UNIVERSITÄT DORTMUND

PROJEKTGRUPPE 492 ENDBERICHT

Videobasierte Detektion und Identifikation von Personen (ViPer)

PG Teilnehmer:
Colin Doert, Kai Glittenberg, Rolf Harren, Marius Hennecke,
Christian Kleine-Cosack, Nicolas Luck, Fabian Nasse, Tom Paschenda,
Bastian Pfleging, Tobias Ramforth, Jiasong Shi, Kui Zhang

Betreuer:
Prof. Dr.-Ing. Gernot Fink
Dr.-Ing. Thomas Plötz
Dipl.-Ing. Jan Richarz

Datum:
30. März 2007

Inhaltsverzeichnis

1 Einleitung

Die Interaktion zwischen Mensch und Maschine im nicht-technischen Alltag bedarf meist eines Eingabegerätes, in dessen Benutzung der Mensch zunächst eingewiesen werden muss. Verfahren zur Interaktion orientieren sich jedoch in der Regel an den Bedürfnissen der Maschine, nicht an denen des Menschen. Der Mensch lernt, mit der Maschine zu interagieren und nicht andersherum.

Ein Ziel der Entwicklung intelligenter Interaktionssysteme ist es, dem Menschen Restriktionen zu ersparen und stattdessen die Maschine mit der Fähigkeit auszustatten, die Modalitäten des Menschen zu verstehen. Damit eine derartige Interaktion möglich ist, muss das intelligente System über Sensorik verfügen, die es ihm erlaubt, die für die Interaktion erforderlichen Informationen zu gewinnen. Des Weiteren wird Software benötigt, um diese Informationen entsprechend verarbeiten zu können. Eine Anforderung an ein solches System besteht darin, die Anwesenheit eines Menschen und dessen Identität feststellen zu können.

Eine Möglichkeit bestünde darin, die interagierende Person mit künstlichen Merkmalen wie beispielsweise RFID-Tags auszustatten. Unser Ziel ist es jedoch, die Interaktion für den Menschen so restriktionslos wie möglich zu gestalten. Hier bietet sich die Aufnahme und Interpretation von Bildinformationen aus der Umgebung an. So lässt sich feststellen, wann Personen anwesend sind und wo sich diese befinden. Die Identität kann durch die Erfassung des Gesichts per Kamera festgestellt werden.

Das Szenario, in dem die Interaktion stattfindet, ist in diesem Projekt ein intelligenter Raum. Dieser soll in der Lage sein, mit den in ihm befindlichen Personen zu interagieren, ohne dass sich die Personen technischer Hilfsmittel bedienen müssen.

Dieses Kapitel beschreibt zunächst die Ziele, die das im Rahmen der Projektgruppe entwickelte ViPer-System erreichen soll - dabei steht ViPer für die videobasierte Detektion und Identifikation von Personen. Im Anschluss werden ähnliche Projekte aufgezeigt und beschrieben. Im dritten Abschnitt wird ein Überblick über das System gegeben.

1.1 Problemstellung

Das Ziel dieser Projektgruppe ist ein System, das auf der Basis von Videodaten in der Lage ist, die Anwesenheit von Personen und deren Identität festzustellen, sowie ihre Bewegungen im Raum zu verfolgen, um ihre Position zu jedem Zeitpunkt zu kennen. Ein solches System schafft die Grundlage für vielseitige Anwendungen, wie zum Beispiel eine automatische Begrüßung oder eine Anwesenheitsprotokollierung.

Das Projektziel setzt sich aus vier Teilaufgaben zusammen

1. Die anwesenden Personen sollen detektiert werden. (*Detektion*)
2. Die Bewegung der detektierten Personen soll verfolgt werden. Die Verfolgung von Objekten in Bildern wird auch als *Tracking* bezeichnet.
3. Die Identität der Personen soll durch bildbasierte *Identifikation* festgestellt werden.
4. Die Informationen von Detektion, Tracking und Identifikation sollen zusammengeführt und koordiniert werden.

Die Aufnahme und Verarbeitung von Bildinformationen ist mit Unsicherheit und Fehlern behaftet. Während das Ziel der Teilsysteme in der Fehlerminimierung besteht, soll das Gesamtsystem die Interpretation so gestalten, dass solche Aufnahme- und Verarbeitungsfehler zu keinem Fehler bezüglich des Gesamtverhaltens führen. Dies ist eine zentrale Herausforderung bei der Integration der Teilsysteme.

Jede der Teilaufgaben wird von einem eigenen Modul innerhalb von ViPer durchgeführt. Abschnitt 1.3 gibt einen Überblick über die Module und deren Zusammenspiel.

1.2 Verwandte Arbeiten

Intelligente Mensch-Maschine-Interaktionssysteme sind Gegenstand zahlreicher weltweiter Forschungsprojekte.

Das Georgia Institute of Technology entwickelt mit dem Aware Home[1] ein intelligentes Haus, das die Anwesenheit der in ihm wohnenden Menschen bemerkt und deren derzeitige Tätigkeit erkennt, um sie zu unterstützen. Bei dem dort durchgeführten »Aging in Place« Projekt liegt der Schwerpunkt auf Hilfe für Senioren, beispielsweise durch das Finden gesuchter Objekte oder die Überwachung des Gesundheitszustandes.

Die Steuerung des intelligenten Systems geschieht über eine Gestenerkennung. Zur Detektion, zum Tracking und zur Identifikation sind mehrere Mechanismen vorgesehen. Darunter sind RFID Chips, Verarbeitung von Bildinformationen von Kameras, sowie die Identifikation und das Tracking durch Analyse des Schrittmusters [OJAD00].

Das CHIL Projekt[2] versucht ebenfalls, intelligente Systeme zu entwickeln, die mit ihnen interagierende Person beobachten und durch die Interpretation von Sensorinformationen verstehen können. Auf Grundlage eines solchen Systems sind dann Dienste vorgesehen, die eine intelligente Umgebung erbringen kann. Geplant ist unter anderem ein Protokollierungsdienst, der den Verlauf einer Konferenz speichern soll sowie ein System, das dem Vortragenden Informationen über das Aufmerksamkeitsniveau seiner Zuhörer geben soll. Zahlreiche Technologien werden im Rahmen des Projekts entwickelt. Darunter ein Verfahren zur Identifikation anhand von Audio- und Videoinformationen [SPP05], sowie ein 3D-Tracker, der ebenfalls Audio- und Videodaten kombiniert [KST+06].

1 http://www.awarehome.gatech.edu/projects/

2 http://chil.server.de

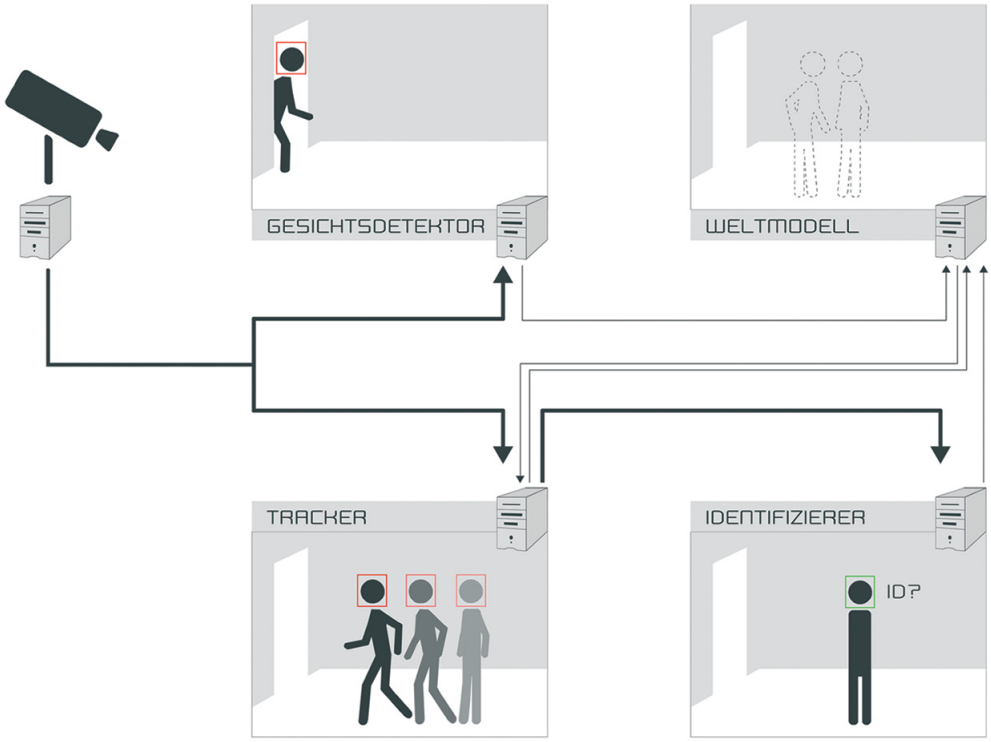


Abbildung 1.1: Der Aufbau des ViPer-Systems. Die Pfeile geben den Informationsfluss wieder. Dicke Pfeile transportieren Bilder, dünne Steuerinformationen. Die abgebildete Architektur erlaubt auch den Einsatz mehrerer Kameras: An eine zusätzliche Kamera würde ein weiterer Detektor mit einem weiteren Tracker angeschlossen werden.

In den Kapiteln zur Detektion, zum Tracking und zur Identifikation finden sich jeweils zahlreiche Hinweise auf verwandte Arbeiten zu den jeweiligen Themen.

1.3 Überblick über das ViPer-System

Im Folgenden wird die Struktur des ViPer-Systems beschrieben. Dabei wird auf die Schnittstellen der einzelnen Module und deren Einordnung in das Gesamtsystem eingegangen. Für eine genaue Beschreibung der theoretischen Grundlagen und Implementierung der Module sei auf die entsprechenden Kapitel verwiesen.

Abbildung 1.1 zeigt die Module und den Informationsaustausch im System. Die Komponenten werden nun in der Reihenfolge des Informationsflusses beschrieben.

1.3.1 Kameramanager

Der Kameramanager versorgt das System mit Bildern. Dieses Modul verschafft sich Bilder von einer Quelle (zum Beispiel einer Kamera) und gibt die Bilder an den Tracker

und den Detektor weiter. Soll mehr als eine Kamera verwendet werden, so kann dem System ein weiterer Kameramananager mit angeschlossenem Detektor und Tracker hinzugefügt werden. Das Weltmodell muss die von den verschiedenen Kameras gewonnenen Informationen dann integrieren. Im Rahmen dieser Projektgruppe wird allerdings immer nur eine Kamera verwendet.

1.3.2 Gesichtsdetektor

Der Gesichtsdetektor erhält einzelne Bilder und sucht Bildausschnitte, die ein Gesicht enthalten. Eingabe ist also ein Bild und Ausgabe ist eine Liste von Rechtecken, die an das Weltmodell weitergeschickt wird.

1.3.3 Tracker

Der Tracker benötigt eine Folge von Bildern als Eingabe. Seine Aufgabe ist es, Objekte von Bild zu Bild zu verfolgen. Er wird vom Weltmodell auf den Koordinaten eines Bildausschnitts, der ein Gesicht enthält, initialisiert. Dieses Gesicht soll dann vom Tracker verfolgt werden. Die Ergebnisse der Verfolgung werden in Form von Rechteckkoordinaten an das Weltmodell weitergegeben.

1.3.4 Weltmodell

Das Weltmodell besteht aus einer logischen Repräsentation der Personen im Raum, sowie Algorithmen zur Aktualisierung dieser Repräsentation. Es erhält die Ergebnisse des Detektor-Moduls sowie die Nummer des Bildes, auf dem diese Ergebnisse ermittelt wurden. Jedes Rechteck, das der Gesichtsdetektor versendet, stellt ein Gesicht, also eine Person, dar. Das Weltmodell speichert die Position der Personen im Bild und weist den Tracker an, die vom Detektor erkannten Gesichter zu verfolgen.

Neben den Resultaten der Detektion erhält das Weltmodell auch die Resultate des Trackings. Es weiß also immer, welche Bildausschnitte getrackt werden, und welchen Objekten diese zugeordnet sind.

Darüber hinaus ist das Weltmodell dafür verantwortlich, die Identifikation der verfolgten Objekte anzustoßen. Das Ergebnis der Identifikation wird dann durch das Weltmodell dem entsprechenden Objekt zugeordnet.

Das Weltmodell muss davon ausgehen, dass die Ergebnisse der Detektion und des Trackings mit Fehlern behaftet sind und dies berücksichtigen. Abschnitt 5.1 beschreibt das Weltmodell im Detail.

1.3.5 Identifizierer

Der Identifizierer arbeitet auf einzelnen Bildern und stellt eine Beziehung zwischen Bild und einem Eintrag der Gesichtsdatenbank her. Eingabe ist ein Gesichtsbild und Ausgabe der Name einer Person aus der Datenbank. Dabei ist der Identifizierer auch in der Lage, Bildausschnitte, die sich nicht zur Identifikation eignen zu erkennen und zurückzuweisen.

2 Gesichtsdetektion

Den ersten Schritt auf dem Weg zur Identifikation von Personen in einem videobasierten System stellt die Detektion dieser Personen dar. In diesem Kapitel soll der theoretische Hintergrund der von uns gewählten Detektionsmethode näher betrachtet werden. Diese Methode hat sich in verschiedenen Anwendungen (vgl. z.B. [Pet06]) als effizient und zuverlässig erwiesen. Dabei gehen wir auf notwendige Basisalgorithmen sowie deren Einordnung in den Gesamtdetektionskontext ein.

2.1 Problembeschreibung

Um die Präsenz von Menschen in einem lokal beschränkten Bereich festzustellen, gibt es verschiedene Möglichkeiten, die alle eines technisch (meist elektronisch) detektierbaren *Tags* bedürfen. Damit stellt die Detektion keine große Herausforderung mehr dar und verlangt zudem von allen zu identifizierenden Personen das unbedingte Mitführen eines solchen Erkennungsmerkmals. Sollen Menschen unabhängig von solchen Hilfsmitteln detektiert werden, muss die Detektion auf Attribute des menschlichen Körpers (z. B. Gestalt, Größe) oder auf typisch menschliche Eigenheiten (z. B. Sprache, Bewegung) fokussiert werden. Das Gesicht ist das auffälligste menschliche Merkmal und dient hier als Ausgangspunkt, um die Präsenz von Menschen zu erkennen. Als Eingabe dienen Bilder auf denen Gesichter zu sehen sein können. Ziel ist es, alle im Bild vorkommenden Gesichter zu finden und dabei möglichst keine Nicht-Gesichter fälschlicherweise als Gesicht zu erkennen. Dabei kann man die Gesichtsdetektion auf Bildern als Lösung eines Zwei-Klassen-Problems auffassen. Jeder beliebige Bildausschnitt muss einer der beiden Klassen Gesicht oder Hintergrund zugeordnet werden. Hierbei werden die von der Kamera gelieferten Bilder mit Mustererkennungsverfahren verarbeitet, um Position und Ausmaße der Gesichter zu ermitteln. Wie von Yang et al. [YKA02] erwähnt, bestehen die wesentlichen Herausforderungen der Gesichtsdetektion aus:

1. Variation der Kopfposition relativ zur Kamera
2. Dem eventuellen Vorhandensein von strukturverändernden Komponenten wie Bärten oder Brillen
3. Variation des Gesichtsausdrucks, der die Gesamterscheinung des Gesichts verändert
4. Überdeckungen z. B. auf Bildern die verschiedene Personen zeigen, können sich Gesichter gegenseitig überdecken
5. Variation der Orientierung des Bildes um die Optische Achse (Rotation)
6. Änderungen der Abbildungseigenschaften z. B. Variationen in der Beleuchtung oder Sensorrauschen

Jede dieser Schwierigkeiten stellt sich auch in unserem Szenario dar.

2.2 Methodenübersicht

Verfahren zur Gesichtsdetektion finden, neben der Aufgabenstellung eines interaktiven Konferenzraums, in vielen Anwendungsbereichen Beachtung. So reicht die Spanne der Anwendungen von Überwachungs- und Zutrittssicherungssystemen bis zu, auf visueller Eingabe basierenden, interaktiven Anwendungen und Spielen. Hinzu kommt, dass der gegenwärtige Stand in der Hardwareentwicklung erlaubt, derartige Anwendungen mittlerweile in Echtzeit zu betreiben. Das große Interesse erklärt, warum Gesichtsdetektion vielfach Gegenstand aktueller Forschung ist und verschiedene Ansätze entwickelt wurden. Dem Übersichtspaper von Yang et al. [YKA02] entsprechend lassen sich die Verfahren grob in die nachfolgend beschriebenen Klassen unterteilen.

2.2.1 Wissensbasierte Methoden

Die grundlegende Idee der *wissensbasierten Methoden* ist die Annahme, dass Erkenntnisse über den Aufbau des menschlichen Gesichts in Form von einfachen Regeln, z. B. bezogen auf relative Abstände, Positionen und anderen Eigenschaften von Gesichtsmerkmalen, ausgedrückt werden können. Üblicherweise werden hierfür in einem *Top-down*-Ansatz zuerst einfache Gesichtscharakteristika, wie Helligkeitsunterschiede oder Kanten, untersucht. Falls der betrachtete Ausschnitt als Gesicht in Frage kommt, werden in den nächsten Schritten immer komplexere Regeln und Merkmale, bis hin zu den charakteristischen Gesichtseigenschaften wie Augen, Nase und Mund zu Rate gezogen. Die Schwierigkeiten in der Umsetzung solcher Verfahren bestehen vor allem in der problematischen Übersetzung menschlichen Wissens in wohldefinierte Regeln, denn offensichtlich führen zu strenge Regeln zu fälschlich nicht detektierten Gesichtern und zu allgemeine Regeln zu Gesichtsdetektionen auf dem Hintergrund. Als Beispiel für ein Verfahren dieser Klasse kann die, auf Regeln für unterschiedlich aufgelöste Versionen eines Bildes (*Multiresolution*) basierende, Methode von Yang und Huang [YH94] angeführt werden. In ihr werden, beginnend mit einem niedrig aufgelösten Bild, zunächst einfache Regeln (beispielsweise bezüglich der Helligkeitsverteilung), welche mit steigender Auflösung immer komplexer werden, angewandt. Sobald eine Regel verletzt wird, kann das Bild als *Nicht-Gesicht* verworfen werden und die rechnerisch aufwändigeren Regeln der folgenden Stufen müssen nicht mehr ausgewertet werden.

2.2.2 Merkmalsbasierte Methoden

Die zweite Klasse von Gesichtsdetektionsverfahren wird von Yang als *merkmalsbasierte Methoden* bezeichnet. Entsprechend der Annahme, dass Menschen unabhängig von Beleuchtung und Pose Gesichter problemlos erkennen können, wird hierbei auf die Existenz von diesbezüglich relativ invarianten Merkmalen geschlossen. Mögliche Merkmale dieser Art können die offensichtlichen Gesichtsmerkmale wie Augen und Mund, aber auch Farbe oder Textur der Haut sein. Im Gegensatz zu den wissensbasierten Methoden arbeiten

die Verfahren dieser Gruppe nach einem *Bottom-up*-Ansatz. Zuerst werden die einzelnen Merkmale detektiert, wobei üblicherweise eine Kantendetektion zum Einsatz kommt, wie in den Verfahren [LBP95], [YC97]. Danach werden diese zu Gesichtskandidaten gruppiert. Anschließend kann mit Hilfe von statistischen Modellen über Gesichtsrelationen die Existenz eines Gesichtes nachgewiesen werden. Hierbei gibt es auch Verfahren, die verschiedene Merkmalsarten kombinieren (vgl. [KK96]).

2.2.3 Template Matching

Eine weitere Gruppe der Gesichtsdetektionsverfahren basiert auf der Technik des *Template Matchings*. Hierbei wird im einfachsten Fall eine allgemeine Gesichtsvorlage, die beispielsweise als Durchschnitt über verschiedene Gesichtsbilder berechnet werden kann, benötigt. Es kann somit die pixelweise Differenz zwischen einem zu untersuchenden Ausschnitt und dem Durchschnittsgesicht bestimmt werden. Die Entscheidung zwischen Gesicht und Nicht-Gesicht erfolgt anschließend durch den Vergleich dieser Differenz mit einem gewählten Schwellwert. Ebenso können mit Hilfe von Korrelationsverfahren potentielle Positionen des Gesichtstemplates ermittelt werden. Offensichtlich sind derartige Verfahren einfach zu implementieren, sie führen aber nicht zu robusten Ergebnissen. Zwar können Abweichungen zum Vergleichsmuster, wenn sie das gesamte Bild beeinflussen, wie Änderungen in der Lichtintensität, vielfach durch Normalisierung beseitigt werden. Falls aber nur Bildbereiche betroffen sind oder Größe, Pose und Form des Gesichtes sich zu sehr unterscheiden, werden nur sehr schlechte Detektionsergebnisse erzielt. Aus diesem Grund erweitern höherentwickelte Ansätze die einfachen Durchschnittgesichtsmuster um Ideen wie multiple Auflösungen, Größen oder Subtemplates. Außerdem werden die vordefinierten Vorlagen (vgl. [CTB92]) oft durch verformbare Vorlagen ersetzt (vgl. [LTC95]). Entsprechende Techniken erzielen wesentlich robustere Detektionsergebnisse und sind weniger abhängig von stabilen Bilderfassungsbedingungen.

2.2.4 Ansichtsbasierte Methoden

Die letzte Klasse von Gesichtsdetektionsverfahren, die hier vorgestellt werden soll, wird von Yang als *ansichtsbasierte Methoden* bezeichnet. Im Gegensatz zu den vorherigen Ansätzen werden hierfür keine expliziten Informationen über das menschliche Gesicht benötigt, um einen Klassifikationsalgorithmus zu entwickeln. Stattdessen wird eine Entscheidungsfunktion mit Hilfe von statistischen und Maschinenlern-Verfahren auf einer Trainingsmenge von Gesichts- und Nichtgesichtsbildern gelernt. Aufgrund der hohen Dimensionalität des Bildraums (Anzahl der Pixel in einem Bild) wird zur Bestimmung der Entscheidungsfunktion eine effiziente Unterraum-Repräsentation benötigt. Die Entscheidungsfunktion korrespondiert in diesem Unterraum zu einer Fläche oder Hyperebene und trennt somit Gesichter von Nicht-Gesichtern. Verfahren dieser Gruppe basieren unter anderem auf der Hauptkomponentenanalyse (*Eigenface*-Ansatz, siehe Abschnitt 4.2.1), neuronalen Netzen [RBK98] oder Hidden Markov Modellen [RKK+98].

2.3 Umsetzung und Erweiterung

2.3.1 Viola & Jones Gesichtsdetektion

Die Viola & Jones Gesichtsdetektionsmethode, wie in [VJ01] vorgestellt, ist ein Beispiel für ein ansichtsbasiertes Verfahren. Es ist nicht auf den Bereich der Gesichtsdetektion beschränkt, sondern findet vielfach Anwendung in der Erkennung von Objekten, deren Orientierung nicht zu stark variiert. Wir haben das Verfahren gewählt, da es sich, nach dem Training auf einer angemessen großen Menge an Trainingsbildern, durch gute Erkennungsraten und nahezu Echtzeitverhalten auszeichnet. Vor allem die hohe Geschwindigkeit und Robustheit gaben den Ausschlag für den Einsatz in unserem Detektor-Modul, weil somit zeitnah die notwendigen Informationen für die Tracking- und Identifikationsprozedur zur Verfügung gestellt werden können.

