Benutzer, die Lerndaten für die Verfahren selbst zu generieren oder berechnete Daten zu korrigieren.

5.1.2 Das Programm `FinderTestIt`

Zum einfachen Testen des Systems wurde der *Ground-Truth*–Gesichtsfinder `FinderTestIt` implementiert. Dieser liefert zu einem von `PredictIt` angegebenen Bildausschnitt, ob ein Gesicht gefunden wurde oder nicht (inkl. der dazugehörigen Daten).

Im Gegensatz zu einem realen Gesichtsfinder, greift `FinderTestIt` zur Bestimmung der Gesichter auf eine Datenbasis zurück. In dieser Datenbasis sind zu jedem Bild die entsprechenden Positionen der Gesichter gespeichert. `FinderTestIt` ist daher in der Lage, einen optimalen Gesichtsfinder zu simulieren und erlaubt so, die eingesetzten Verfahren zu testen, ohne daß die Besonderheiten eines realen Gesichtsfinders in die Bewertung der Ergebnisse mit einfließen.

Ein weiterer Vorteil ist die sehr kurze Antwortzeit von `FinderTestIt`; der verwendete reale Gesichtsfinder benötigt je nach Güte der Suche zwischen 10 und 30 Sekunden, um in einem Bild oder Bildausschnitt ein Gesicht zu finden.

Ferner setzt `FinderTestIt` auf dieselbe Schnittstelle auf wie ein realer Gesichtsfinder. Das hat einerseits den Vorteil, daß er einfach durch einen realen Gesichtsfinder ersetzt werden kann, und andererseits wird die Schnittstelle zu `PredictIt` getestet.

5.1.3 Der Gesichtsfinder `BananaPeal`

Der Gesichtsfinder `BananaPeal` wurde am Institut für Neuroinformatik in Bochum entwickelt und war nach einigen kleineren Erweiterungen (Einbettung der Schnittstelle, Bereitstellung der benötigten Funktionalität) einsetzbar. Er erfüllt zu einem großen Teil die Anforderungen, die als Voraussetzungen für ein funktionierendes System aus `PredictIt`/Gesichtsfinder aufgestellt worden sind.

`BananaPeal` beruht auf dem Konzept von generalisierten Gaborwavelets, die Krüger und Peters (1997) wegen ihrer Krümmung Bananawavelets ge-

Abbildung 5.2: i–iv) Verschiedene Gesichtsbeispiel, v) die gelernte Repräsentation. Die Abbildung stammt aus Krüger und Peters (1997).

tauft haben. Es ist allgemein zur Suche von Objekten in komplexen Szenen entworfen worden und wird im Rahmen der Diplomarbeit als Gesichtsfinder eingesetzt.

Zuerst müssen für die zu findenden Objekte entsprechende Repräsentationen gelernt werden. Diese Repräsentationen enthalten nur die wichtigsten Merkmale der gelernten Objekte. Die Merkmale werden an Hand von Beispielen gelernt (s. Abbildung 5.2).

Aus jedem Beispiel werden die markanten Merkmale extrahiert. Anschließend wird aus der Summe aller dieser Merkmale die Repräsentation gebildet, indem die unwichtigen Merkmale herausgefiltert werden.

Die so gewonnene Repräsentation wird dann verwendet, um mittels elastischem Graphmatching (Lades et al., 1993) ein Objekt in einer Szene zu finden. Ein Beispiel mit drei unterschiedlich großen Repräsentationen ist in Abbildung 5.3 zu sehen.

Kann das Objekt in unterschiedlichen Größen in der zu durchsuchenden

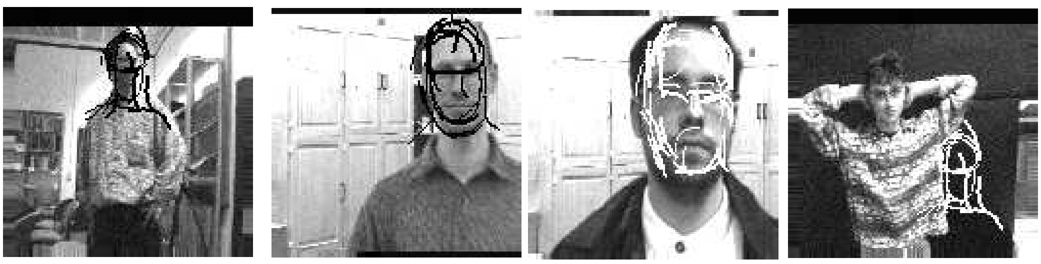


Abbildung 5.3: Die Gesichtsfindung mit drei unterschiedlich großen, automatisch gelernten Repräsentationen. Das Gesicht im letzten Bild wurde durch die unübliche Armposition nicht gefunden. Die Abbildung stammt aus Krüger und Peters (1997).