Wie bereits erwähnt, werden für das Training Bilder von Gesichtern und Nicht-Gesichtern benötigt, wobei für Erstere möglichst ein Ausschnitt des Gesichts gewählt wird, der die typischen Gesichtsattribute beinhaltet, ohne dabei die stark variierenden Randbereiche wie das Haar oder den Hintergrund mit zu umfassen. Die zur Unterscheidung zwischen Gesichtern und Nicht-Gesichtern eingesetzten Merkmale ähneln den einfachen Haar-Features (vgl. [POP98, MPP01]) und bestehen aus rechteckigen Blöcken (Abb. 2.2(a)). Da diese in Art, Größe und Position variieren können, existiert eine große Anzahl möglicher Ausprägungen. Zur Durchführung des Trainings benutzt der Viola & Jones Gesichtsdetektor ein Boosting-Verfahren das als Adaboost bezeichnet wird. Es bestimmt die Teilmenge von Blockfiltern (Blockmerkmal auf einen Bildausschnitt angewandt), welche auf der Trainingsmenge effizient zwischen den beiden Bildklassen unterscheiden kann. Um die Berechnungseffizienz des Gesichtsdetektors zu erhöhen, werden im Klassifikationsprozess die Filter hierarchisch in einer Kaskade angeordnet und abgearbeitet. Die unteren Ebenen der Kaskade (wenige Blockfilter) verwerfen Bilder, die eindeutig keine Gesichter beinhalten, während die höheren Ebenen (viele Blockfilter) potentielle Gesichter aufwändiger untersuchen.

Merkmale

Bei den im Viola & Jones Ansatz benutzten Merkmalen handelt es sich um einfache Blockfilter. Jeden dieser Filter kann man als eine Kombination aus schwarzen und weißen Rechtecken ansehen, die auf ein Bild angewandt werden, indem die Summe der Pixel in den weißen Bereichen von der Summe der Pixel in den schwarzen Bereichen subtrahiert wird. Wie in Abbildung 2.2(a) zu sehen, gibt es verschiedene Grundanordnungen dieser Blöcke, welche die Detektion von horizontalen, vertikalen oder diagonalen Kanten, Linien oder Helligkeitsunterschieden ermöglichen. Trotz der geringen Anzahl an Basisfiltern wird die Zahl der möglichen Ausprägungen sehr groß, da jeweils sämtliche Variationen in Größe und Position auf ein Bild angewandt werden können. Für die Berechnung der Merkmalswerte müssen Summen über Pixel gebildet werden. Mit einem trivialen Ansatz würden so für leicht verschobene Filter viele Additionen mehrfach durchgeführt werden, was offensichtlich sehr ineffizient ist.

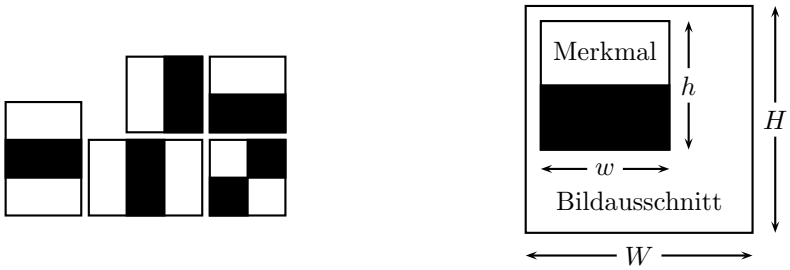


Abbildung 2.1: Basis Blockmerkmale mit verschiedenen Ausprägungen/Orientierungen und ihre Platzierung im Bildausschnitt

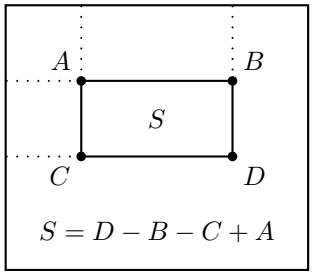


Abbildung 2.2: Integralbild zur effizienten Berechnung von Pixelsummen (Durch Vorberechnung der kumulierten Pixelwerte kann die Berechnung von Pixelsummen über beliebige Bildausschnitte auf vier elementare Rechenoperationen reduziert werden.)

Integralbild

Um die Neuberechnung bestimmter Summen zu vermeiden benutzt der Viola & Jones Ansatz eine Repräsentation des Bildes in der Form eines Integralbildes (Abb. 2.2). Nach

$$I(x,y) = \sum_{k=0}^{x}\sum_{l=0}^{y} i(k,l) \qquad i(k,l) : \text{Helligkeitswert des Pixel } (k,l)$$

entspricht jedes Element (x, y) des Integralbildes der Pixelsumme aller Pixel, die im Rechteck von $(0, 0)$ bis (x, y) im Originalbild liegen. Diese Repräsentation kann in linearer Zeit vorberechnet werden. Mit Hilfe der gespeicherten Pixelsummen kann die Summe über jedes beliebige Rechteck im Originalbild ausgedrückt werden als

$$S(x_1, y_1, x_2, y_2) = I(x_2, y_2) - I(x_1, y_2) - I(x_2, y_1) + I(x_1, y_1)$$

und somit wesentlich effizienter berechnet werden.

Lernverfahren

Das Lernverfahren, das von Viola & Jones benutzt wurde, basiert auf dem erstmals von Shapire [Sch90] eingeführten Boosting. Das wesentliche Prinzip hierbei ist eine Klassifi-

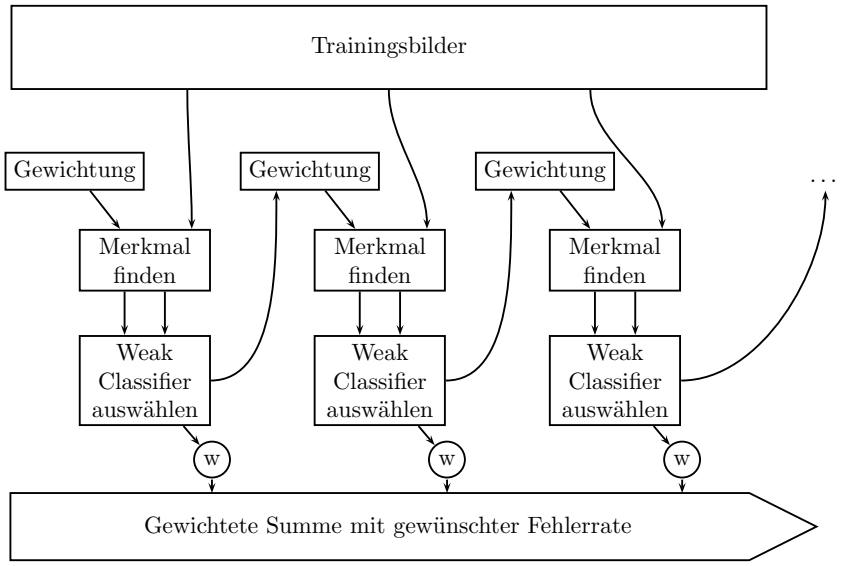


Abbildung 2.3: Training eines Klassifikators mit variierender Gewichtung der Trainingsbilder; Für eine Gewichtung der Trainingsbilder wird jeweils ein Merkmal zur Unterscheidung (Gesicht/Nicht-Gesicht) bestimmt. Für *schwierige* Bilder wird die Gewichtung erhöht, der Vorgang wiederholt.

kation auf Grundlage eines Mehrheitsentscheides verschiedener Experten, die jeweils auf verschiedenen Trainingsmengen gelernt haben. Die Trainingsmengen müssen hierbei bestimmte Kriterien erfüllen, um sicherzustellen, dass die Experten (oder schwachen Klassifikatoren) sich für ihre Entscheidung auf verschiedene Eigenschaften oder Merkmale konzentrieren. Bei einer großen Anzahl schwacher Klassifikatoren ist es somit ausreichend, wenn ihre Fehlerwahrscheinlichkeit jeweils nur etwas kleiner als $0,5$ – also leicht besser als zufällig – ist, um einen wesentlich geringeren Gesamtfehler zu erhalten. Um drei schwache Klassifikatoren zu trainieren, könnte man beispielsweise den ersten h_1 auf einer Teilmenge aller Bilder und den zweiten h_2 auf einer Teilmenge, auf der h_1 mit $50\,\%$ Wahrscheinlichkeit falsch entscheidet, trainieren. h_3 würde schließlich auf einer anderen Teilmenge trainiert werden, auf der h_1 und h_2 nie die gleiche Entscheidung treffen. Indem man das Training so durchführt, würden h_1 und h_2 sich auf verschiedene Aspekte konzentrieren und h_3 würde sicherstellen, dass sogar, falls h_1 oder h_2 falsch liegen, die richtige Entscheidung getroffen wird. Ein Problem, was hierbei allerdings auftreten könnte, ist, dass h_2 oder h_3 nur kleine Trainingsmengen erhalten, falls h_1 schon eine sehr gute Fehlerrate erzielt und somit wenige Fehler macht. Diese Fehler wären für die Konstruktion der weiteren Trainingsmengen im beschriebenen Verfahren notwendig. Trotz ihrer kleinen Trainingsmengen hätten h_2 und h_3 in diesem Fall den gleichen Einfluss auf das Endergebnis, da eine Mehrheitsentscheidung durchgeführt würde.

AdaBoost

Um dieses Problem zu vermeiden, wird für das Viola & Jones Detektionsverfahren der als AdaBoost bekannte erweiterte Boosting-Algorithmus nach Freund und Schapire [FS95] eingesetzt. Es ist wahrscheinlich der momentan am weitesten verbreitete Boosting-Algorithmus, weshalb auch weitere Abwandlungen existieren. AdaBoost leidet nicht unter dem Problem der schrumpfenden Trainingsmengen und kann wiederholt ausgeführt werden, bis der Fehler auf der Trainingsmenge einen bestimmten Grenzwert unterschreitet. Im Gegensatz zum allgemeinen Boosting wird hierbei eine gewichtete Trainingsmenge benutzt. Zu Beginn haben alle Bilder die selbe Gewichtung. In jeder Iteration des Algorithmus wird ein neuer schwacher Klassifikator, bestehend aus Basismerkmal, Position, Größe und Grenzwert bestimmt, indem der gewichtete Klassifikationsfehler über der Trainingsmenge minimiert wird. Sobald die minimale Kombination von Basismerkmal, Position, Größe und Grenzwert gefunden ist, wird dieser Klassifikator entsprechend seines Fehlers auf der Trainingsmenge gewichtet und den Bildern der Menge werden neue Gewichte zugewiesen. Hierbei werden die Gewichte der richtig klassifizierten Bilder reduziert und die der falsch erkannten Bilder erhöht. Somit erhält die richtige Klassifikation der schwierigeren Bilder für den nächsten zu findenden schwachen Klassifikator eine höhere Priorität. Diese Prozedur kann iteriert werden, bis der Fehler die gewünschte Schranke unterschreitet. Die Gesamtentscheidung entspricht dann der gewichteten Summe über die Antworten aller schwachen Klassifikatoren. Algorithmus 2.1 und Abbildung 2.3 zeigen die von Viola & Jones genutzte Variante.

Detektionskaskade

Viola & Jones benutzen eine hierarchische Anordnung von Klassifikatoren um eine hohe Performanz der Gesichtsdetektionsmethode zu erzielen (Abb. 2.4). Auf jedem Level der Detektionskaskade wird ein starker Klassifikator eingesetzt, der aus mehreren, mit AdaBoost trainierten, schwachen Klassifikatoren besteht. Die Anzahl dieser schwachen Klassifikatoren steigt von Level zu Level. Auf jeder Ebene kann ein Bild entweder als Nicht-Gesicht verworfen oder zur nächsten Ebene weiter gereicht werden. Somit werden Bilder, die eindeutig keine Gesichter enthalten, frühzeitig erkannt und Bilder, die Gesichtern ähneln, werden genauer untersucht. Nur falls ein Bild alle Level durchläuft, wird es als Gesicht akzeptiert.

2.3.2 Hautfarbendetektion

Erste Experimente in dem vorgegebenen Szenario dieses Projektes zeigten, dass Gesichter zwar zuverlässig erkannt werden, sich jedoch auch viele Nicht-Gesichter unter den Detektorergebnissen befinden. Um die Anzahl dieser False-Positives zu verringern, haben wir das Verfahren von Viola & Jones um eine Hautfarbenverifikation erweitert. Bildausschnitte, die von dem Viola & Jones Verfahren als mögliche Gesichter identifiziert sind, basierend auf der Graustufendarstellung des jeweiligen Bildes, werden anschließend auf das Vorhandensein von Hautfarbwerten im entsp. Bildausschnitt des Farbbildes untersucht. Den einzelnen Pixeln des Bildausschnitts wird ein Wahrscheinlichkeitswert zu-

- Gegeben sind die Trainingsbilder $(x_1, y_1), \ldots, (x_n, y_n)$ wobei y_i die Bilder als negative ($y_i = 0$) oder positive ($y_i = 1$) Beispiele kennzeichnet.
- Initialisiere Bild Gewichte:

$$w_i^1 = \begin{cases} \frac{1}{2m} & \text{, wenn } y_i = 0 \\ \frac{1}{2l} & \text{, wenn } y_i = 1 \end{cases}$$

  hierbei entsprechen m und l den Anzahlen der negativen und positiven Beispiele
- Für $t = 1 \ldots T$
  - Normalisiere Gewichte:

$$w_i^{t+1} = \frac{w_i^{t+1}}{\sum_i w_i^{t+1}}$$

    so dass w_t einer Wahrscheinlichkeitsverteilung entspricht.
  - Für jedes Feature, j, wird ein Klassifikator h_j trainiert. Sein Fehler wird unter Berücksichtigung von w_t bestimmt:

$$\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$$

  - Wähle den Klassifikator, h_i, mit dem geringsten Fehler ε_t.
  - Aktualisiere Gewichte:

$$w_i^{t+1} = w_i^t \left(\frac{\varepsilon_t}{1 - \varepsilon_t} \right)^{1-e_i}$$

    wobei $e_i = 0$, falls x_i richtig klassifiziert wurde und sonst $e_i = 1$ gilt.
- Der starke Klassifikator ergibt sich als Gemeinschaftsentscheidung:

$$\Rightarrow H_T(x) = \begin{cases} 1 & \text{wenn } \sum_{t=1}^T \alpha \cdot h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{sonst} \end{cases}$$

  mit $\alpha_t = \log \frac{1-\varepsilon_t}{\varepsilon_t}$

Algorithmus 2.1: AdaBoost

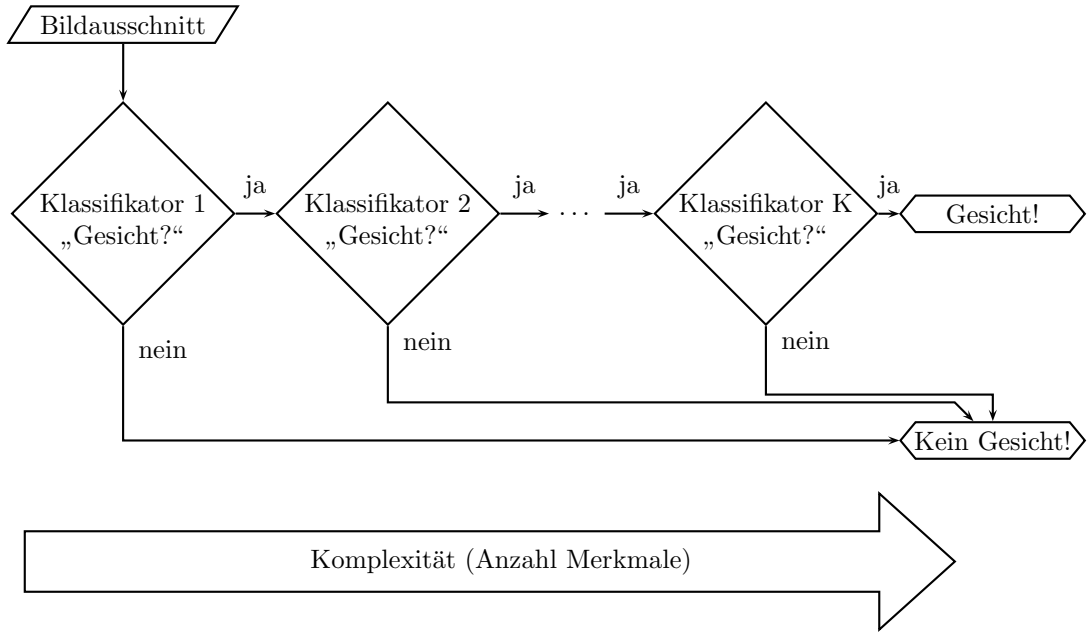


Abbildung 2.4: Violas & Jones Detektionskaskade; Auf jeder Stufe werden Nicht-Gesichter verworfen und Gesichter zur nächst-höheren Stufe weitergereicht. Die Klassifikatoren jeder Stufe setzen sich aus mehreren schwachen Klassifikatoren zusammen. Die Anzahl steigt mit dem Stufenindex an.

gewiesen, der das Vorhandensein von Hautfarbe beschreibt. Liegt die durchschnittliche Wahrscheinlichkeit aller Pixel im Bildausschnitt über einem gegebenen Schwellwert, wird der Ausschnitt als Gesichtsdetektion akzeptiert, ansonsten verworfen.

Für die Erkennung von Hautfarbe verwenden wir den H-Kanal des in den HSV-Raum konvertierten Bildes. Basierend auf einem Referenzbild wird ein Farbhistogramm der für hautfarbe typischen H-Werte erstellt. Mit diesem Histogramm wird die Wahrscheinlichkeitsprojektion für die einzelnen Pixel eines Bildausschnitts durchgeführt. Eine Beschreibung der positiven Eigenschaften des H-Kanals bzgl. der Hautfarbe sowie eine detaillierte Darstellung der für die Ermittlung der Wahrscheinlichkeitswerte notwendigen Berechnungen findet sich in Kapitel 3.

3 Tracking

Nachdem unser System nun in der Lage ist, Gesichter zu detektieren, ist der nächste wichtige Schritt hin zu einem videobasierten Detektions- und Identifikationssystem, dass wir diese detektierten Objekte verfolgen und somit jederzeit wiederfinden können. Diese Tätigkeit wird als Tracking bezeichnet und bildet den zweiten Hauptbestandteil dieser Projektgruppe, welcher in diesem Kapitel erläutert wird.

Neben der Klassifikation von Objekten ist deren Verfolgung in einer Sequenz von Bildern von besonderer Bedeutung. Sie ermöglicht es, gewonnene Informationen über erkannte Objekte diesen auch zu den Zeitpunkten zuzuordnen, in denen eine erneute Klassifikation nicht möglich ist. Wird eine Person beispielsweise anhand ihres Gesichtes detektiert und identifiziert, können diese Informationen auch zu den Zeitpunkten zugeordnet werden, in denen das Gesicht der Person nicht oder nur teilweise im Bild präsent ist, der Kopf der Person jedoch weiterhin verfolgt wird.

In diesem Kapitel werden sowohl die theoretischen Grundlagen verschiedener Tracking-Verfahren, als auch die in diesem Projekt entwickelten Erweiterungen vorgestellt und diskutiert.

3.1 Problembeschreibung

Das Wort Tracking bezeichnet die kontinuierliche Verfolgung der Position von Objekten, wie Personen, Autos oder Gesichtern, in einer Folge von Videobildern. Im Folgenden werden die dabei zu lösenden Problemstellungen diskutiert und verschiedene Lösungsansätze aus der Literatur präsentiert.

Beim Tracking ergeben sich einige grundlegende Probleme, die zu lösen sind, unabhängig davon, welches Verfahren genutzt wird. Zu jedem der Probleme existieren unterschiedliche Lösungsansätze, die in den verschiedenen Verfahren Anwendung finden. Diese Verfahren unterscheiden sich teils deutlich in ihrer Möglichkeit, einzelne dieser Tracking-spezifischen Probleme zu bewältigen. Im Folgenden werden nun diese Hauptprobleme des Trackings vorgestellt.

Die Hauptproblematiken sind:

- Es muss sichergestellt werden, dass mit Objektänderungen, also Änderungen des Aussehens über die Zeit, umgegangen werden kann. Mögliche Veränderungen sind Drehungen, Verformungen und Beleuchtungsänderungen.
- Wie lassen sich Objekte unterscheiden, die in ihrer Repräsentation sehr ähnlich sind? Hierbei kommt erschwerend hinzu, wie mit Verdeckungen dieser Objekte umzugehen ist.
- Wie genau ist ein Trackingverfahren, wie zuverlässig ist dieses Verfahren und wie genau kann man sich auf die Ergebnisse dieses Verfahrens verlassen? Was passiert

wenn die Trajektorie von Verdeckungen etc. betroffen ist? Wie evaluiert man die Ergebnisse?

- Wie effizient ist ein Verfahren? Ist ein Verfahren echtzeitfähig?

Abhängig von den Anforderungen, die an den Tracker gestellt werden, kommen noch folgende Probleme hinzu:

- Was ist ein Objekt, wie repräsentiert man dieses und wie sieht die interne Modellierung aus? (z. B. Blob oder Histogramm)
- Welches Wissen muss der Tracker über die Umgebung sowie über das Objekt haben?

Dies sind die Hauptproblematiken, die es beim Tracking zu lösen gilt.

3.2 Methodenübersicht

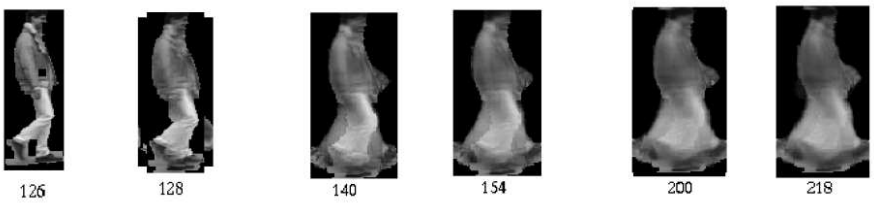
Die Realisierung eines robusten Trackers erfordert den Einsatz verschiedener Methoden: Das zu trackende Objekt muss in angemessener Weise modelliert werden, so dass es in den visuellen Daten gut repräsentiert ist. Des Weiteren muss die neue Position des Objektes möglichst genau ermittelt werden und die Möglichkeit gegeben sein, die Ungenauigkeiten sowohl in der Objektrepräsentation als auch bei der Positionsvorhersage zu behandeln. In diesem Abschnitt werden grundlegende Methoden für diese Aufgaben vorgestellt.

3.2.1 Regionsbasierte Objektrepräsentation

Bei diesem Ansatz werden bewegte Objekte durch Zuordnung der Pixel zu den einzelnen Objektregionen getrackt. Es wird ein dynamisches, sich über die Zeit veränderndes Modell für die Objektrepräsentation genutzt, ein sogenanntes Blob-Modell. Dieses kann aus einer einzigen Region bestehen, meist jedoch aus mehreren Regionen und deren Lage zueinander. Vorteil dieses Verfahrens ist die dynamische Anpassung des komplexen Objektmodells, so dass auch sich stark ändernde Objekte verfolgt werden können. Der Nachteil dieses komplexen Modells ist hingegen die aufwändige Initialisierung für ein Objekt. Ein Beispiel eines regionsbasierten Trackers findet sich in [IM01].

3.2.2 Histogrammbasierte Objektrepräsentation

Bei einem histogrammbasierten Tracking findet im Gegensatz zu einer regionsbasierten Methode keine Zuordnung von einzelnen Pixeln zu dem zu trackenden Objekt statt. Das zu trackende Objekt wird hier durch ein Histogramm des entsprechenden Bildbereichs repräsentiert und verfolgt. Das Histogramm selbst kann bei diesem Verfahren sowohl in einer statischen, wie auch einer dynamischen Variante vorliegen: Das Modell (Histogramm) wird stets zu Beginn des Trackings initialisiert und dann als statische Objektrepräsentation verwendet, alternativ wird das Histogramm im Laufe der Objektverfolgung an Änderungen des Objektes dynamisch angepasst. Vorteil dieser Verfahren ist, dass die

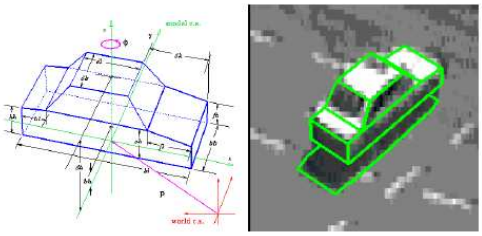


(a) Beispiel eines Blob-Region-Trackers. Die Abbildung zeigt, wie sich das Modell über die Zeit anpasst. Es werden immer mehr Pixel zugeordnet (aus [HHD98]).