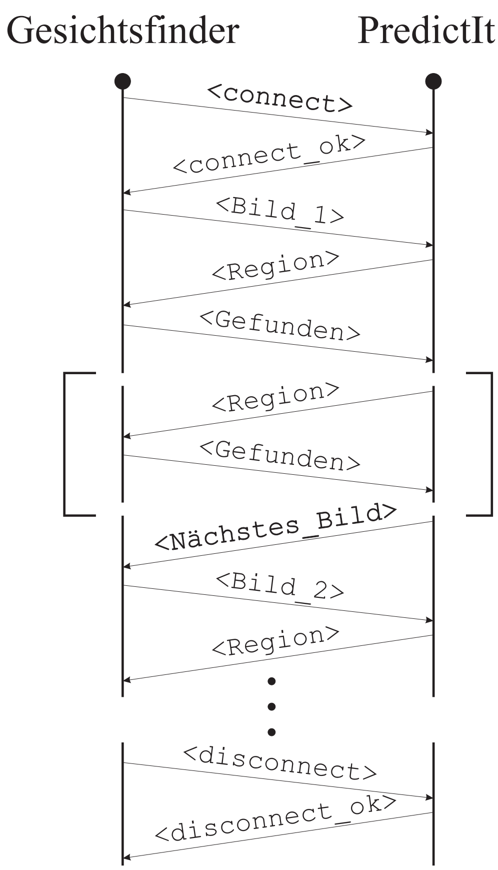


Abbildung 5.4: Das Kommunikationsprotokoll zwischen dem Gesichtsfinder und `PredictIt`.

Szene erscheinen, ist es sinnvoll, entsprechende Repräsentationen zu lernen. Jede einzelne Repräsentation wird auf dem Bild gematcht. Die Repräsentation mit der größten Ähnlichkeit bestimmt das Objekt.

5.1.4 Die Kommunikation der Komponenten

Um den Datenaustausch zwischen den Programmen zu ermöglichen, wurde ein einfaches Protokoll entworfen, das die Kommunikation regelt (s. Abbildung 5.4).

Nach dem Verbindungsaufbau zwischen den beiden Programmen (`connect`) sendet der Gesichtsfinder eine Bildidentifikation[1] zu `PredictIt`.

[1]Die Bildidentifikation besteht aus dem Dateinamen des Bildes mit kompletter Pfadangabe. Die in Bearbeitung befindlichen Bilder konnten durch diese Information in `Pre-`

Daraufhin schickt `PredictIt` einen Bildausschnitt zurück und empfängt im Gegenzug wo und mit welcher Konfidenz ein Gesicht in dem Ausschnitt gefunden wurde. Sind gewisse Voraussetzungen erfüllt, fordert `PredictIt` das nächste Bild. Dieser Vorgang wiederholt sich so lange, bis keine Bilder mehr vorhanden sind oder ein anderer Grund zur Beendigung vorliegt. Zum Abschluß sendet der Gesichtsfinder ein Kommunikationsende (`disconnect`).

Da nahezu alle Programme, die in letzter Zeit am Institut für Neuroinformatik erstellt wurden, mit einer standardisierten Oberfläche auf Tcl/Tk-Basis arbeiten, bot es sich an, den Datenaustausch zwischen dem Gesichtsfinder und `PredictIt` durch einen in Tcl/Tk implementierten Mechanismus (`send`) durchzuführen. Dadurch ist es möglich, die beiden Programme parallel auf verschiedenen Rechnern auszuführen und trotzdem eine einfach zu bedienende Schnittstelle zu nutzen.

5.2 Das Zusammenspiel von `PredictIt` mit einem realen Gesichtsfinder

Die Arbeitsweise von `BananaPeal` ist folgendermaßen: es sucht in einem Bild nach einem Gesicht und liefert auf jeden Fall eine Gesichtsposition zurück. Diese Information ist mit einem Wert versehen, der Konfidenz. Die Konfidenz ist aus dem Intervall $[0, 1]$. Je näher der Wert an 1 liegt, desto ähnlicher war die gefundene Gesichtsposition einer verwendeten Repräsentation. Ein Gesicht wird als richtig erkannt eingestuft, wenn die Position stimmt und die Konfidenz hoch genug ist.

5.2.1 Die Schwierigkeiten

Die Daten der Trainingsmengen wurden nicht durch `BananaPeal` erzeugt, sondern mit der Hand markiert. Dieses Vorgehen erzeugt Abweichungen, da der Gesichtsfinder dasselbe Gesicht unterschiedlich in der Szene markiert. Bei ersten Tests wurden allerdings keine Nachteile dadurch festgestellt; sonst wäre eine Konvertierung notwendig geworden.

Damit ein Gesicht überhaupt von `BananaPeal` gefunden werden kann, muß es sich einerseits komplett in dem von `PredictIt` vorhergesagten Ausschnitt befinden, andererseits darf es nicht zu nah am Rand sein. Daher wurde die Fläche des Ausschnitts vervierfacht, indem die Breite und Höhe jeweils verdoppelt wurden; der Mittelpunkt wurde nicht verändert.

`dictIt` angezeigt werden. Dies ist für die Vorhersage nicht nötig, war aber sehr hilfreich bei der Interpretation der Ergebnisse.