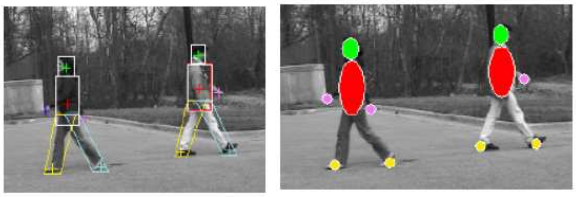
(b) Beispiel eines Blob-Trackers, dargestellt sind das original Kamerabild und die Zuordnung der Pixel zu den Modellen, innerhalb des Bildes (aus [IM01]).

Abbildung 3.1: Die Abbildung zeigt zwei unterschiedliche Modelle, wie sie von Blob-Trackern genutzt werden. Im ersten Teilbild wird die Anpassung eines Modells über die Zeit dargestellt, es werden immer mehr Pixel, dem Modell, zugeordnet. Das zweite Teilbild stellt ein original Kamerabild dar und zeigt, wie die Pixel, den verschiedenen Modellen zugeordnet werden.

Abbildung 3.2: Das Histogramm wird hier aus der Ellipse errechnet (aus [CRM03]).



(a) 3D-Model-Tracker eines Autos (aus [Kol96]).



(b) Model-Tracker mit unterschiedlichen 2D-Modellen zum Tracken einer Person(aus [HHD98]).

Abbildung 3.3: Verschieden Model-Tracker. Das erste Teilbild zeigt einen Model-Tracker der mit 3D-Modellen arbeitet. Das zweite Teilbild stellt Model-Tracker mit unterschiedlichen 2D-Modellen vor.

Initialisierung der Modelle sehr einfach ist. Nachteil ist die Anfälligkeit der Histogramme gegenüber Beleuchtungs- und Objektänderungen. Die dynamische Variante trägt dieser Problematik Rechnung. Hier ist es allerdings problematisch, zu entscheiden, auf welche Weise die Modelle (Histogramme) über welchen Zeitraum anzupassen sind. Potentiell besteht bei der Anpassung des Histogramms die Gefahr, dass ursprüngliche Merkmale des Objektes im Histogramm verloren gehen und eine Anpassung des Histogramms an objektfremde Merkmale geschieht. Als Beispiele siehe [FT97], [WADP97], [HW94], [SP96], [SB91], [CRM03].

3.2.3 Modellbasierte Objektrepräsentation

Der Ansatz des modellbasierten Trackings untersucht die vorliegenden Daten auf gewisse Bildmerkmale und segmentiert das Bild entsprechend. Daraufhin werden die Merkmale mit zuvor erstellten 3D-Modellen der zu trackenden Objekte verglichen. Diese Objekte bestehen z. B. für starre, sich nicht ändernde Objekte aus einem einfachen, statischen Gittermodell und für dynamische, nicht-starre Objekte aus einem Modell, welches dynamische Änderungen erlaubt (siehe [HHD98], [Kol96]). Bei den modellbasierten Verfahren ist die Initialisierung sowie die Suche der Trajektorien aufgrund der verwendeten, teilweise komplexen 3D-Modelle sehr aufwändig und rechenlastig.

3.2.4 Bewegungsvorhersage

Das Grundprinzip aller Trackingverfahren, unabhängig von der gewählten Modellierung, ist es, aus einer bekannten oder mit einer gewissen Wahrscheinlichkeit angenommenen Objektposition und den aktuellen visuellen Daten die momentane Objektposition zu approximieren. Zu diesem Zweck kommen verschiedene Methoden zum Einsatz, wie der Kalman-Filter oder der Mean-Shift-Algorithmus. Abhängig vom Einsatzgebiet des Trackers und der Objektmodellierung werden Bewegungsschätzer auch mit Partikelfiltern oder komplexen Bewegungsmodellen kombiniert. Auf diese Weise kann das a-priori-Wissen über das zu verfolgende Objekt, insbesondere die zu erwartenden Bewegungsmuster, genutzt werden, um auch schwierige Situationen zu bewältigen. Einige dieser Verfahren werden im Folgenden in Zusammenhang mit den vorgestellten Tracking-Verfahren im Detail erläutert.

3.3 Trackingverfahren

Ein Tracking-Verfahren kombiniert verschiedene der vorgestellten Methoden, um auch unter schwierigen Bedingungen die Verfolgung beliebiger Objekte zu erreichen und den Problemstellungen des Trackings zu begegnen. Die Auswahl der nachfolgend detailliert vorgestellten Verfahren orientiert sich an dem für dieses Projekt definierten Anforderungsprofil des Tracking-Moduls.

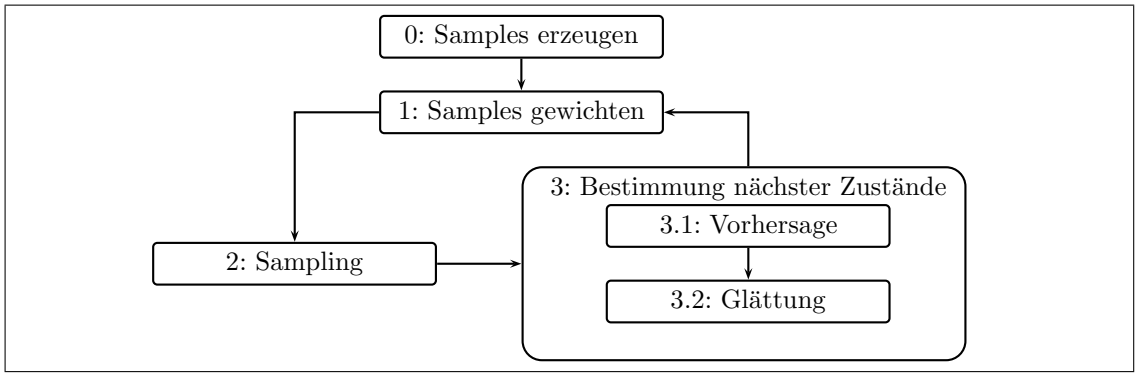
3.3.1 Condensation

Einführung

Der Condensation-Algorithmus basiert auf Partikelfiltern, die auch unter dem Namen sequentielle Monte-Carlo-Methode (SMC) bekannt sind. Dies sind stochastische Verfahren zur Zustandsschätzung in einem dynamischen Prozess, wobei die Dynamik dieser Systeme nur ungefähr bekannt bzw. unvollständig zu beobachten ist. Beim Partikelfilter werden N Hypothesen (Partikel) beobachtet, die eine Vorhersage aller Bewegungen ermöglichen sollen. Genau N Partikel (Samples) werden genutzt, um Laufzeitgarantie und aussagekräftiges Gewichtungsverhalten zu gewährleisten. Die Partikel werden durch entsprechende Zustandsvektoren dargestellt:

$$S = \left\{ s^{(1)}, \ldots, s^{(N)} \right\}$$

Des Weiteren werden ein Bewegungs-, ein Objekt- und ein Beobachtungsmodell benötigt, um ein Gesamtmodell zu bilden. Die Idee hierbei ist, eine Näherungslösung unter Verwendung dieses komplexen Modells zu finden. Als Annahme gilt, dass das »interessante« Modell zu Anfang bekannt ist. Im Folgenden wird der Condensation-Algorithmus, nach [IB98], als Beispiel für einen Partikelfilter vorgestellt.



Algorithmus 3.1: Condensation – Verfahrensablauf

Condensation: Verfahrensablauf – Überblick

Der Ablauf des Condensation Algorithmus ist in Algorithmus 3.1 dargestellt. Die einzelnen Schritte werden im Folgenden erläutert.

Condensation: Samples erzeugen – Initialisierung

Im ersten Schritt, zum Zeitpunkt $t = 0$, werden die Partikel erzeugt, wobei jeder Partikel anfänglich die gleiche Gewichtung bekommt. Die Partikel werden über das Bild gleichverteilt, oder um den vermuteten Aufenthaltsort der Trajektorie herum, falls a-priori-Wissen verfügbar ist:

1. Generiere N (Zustands-)Partikel $\left\{s_0^{(1)}, \ldots, s_0^{(n)}\right\}$ entsprechend der anfänglichen Dichte $p(x_0)$.
2. Ordne jedem Partikel das gleiche Gewicht zu $\pi_0^{(1)}, \ldots, \pi_0^{(n)}$.

Condensation: Auswahl der Partikel – factored sampling

In diesem Schritt sind die Partikel und Gewichtungen aus dem vorherigen Schritt bekannt. Es erfolgt nun ein Resampling, d. h. es wird eine neue Partikelmenge bestimmt. Beim Resampling werden Partikel mit niedrigem Gewicht verworfen und Partikel mit hohem Gewicht evtl. mehrfach verwendet. Es wird nun durch »Ziehen mit Zurücklegen« die neue Partikelmenge bestimmt:

1. Ziehen eines Samples $s_{t-1}^{(i)}$ aus der Partikelmenge S_{t-1} (vom Zeitpunkt $t-1$)
2. Übernahme dieses Partikels mit einer Wahrscheinlichkeit $\pi_{t-1}^{(i)}$ entsprechend der Gewichtung des Samples in die neue Partikelmenge S_t
3. Wiederhole ab 1. solange, bis die neue Partikelmenge S_t N Partikel enthält

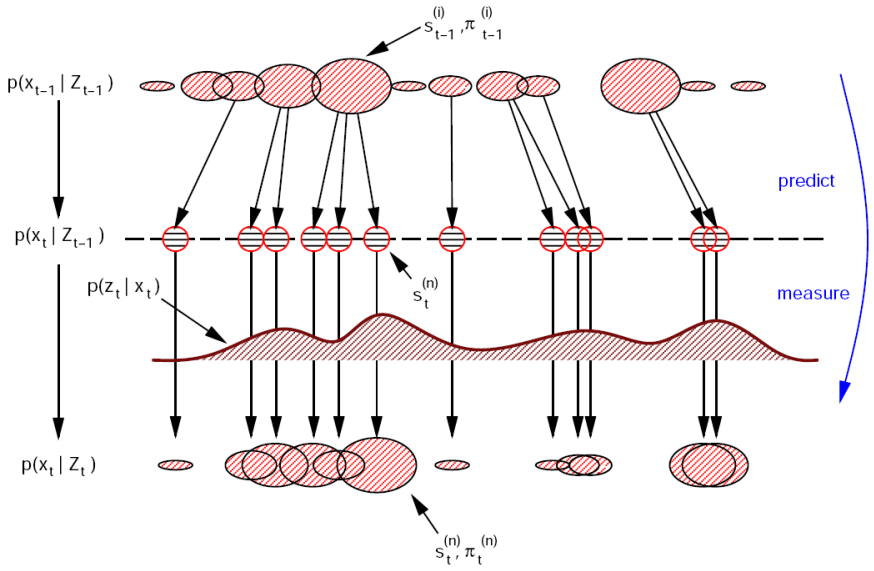


Abbildung 3.4: Hier werden die Partikel dem Sampling unterzogen und die Bewegung wird vollzogen. Identische Partikel werden gleich bewegt. Anschließend wird Rauschen hinzugefügt um diese zu trennen. Zuletzt werden die Partikel mit Hilfe des Bewegungsmodells gewichtet (aus [IB98]).

Condensation: Vorhersage – Bewegung der Partikel

In diesem Schritt wird nun jeder Partikel nach dem Bewegungsmodell $p\,(x_t|x_{t-1})$ deterministisch bewegt, wobei identische Samples identisch behandelt werden. Identische Samples werden dann durch Überlagerung von Rauschen getrennt. Siehe Abbildung 3.4.

Condensation: Update – Messung/Gewichtung

Nachdem die Partikelmenge nun komplett ist, muss sie sinnvoll gewichtet werden. Hierzu wird die Messung z mit einbezogen. Die Partikel werden dann mit der Wahrscheinlichkeit des Beobachtungsmodells $p(z_t|x_t = s_t^{(n)})$ gewichtet. Zusätzlich werden die Gewichte normiert, sodass sie wieder eine Wahrscheinlichkeitsdichte repräsentieren. Somit ist nun die Verteilung zum Zeitpunkt t bekannt. Siehe Abbildung 3.4.

Condensation: Ermittlung der Objektposition

Die Position des gesuchten Objekts wird nun aus den Partikeln geschätzt, siehe Abbildung 3.5. Mit den Samples $s_t^{(i)} = (x, y, \dot{x}, \dot{y}) \in \mathcal{R}^4$ ergibt sich dann die geschätzte Position $(x_{e,t}, y_{e,t})^T$ als gewichteter Mittelwert der Einzelhypothesen:

$$\begin{pmatrix} x_{e,t} \\ y_{e,t} \end{pmatrix} = \sum_{i=1}^{N} \begin{pmatrix} 1&0&0&0 \\ 0&1&0&0 \end{pmatrix} s_t^{(i)}$$

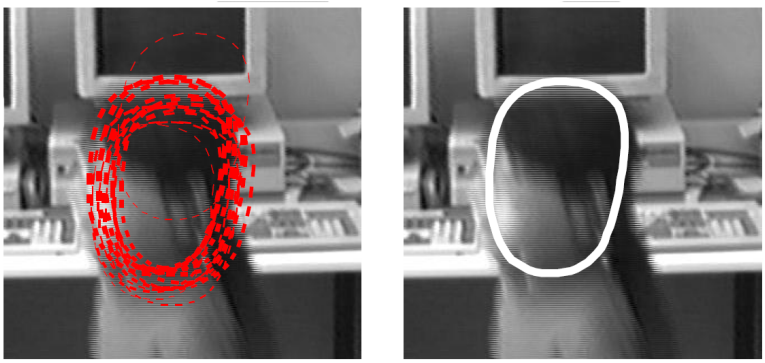


Abbildung 3.5: Abschätzung der Position. Im ersten Teilbild sind die verschiedenen Positionen die durch die Partikel gegeben sind rot eingezeichnet. Im zweiten Teilbild ist die gemittelte Position zu sehen.

Ein großer Nachteil des Verfahrens ist, dass es Bewegungen geben kann, die vom Bewegungsmodell nicht vorhergesagt werden. Zudem steigt der Rechenaufwand mit zunehmender Partikelzahl und der Dimensionen des Zustands- und Partikelvektors. Da die Genauigkeit des Verfahrens ebenfalls von diesen Faktoren abhängt, entsteht hier ein Trade-off zwischen der Genauigkeit und dem Rechenaufwand. Bei linearen Vorgängen ist der Einsatz eines einfacheren Verfahrens, wie eines Kalman-Filters, schneller und genauer.

Vorteile des Verfahrens liegen in der Robustheit gegenüber Störungen. Weil die Partikelanzahl festgelegt ist, gibt es eine Laufzeitgarantie. Zudem eignet sich das Verfahren auch zur Vorhersage nichtlinearer Vorgänge, im Gegensatz beispielsweise zum Kalman-Filter.

3.3.2 Kernel Based Tracking – Mean Shift

Einführung

Kernel based Tracking nach [CRM03] ist ein merkmalbasiertes (feature based) Trackingverfahren, d. h. es findet keine aufwändige Modellierung des Objektes statt, sondern das Tracking erfolgt direkt auf den Bildmerkmalen. Bei diesem Verfahren wird für die Modelle ein isotropischer Kernel genutzt. Ein Kernel ist eine Gewichtsfunktion, welche die Pixel innerhalb des Modells vom Zentrum nach außen hin abschwächend gewichtet. Das Modell ist durch eine ellipsenförmige Region im Bild repräsentiert und besteht aus einem Zielmodell sowie aus Zielkandidaten. Das Zielmodell ist das Modell im Bild $n-1$, das im Bild n wiedergefunden werden soll. Der Zielkandidat bzw. die Zielkandidaten sind die möglichen Modelle (Ziele) im Bild n. Durch die kernelbasierte Gewichtung werden Pixel am äußeren Rand weniger stark berücksichtigt. Diese sind am unzuverlässigsten, da sie häufig von Verdeckungen und Hintergrundinterferenz betroffen sind.

Der Kernel eines Profils K ist eine Funktion $k : [0, \infty) \rightarrow \mathcal{R}$, so dass

$$K(x) = k\left(\|x\|^2\right) \quad \text{, mit}$$
$$k(x) = \begin{cases} \frac{1}{2}C_d^{-1}(d+2)(1-x) & \text{wenn } x \leq 1 \\ 0 & \text{sonst} \end{cases}$$

Hierbei ist x der Pixel, dessen Gewicht berechnet werden soll. $\|x\|$ ist die Entfernung zum Zentrum. Es gilt $d = 2$, da die Position des Pixels durch einen 2-dimensionalen Raum beschrieben wird. C_d ist schließlich die Anzahl der Pixel die sich im Kernel befinden.

Zielrepräsentation (Modellrepräsentation)

Kernelbasiertes Tracking kann auf Basis verschiedener Merkmale zur Modellrepräsentation erfolgen. Es ist lediglich zu beachten, dass ein Merkmal gewählt wird, welches sich durch eine probabilistische Dichtefunktion darstellen lässt. Um einen geringen Rechenaufwand zu gewährleisten, sollten diskrete Dichten verwendet werden (z. B. m-Bin-Histogramme). Um die Dichtefunktion des Modells nun zu berechnen, werden die Dichtefunktion des Merkmals sowie des Kernels miteinander kombiniert.

Zielmodell:

$$\hat{q} = \hat{q}_u \qquad \text{für } u = 1 \ldots m$$

ist die probabilistische Dichtefunktion des Zielmodells q.

Zielkandidat:

$$\hat{p}(y) = \hat{p}_u(y) \qquad \text{für } u = 1 \ldots m$$

ist die probabilistische Dichtefunktion des Zielkandidaten p.

Ähnlichkeitsfunktion von Zielmodell und -kandidat:

$$\hat{\rho} \equiv \rho[\hat{p}(y), \hat{q}]$$

Die Funktion $\hat{\rho}$ ist hier als Likelihood-Funktion zu verstehen; die lokalen Maxima dieser Funktion geben die mögliche Präsenz eines Objekts im zweiten Bild an, dessen Repräsentation ähnlich zum Zielmodell $\hat{q}$, welches im ersten Bild definiert wurde, ist. Diese Ähnlichkeitsfunktion wird mit Hilfe des Kernels reguliert.

Ziellokalisierung (MeanShift-Prozedur)

Gegeben ist das Zielmodell sowie dessen Position y_0 im vorherigen Bild. Nun wird das Ziel im aktuellen Frame n an der alten Position y_0 vom vorherigen Frame $n-1$ initialisiert. Danach wird das lokale Maximum für die Dichtefunktion des Modells mit Hilfe der

MeanShift-Prozedur gesucht. Da die MeanShift-Funktion unendlich konvergent ist, werden zwei Abbruchkriterien genutzt. Erstens wird die maximale Anzahl der Iterationen beschränkt (Standardwerte sind 20 bis 25 Wiederholungen). Zweitens wird eine Fehlerschranke verwendet, um zu verhindern, dass bei bereits pixelgenauer Bestimmung des Ziels aufgrund der unendlichen Konvergenz der Funktion, die Prozedur weitersucht.

MeanShift-Funktion:

$$\hat{y_1} = \frac{\sum_{i=1}^{n_h} x_i w_i g\left(\left\|\frac{\hat{y_0}-x_i}{h}\right\|^2\right)}{\sum_{i=1}^{n_h} w_i g\left(\left\|\frac{\hat{y_0}-x_i}{h}\right\|^2\right)}$$

Die Funktion errechnet den neuen Mittelpunkt, $\hat{y_1}$, dies ist der gewichtete Mittelwert über n_h Beobachtungen. Da ein Kernel verwendet wird, dessen Ableitung eine Konstante ist, ein möglicher Kernel wäre der Epanechnikovkernel, wird es zu einem einfachen gewichteten Mittelwert die neue Position zu errechnen.

Um den oben genannten Problemen zu begegnen, die beim Tracking auftreten, wird noch eine Größenanpassung des Zielmodells vorgenommen, nachdem das Modell wiedergefunden wurde. Dadurch wird den möglichen Änderungen des Modells über die Zeit Rechnung getragen. Der Algorithmus wird zu diesem Zweck dreimal mit unterschiedlichen Werten für h durchgeführt. Damit die Größenanpassung nicht zu stark ausfällt, wird die maximale Größenänderung pro Schritt ebenfalls begrenzt:

1. $h = h_{prev}$
2. $h = h_{prev} + \Delta h$
3. $h = h_{prev} - \Delta h$

Der neue Wert für h ergibt sich dann folgendermaßen (Standardwert für γ ist $\gamma = 0{,}1$) :

$$h_{new} = \gamma \cdot h_{opt} + (1 - \gamma) h_{prev}$$

Die Vorteile des Kernel Based Trackings liegen vor allem in seiner Geschwindigkeit, aufgrund der Einfachheit und Schnelligkeit der MeanShift-Prozedur. Die Schwierigkeit besteht darin, ein Merkmal zu finden, welches den Anforderungen an den Tracker genügt und zugleich durch eine diskrete probabilistische Verteilungsfunktion dargestellt werden kann.

3.4 Umsetzung und Erweiterung

Das Tracking-Modul in diesem Projekt hat die Aufgabe, Personen in einer geschlossenen Umgebung zu verfolgen. Gleichzeitig muss ein Zusammenarbeiten des Tracking-Moduls mit dem Identifizierer- und Detektor-Modul sichergestellt sein. Es bietet sich an, einzelne Tracker auf Basis der Detektor-Ergebnisse zu initialisieren. Ein Effekt dieses Vorgehens ist, dass die zu trackenden Objekte zum Zeitpunkt der Initialisierung zu großen Teilen

hautfarbene Farbwerte besitzen. Aus diesem Grund entschieden wir uns, ein farbhistogrammbasiertes Verfahren für die Verfolgung der Personen einzusetzen, dessen Basis das für diese Zwecke entwickelte CAMShift-Verfahren ist.

3.4.1 CAMShift

Der CAMShift-Algorithmus (*Continously Adaptive MeanShift*) nach [Bra98] ist ein besonders schnelles und robustes Trackingverfahren auf diskreten Wahrscheinlichkeitsdichteverteilungen. Die Objektlokalisierung bei diesem Verfahren basiert auf einer Erweiterung des MeanShift-Algorithmus um eine kontinuierliche, adaptive Anpassung des Suchfensters. Verschiedene Faktoren haben dazu geführt, dieses Verfahren für die Implementierung des Tracker-Moduls zu verwenden. Die entscheidenden Argumente neben Robustheit und Geschwindigkeit waren das reibungslose Zusammenspiel der Objektinitialisierung mit dem Gesichtsdetektor-Modul, sowie die Möglichkeit der Nutzung von Farbhistogrammen, welche sich insbesondere für die vorgegebene Problemstellung eignen. In unserer Implementierung des Tracking-Moduls werden entsprechende Farbhistogramme als Look-Up-Tabelle für eine Dichteverteilung genutzt. Der CAMShift nutzt dann den MeanShift-Algorithmus, um ein Maximum in der Wahrscheinlichkeitsverteilung zu finden und darauf basierend die neue Objektposition und Ausdehnung zu bestimmen. Im Vergleich zu anderen histogrammbasierten Verfahren ist der Rechenaufwand des CAMShift-Algorithmus deutlich niedriger.

Ablauf

In Abbildung 3.6 wird der Ablauf des CAMShift-Verfahrens dargestellt. Im Weiteren wird auf diesen Verlauf genauer eingegangen.

Jedes Videobild wird mit Hilfe eines Histogrammmodells in eine Wahrscheinlichkeitsverteilung konvertiert. Der graue Kasten in 3.6 repräsentiert den MeanShift-Algorithmus, mit dessen Hilfe das Zentrum und die Größe des zu trackenden Objekts gefunden wird. Diese Ergebnisse werden genutzt um das Suchfenster für das nächste Bild zu skalieren und zu platzieren. Für jeden Trackingschritt wird der Prozess wiederholt.

Histogramm

Ziel ist es, eine möglichst gute, genaue und robuste Farbrepräsentation des zu trackenden Objekts zu erhalten. Dazu wird ein 1D-Farbhistogram auf dem H-Kanal des HSV-Farbraums genutzt. Der HSV-Farbraum stimmt mit der Projektion entlang der diagonalen Linie vom weißen ins schwarze des RGB-Farbraums überein (siehe Abbildung 3.7).

Hieraus entsteht dann der kegelförmige HSV-Farbraum, wie er in der Abbildung 3.9(a) und 3.9(b) dargestellt ist. Mit abnehmender Helligkeit (V-Achse) wird der Kegel kleiner, was äquivalent zu kleineren, dunkleren Unterwürfeln im RGB-Farbraum führt. (siehe Abbildung 3.7, der kleinere graugefärbte Würfel)

Für die Histogrammbildung wird nur der H-Kanal, der die Farbe repräsentiert, genutzt. Es wird also lediglich ein 1D-Histogramm, aufgeteilt auf 16 Bins, erstellt. Dieses Histogramm wird einmalig von dem zu trackenden Objekt gebildet und repräsentiert

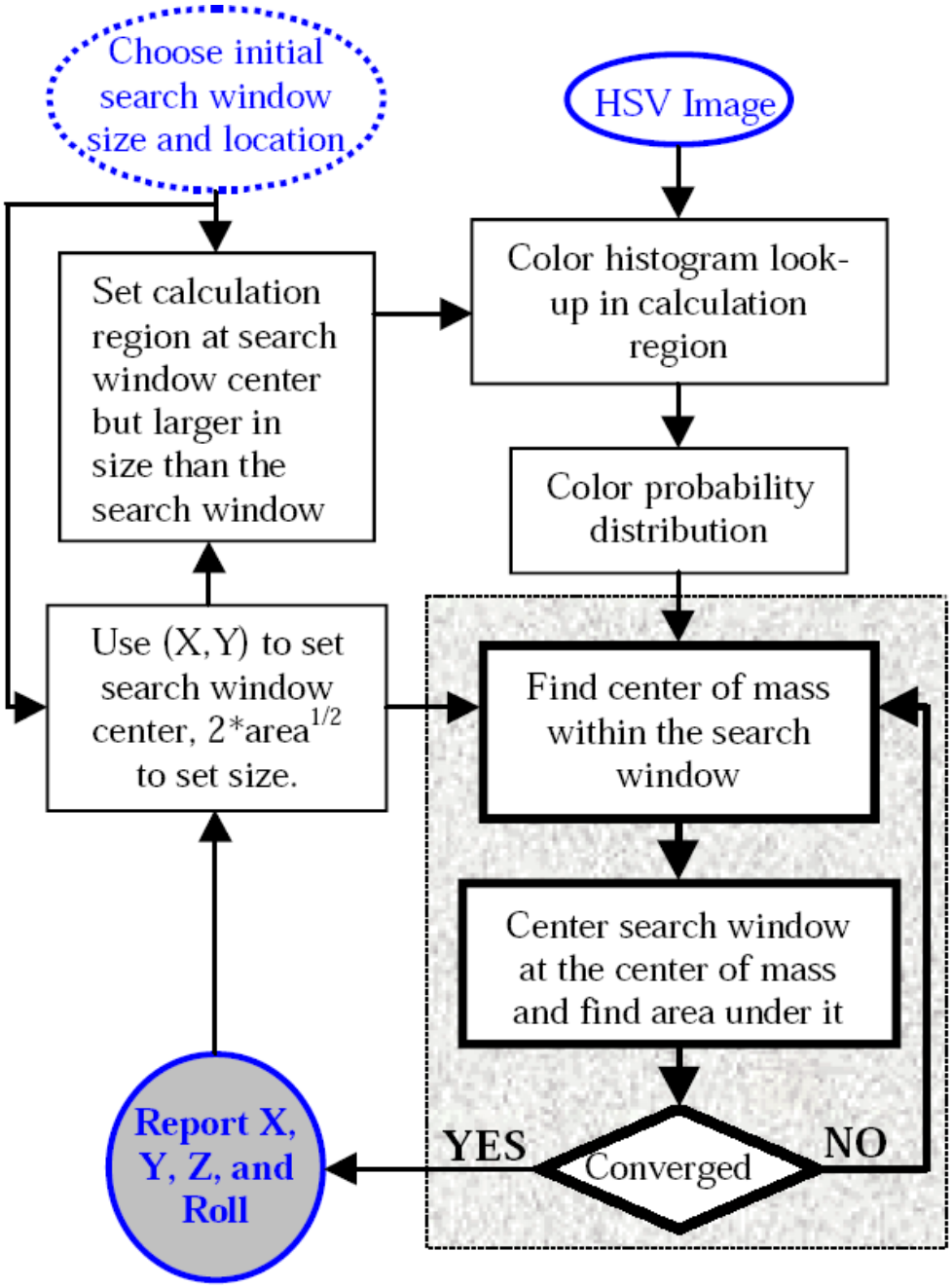


Abbildung 3.6: Ablauf des CAMShift-Algorithmus (aus [Bra98]).

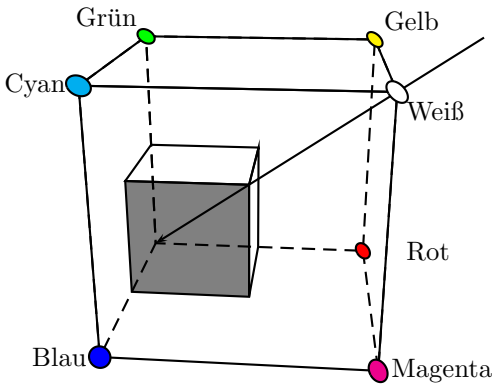
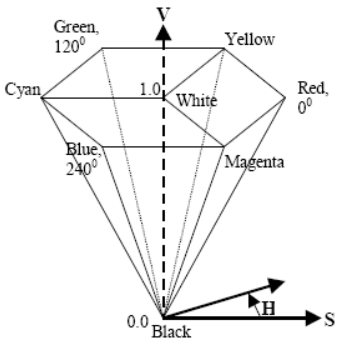
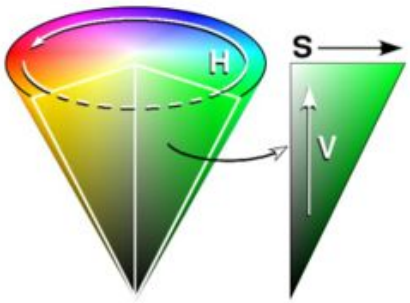


Abbildung 3.7: RGB-Farbwürfel. Der HSV-Farbraum stimmt mit der Projektion entlang der diagonalen Linie überein

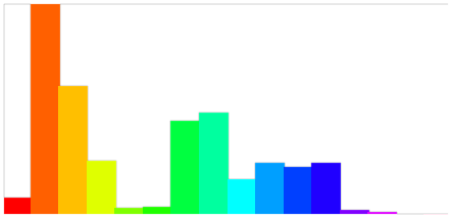


(a) Kegelförmiger HSV-Farbraum, zu sehen ist wie die Variablen H (Hue), S (Saturation), und V (Helligkeit) einfließen (aus [Bra98])

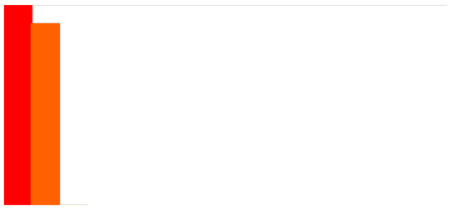


(b) Kegelförmiger HSV-Farbraum (farbig)

Abbildung 3.8: Darstellung des HSV-Farbraums. Das erste Teilbild zeigt wie die Variablen H, S und V einfließen. Das zweite Teilbild stellt dies anschaulicher, farbig dar.



(a) Ein Histogramm das schlecht zum Tracken geeignet ist, da es nicht sehr markant ist und sehr viele Farbanteile darin enthalten sind



(b) Ein Histogramm welches Hautfarben repräsentiert, wie es zum tracken genutzt wird

Abbildung 3.9: Beispiele für 1D-Hue-Histogramme. Das erste Teilbild zeigt ein Histogramm, das sehr schlecht zum Tracken geeignet ist, da es nicht sehr markant ist. Das zweite Teilbild zeigt ein Histogramm, das auf Hautfarben erstellt wurde und zum Tracken genutzt wird

somit modellartig das Objekt. Da der Hue-Kanal die Farbe repräsentiert und dieser unabhängig gegenüber Beleuchtungsänderungen ist, ist das resultierende Histogramm sehr robust. Zusätzlich bietet dieser Ansatz den Vorteil, dass Hautfarben alle den gleichen H-Wert haben, egal ob jemand eine helle oder dunkle Hautfarbe hat. Dunkle Hauttypen haben lediglich einen stärkeren Saturationwert (S) als helle Hauttypen. Dieser Kanal wird bei dem 1D-Histogramm jedoch nicht betrachtet. Somit wird mit dem Hue-Histogramm eine sehr weite Palette von Hautfarben abgebildet. Da wir Gesichter tracken wollen, bietet dies eine sehr robuste und einfache Art, eine modellhafte Repräsentation des zu trackenden Objekts zu bilden. Hierbei kann ein Problem entstehen, wenn die Helligkeit sehr gering ist, da dann die Sättigung ebenfalls nahe 0 ist (siehe Abbildung 3.9(a)). Das hat zur Folge, dass der Hue-Wert in diesem Bereich sehr stark verrauscht ist. Ähnliche Probleme ergeben sich, wenn die V-Werte sehr hoch sind und die S-Werte sehr niedrig. Darum werden Pixel mit sehr niedrigen V-Werten, sehr hohen V-Werten und sehr niedrigen S-Werten gar nicht beachtet.

Wahrscheinlichkeitsverteilung – Backprojection

Das Histogramm, das im ersten Schritt einmalig berechnet wurde, wird nun für jedes weitere Bild als Look-Up-Tabelle genutzt, um eine Wahrscheinlichkeitsverteilung (Backprojection) des Bildes zu berechnen (siehe Abbildung 3.10). Jedem Pixel eines Bildes wird ein Wahrscheinlichkeitswert zugeordnet. Hierbei reicht der Wahrscheinlichkeitswert eines Pixels in diskreten Schritten von 0,0 bis zu einem Wert von 1,0. Für 8-Bit Hue-Werte, wie wir sie nutzen, bedeutet dies, dass die Werte eine Spanne von 0 bis 255 haben.

Tracking-Prozedur

Das Tracking läuft nun folgendermaßen ab: Grundlegend wird der MeanShift-Algorithmus (siehe Abbildung 3.6 grauer Kasten) genutzt. Mit dessen Hilfe werden Maxima

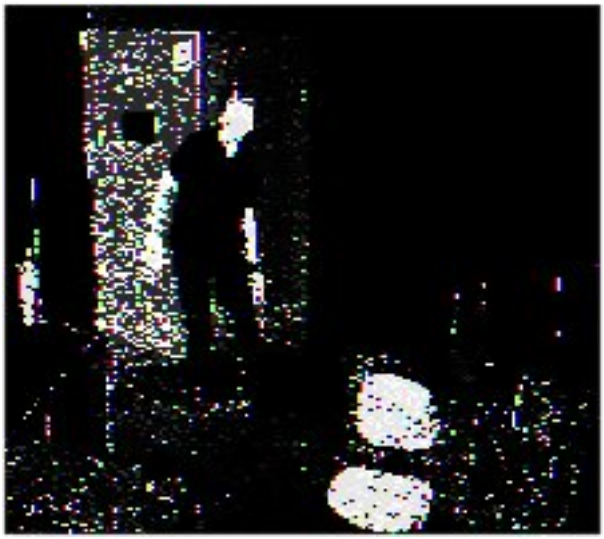


Abbildung 3.10: Beispiel einer Backprojection. Hierzu wurde ein 1D-Hue-Histogramm als Look-up-Tabelle, zum Erstellen der Backprojection genutzt

in der Backprojection gefunden. Diese repräsentieren mit hoher Wahrscheinlichkeit die Objekte, die getrackt werden sollen, da sie mit hoher Wahrscheinlichkeit den Farben aus dem Hue-Farbhistogramm entsprechen, welches von dem zu trackenden Objekt erstellt wurde. Der Ablauf des CAMShift-Algorithmus sieht dann folgendermaßen (siehe Abbildung 3.6) aus:

1. Initialisierung:
   - Wähle die initiale Position des Suchfensters. Das Suchfenster wird in ViPer im ersten Schritt durch den Worldkoordinator auf Basis der Ergebnisse des Detektor-Moduls positioniert.
   - Berechne einmalig das Hue-Histogramm an dieser Stelle.
2. Schritte für jedes Bild:
   - Berechne die Backprojection des aktuellen Bildes n.
   - Platziere das Suchfenster. Dies wird an der Stelle zentriert, wo im Bild $n-1$ das Zentrum des Objekts lag.
   - Wende den MeanShift-Algorithmus an.
   - Passe die Suchfenstergröße so anp, wie sie im MeanShift-Schritt bestimmt wurde.
   - Wiederhole die beiden vorherigen Schritte, bis Konvergenz erreicht ist, sich die Position also weniger als eine vorgegeben Schranke ändert.

In Abbildung 3.12(a) ist der CAMShift-Ablauf grafisch veranschaulicht. Hierbei stellt der rote Graph eine Wahrscheinlichkeitsverteilung dar, wobei der breitere rote Bereich

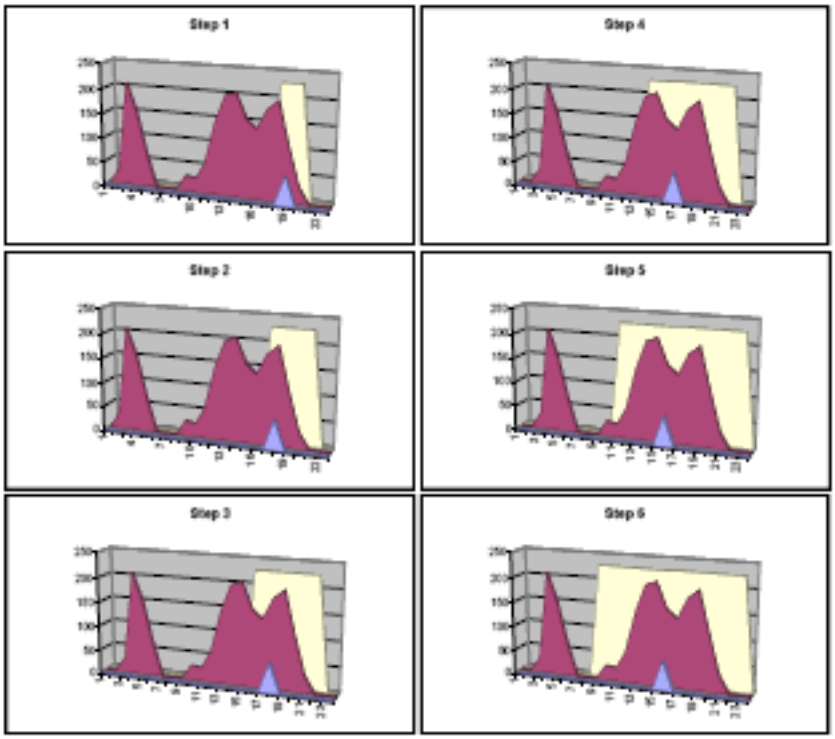
ein Gesicht und der schmalere rote Bereich eine Hand in der Verteilung darstellen. Der gelbe Bereich ist das CAMShift-Suchfenster. Der blaue Bereich ist das Zentrum des Suchfensters. Zu sehen ist, dass der CAMShift-Algorithmus das nächste Maximum wiederfindet. Sofern sich von Bild zu Bild das zu trackende Objekt nicht sehr weit (nicht weiter als die Hälfte der Objektgröße) verschiebt, wird dieses Maximum in der Verteilung wiedergefunden. Andere Maxima, wie hier z. B. die Hand, beeinträchtigen den Algorithmus dabei nicht. Abbildung 3.12(a) zeigt CAMShift zu Beginn, beim ersten Durchlauf, zur Bestimmung einer Suchfenstergröße. In Abbildung 3.12(b) ist zu sehen, wie sich der Algorithmus von Bild zu Bild verhält. Hierbei sieht man, dass sich die Verteilung nach links verschoben hat und sich die Form etwas geändert hat. (im Vergleich zu Abbildung 3.12(a)) Weiterhin ist zu sehen, dass das Suchfenster an der alten Position initialisiert wird. In einer Iteration verschiebt der CAMShift-Algorithmus das Suchfenster auf die richtige Position.

Vor- und Nachteile

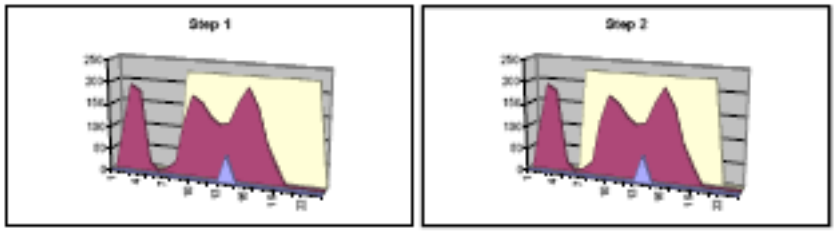
Die Vorteile des CAMShift-Algorithmus sind seine Robustheit gegenüber Störungen im Bild, seine Performanz hinsichtlich Geschwindigkeit sowie die Möglichkeit, auf Basis einer histogrammbasierten Wahrscheinlichkeitsverteilung das Objekt zu verfolgen. Dies macht ein rechenintensives, komplexes Objektmodell überflüssig und ermöglicht die Initialisierung eines Objektes auch ohne vorherige Segmentierung des Bildes: Die Ergebniskoordinaten des Gesichtsdetektors können direkt für die Initialisierung eines Trackers verwendet werde, ohne Wissen über die genauen Objektgrenzen. Des Weiteren ermöglicht die Eigenschaft des zu trackenden Objekts zum Zeitpunkt der Initialisierung trotz der Ungenauigkeiten der Detektorresultate bzgl. der Objektgrenzen eine robuste Initialisierung: Die Erstellung des Objektmodells, das Farbhistogramm des detektierten Gesichtes, erfolgt auf Basis eines inneren Ausschnitts des vom Detektor gelieferten Bildausschnitts. Dieses ist aufgrund der Beschaffenheit des menschlichen Gesichtes repräsentativ für das gesamte Gesicht, also für das zu trackende Objekt.

Die Vorteile des statischen Farbhistogramms auf Basis des H-Kanals des in den HSV-Raum konvertierten Bildes sind die Einfachheit des resultierenden Objektmodells, der daraus resultierende geringe Rechenaufwand in den verschiedenen Verarbeitungsschritten des Algorithmus, sowie eine Robustheit des Objektmodells bezüglich Beleuchtungsänderungen. Es hat sich gezeigt, dass insbesondere die H-Werte menschlicher Hauttöne nur wenig von Beleuchtungsänderungen beeinflusst sind.

Erste Experimente innerhalb des vorgegebenen Szenarios lieferten jedoch nur unzureichende Ergebnisse. Während die Performanz des Trackers hinsichtlich der Geschwindigkeit den Anforderungen entsprach, zeigten sich zwei Probleme bzgl. der Trackingqualität. Es stellte sich zum einen heraus, dass das Auflösungsvermögen des verwendeten H-Histogramms bei der Initialisierung eines Objektes vor ähnlich farbenem Hintergrund nicht ausreichend ist, und auch ein detaillierteres Farbhistogramm nicht die geforderte Robustheit liefert. Das zweite Problem liegt in den Eigenschaften des verwendeten statischen Objektmodells begründet. Da Objektveränderungen nicht erfasst werden, konnten Personen nicht fortlaufend getrackt werden, wenn diese sich von der Kamera wegdrehten.



(a) Suchfensteranpassung des CAMshift-Algorithmus zu Beginn des Algorithmus (aus [Bra98])



(b) Verschiebung und Anpassung des Suchfensters beim Tracken von Bild zu Bild (aus [Bra98])

Abbildung 3.11: Anpassung des CAMShift-Suchfensters über die Zeit. Das erste Teilbild zeigt, wie das Suchfenster zu Beginn des Algorithmus angepasst wird. Das zweite Teilbild zeigt, wie das Suchfenster von Bild zu Bild verschoben und angepasst wird.

Die Fähigkeit, solche Situationen erfolgreich zu bewältigen, ist jedoch eine der Hauptanforderungen an den Tracker in diesem Projekt.

Aufgrund der beschriebenen vorteilhaften Eigenschaften, entschieden wir uns, an dem Einsatz des Camshift-Algorithmus festzuhalten. Um den aufgetretenen Problemen zu begegnen, erweiterten wir das verwendete Objektmodell.

3.4.2 Erweiterung des Objektmodells

Um die an das Tracking-Modul gestellten Anforderungen zu erfüllen, erweitern wir das Objektmodell – das statische H-Histogramm – um eine weitere, dynamische Komponente. Gleichzeitg führen wir eine adaptive Gewichtung der einzelnen Komponenten des Objektmodells ein und erreichen auf diese Weise eine Kombination der jeweils positiven Eigenschaften der beiden Komponenten.

Dynamisches HS-Histogramm

Das bisherige Objektmodell besteht aus einem eindimensionalen, statischen H-Histogramm des detektierten Gesichts. Dieses Modell wird nun um ein zweidimensionales Farbhistogramm auf Basis des H- und S-Kanals des HSV-Farbraums erweitert. Die erhöhte Auflösungseigenschaft des mehrdimensionalen Histogramms ermöglicht die Behandlung schwieriger Situationen, in denen auf Basis des H-Histogramms der Hintergund nur unzureichend vom Objekt zu unterscheiden ist. Insbesondere während der Initialisierung des Trackers, basierend auf der ungenauen Objektbeschreibung des Detektor-Moduls, führt die erhöhte Auflösungseigenschaft des zweidimensionalen Histogramms zu wesentlich robusteren Ergebnissen. Das dynamische HS-Histogramm wird zudem über die Zeit kontinuierlich an die Veränderungen des Objektes angepasst. Auf diese Weise ist es möglich, Veränderungen des Objektes – beispielsweise Drehungen des Gesichts weg von der Kamera – im Objektmodell zu repräsentieren. Mittels des dynamischen HS-Histogramms wird parallel zur Wahrscheinlichkeitsprojektion auf Basis des statischen H-Histogramms eine zweite Projektion berechnet.

Die Anpassung des Histogramms erfolgt stets basierend auf der angenommenen Position des zu trackenden Objekts. Der damit verbundenen Unsicherheit begegnet das erweiterte Objektmodell wie folgt: Einerseits erfolgt die Anpassung auf Basis der Tracking-Ergebnisse einer vogegebenen Zeitspanne und nicht abrupt von Bild zu Bild. Sofern das Objekt während der Verfolgung nicht verloren wird, können Ungenauigkeiten der Trackingergebnisse auf diese Weise kompensiert werden. Andererseits wird das statische H-Histogramm nicht verworfen, sondern fortwährend berücksichtigt. Dies führt zu einer Stabilisierung der Trackingergebnisse, sobald ein Teil des Gesichtes im Bild sichtbar ist oder sichtbar wird.

Adaptive Gewichtung der Wahrscheinlichkeitsverteilungen

Die Kombination der beiden Komponenten des Objektmodells erfolgt adaptiv. Im ersten Tracking-Schritt nach der Initialisierung des Objekts werden beide Komponenten in gleichem Maße zur Objektverfolgung verwendet. Am Ende eines jeden Tracking-Schritts

wird eine Gewichtung der beiden Wahrscheinlichkeitsverteilungen des jeweiligen Histogramms für den nächsten Schritt berechnet. Diese Gewichtung ist abhängig von der Ausprägung des Objekts in der jeweiligen Projektion: Je besser ein Histogramm das Objekt vom Hintergrund und anderen Objekten in direkter Nähe trennt, desto stärker wird dieses bei der Kombination der beiden Wahrscheinlichkeitsverteilungen gewichtet. Der statischen Komponente, die in diesem Fall die Hautfarbe des Objektes repräsentiert, wird dabei ein definiertes Mindestgewicht zugewiesen, um die beschriebene Stabilisierung des dynamischen Modells zu erreichen. Die Berechnung der für das Tracking genutzen Wahrscheinlichkeitsverteilung aus den Projektionen des H- und HS-Histogramms ergibt sich wie folgt:

Es sei

$$X = \{1, 2, \ldots, x_{max}\}, \qquad x_{max} : \text{Bildauflösung horizontal}$$
$$Y = \{1, 2, \ldots, y_{max}\}, \qquad y_{max} : \text{Bildauflösung vertikal}$$

$$\text{Obj}(x, y) = \begin{cases} 1 & , \text{ wenn } (x, y) \text{ zum Objekt gehört} \\ 0 & , \text{ sonst} \end{cases}$$

$$\text{Env}(x, y) = \begin{cases} 1 & , \text{ wenn } (x, y) \text{ zur definierten Umgebung des Objekts gehört} \\ 0 & , \text{ sonst} \end{cases}$$

Die Qualität Q der Projektion zu einem Histogramm $Hist$ bzgl. der Repräsentiertheit des Objektes in der direkten Umgebung berechnet sich wie folgt:

$$Q_{Hist} = \frac{\sum_X \sum_Y (\text{Obj}(x, y) \cdot \text{Proj}(x, y))}{\sum_X \sum_Y ((\text{Obj}(x, y) + \text{Env}(x, y)) \cdot \text{Proj}(x, y))}$$

Dabei stellt Proj(x, y) den Wert der Wahrscheinlichkeitsprojektion zum Histogramm $Hist$ am Bildpunkt (x, y) dar. Damit ergibt sich die Gewichtung W der einzelnen Projektionen des H- und HS-Histogramm wie folgt:

$$W_H = \max\left(\text{Mindestgewicht}, \frac{Q_H}{Q_H + Q_{HS}}\right)$$
$$W_{HS} = (1 - W_H)$$

3.4.3 Erweiterter Algorithmus

Das Ergebnis dieser Erweiterungen ist ein histogrammbasierter Tracker, der den Anforderungen dieses Projektes entspricht. Personen werden anhand ihrer Gesichter zuverlässig getrackt, Beleuchtungsänderungen in zu erwartendem Maße sind unproblematisch und

Objektänderungen in Form von Kopfdrehungen führen nicht zu einem Abbruch der Objektverfolgung. Der erhöhte Rechenaufwand durch die Hinzunahme eines dynamischen Farbhistogramms und die dadurch notwendige gewichtete Verrechnung der Wahrscheinlichkeitsprojektionen führen zu keiner Beeinträchtigung hinsichtlich der Geschwindigkeit des Tracker-Moduls. Der Mehraufwand beschränkt sich auf einen zusätzlichen Vor- und Nachbearbeitungsschritt. Der Rechenaufwand für die iterative CAMShift-Prozedur erhöht sich nicht.

Ein positiver Nebeneffekt der beschriebenen Erweiterung ist zudem die Tatsache, dass die zusätzlichen Berechnungen ein zuverlässiges Abbruchkriterium für die Objektverfolgung liefern. Wird ein Objekt verloren, so führt dies zu einer starken Veränderung des dynamischen Histogramms und einer resultierenden geringen Bewertung dieser Komponente. Gleichzeitig erhält auch das statische Histogramm bei falscher Positionsannahme des Objektes eine niedrige Bewertung. In Kombination bieten diese beiden Indikatoren eine zuverlässige Abbruchbedingung. Eine Unterscheidung verschiedener, ähnlicher Objekte ist auf Ebene der Merkmale innerhalb des Trackers nicht möglich. Die Behandlung solcher Fälle ist einer übergeordneten Instanz, im Falle von ViPer dem Weltmodell überlassen.

4 Identifikation

Der letzte Schritt zu einem erfolgreichen videobasierten Detektions- und Identifikationssystem stellt die eigentliche Identifikation, also die eindeutige Zuordnung eines Bildes zu einer Person, dar. Sie ist nach der Detektion und dem Tracking der dritte Hauptbestandteil dieser Projektgruppe und wird in diesem Kapitel erläutert.

Wir haben uns für die bildbasierte Gesichtsidentifizierung entschieden, obwohl sie nach heutigem Stand der Technik anderen biometrischen Verfahren, wie dem Vergleich des Fingerabdrucks oder dem Irisscan, im Bezug auf die Zuverlässigkeit noch unterlegen ist. Sie hat den Eingangs erwähnten Vorteil, dass es den Anwender keinen bestimmten Restriktionen unterwirft. Neben den immer schneller werdenden Prozessoren, die den Anforderungen komplexer Identifizierungsalgorithmen allmählich gerecht werden, ist das mit ein Grund, warum auf diesem Gebiet seit einigen Jahren rege geforscht wird.

4.1 Problembeschreibung

Nach Zaho et al.[ZCPR03] kann das Gesichtsidentifizierungsproblem wie folgt beschrieben werden: Gegeben ist ein Video oder ein Bild einer Szene. Mit Hilfe einer Datenbank, die Gesichter gespeichert hat, sollen eine oder mehrere Personen in dieser Szene identifiziert bzw. verifiziert (letzteres beinhaltet die Möglichkeit einer unbekannten Person) werden. Im Hinblick auf das Gesamtsystem wird im Folgenden davon ausgegangen, dass in vorangegangen Arbeitschritten die Szene bereits reduziert wurde und als Eingabe nur noch Ausschnitte vorliegen, die Gesichter zeigen und möglichst wenig Hintergrundinformationen enthalten. Wie zutreffend diese Annahme ist, hängt zum einen von der Qualität des Detektionsverfahrens und zum anderen von den Auswahlkriterien, anhand derer entschieden wird welche Bilderausschnitte an den Identifizierer weitergeleitet werden, ab. Es besteht also eine gewisse Restwahrscheinlichkeit, dass den Identifizierer Bildausschnitte erreichen, die keine (vollständigen) Gesichter enthalten. Der Identifizierungsalgorithmus sollte solche Eingaben in einem frühen Arbeitsschritt erkennen und zurückweisen können.

Einige Gegebenheiten der Szene, vor allem Variationen in der Beleuchtung und der Orientierung des Gesichts zur Kamera, erschweren die Identifizierung. In der Regel geht deshalb dem eigentlichen Identifizierungsverfahren ein Normalisierungsprozess voraus, der versucht diese erschwerenden Faktoren auszugleichen. Dieses Thema wird in Abschnitt 4.3 behandelt.

4.2 Methodenübersicht

Identifizierungsverfahren kann man grob in zwei Klassen unterteilen: in merkmalsbasierte und holistische Verfahren.

Merkmalsbasierte Verfahren suchen in dem Gesichtsbild nach vorher festgelegten charakteristischen Punkten, wie Augen, Nasenspitze, Mundwinkel usw. und extrahieren an diesen Stellen jeweils ein Merkmal. Die Gesamtheit der extrahierten Merkmale bilden einen Merkmalsvektor, der das Gesicht repräsentiert. Um ein Gesicht schließlich zu identifizieren, wird sein Merkmalsvektor mit den Merkmalsvektoren bekannter Personen verglichen. Merkmalsbasierte Verfahren nutzen also gezielt Domänenwissen über das menschliche Gesicht und werden deshalb auch wissensbasierte Verfahren genannt. Ein merkmalsbasierter Ansatz, der vor allem in den frühen Anfängen der Mustererkennung verfolgt wurde, ist die Geometrische Analyse [BP93]. Bei diesem Verfahren werden mit Hilfe von Gradientenoperatoren Kantenbilder generiert. Aus den Kantenbildern werden Histogramme erstellt, deren lokale Extrema Aufschluss über die Lage der charakteristischen Punkte geben sollen. Aus diesen Daten werden dann Merkmale wie Höhe und Breite des Mundes und der Nase sowie Position und Abstand der Augen gewonnen. Das Gesicht wird also geometrisch vermessen. Dieses Verfahren hat einige schwerwiegende Schwächen. Zunächst einmal müssen alle Merkmale klar erkennbar sein, was ein Bild mit sehr hoher Auflösung erforderlich macht. Außerdem müssen, damit die geometrischen Daten aussagekräftig sind, alle Gesichter möglichst genau aus dem selben Winkel aufgenommen werden. Beides ist in einem Szenario mit realistischen Bedingungen nicht möglich. Hinzu kommt noch, dass ein Merkmalsvektor nur sehr wenige Einträge (i. d. R. weniger als hundert) enthält und somit nicht sehr zuverlässig ist. Die Unzulänglichkeiten der Geometrischen Analyse zeigen sehr schön, was ein moderner merkmalsbasierter Algorithmus besser machen muss: Die Merkmale müssen auch bei einer niedrigen Auflösung sicher lokalisiert werden können. Dies sollte auch bei (zumindest kleinen) Variationen des Aufnahmewinkels möglich sein. Außerdem muss der Informationsgehalt, der aus einem Merkmal gewonnen wird, ausreichend hoch sein. Das *Elastic Bunch Graph Matching (EBGM)* ist ein Verfahren, das diese Anforderungen erfüllt. Es wird in Abschnitt 4.2.2 vorgestellt.

Holistische Verfahren greifen nicht auf Domänenwissen zurück, sondern betrachten das Gesichtsbild als ganzheitlichen Informationsvektor. Als ein einfacher Ansatz wäre hier zunächst das Holistic Template Matching zu nennen [BJ81]. Hierbei werden jeweils die von den bekannten Personen erstellten Informationsvektoren wie Schablonen über das zu verifizierende Gesichtbild gelegt und anhand von Abweichungs- oder Korrelationsberechnungen die Ähnlichkeit bestimmt. Der Ähnlichkeitswert wird maximiert indem die Schablone innerhalb eines bestimmten Suchbereiches verschoben wird. Dieser naive Ansatz hat ebenfalls einige Schwächen. Zunächst einmal muss der Aufnahmewinkel des zu identifizierenden Bildes möglichst genau mit dem Winkel, in dem die Schablonen aufgenommen wurden, übereinstimmen. Weiterhin ist das Verfahren anfällig gegen Variationen der Lichtverhältnisse, da die Ähnlichkeitswerte mit Änderung der Beleuchtung starken Schwankungen unterliegen. Außerdem wird bei dem Vergleich der Informationsvektoren nicht zwischen der Aussagekraft der einzelnen Vektorelemente im informationstheoretischem Sinne differenziert. Beispielsweise haben Bildausschnitte der Wange oder der Stirn einen relativ hohen Redundanzanteil und variieren von Person zu Person nicht sonderlich stark. Trotzdem werden sie bei der Berechnung der Ähnlichkeit mit gleicher Gewichtung und gleichem Rechenaufwand einbezogen wie die viel aussagekräf-

tigere Augenpartie. Die Idee, die Dimension des Informationsvektors zu reduzieren, und das bei möglichst geringem Verlust an Informationen, die für das individuelle Gesicht bestimmend sind, führt zur Unterraumanalyse [SM04]. Hier ist vor allem die *Hauptachsentransformation* (Principal Component Analysis, PCA) zu nennen [MP01], welche beim *Eigenface*-Verfahren zum Einsatz kommt.

Ansätze, die versuchen zwei (oder mehr) Identifizierungsverfahren zu kombinieren, nennt man hybride Verfahren. Mit einer geschickten Verknüpfung zweier Verfahren ist es unter Umständen möglich, bessere Ergebnisse zu erzielen, als mit den jeweiligen Verfahren für sich betrachtet. Beispielsweise wird in [WMR01] versucht, mit einem merkmalsbasiertem Ansatz zunächst die Rotation des Gesichtes zu normalisieren. Die eigentliche Identifizierung erfolgt anschließend mit einer holistischen Methode.

4.2.1 Eigenface-Verfahren

Beim *Eigenface*-Verfahren [TP91] werden Gesichtsbilder als hochdimensionale Vektoren aufgefasst. Die Dimension eines solchen Vektors ergibt sich bei $n \times n$ großen Bildern durch Aneinanderreihung aller Spalten zu n^2.

Gesichtsbilder nehmen nicht den gesamten zur Verfügung stehenden n^2-dimensionalen Raum ein, sondern sammeln sich in einem kleineren Unterraum. Dies liegt in der relativen Ähnlichkeit von Gesichtsbildern, im Gegensatz zu zufälligen Bildkonfigurationen begründet. Diesen Umstand nutzt das Eigenface-Verfahren zur Dimensionsreduktion mittels Hauptachsentransformation aus. Dieser so gefundene Unterraum wird auch Gesichtsraum genannt und wird von den m' größten Eigenfaces, den Eigenvektoren der Trainingsgesichter-Menge, aufgespannt.

Durch Projektion des Gesichtsbilds einer bekannten Person auf den so berechneten Gesichtsunterraum kann die m'-dimensionale Koordinate des Bildes in einer Datenbank zum späteren Vergleich abgelegt werden. Die Identifikation einer unbekannten Person erfolgt nun durch Projektion des Gesichtsbilds auf den Gesichtsunterraum und anschließender euklidischer Abstandsberechnung zu allen bekannten Gesichtern.

Hauptachsentransformation

Eine Hauptachsentranformation findet die Hauptausrichtungen und -ausdehnungen einer Datenmenge durch Bestimmung der Eigenvektoren respektive Eigenwerte der Kovarianzmatrix der Datenmenge.

Für die Hauptachsentransformation muss der Ursprung des zu transformierenden Koordinatensystems in den Mittelpunkt der Trainingsbilder verschoben werden. Dies geschieht durch Subtraktion des Durchschnittsgesichts $\Psi = \frac{1}{m}\sum_{i=1}^{m}\Gamma_i$ von jedem Trainingsbild $\Phi_i = \Gamma_i - \Psi$. Dann sind Φ_i die zentrierten Trainingsbilder mit denen die eigentliche Hauptachsentransformation durchgeführt wird.

Die Kovarianzmatrix C, deren Eigenvektoren bestimmt werden müssen, ist

$$C = \frac{1}{m}\sum_{i=1}^{m}\Phi_i\Phi_i^T = AA^T$$

wobei $A = [\Phi_1 \Phi_2 \dots \Phi_m]$ aus allen zentrierten Gesichtsbildern besteht. Die Berechnung der Eigenvektoren und Eigenwerte erfolgt durch Lösung von

$$C\vec{u}_i = \nu_i \vec{u}_i$$

Die Berechnung der Eigenvektoren $\vec{u}_i$ der Kovarianzmatrix wäre allerdings auf diese Art und Weise viel zu aufwändig ($\dim(C) = n^2$) . Da die Anzahl der Trainingsbilder im Normalfall viel geringer ist als die Dimension des gesamten Bildraumes ($m \ll n^2$) und dadurch nur m wesentliche Eigenvektoren existieren, kann der Rechenaufwand verringert werden. Fragt man nämlich nach den Eigenvektoren $\vec{v}_i$ von $A^T A$ ist

$$A^T A\vec{v}_i = \mu_i \vec{v}_i$$

zu lösen. Durch Multiplikation beider Seiten mit A erhält man

$$AA^T A\vec{v}_i = \mu_i A\vec{v}_i$$

wodurch man die Eigenvektoren $\vec{u}_i$ von C durch

$$A\vec{v}_i = \sum_{k=1}^{m} \vec{v}_{ik}\Phi_k, \quad i = 1, \dots, m$$

berechnen kann. Auf diesem Wege sind nur die Eigenvektoren einer m-dimensionalen statt einer n^2-dimensionalen Matrix zu berechnen.

Um die Dimension des Gesichtsraums weiter zu verkleinern, kann man die Eigenwerte nutzen. Diese geben die Größe der Varianz und damit den Informationsgehalt in Richtung des zugehörigen Eigenface an. Aufgrund dessen werden nur die $m' < m$ Eigenfaces mit den m' größten Eigenwerten zur Identifikation benutzt.

Identifikation

Zur Identifizierung einer unbekannten Person wird ihr Gesichtsbild Γ in den m'-dimensionalen Gesichtsraum projiziert.

$$\omega_k = \vec{u}_k^T(\Gamma - \Psi), \quad k = 1, \dots, m'$$

Die Koordinate Ω von Γ bezüglich des Gesichtsraums ergibt sich durch alle Eigenface-Gewichte $\Omega_T = [\omega_1, \omega_2, \dots, \omega_{m'}]$. Um der unbekannten Person einen Namen zuzuordnen wird der euklidische Abstand

$$\varepsilon_k^2 = |\Omega - \Omega_k|^2$$

zu allen bekannten Gesichtsvektoren Ω_k berechnet. Die Person gilt als erkannt, wenn mindestens einer dieser Abstände unter einer vorher festgelegten Fehlerschranke liegt.

Die Identifikationsgüte, die man mit Eigenfaces erlangen kann ist abhängig von der Qualität der zu identifizierenden Bilder. Diese müssen die gleiche Position im Bildausschnitt sowie ähnliche Pose und Beleuchtungscharakteristik wie die bekannten Gesichter aufweisen, damit die Koordinate Ω der Gesichtsraum-Projektion nah genug an ein bekanntes Ω_k kommt. Bereits kleine Veränderungen in einem Bild können eine Identifikation unmöglich machen. Daher ist eine Normalisierung aller Gesichtsbilder vor der Durchführung oben genannter Schritte unumgänglich. Die in ViPer genutzte Eigenface-Normalisierung ist in Abschnitt 4.3 beschrieben.

4.2.2 Elastic Bunch Graph Matching

Das *Elastic Bunch Graph Matching* (EBGM) ist ein merkmalsbasiertes Identifikationsverfahren [WFKM96]. Die Merkmalspunkte bilden dabei einen *Gesichtsgraphen*, der aus einem Trainingsgraphen (*Bunchgraphen*) durch Verschiebung, Verzerrung und Drehung hervorgeht, so dass der Gesichtsgraph die zuvor trainierten Merkmale repräsentiert. Der für diesen Schritt verwendete Algorithmus wird im Abschnitt »Lokalisierung« beschrieben.

Ein Merkmal wird bei diesem Verfahren durch eine Menge von Wavelet-Komponenten, einem sogenannten *Jet*, beschrieben, die das um ein Merkmal herum lokalisierte Spektrum darstellen. Diese Jets können durch Ähnlichkeitsmetriken zu einer Gesamtähnlichkeit zweier Graphen zusammengefasst werden, womit eine Identifizierung möglich wird.

Im Folgenden wird die Berechnung der Jets, die Lokalisierung der Merkmale in einem Gesicht und das Vorgehen bei der Identifikation einer Person detailliert erläutert. Abschließend wird die semi-automatische Erzeugung des während der Lokalisierung benötigten Bunchgraphen beschrieben.

Merkmalsextraktion

Die beim EBGM verwendeten Merkmale werden durch lokalisierte Spektren repräsentiert. Diese erhält man durch eine Wavelet-Transformation, welche folgende Form hat

$$\mathcal{J}_j(\vec{x}) = \int \mathcal{I}(\vec{x}')\mathcal{W}_j(\vec{x} - \vec{x}')\mathrm{d}^2\vec{x}'$$

Dabei ist $\mathcal{I}(\vec{x})$ jeweils ein Grauwert des Bildes und $\mathcal{W}_j$ die j-te Gabor-Wavelet-Konfiguration mit der dieser Punkt gefaltet wird. Um das Spektrum ausreichend genau beschreiben zu können muss obiger Schritt mit einer Reihe verschiedener Wavelets durchgeführt werden. Dies motiviert den Begriff des *Jets*. Ein Jet $\mathcal{J}$ besteht aus allen komplexen Faltungskoeffizienten $\mathcal{J} = \{\mathcal{J}_j(\vec{x})\}$. Die verwendeten Gabor-Wavelets $\mathcal{W}_j$ werden dabei so gewählt, dass sie den gewünschten Bereich um einen Gesichtspunkt in Größe und Ausrichtung gleichmäßig abdecken.

Gabor-Wavelets

Gabor-Wavelets sind biologisch motiviert [Dau88] und gehen aus einer ebenen Welle hervor, welche durch ein Gauß-Fenster örtlich begrenzt wird. Der von uns implementierte EBGM-Algorithmus bedient sich der Gabor-Wavelet-Darstellung aus [Bol03].

$$\begin{aligned}\mathcal{W}(\vec{x}, \theta, \lambda, \varphi, \sigma, \gamma) &= \exp\left(-\frac{x_1'^2 + \gamma^2 x_2'^2}{2\sigma^2}\right)\exp\left(i\left(2\pi\frac{x_1'}{\lambda} + \varphi\right)\right)\\ x_1' &= x_1\cos(\theta) + x_2\sin(\theta)\\ x_2' &= -x_1\sin(\theta) + x_2\cos(\theta)\end{aligned}$$

Die Parameter steuern folgende Eigenschaften des Wavelets:

θ steuert die Rotation des Wavelets. Dieser Parameter sollte zwischen 0 und π liegen, da größere sowie kleinere Werte redundant sind aufgrund der Symmetrie der Wavelets.

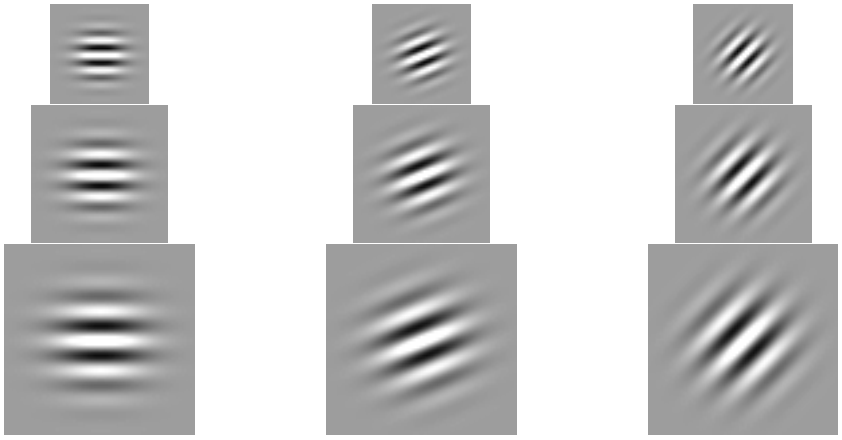


Abbildung 4.1: Auswahl von 9 Gabor-Wavelets aus dem Bolme Parameter-Set für drei verschiedene Frequenzen λ und drei verschiedene Winkeln θ

λ gibt die Frequenz des im Wavelet verwendeten Sinusoids an.

φ bestimmt die Phase des Wavelets. Dabei sollte $\varphi \in [0, \frac{\pi}{2}]$.

σ ist der Radius des begrenzenden Gauß-Fensters. Er bestimmt also den Bereich des Bildes der auf das Faltungsergebnis Einfluss hat. Dieser Parameter ist normalerweise proportional zur Frequenz des Wavelets $\sigma = c\lambda$, so dass Wavelets unterschiedlicher Größe und Frequenz skalierte Versionen voneinander sind.

γ steuert das Seitenverhältnis des begrenzenden Gauß-Fensters. Das von uns benutzte Parameter-Set macht von diesem Parameter keinen Gebrauch, d. h. $\gamma = 1$.

Wir verwenden das von Bolme [Bol03] vorgeschlagene Parameter-Set, welches acht Ausrichtungen, fünf Frequenzen und zwei Phasen nutzt um auf insgesamt 80 Wavelet-Filterkerne zu kommen. Dies ergibt pro Jet 40 komplexe Koeffizienten.

Das Bolme Parameter-Set
Das Bolme Parameter-Set verwendet folgende Werte für die einzelnen Parameter:

$\theta \in \left\{0, \frac{\pi}{8}, \frac{2\pi}{8}, \frac{3\pi}{8}, \frac{4\pi}{8}, \frac{5\pi}{8}, \frac{6\pi}{8}, \frac{7\pi}{8}\right\}$

$\lambda \in \left\{0, 4\sqrt{2}, 8, 8\sqrt{2}, 16\right\}$

$\varphi \in \{0, \frac{\pi}{2}\}$

$\sigma = \lambda$

$\gamma = 1$

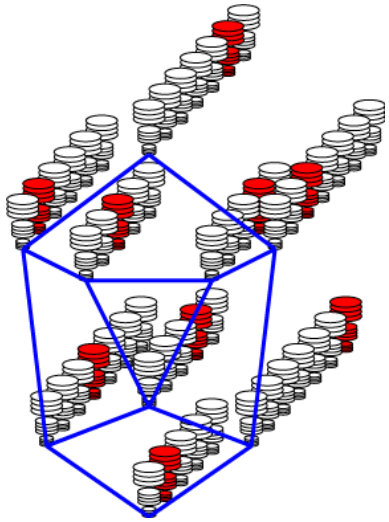


Abbildung 4.2: Der Bunchgraph repräsentiert Informationen aus verschiedenen Gesichtern.

Graphendarstellung

Wie Eingangs erwähnt, wird beim EBGM-Verfahren ein Gesicht mittels eines Graphen $\mathcal{G}(N, E)$ repräsentiert. Die Knoten bezeichnen dabei die Merkmale. Entsprechend gehören zu jedem Knoten die Koordinate und der Jet eines Merkmals – siehe Abbildung 4.3. Die Kanten sind jeweils mit dem euklidischen Abstand ihrer Knoten beschriftet.

Eine wichtige Datenstruktur, die für die Lokalisierung der Merkmale eingesetzt wird, ist der *Bunchgraph* $\mathcal{G}^{\mathcal{B}m}$ (Abb. 4.2). Jeder Merkmalsknoten des Bunchgraphen referenziert eine Liste mit mehreren Jets, die von verschiedenen Gesichtsbildern im Rahmen eines Trainings extrahiert wurden. Diese Listen werden *Bunches* genannt und decken für das jeweilige Merkmal ein ausreichend großes Spektrum verschiedener Merkmalsausprägungen ab. Will man schätzen, wie nah sich ein extrahierter Jet an der tatsächlichen Position des gesuchten Merkmals befindet, wird der ähnlichste Jet aus dem entsprechenden Bunchen gesucht und die Distanz zu diesem geschätzt. Der ähnlichste Jet aus einem Bunch wird entsprechend als *lokaler Experte* bezeichnet. Die Koordinaten und Abstände des Bunchgraphen sind geometrische Mittel und repräsentieren damit die durchschnittliche Graphengeometrie.

Aus dem Bunchgraph kann der *Durchschnittsgraph* berechnet werden. Hierfür wird aus jedem Bunch ein durchschnittlicher Jet gebildet, indem die arithmetischen Mittel über die Koeffizienten der Jets berechnet werden.

Lokalisierung

Beim Algorithmus zur Merkmalslokalisierung wird ein vorinitialisierter Gesichtsgraph auf das zu verifizierende Gesichtsbild gelegt. Dieser Graph wird als *Bildgraph* bezeichnet und unter Beschränkung der Freiheitsgrade bewegt oder skaliert, um die Koordinaten

der Knoten an die tatsächlichen Koordinaten der Merkmale im Gesichtsbild anzupassen. Für die Initialisierung des Bildgraphen werden die Durchschnittskoordinaten des Bunchgraphen genommen. Entsprechend dieser Koordinaten wird aus dem Gesichtsbild für jeden Knoten ein Jet extrahiert. Die Anpassung der Koordinaten erfolgt gemäß Algorithmus 4.1.

1. Approximieren der Gesichtsposition
2. Anpassung der Position und Größe
3. Anpassung des Seitenverhältnisses
4. Lokale Verformung

Algorithmus 4.1: Schritte der Merkmalslokalisierung des EBGM-Verfahrens

Im Folgenden werden die einzelnen Schritte detailliert erläutert.

Approximieren der Gesichtsposition
Um die Position des Gesichtes grob zu approximieren, wird der Schwerpunkt des Bildgraphen innerhalb eines mittig liegenden Quadrates vorgegebener Größe mit einer festen Schrittweite bewegt. Dabei ändert sich der Abstand der Knoten zueinander nicht. An jeder Position wird ein Satz Jets extrahiert und mit den Jets des Durchschnittsgraphen verglichen. Dazu wird folgende Ähnlichkeitsfunktion benutzt,

$$\mathcal{S}_a\left(\mathcal{J},\mathcal{J}'\right) = \frac{\sum_j a_j a'_j}{\sqrt{\sum_j a_j^2 \sum_j {a'_j}^2}}$$

wobei a_j und a'_j die Amplituden der Koeffizienten der Jets $\mathcal{J}$ sowie $\mathcal{J}'$ bezeichnen.

Bei visuellen Signalen unterliegt die Amplitude nicht so starken Schwankungen wie die Phase und eignet sich deshalb für eine grobe Positionsbestimmung besser. Anschließend werden mit halbierter Schrittweite die benachbarten Positionen um den Punkt mit dem höchsten Ähnlichkeitswert auf gleiche Weise überprüft. Die Position mit dem höchsten Ähnlichkeitswert wird als Ausgangspunkt für den nächsten Schritt benutzt.

Anpassung der Position und Größe
In diesem Schritt werden zwei Skalierungsstufen an jeweils vier Punkten überprüft. Bei der einen Stufe wird der Bildgraph etwas vergrößert, bei der anderen Stufe etwas verkleinert. Die vier Punkte liegen auf den Ecken eines Quadrates mit festgelegter Größe um den Ausgangspunkt. Insgesamt gibt es also acht Kombinationen, die jeweils wie folgt gehandhabt werden: Es wird ein Satz Jets extrahiert. Zu jedem Jet wird aus dem entsprechenden Bunch des Bunchgraphen der lokale Experte bestimmt. Bei der Berechnung der Ähnlichkeiten wird diesmal die Phase berücksichtigt:

$$\mathcal{S}_\phi\left(\mathcal{J},\mathcal{J}'\right) = \frac{\sum_j a_j a'_j \cos\left(\phi_j - \phi'_j - \vec{d}\,\vec{k}_j\right)}{\sqrt{\sum_j a_j^2 \sum_j {a'_j}^2}}$$

Die Entfernung $\vec{d}$ der Jets zu ihren jeweiligen lokalen Experten wird nach folgender Vorschrift geschätzt:

$$\vec{d} = \frac{1}{\Gamma_{xx}\Gamma_{yy} - \Gamma_{xy}\Gamma_{yx}} \times \begin{pmatrix} \Gamma_{yy} & -\Gamma_{yx} \\ -\Gamma_{xy} & \Gamma_{xx} \end{pmatrix} \begin{pmatrix} \Phi_x \\ \Phi_y \end{pmatrix}$$

$$\Phi_x = \sum_j a_j a'_j k_{jx} \left(\phi_j - \phi'_j\right)$$
$$\Gamma_{xy} = \sum_j a_j a'_j k_{jx} k_{jy}$$

Dabei sind Φ_y, Γ_{xx}, Γ_{yx}, Γ_{yy} analog definiert.

Der Bildgraph wird dann so verschoben und skaliert, dass die Summe der Quadrate dieser geschätzten Entfernungen minimal wird. Schließlich wird die Ähnlichkeit des gesamten Bildgraphen zum Bunchgraph wie folgt bestimmt:

$$\mathcal{S}_\mathcal{B}\left(\mathcal{G}^\mathcal{I}, \mathcal{B}\right) = \frac{1}{N} \sum_n \max_m \left(\mathcal{S}_\phi \left(\mathcal{J}_n^\mathcal{I}, \mathcal{J}_n^{\mathcal{B}m}\right)\right) - \frac{\lambda}{E} \sum_e \frac{\left(\Delta\vec{x}_e^\mathcal{I} - \Delta\vec{x}_e^\mathcal{B}\right)^2}{\left(\Delta\vec{x}_e^\mathcal{B}\right)^2}$$

Dabei bezeichnet $\mathcal{B}$ den Bunchgraph, $\mathcal{G}^\mathcal{I}$ den Bildgraph und $\max_m \mathcal{S}_\phi$ die Ähnlichkeit zu den jeweiligen lokalen Experten. Der zweite Summand lässt die Geometrie der Graphen in die Ähnlichkeitsbetrachtung einfließen. Dies wird erst im vierten Schritt benötigt, womit hier $\lambda = 0$ gesetzt wird. Von den acht Kombinationen wird der Graph mit dem höchsten Ähnlichkeitswert als Ausgangsgraph für den dritten Schritt benutzt.

Anpassung des Seitenverhältnisses
Dieser Schritt funktioniert genau wie Schritt zwei, nur das die Koordinaten getrennt voneinander betrachtet werden. Es wird also abwechselnd nur auf der vertikalen oder auf der horizontalen Achse verschoben und nur eine Koordinate wird zur selben Zeit skaliert. Auf diese Weise wird versucht das Seitenverhältnis anzupassen.

Lokale Verformung
Dieser Schritt betrachtet die einzelnen Knoten getrennt voneinander. In einer pseudozufälligen Reihenfolge werden die einzelnen Knoten wie folgt bearbeitet: Es wird der lokale Experte bestimmt, der Abstand zu diesem geschätzt und der Knoten entsprechend verschoben. An der neuen Position wird ein Jet extrahiert. Anschließend wird die Ähnlichkeit des Gesichtsgraphen zum Bunchgraph erneut berechnet. Da durch die Verschiebung einzelner Knoten die Geometrie des Graphen verzerrt werden kann, wird von den Ähnlichkeitswerten die Abweichungen der Kantenlängen von der Durchschnitssgeometrie abgezogen. Zur Berechnung der Ähnlichkeit unter Betrachtung der Graphengeometrie wird $\mathcal{S}_\mathcal{B}\left(\mathcal{G}^\mathcal{I}, \mathcal{B}\right)$ verwendet, wobei nun $\lambda > 0$ bestimmt, wie stark die Gemoetrie berücksichtigt werden soll.

Der aus den Lokalisierungsschritten resultierende Bildgraph wird als Gesichtsgraph übernommen und repräsentiert nun das zu verifizierende Gesicht.

Identifizierung

Sind die Merkmale des Gesichtbildes erst einmal gefunden, gestaltet sich die Identifizierung einfach. Der lokalisierte Gesichtsgraph $\mathcal{G}_\mathcal{I}$ wird mit den Gesichtsgraphen $\mathcal{G}_\mathcal{M}$ bekannter Personen verglichen. Dabei wird jeweils die Graphenähnlichkeit $\mathcal{S}_\mathcal{G}$ berechnet.

$$\mathcal{S}_\mathcal{G} = \frac{1}{N} \sum_n \mathcal{S}_a \left(\mathcal{J}_n^\mathcal{I}, \mathcal{J}_n^\mathcal{M} \right)$$

Die Person mit dem höchsten Ähnlichkeitswert gilt dann als identifiziert. Um zu vermeiden, dass unbekannte Personen automatisch einer Person aus der Datenbank zugewiesen werden, wird anhand der Graphenähnlichkeiten $\mathcal{S}_\mathcal{G}$ entschieden, ob es eine Person aus der Datenbank ist oder es sich um eine unbekannte Person handelt. Auf diese Weise wird gleichzeitig das Zurückweisen von ansonsten fehlerhaft identifizierten Bildern ermöglicht. Diese weisen nämlich eine vergleichbare Verteilung der Gesichtsähnlichkeiten $\mathcal{S}_\mathcal{G}$ auf, wie die Bilder unbekannter Personen. Konkret wird anhand dreier Kriterien entschieden, ob die Person als bekannt gilt und identifiziert wird oder als unbekannt zu klassifizieren ist. Das erste Kriterium besagt, dass die höchste Graphenähnlichkeit $\mathcal{S'}_\mathcal{G}$ einen bestimmten Mindestwert annehmen muss. Dies stellt sicher, dass nur dann Bilder zur Identifizierung akzeptiert werden, wenn überhaupt eine Person eine hinreichende Ähnlichkeit aufweist. Als zweites wird überprüft, ob der Abstand zwischen der höchsten Graphenähnlichkeit $\mathcal{S'}_\mathcal{G}$ und der zweithöchsten Graphenähnlichkeit $\mathcal{S''}_\mathcal{G}$ einen bestimmten Mindestwert übersteigt. Ist dies nämlich nicht der Fall, so ist keine hinreichende Trennung zwischen den Personen möglich und das Bild muss als unbekannt klassifiziert werden. Um eine noch bessere Trennung der unbekannten von den bekannten Gesichtern zu ermöglichen, kann schließlich eine lineare Trennung anhand der beiden obigen Merkmale vorgenommen werden. Diese Trennung ist motiviert durch die Annahme, dass bei einer ausgesprochen hohen Graphenähnlichkeit $\mathcal{S'}_\mathcal{G}$ zu einer Person die Distanz zur zweiten Person eher unbedeutend ist während bei einer geringen Graphenähnlichkeit $\mathcal{S'}_\mathcal{G}$ ein großer Abstand zur zweitähnlichsten Person eine richtige Identifizierung andeutet.

Training

Ein wichtiger Aspekt des EBGM-Algorithmus ist das Training des Bunchgraphen. Dieses geschieht in einem interaktiven Prozess, also mit manueller Unterstützung, der im Laufe des Trainings immer besser automatisiert wird. Die Merkmalspositionen im Gesicht werden in einem ersten Schritt manuell festgelegt. Dies geschieht in einem Trainingsfenster, in dem ein Gesichtsbild angezeigt wird – siehe Abbildung 4.3. Nachdem der Bunchgraph manuell in das Trainingsfenster gezeichnet wurde, werden die Gaborjets an den entsprechenden Stellen berechnet und der so erzeugte Gesichtsgraph gespeichert. Die Person ist somit durch diesen Gesichtsgraphen in die Datenbank aufgenommen worden. Darüber hinaus stehen die Merkmalspositionen und Gaborjets jedoch auch dem Bunchgraphen zur Verfügung, der somit zu diesem Zeitpunkt aus einem einzigen Jet pro Merkmalsposition besteht. Im nächsten Schritt wird dieser Bunchgraph bereits zur Lokalisierung der Merkmalspositionen des nächsten Bildes benutzt. In der Regel werden

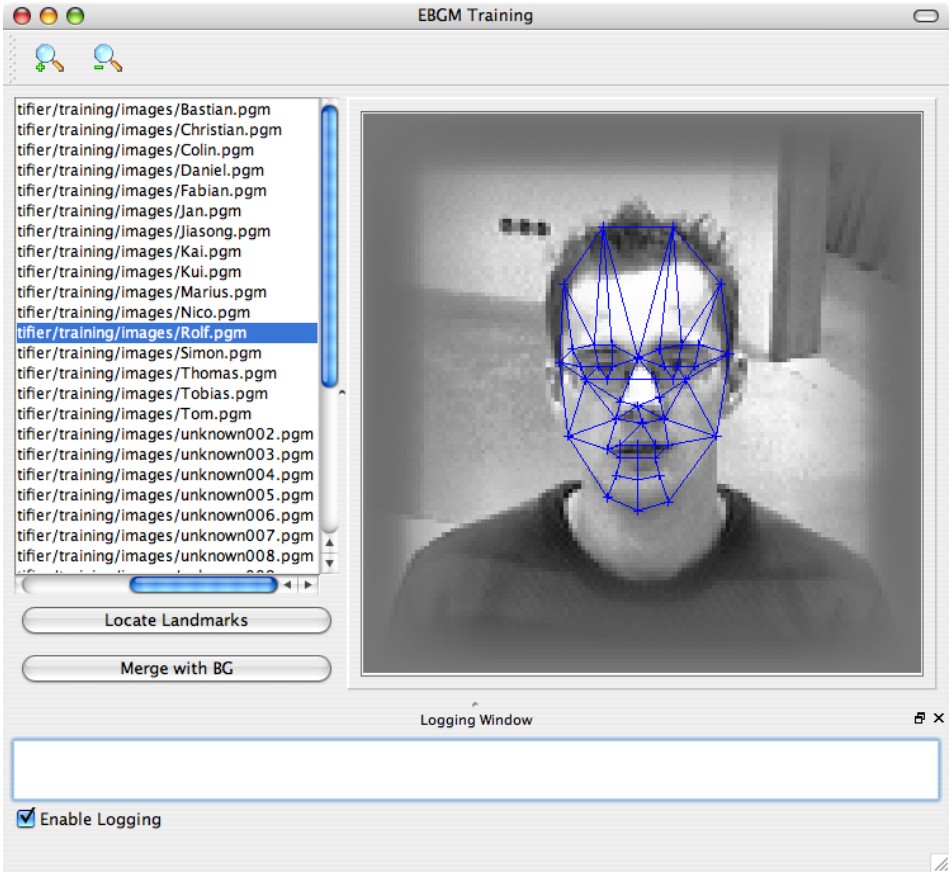


Abbildung 4.3: Das Trainingsfenster mit dem Gesichtsgraphen einer Person. Der Graph kann in dem Fenster manuell angepasst werden bevor er der Datenbank und dem Bunchgraphen hinzugefügt wird

die Positionen noch nicht so gut getroffen, da der Bunchgraph noch keine Variationen in den Merkmalen abdeckt. Durch manuelles Verschieben des Graphen auf die tatsächlichen Positionen der Merkmale wird sichergestellt, dass beim Training die richtigen Jets extrahiert werden. Auch nach diesem Schritt wird der Gesichtsgraph der neuen Person in der Datenbank abgelegt und gleichzeitig der Bunchgraph erweitert. In jedem folgenden Schritt ist der Bunchgraph somit vollständiger und verhilft der Lokalisierung zu immer besseren Ergebnissen. Nach der manuellen Korrektur der Lokalisierung werden die neuen Gesichtsgraphen jeweils in die Datenbank aufgenommen und der Bunchgraph aktualisiert.

Am Ende des Trainingsprozesses steht ein vollständiger Bunchgraph mit etwa 50–70 Jets pro Merkmalsposition. Außerdem wurden die Gesichtsgraphen der Personen erzeugt und in der Datenbank abgelegt. Das Training kann jederzeit unterbrochen und später fortgesetzt werden. Ebenso können nachträglich Personen in die Datenbank eingefügt werden.

4.3 Umsetzung und Erweiterung

Für die Umsetzung des Identifizierers haben wir uns für das Elastic-Bunch-Graph-Matching entschieden. Das Verfahren kann Variationen in Skalierung und Rotation bewältigen und die Wavelet-Merkmale sind robust gegenüber Änderungen der Beleuchtung. Das macht es zu einem geeigneten Verfahren für die Gesichtsidentifizierung unter erschwerten Bedingungen.

Kritisch für die Laufzeit ist die Extraktion der Merkmale, speziell die Faltung mit den Filtermasken. Durch High-Level-Optimierungen, insbesondere durch Loop-Unrolling und Randfallbetrachtungen, konnte die Faltungsfunktion deutlich beschleunigt werden. Zusätzliche Optimierungen für den Prozessor der Zielplattform brachten einen weiteren Geschwindigkeitszuwachs, so dass für ein Echtzeitsystem akzeptable Laufzeiten erreicht werden konnten. Des Weiteren haben wir in das Lokalisierungsverfahren zwischen den einzelnen Schritten Abbruchkriterien festgelegt. Wird ein gewisser Ähnlichkeitsschwellwert nicht erreicht wird die Identifizierung vorzeitig abgebrochen, das Gesichtsbild verworfen und es wird davon ausgegangen, dass der betrachtete Bildausschnitt gar kein (vollständiges) Gesicht enthält und dem Identifizierer fälschlicherweise übergeben wurde. Damit wird zum einen ermöglicht die Fehlerrate des Gesamtsystems zu reduzieren, da fehlerhafte Detektionen nachträglich zurückgewiesen werden können, zum anderen wir die Performanz erhöht, da durch einen frühen Abbruch keine unnötige Rechenzeit für solche Eingaben verbraucht wird.

Um die Qualität der Merkmalslokalisierung zu erhöhen, haben wir zwei Erweiterungen umgesetzt. Die erste Erweiterung erlaubt zusätzliche rotatorische Freiheitsgrade. Es wird versucht die Rotation des Gesichts um drei verschiedene Achsen zu bestimmen. Neben der Rotation in der Bildebene, werden Rotationen um die vertikale und horizontale Achse in die Bildebene hinein mit einbezogen. Für diese räumlichen Rotationen haben wir eine einfache planare Abbildungsfunktion von drei nach zwei Koordinaten benutzt. Die zweite Erweiterung wurde in [WFKM96] vorgeschlagen. Hierbei wird das Lokalisierungsverfahren zwei mal durchgeführt und für jeden Durchlauf wird ein eigener Bunchgraph verwendet. Der erste Durchlauf dient dazu, die Silhouette des Gesichts und Kopfbereiches zu lokalisieren. Entsprechend betrachtet die verwendete Graphenstruktur viele Merkmale im äußeren Gesichtsbereich. Für die Identifizierung sind aber vor allem Merkmale im inneren Gesichtsbereich, also Augen, Nase, Mund usw. entscheidend. Deswegen wird beim zweiten Durchlauf eine Graphenstruktur mit vielen Merkmalen im inneren Gesichtsbereich verwendet. Der Image-Graph kann dabei wesentlich präziser initialisiert werden, da er jetzt an der zuvor bestimmten Silhouette ausgerichtet werden kann.

Normalisierung

Vor der Identifikation ist einen Normalisierung der Bilddaten nötig, um z. B. Beleuchtungs-, Größen- und Drehungsunterschiede auszugleichen. Dabei unterscheiden sich die angewandten Normalisierungsschritte je nach Identifikationsalgorithmus. Es werden nun die Normalisierungsschritte für den EBGM-Algorithmus vorgestellt.

1. Verschieben des durchschnittlichen Pixelwertes:

   Bildpunkte außerhalb des Bildes werden mit dem Pixelwert 0 angenommen. Daher wird der Durchschnittswert aller Pixel auf 0 normiert, um hohe Frequenzen an den Rändern zu vermeiden.

2. Lineares Gewichten der Pixel an den Rändern des Bildes:

   Noch besser werden hohe Frequenzen an den Bildrändern, die unweigerlich starken Einfluss auf die Gaborjets hätten, vermieden, indem die Pixel an den Rändern geringer gewichtet werden. Konkret wird dies über einen 30 Pixel breiten Bereich an den Rändern gemacht, in dem der Pixelwert mit $d/30$ gewichtet wird, wobei d der Abstand zum Bildrand ist.

3. Geometrische Transformation:

   Der Worldkoordinator veranlasst das Identifizieren von Bildausschnitten in der Regel dann, wenn bereits der Detektor dort ein Gesicht erkannt hat. Daher sollte sich das Gesicht, falls tatsächlich eines enthalten ist, in der Mitte des Bildes befinden. Es genügt somit eine Skalierung des Bildes auf eine einheitliche Größe von 128×128 Pixeln, um die Gesichter auf annähernd gleiche Größe zu bringen. Diese Größe wird bereits in [Bol03] verwendet und hat sich als geeignet herausgestellt.

4. Kontrastnormalisierung:

   Der Kontrast des Bildes wird über eine Histogrammglättung normalisiert. Diese dient dazu, die Beleuchtungsunterschiede auszugleichen.

5. Verschieben des durchschnittlichen Pixelwertes und Skalierung der Pixelwerte:

   Es wird eine erneute Durchschnittsanpassung auf null vorgenommen. Dies ist nötig, da sich durch die vorhergehenden Schritte der Durchschnitt bereits sehr verändert haben könnte. Danach werden die Pixel derart skaliert, dass sie eine Standardabweichung von eins haben.

Abschließend wird erneut ein lineares Gewichten der Pixel an den Rändern vorgenommen, da sich der Effekt der ersten Gewichtung ebenfalls mit den letzten Schritten stark geändert haben kann.

Ein wesentlicher Unterschied dieser Normalisierung gegenüber dem Standard-EBGM-Algorithmus aus [Bol03] besteht darin, dass bei der geometrischen Transformation nicht davon ausgegangen wird, dass die Augenkoordinaten bekannt sind. Ist dies nämlich der Fall, lassen sich die Augenkoordinaten an eine feste Position im Bild skalieren, wodurch auch die Größe des Gesichts normiert wird. Da die Augenkoordinaten nicht bekannt sind, begnügen wir uns mit einer Skalierung auf die Standardgröße. Der Nachteil wird jedoch dadurch aufgewogen, dass unsere Bilder in der Regel dann vom Worldkoordinator verschickt werden, wenn der Detektor bereits ein Gesicht erkannt hat und die Gesichter daher in der Regel zentriert und im gleichen Größenverhältnis sind.

5 Realisierung des ViPer-Gesamtystems

Die in den drei vorangegangenen Kapiteln beschriebenen Module Detektor, Tracker und Identifizierer müssen zur Realisierung des Gesamtsystems miteinander kommunizieren. Die Ergebnisse werden in einer geeigneten Repräsentation zusammengeführt, der Einsatz der Module wird koordiniert. Diese Aufgaben werden in ViPer von einem Weltmodell übernommen, das zusammen mit der Kommunikationsstruktur im ersten Abschnitt des Kapitels vorgestellt wird. Für die Umsetzung von ViPer wird die Programmiersprache C++, sowie mehrere Bibliotheken eingesetzt. Die Anforderungen, die zur Wahl dieser Mittel geführt haben sowie der Entwicklungsprozess werden im zweiten Teil, die Werkzeuge selbst im dritten Teil des Kapitels erörtert.

5.1 Kommunikationsstruktur und Weltmodell

Die Module des ViPer-Systems sind in soweit unabhängig voneinander, als dass sie als eigenständige Prozesse auf unterschiedlichen Rechnern laufen können und nur lokal mit ihren Eingaben arbeiten müssen, um ihre Ausgaben zu erzeugen. Die Vorteile dieser Modularisierung sind einerseits, dass das System leicht auf mehrere Rechner verteilt werden kann, und andererseits, dass die einzelnen Komponenten kleiner sind, als eine monolithische Software und sich daher leichter warten lassen. Darüber hinaus ist es einfacher, eine enge Kopplung der Module zu vermeiden. Der Nachteil der Modularisierung ist der entstehende Kommunikationsbedarf. Bilder, Ergebnisse und Steuerinformationen müssen zwischen den Modulen verschickt werden.

5.1.1 Kameramanager

Der Kameramanager verschafft sich Bilder von einer Kamera oder einem Verzeichnis von zuvor gespeicherten Bildern. Er schickt die Bilder sowohl an den Tracker als auch an den Detektor und stellt sicher, dass der Detektor nur Bilder erhält, die auch der Tracker bekommen hat. Dabei wird nach einem Request-Response Protokoll gearbeitet: Der Kameramanager erhält von den angeschlossenen Modulen eine Anfrage und schickt daraufhin das aktuelle Kamerabild an das Modul, von dem die Anfrage stammt. Die Bilder könnten hier auch mit einem Push-Protokoll verschickt werden. In diesem Fall würde der Kameramanager die Bilder so schnell verschicken, wie sie von der Kamera oder der Festplatte geliefert werden können. Da aber die angeschlossenen Module (insbesondere der Detektor) nicht jedes Bild verarbeiten können, würde es hier zu einem Datenstau kommen. Genaue Informationen zur Framerate der Übertragung und der Verarbeitungsgeschwindigkeit der einzelnen Module sind in Abschnitt 6 aufgeführt.

5.1.2 Detektor

Der Gesichtsdetektor erhält ein Bild vom Kameramanager und gibt die gefundenen Gesichter als Liste von Rechtecken an das Weltmodell weiter. Das Weltmodell erhält dabei lediglich die Koordinaten der gefundenen Gesichter. Es werden keine Bilder an das Weltmodell verschickt. Dabei erhält der Detektor nicht jedes Kamerabild, weil dies seine Verarbeitungsgeschwindigkeit übersteigen würde. Nach der Detektion wird das Eingabebild verworfen, eine Speicherung findet nicht statt.

5.1.3 Tracker

Der Tracker erhält seine Bilder ebenfalls vom Kameramanager. Seine Aufgabe ist es, Objekte von Bild zu Bild zu verfolgen. Innerhalb des Tracker-Moduls besteht die Möglichkeit, mehrere Tracker-Threads zu starten, um so auch mehrere Objekte gleichzeitig zu verfolgen. Der Tracker wird durch das Weltmodell auf den Bildausschnitten initialisiert, die vom Detektor als Gesicht erkannt worden sind. Zusätzlich zu den Koordinaten des Bildausschnittes bekommt der Tracker die Nummer des Bildes, in dem die Detektion stattgefunden hat. Das Modul startet dann einen Tracker-Thread auf dem entsprechenden Bild und initialisiert ihn mit den erhaltenen Koordinaten. Da die Detektion der Gesichtsrechtecke einige Zeit benötigt, handelt es sich bei dem Bild, auf dem das Tracking begonnen werden soll, nicht um das aktuelle Bild. Daher muss der Tracker die Bilder zwischenspeichern, die er vom Kameramanager erhält. Dies geschieht in einem Ringpuffer, so dass immer die Bilder für einen bestimmten Zeitabschnitt verfügbar sind. Da der Tracker-Thread auf Bildern aus dem Puffer arbeitet (und dies mit einer höheren Framerate als der Kameramanager), kann er sich bis zum aktuellen Bild vorarbeiten. Ist dies geschehen, wird die Framerate des Tracker-Threads an die Framerate des Bildempfangs angepasst. Die Ergebnisse des Trackings werden in Form von Rechteckkoordinaten an das Weltmodell weitergegeben.

5.1.4 Identifizierer

Der Identifizierer stellt eine Beziehung zwischen einem Bildausschnitt und einem Eintrag der Gesichtsdatenbank her. Er erhält seine Bilder aus dem Ringpuffer des Tracker-Moduls. Die Identifizierung eines Gesichtes wird jedoch vom Weltmodell angestoßen. Das Weltmodell sendet also den Identifizierungsauftrag zunächst an das Tracker-Modul, wo der passende Bildausschnitt aus dem Ringpuffer geholt und dann an den Identifizierer weitergegeben wird. Dieser schickt das Ergebnis, d.h. den Namen der erkannten Person, an das Weltmodell, das den Namen mit dem entsprechenden Objekt verknüpft. Dabei ist der Identifizierer auch in der Lage, Bildauschnitte, die für die Identifikation ungeeignet sind (beispielsweise, weil der Kopf nur im Profil zu sehen ist) zu erkennen und dem Weltmodell dies mitzuteilen.

5.1.5 Weltmodell

Das Weltmodell besteht aus einer logischen Repräsentation der Personen im Raum, sowie Algorithmen zur Aktualisierung dieser Repräsentation.

Es erhält die Ergebnisse des Detektormoduls sowie die Nummer des Bildes, auf dem diese Ergebnisse ermittelt wurden. Jedes Rechteck, das der Gesichtsdetektor versendet, stellt ein Gesicht, also eine Person, dar. Das Weltmodell speichert die Position der Personen im Bild und weist den Tracker an, die vom Detektor erkannten Gesichter zu tracken, wenn diese nicht bereits getrackt werden. Das Weltmodell nutzt also die Ergebnisse der Detektion, um den Tracker auf den entsprechenden Bildausschnitten zu initialisieren.

Neben den Resultaten der Detektion erhält das Weltmodell auch die Resultate des Trackings. Es weiß also immer, welche Bildausschnitte getrackt werden, und welchen Objekten diese zugeordnet sind.

Darüber hinaus ist das Weltmodell dafür verantwortlich, die Identifikation der verfolgten Objekte anzustoßen: Bei der erstmaligen Erkennung einer Person weist das Weltmodell den Tracker an, den entsprechenden Bildausschnitt an den Identifizierer zu senden. Das Ergebnis der Identifikation wird dann durch das Weltmodell dem entsprechenden Objekt zugeordnet. Da das Weltmodell über keine Bilder verfügt, wird ein Umweg über den Tracker gemacht, denn im Ringpuffer des Trackers liegen die aktuellsten Bilder vor.

Das Weltmodell muss davon ausgehen, dass die Ergebnisse der Detektion und des Trackings mit Fehlern behaftet sind. Um Fehldetektionen (d.h. die Erkennung eines Nicht-Gesichtes als Gesicht) des Detektors auszugleichen, wird ein detektiertes Gesicht erst dann zu einem Objekt, wenn das Gesicht mehrfach hintereinander an etwa der gleichen Stelle erkannt wurde. Beim Tracker kommt es vor, dass dieser den Kopf einer Person aufgrund einer zu schnellen Bewegung verliert und dann einen Teil des Bildhintergrundes trackt. Um diesem Problem zu begegnen, hat jedes Objekt im Weltmodell eine bestimmt Lebensdauer und der Tracker besitzt Funktionalität zur Bestimmung der Wahrscheinlichkeit, mit der das zu verfolgende Objekt soeben verloren wurde. Diese Lebensdauer kann durch erneute Detektion eines Gesichts an der Position, wo sich das Objekt befindet, verlängert werden. Wird bei einem Objekt, das vom Tracker als *wahrscheinlich verloren* gekennzeichnet ist, für eine bestimmte Zeit kein Gesicht detektiert, so wird es aus dem Weltmodell entfernt und nicht mehr länger verfolgt. Dieses Verfahren ermöglicht es, Tracker, die ihr Objekt verloren haben, abzuschalten.

5.2 Anforderungen und Entwicklungsprozess

Das ViPer-System soll auf aktueller Desktophardware in angemessener Geschwindigkeit lauffähig sein. Dabei ist die Performanz eine notwendige Bedingung für das Funktionieren des Systems: Ist beispielsweise der Tracker nicht in der Lage, die Bilder in Echtzeit zu verarbeiten, so könnten allenfalls aufgenommene Sequenzen betrachtet werden. Dies ist für das Szenario des intelligenten Raums jedoch nicht ausreichend. Darüber hinaus hat die Performanz des ViPer-Systems direkten Einfluss auf die Qualität der Ergebnisse: je mehr Bilder das System in einer bestimmten Zeit verarbeiten kann, umso geringer wird die Wahrscheinlichkeit, dass der Tracker Personen verliert und dem Detektor Gesichter

entgehen. Eine weitere Rolle spielt die Beobachtbarkeit des Systems. Dabei soll nicht nur das endgültige Verhalten (beispielsweise die Begrüßung einer Person), sondern auch die Ergebnisse der einzelnen Module betrachtet werden können. Es wird also eine GUI zum Steuern und Analysieren des Systems erstellt.

Der Entwicklungsprozess besteht in einem Prototyping-Verfahren. Zunächst wird ein lauffähiger Prototyp des ViPer-Systems erstellt, der dann schrittweise verbessert wird und schließlich in ein fertiges Systems mündet. Der Vorteil eines solchen Vorgehens besteht darin, dass bereits nach kurzer Zeit ein Programm zur Verfügung steht und Probleme in der Architektur und der Implementierung früh zu Tage treten. Demgegenüber steht der Nachteil, dass einige Teile eines solchen Systems den Anforderungen nicht genügen und im Laufe der inkrementellen Weiterentwicklung verworfen und neu implementiert werden müssen. Da der Prototyp später das fertige System sein wird, muss dieser in der gleichen Sprache implementiert werden, in der auch das fertige Produkt implementiert sein soll. Zur zügigen Erstellung eines Prototypen wurde für die fachlichen Teilprobleme auf fertige Lösungen zurückgegriffen, die dann ausgetauscht oder erweitert wurden.

Diese Anforderungen beeinflussen unmittelbar die Wahl der eingesetzten Werkzeuge, allen voran die Programmiersprache.

5.3 Eingesetzte Werkzeuge

5.3.1 Die C++ Programmiersprache

Die Programmiersprache stellt das zentrale Entwicklungswerkzeug dar. Die objektorientierte Sprache C++ verbindet eine gegenüber dem nicht objektorientierten C höhere Ausdruckskraft [McC03] mit der Möglichkeit ebenso performante Programme zu schreiben. Gegenüber anderen Allzwecksprachen wie Python oder Java ist die Standardbibliothek in C++ eher klein und beschränkt sich im Wesentlichen auf die Bereitstellung von allgemeinen Datenstrukturen und Algorithmen. Insbesondere gibt es standardmäßig keine Unterstützung zur Erstellung von GUI-Anwendungen. Die Möglichkeiten zur Benutzung von Threads und zur Netzwerkkommunikation sind ebenfalls in der Standardbibliothek nicht vorhanden. Es existieren allerdings zahlreiche Bibliotheken von Drittanbietern, in denen die eben genannten grundlegenden Funktionalitäten zur Verfügung stehen. Zudem gibt es Bibliotheken, die Algorithmen und Datenstrukturen sowie teilweise komplette Verfahren aus der Bildverarbeitung implementieren, mit denen die Erstellung eines Prototypen erheblich beschleunigt werden kann. Wegen der Kompatibilität zu C ist es unerheblich, ob die Bibliotheken in C oder C++ geschrieben sind.

C++ Compiler stehen für alle gängigen Plattformen zur Verfügung. Ein C++ Programm ist allerdings an eine Plattform gebunden, sobald Betriebssystemaufrufe genutzt werden. Soll die Unabhängigkeit erreicht werden, empfiehlt sich die Verwendung einer plattformunabhängigen Bibliothek, die die Betriebssystemaufrufe kapselt.

5.3.2 Qt – Eine plattformunabhängige C++-Bibliothek

Die Qt-Bibliothek[1] der Firma Trolltech stellt Funktionalitäten aus nahezu jedem Bereich zur Verfügung. ViPer verwendet die Qt-Klassen zur Netzwerkprogrammierung, um die Kommunikation zwischen den einzelnen Modulen zu realisieren. Die Oberflächen der ViPer-Module werden mit dem Qt-Designer, einem Tool zum Erstellen von GUIs, und den entsprechenden GUI-Klassen von Qt erzeugt. Im Tracker-Modul läuft jede Tracker-Instanz in ihrem eigenen Thread. Auch dafür werden Klassen von Qt eingesetzt. ViPer benutzt die Open-Source Version von Qt 4.1, die unter der GPL verbreitet wird.

5.3.3 boost – Eine Sammlung von C++-Bibliotheken

boost[2] ist eine Sammlung von Open-Source C++-Bibliotheken, die teilweise in der kommenden C++ Standard-Bibliothek enthalten sein werden [Com05]. ViPer setzt boost::program_options ein, um Kommandozeilenargumente und Konfigurationsdateien zu verarbeiten. Die Bibliothek boost::serialization wird genutzt, um Objekte vor der Übertragung über das Netzwerk in einen Stream zu serialisieren. boost wird unter der *Boost Software License* verbreitet, die sowohl kommerzielle als auch nicht-kommerzielle Benutzung erlaubt.

5.3.4 OpenCV – Eine Open Source Bibliothek zur Bildverarbeitung

Bei OpenCV[3] handelt es sich um eine in C und C++ geschriebene Open-Source Bibliothek zur Bildverarbeitung, die von Intel entwickelt wird. OpenCV enthält eine Datenstruktur für Bilder, sowie Funktionen um Bilder zu laden, zu speichern oder zu verändern. Es bietet außerdem die Möglichkeit, Bilder von angeschlossenen Kameras einzulesen.

Darüber hinaus sind komplette Algorithmen in OpenCV enthalten. Für den ersten Prototypen von ViPer werden die OpenCV-Gesichtsdetektion und der CAMShift-Tracker eingesetzt. Die Gesichtsdetektion wird mit Hilfe des Viola & Jones-Algorithmus [VJ01] durchgeführt. In OpenCV werden bereits einige Kaskaden mitgeliefert, so dass das Training einer eigenen Kaskade für die Erstellung des Prototypen nicht erforderlich ist. Die Lizenz von OpenCV erlaubt die kommerzielle und nicht-kommerzielle Benutzung.

1 http://www.trolltech.com/products/qt/

2 http://www.boost.org

3 http://www.intel.com/technology/computing/opencv/

6 Evaluation

6.1 Evaluation der Gesichtsdetektion

Das folgende Kapitel beinhaltet die Evaluierung des Detektor-Moduls im Einzelbetrieb. Es umfasst eine Beschreibung der für die Evaluierung variierten Parameter und der benutzten Bildsequenz sowie die Präsentation und Interpretation der Ergebnisse.

Evaluierungsparameter des OpenCV Detektors

Es gibt eine Reihe von Faktoren, die das Detektionsergebnis beeinflussen. Neben der aus dem Training auf einer bestimmten Bildmenge hervorgegangenen Haarkaskade, welche den signifikantesten Einfluss auf die Leistung des von uns eingesetzten OpenCV Haardetektors hat, gibt es in der vorliegenden Implementierung folgende Parameter die zur Evaluation variiert werden können:

- Minimale Größe des Suchfensters

  Wie in Kapitel 2 beschrieben, werden während des Detektionsprozesses Ausschnitte des zu untersuchenden Bildes mit Hilfe einer Klassifikatorkaskade in Gesichts- und Nichtgesichtsbilder unterschieden. Die Auswahl der Ausschnitte erfolgt durch ein in Größe und Position variierendes Suchfenster. Als untere Schranke für die minimale Größe dieses Suchfensters bietet sich die Dimension der während des Kaskadentrainings benutzten Trainingsbilder an (in unserem Fall 24×24 bzw. 20×20 Pixel). Kleinere Gesichter können mit einer solchen Kaskade nicht gefunden werden. Offensichtlich steigt mit der Verringerung dieses Faktors die Zahl der untersuchten Ausschnitte und somit auch die Zahl der Detektionen und False Positives (Fehldetektionen) an. Da für die erhöhte Anzahl an Ausschnitten die Rechenzeit in gleichem Maße wächst, empfiehlt es sich, diesen Parameter entsprechend der geringsten im Szenario erwarteten Gesichtsgröße zu setzen. In der Einsatzumgebung des ViPer-Systems gibt es allerdings keine untere Beschränkung für die Größe der Gesichter, weshalb wir diesen Wert im Folgenden entsprechend der Trainingsdaten wählen.

- Skalierungsfaktor für das Suchfenster

  Neben der minimalen Größe besteht auch die Möglichkeit den Skalierungsfaktor des Suchfensters zu wählen. Er hat die Form $1 + \varepsilon$ und bestimmt das Wachstum des Fensters nach jedem Durchlauf. Bei der Auswahl dieses Faktors muss ein Kompromiss zwischen einer hohen Wahrscheinlichkeit, jede mögliche Skalierung eines Gesichtes abzudecken – was einem kleinen ε entspricht – und einer niedrigen Verarbeitungszeit pro Bild – was mit einem großen ε einher geht – gefunden werden.

Für unsere Evaluierung untersuchen wir die Faktoren von $1,05$ bis $2,0$ mit einer Schrittweite von $0,05$. Bei ähnlichen Detektionsergebnissen ist aufgrund der besseren Performanz jeweils der größere Faktor zu bevorzugen.

- Minimale Anzahl an benachbarten Detektionen

  Zur Reduzierung von Fehldetektionen bietet die hier eingesetzte Implementierung des Haardetektors die Möglichkeit, nur Detektionen zu akzeptieren, in deren unmittelbarer Umgebung eine minimale Anzahl weiterer Detektionen gefunden wird. Offensichtlich können bei einer zu großen Wahl dieses Parameters auch richtige Detektionen verworfen werden. Wir untersuchen im Folgenden die Ergebnisse bei 1 bis 5 benötigten Detektionen in der Nachbarschaft.

- Detektionsreduzierung mit Hilfe eines Canny-Kantenfilters

  Ein weiteres Merkmal der OpenCV-Implementierung des Haardetektors zur Reduzierung der Fehldetektionen basiert auf einem Canny-Kantenfilter, der auf den zu untersuchenden Ausschnitt angewandt wird. Dieser wird verworfen, falls die Anzahl der Kanten in seinem Kantenbild über oder unter einem bereits auf Gesichtsbilder optimierten Wert liegt. In unserer Auswertung untersuchen wir, welchen Einfluss das Zuschalten dieser Funktionalität auf unsere Detektionsergebnisse hat.

Evaluierung des Hautfarbenklassifikators

Neben den Merkmalen der OpenCV-Implementierung bietet unsere Erweiterung der Hautfarbenklassifikation ebenfalls die Möglichkeit, die Zahl der Fehldetektionen zu verringern. Dazu werden für die potentiellen Detektionen Hue-Histogramme berechnet und mit dem Histogramm eines manuell selektierten Gesichtsbildes verglichen. Unterscheiden sich die Histogramme zu sehr, wird die Detektion an dieser Stelle verworfen. In unserem Evaluationsszenario variieren wir den Schwellenwert der Histogrammunterschiede von 0 (keine Ähnlichkeit) bis 1 (vollständigen Übereinstimmung) mit einer Schrittgröße von $0,05$.

6.1.1 Die Testsequenz

Zur Erstellung der für die Evaluierung des Detektor-Moduls notwendigen Sequenz aus Testbildern untersuchen wir zunächst, welche Positionen und Orientierungen der Gesichter relativ zur Kamera unter idealen Bedingungen zu detektieren wären. Bezüglich der Rotation und der Neigung des Kopfes ist der Spielraum für die Testbilder stark begrenzt. Da es sich bei den Bildern in den Trainingsmengen ausschließlich um Frontalaufnahmen handelt und das Training in dieser Form keine Unterscheidung verschiedener Orientierungen ermöglicht, können bestenfalls leicht rotierte Gesichter erkannt werden. In Größe und Position hingegen sollten alle Variationen erlaubt sein, solange die Gesichter die Größe der Trainingsbilder (von 20×20 bzw. 24×24 Pixel) nicht unterschreiten. Weiterhin ist von Interesse, wie Gesichter verschiedener Personen detektiert werden und welchen Einfluss unterschiedliche Gesichtsausdrücke auf das Ergebnis haben.

Abbildung 6.1: Beispielbilder der Testmenge (eine Person in verschiedenen Posen)

Um diesen Ansprüchen gerecht zu werden, wählen wir eine Testmenge mit 253 Bildern von insgesamt 45 Personen. Die Bilder wurden mit der Kamera des ViPer-Systems aufgenommen und haben eine Auflösung von 756×556 Pixeln. Sie umfassen für jede Person mindestens drei verschiedene Positionen mit teilweise variierenden Gesichtsausdrücken. Die Gesichter in den Bildern sind minimal 20×30 und maximal 90×160 Pixel groß. Auf allen Bildern werden die Gesichter mit dem Annotationsprogramm anhand von Augen- und Mundpositionen annotiert, so dass eine automatische Auswertung der Detektionsergebnisse möglich wird.

6.1.2 Die Detektionskaskaden

Für unsere Evaluation benutzen wir zwei Kaskaden der OpenCV-Implementierung und zwei Kaskaden, die wir auf einer Trainingsmenge von Bildern aus der CMU PIE-Bilddatenbank [SBB03] trainiert haben. Die an der Carnegie Mellon Univertiy zusammengestellte Datenbank umfasst mehr als 40.000 Bilder von 68 verschiedenen Personen, für die jeweils eine große Auswahl an Aufnahmen aus verschiedenen Perspektiven und unter verschiedenen Beleuchtungen existiert. Für das Training der Kaskaden beschränken wir uns auf die drei frontalen Aufnahmeperspektiven mit natürlich wirkender, gleichmäßiger Beleuchtung, was zu einer Trainingsmenge mit insgesamt 12.300 Gesichtsbildern führt. In diesen Bildern werden von uns mit Hilfe unseres Annotationsprogramms (siehe A.2.6) die Mittelpunkte von Augen und Mund markiert und auf dieser Grundlage Gesichtsausschnitte extrahiert. Das Ausschnittsrechteck wird, vom kleinsten sowohl Augen- als auch Mundmarkierungen enthaltenden Rechteck ausgehend, durch Verdreifachung der Seitenlängen unter Beibehaltung des Mittelpunktes bestimmt. Außerdem wird durch Anpassung einer Seitenlänge sichergestellt, dass das Seitenverhältnis für alle Extraktionen identisch ist. Für das Training werden zudem 6.000 Nichtgesichtsbilder aus Aufnahmen aus der Einsatzumgebung sowie von zufälligen Landschafts- und Gebäudebildern erzeugt.

Im Folgenden werden die Ergebnisse unseres Trainings in Form der zwei Kaskaden *viper_1* und *viper_2* mit der OpenCV-Standardkaskade (*frontalface_default*) und der zweiten alternativen Kaskade (*frontalface_alt2*) verglichen. Bei der ersten Kaskade handelt es sich um eine auf 24×24 Pixel großen Bildausschnitten mit dem *Discrete Adaboost*-Verfahren [Fre95] trainierte lineare Kaskade. Die zweite Kaskade beinhaltet auf 20×20 Pixel großen Ausschnitten mit *Gentle Adaboost* [FHT00] erzeugte Trainingsdaten, welche in einer Baumstruktur durchlaufen werden (vgl. [LKP03]). Die Wahl zu Evaluation fiel

auf diese beiden Kaskaden, weil sie von allen OpenCV-Kaskaden in unserem Szenario die besten Detektionsergebnisse liefern. Unsere beiden Kaskaden sind ebenfalls jeweils eine lineare und eine baumförmige Kaskade. Die Boostingverfahren und die Größe der Bildausschnitte ist bei uns genau umgekehrt gewählt: *viper_1* enthält Gesichter mit einer Größe von20×20 Pixeln, ist linear und mit *Gentle Adaboost* trainiert; *viper_2* baut auf Abschnitten der Größe 24×24 Pixel auf, die in einer Baumstruktur unter Einsatz von *Discrete Adaboost* trainiert worden sind.

6.1.3 Durchführung der Evaluation

Um die Evaluation automatisiert durchzuführen, müssen zum einen die im Detektor-Modul erzeugten Detektionsdaten zur Verfügung stehen, zum anderen muss eine verlässliche Methode gefunden werden, um zu bestimmen, ob es sich bei einer Detektion um ein markiertes Gesicht oder eine Fehldetektion handelt. Wir entscheiden uns dafür, unser Modul nicht um die Funktionalität zur Auswertung zu erweitern, sondern ein zusätzliches Java Programm zu schreiben, das diese Aufgabe übernimmt. Dieses ruft das Detektor-Modul für jede Parameterkombination auf der Sequenz von Testbildern auf und verarbeitet anschließend die in Form einer Textdatei ausgegebenen Detektionen jedes Bildes.

Metrik der Detektionsabweichung

Die Zuordnung von Detektionen zu annotierten Gesichtern, und somit die Bestimmung der Anzahl von erfolgreichen Detektionen und False Positives, basiert in unserem Programm auf einer Abweichungsmetrik, die sich aus den zwei möglichen Fehlerquellen ergibt:

- Positionsabhängiger Fehler

  Das sicherste Merkmal, um zu erkennen, ob sich eine Detektion auf ein annotiertes Gesicht bezieht, ist der räumliche Abstand. Es muss ein geeigneter Grenzwert gefunden werden, mit dem die Trennung zwischen richtigen und falschen Detektionen möglich ist. Da für den Positionsfehler bei stark variierenden Gesichtsgrößen ebenfalls unterschiedliche Größenordnungen zulässig sind, wird vor der Anwendung eines Grenzwertes eine Normierung des Fehlers anhand der Bilddiagonalen des annotierten Gesichts durchgeführt.

$$d_{normiert} = \frac{\sqrt{(x^d_{center} - x^a_{center})^2 + (y^d_{center} - y^a_{center})^2}}{\text{diagonale}(AnnotiertesGesicht)}$$

- Größenabhängiger Fehler

  Weiterhin darf die Größe des detektierten Gesichts nicht zu stark von der Größe des markierten Gesichts aus den Testdaten abweichen. So ist es möglich, dass eine Detektion zwar das Positionskriterium erfüllt, aber ein Vielfaches bzw. nur

einen Bruchteil des gesuchten Gesichts enthält. Um diesen Fall als Fehldetektion zu erkennen, bestimmen wir die Größenähnlichkeit mit:

$$S = \frac{\min(\text{diagonale}(\mathit{DetektiertesGesicht}), \text{diagonale}(\mathit{AnnotiertesGesicht}))}{\max(\text{diagonale}(\mathit{DetektiertesGesicht}), \text{diagonale}(\mathit{AnnotiertesGesicht}))}$$

wobei S im Intervall $(0, 1]$ liegt und der Wert 1 identischen Diagonalenlängen entspricht. Für diesen Fehler bestimmen wir ebenfalls einen Grenzwert, bei dessen Unterschreitung die Detektion als False Positive klassifiziert wird.

Variation der Parameter

Im hier behandelten Evaluationsprozess wird der Detektor von unserem Evaluationsprogramm mit verschiedenen Parametereinstellungen und verschiedenen Detektionskaskaden auf der Testsequenz ausgeführt. Die Parameter für den Skalierungsfaktor, die minimale Anzahl benachbarter Detektionen, das Canny-Pruning und die Hautfarbenklassifkation durchlaufen dabei, wie oben erwähnt, verschiedene Wertebereiche. Zur Reduktion der notwendigen Wiederholungen werden für jede Kaskade die einzelnen Parameter getrennt variiert, wobei für die jeweils fixen Parameter ein Wert gewählt wird, der aufgrund der vorhergegangenen Tests geeignet erscheint. Weiterhin werden zur Verringerung des Rechenaufwands sämtliche Detektionen vor Anwendung der Hautfarbenklassifikation jeweils mit dem zugehörigen Histogrammwert ausgegeben, so dass der Einfluss dieser Erweiterung ohne mehrfache Wiederholung des Detektionsprozesses bestimmt werden kann.

6.1.4 Präsentation der Ergebnisse und Einzelinterpretation

Canny-Kantendetektion (canny pruning, siehe Abb. 6.2)

Bei der Betrachtung des Diagramms zur Evaluation des Kantendetektionsparameters fällt auf, dass es sowohl bei der Anzahl der erfolgreichen Detektionen als auch bei der Anzahl der False Positives keinen Unterschied macht, ob er ein- oder ausgeschaltet ist. Jedoch lässt sich bei aktivierter Canny-Kantendetektion eine Beschleunigung des Detektionsvorganges feststellen. Aus diesem Grund ist dieser Parameter im Folgenden immer eingeschaltet.

Da sich die Gesichtsdetektion für ViPer als generell schnell genug herausgestellt hat, evaluieren wir die Detektionszeiten der einzelnen Kaskaden nicht.

Die meisten True Positives erreicht hier unsere Kaskade *viper_1* mit ca. 150 Detektionen ($= 59,3\,\%$) , wobei sich dieses positive Ergebnis sofort wieder relativiert, denn mit fast 80 False Positives ist sie auch die Kaskade mit der höchsten Fehleranzahl. An zweiter Stelle bezüglich der True Positives befindet sich die zweite alternative OpenCV-Kaskade mit ca. 140 richtigen Detektionen ($= 55,3\,\%$). Diese Kaskade hat gleichzeitig auch die geringste Anzahl an False Positives. An dritter Stelle folgt die Standard-Kaskade der OpenCV-Bibliothek mit etwa 130 True Positives ($= 51,4\,\%$) und einem immer noch guten Wert von 10 False Positives. Ebenfalls ein recht gutes Ergebnis liefert unsere

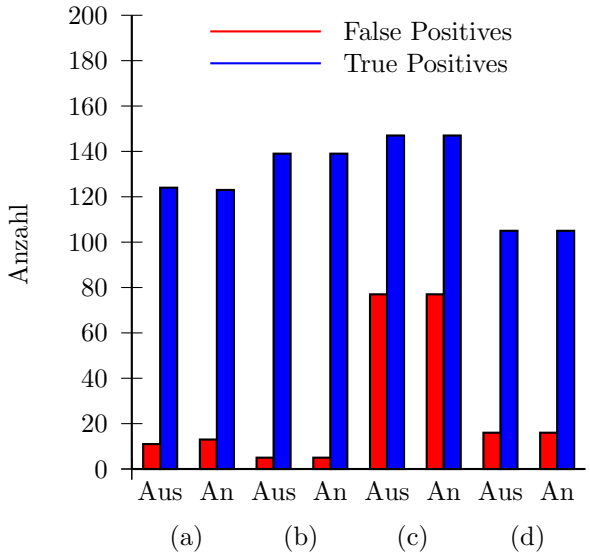


Abbildung 6.2: Evaluation des Canny Pruning Parameters: Detektionslauf mit ein- bzw. ausgeschaltetem Kantendetektor. (a) *frontalface_default*, (b) *frontalface_alt2*, (c) *viper_1*, (d) *viper_2*

eigene Kaskade *viper_2*. Während sie 110 Gesichter korrekt detektiert ($= 43,5\,\%$), findet sie nur etwa 20 Fehldetektionen.

Hautfarbenklassifikator (skin color detection, siehe Abb. 6.3)

Für die Präsentation der Auswirkung des Hautfarbenklassifikators haben wir die absoluten Anzahlen der True Positives und False Positives bei Variation des Parameters innerhalb des Parameterintervalls miteinander ins Verhältnis gesetzt. Das resultierende Diagramm ähnelt einer ROC[1]-Kurve und kann somit sehr gut zur Schwellwertoptimierung genutzt werden. Um das Diagramm und die darin erkennbare Auswirkung der Variation des Hautfarbenfilterschwellwerts zu verstehen, ist es wichtig, zu wissen, dass der Einfluss der Hautfarbenklassifikation von *rechts nach links* zunimmt. Es ist dann klar erkennbar, dass bis zu einem gewissen Punkt jeder Kurve nur die False Positives abnehmen und die True Positives stagnieren. Einerseits zeigt dies deutlich die positive Auswirkung des Hautfarbenklassifikators auf die Güte der Detektionen, denn ohne erfolgreiche Detektionen zu verlieren, werden die Fehler reduziert; andererseits kann direkt aus dem Diagramm der optimale Schwellwert für den Hautfarbenparameter abgelesen werden (die Stelle der Kurve, an dem zum ersten Mal False Positives auf Kosten der True Positives reduziert werden). Um optimale Detektionsergebnisse zu erzielen, kann nun der auf diese Art für jede einzelne Kaskade ermittelte Schwellwert für alle Folgedetektionen genutzt werden. Eventuell ist es - je nach Anwendungszweck - sinnvoll, doch

1 Receiver Operating Characteristic (vgl. [Faw04])

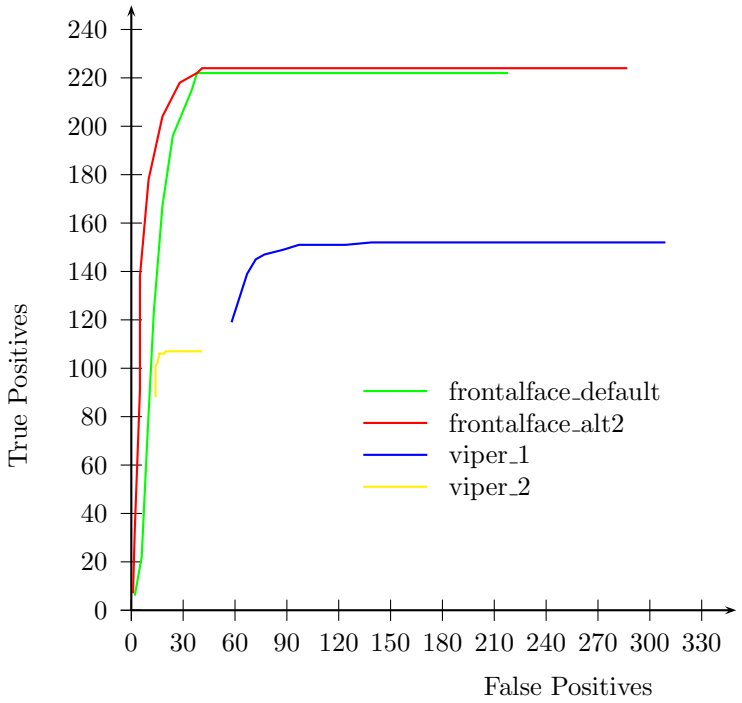


Abbildung 6.3: Evaluation des Hautfarben Parameters: Variation des Einflusses der Hautfarbendetektion von 0 bis 1 in 0,05er Schritten und Bestimmung der True Positives und False Positives mit diesem Wert.

einige True Positives bei gleichzeitiger Reduktion der False Positives zu opfern, um im Verhältnis eine niedrigere Fehlerrate zu erzielen.

Ohne Verlust von True Positives führt die Erhöhung des Parameterwertes bei jeder von uns getesteten Kaskade zu einer erheblichen Verbesserung des Detektionsergebnisses (siehe 6.1), indem die Anzahl an False Positives drastisch gesenkt wird. Bei der Kaskade *frontalface_default* kann so die Anzahl False Positives um 180 Fehldetektionen - von 218 auf 38 - gesenkt werden. Auf die Kaskade *frontalface_alt2* angewandt ergibt sich sogar eine noch stärkere Verbesserung, da die Zahl der False Positives von 287 auf 41 gesenkt werden kann. Bei unseren selbst trainierten Kaskaden zeigt sich ebenfalls eine sehr positive Auswirkung. Die Zahl der False Positives kann bei der Kaskade *viper_1* ohne Einbußen von 309 auf 139 False Positives reduziert werden. Bei der Kaskade *viper_2* verringert sich die Anzahl der fälschlicherweise als Gesicht erkannten Bereiche um mehr als die Hälfte von 41 auf 20.

All diese Verbesserungen relativierten sich jedoch wieder, wenn durch diesen zusätzlichen Filter die Laufzeit der Detektion wesentlich erhöht würden. Zeitmessungen, die wir unabhängig von dieser Evaluation gemacht haben, zeigen, dass die Detektionszeit mit Hautfarbenverifikation nur um etwa 0,25 Prozent erhöht ist im Vergleich zur Detektionszeit ohne Hautfarbenverifikation. Das ist aber auch nicht überraschend, da die Größe des Bildes, auf dem detektiert wird, die Größe der Bildausschnitte, deren Histograme

	ohne Hautfarbenfilter	mit Hautfarbenfilter
frontalface_default	218	38
frontalface_alt2	287	41
viper_1	309	139
viper_2	41	20

Tabelle 6.1: Maximal mögliche Reduktion der False Positives ohne Einbußen von True Positives durch Zuschaltung des Hautfarbenfilter

zur Hautfarbenfilterung bestimmt werden, um ein Vielfaches übersteigt.

Skalierungsfaktor (scale factor, siehe Abb. 6.4, 6.5, 6.6 und 6.7)

Um die Übersichtlichkeit nicht zugunsten der geringeren Diagrammanzahl aufzugeben, gibt es für die Evaluation der Auswirkung des Skalierungsfaktors auf die Detektionsergebnisse jeweils ein Diagramm pro Kaskade. Auf den ersten Blick wirken die Ergebnisse sprunghaft und willkürlich. Bei näherer Betrachtung und unter Einbeziehung des Wissens über die Trainingsmenge erschließt sich jedoch der Zusammenhang zwischen Parameter und Evaluationsergebnis. Zuerst kann zwischen dem Einfluss des Faktors auf unsere Kaskaden und die OpenCV-Kaskaden unterschieden werden: Bei unseren Kaskaden nehmen sowohl True Positives als auch False Positives mit steigendem Faktor ab.

Der Kurvenverlauf für *viper_1* lässt sich durch die Beschreibung der Trainingsmenge, insbesondere der Nicht-Gesichtsbilder, erklären. Im Training für diese Kaskade wurden in feiner Abstufung – angefangen bei 4×4 Pixeln bis hin zur vollen Auflösung der Kamerabilder – Fragmente aus den Nicht-Gesichtsbildern ausgeschnitten und für das Training benutzt. Da ein geringer Wert für den Skalierungsfaktor auch die Vergrößerungsschritte des Detektionsfensters klein hält, kann so am effektivsten detektiert werden, ohne Gesichter aufgrund zu großer Ähnlichkeit zum Hintergrund zu übersehen.

Im Diagramm zur Kaskade *viper_2* ist der Einfluss der Nicht-Gesichtsdaten noch offensichtlicher, denn es wurden im Training ebenso zufällige Nicht-Gesichtsausschnitte wie bei der obigen Kaskade verwendet, jedoch mit dem Unterschied, dass diese mindestens die Größe der Gesichtsbilder der Kaskade haben. Eine Erhöhung des Skalierungsfaktors führt daher schneller zu einem schlechten Detektionsergebnis.

Maxima, die zwischen minimalem und maximalem Skalierungsfaktor auftreten, bilden zufällig passende Werte ab, die sich aus der unregelmäßigen Zusammensetzung der Nicht-Gesichtsdaten ergeben.

Das Detektionsergebnis der OpenCV-Kaskaden schwankt zwischen abwechselnd sehr guten und sehr schlechten Werten. Da wir die Kaskaden nicht selbst trainiert haben, könnten wir nur Spekulationen bzgl. der Trainingsmenge anstellen. Ein Zusammenhang zwischen dieser und dem Verlauf der Kurve besteht jedoch definitiv.

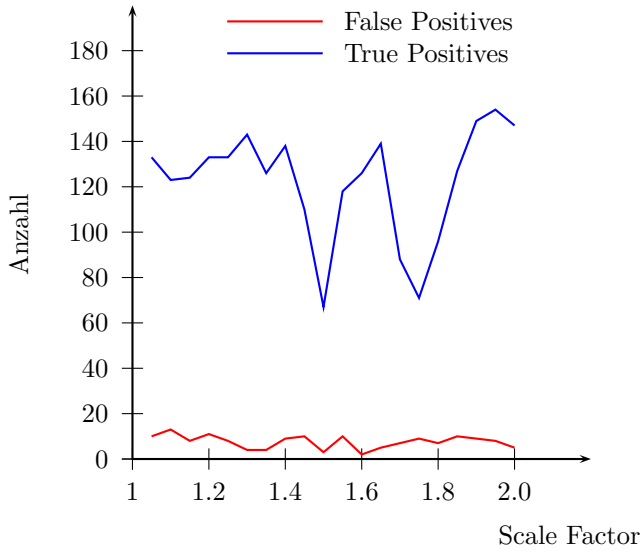


Abbildung 6.4: Evaluation des Skalierungsfaktor Parameters
Kaskade: frontalface_default

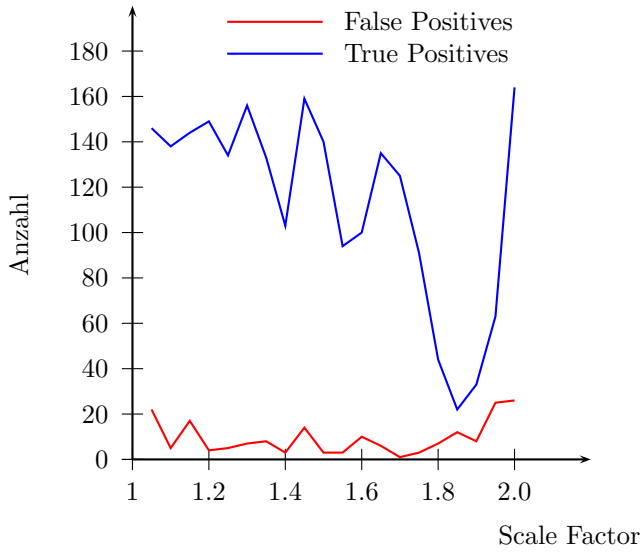


Abbildung 6.5: Evaluation des Skalierungsfaktor Parameters
Kaskade: frontalface_alt2

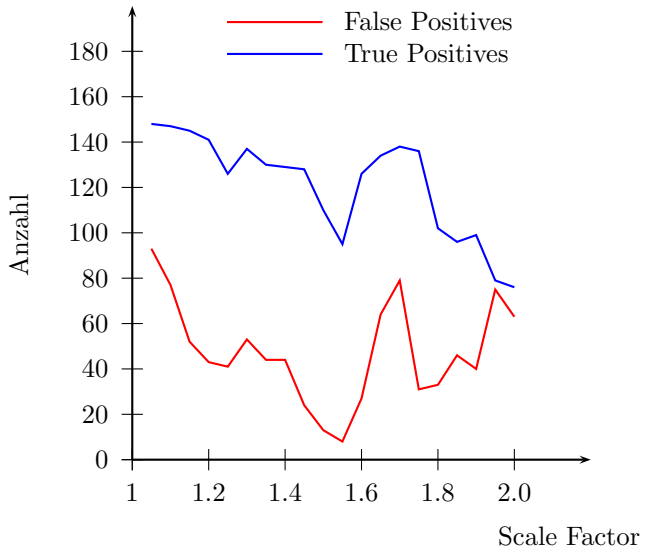


Abbildung 6.6: Evaluation des Skalierungsfaktor ParametersKaskade: viper_1

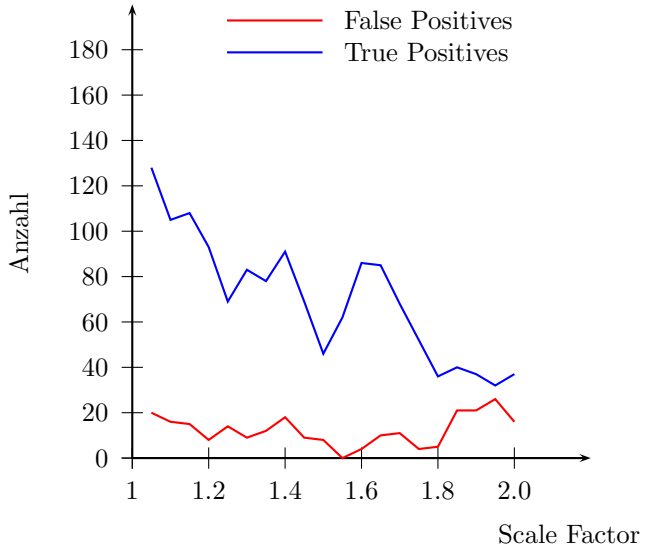


Abbildung 6.7: Evaluation des Skalierungsfaktor ParametersKaskade: viper_2

Minimale Anzahl Detektionen in der Nachbarschaft (minimum neighbours, siehe Abb. 6.8, 6.9, 6.10, 6.11)

Auch bei der Evaluation des Zusammenhanges zwischen der Mindestanzahl der Nachbarschaftsdetektion und der Anzahl an True und False Positives sind die Diagramme nach Kaskaden aufgeteilt. Die erste offensichtliche Relation ist, dass mit der Erhöhung der minimalen Anzahl an Nachbarschaftsdetektion ein Abfallen der True und False Positives einhergeht. Das ist logisch, denn je mehr Detektionen in einem gewissen Bereich benötigt werden, um eine einzelne von ihnen zu rechtfertigen, desto seltener kommt es überhaupt zu einer Detektion. Sichtbar wird auch, dass das Detektionsergebnis durch Erhöhung der benötigten Nachbarschaftsdetektionen hauptsächlich besser wird. Die False Positives werden nämlich um ein Vielfaches schneller reduziert als die True Positives. Somit führt eine Erhöhung dieses Parameters bis zu dem Punkt, an dem die False Positives nahe oder gleich Null sind, zu einer im Verhältnis nur minimal reduzierten Anzahl True Positives.

Durch Erhöhung dieses Parameters von 1 auf 4 kann man bei der OpenCV-Standardkaskade unter Verlust von nur 3 erfolgreichen Detektionen die False Positives auf 0 verringern. Mit der zweiten alternativen Kaskade funktioniert es ähnlich gut: Nach Parametererhöhnung auf 4 existiert von ehemals 5 False Positives kein einziger mehr - jedoch verliert man auch 8 True Positives. Für unsere Kaskade *viper_1* ergeben sich - trotz insgesamt generell etwas schlechterem Ergebnis - noch deutlichere Verbesserungen. Zwischen 1 und 3 Mindestnachbardetektionen ergibt sich eine Reduktion von über der Hälfte an False Positives von 77 auf 29 bei Inkaufnahme von 13 verlorenen True Positives. Bei *viper_2* ist von einer Erhöhung des Parameters abzusehen, da die Kaskade generell eher wenige Detektionen liefert, geht jede Erhöhung zu stark auf Kosten der True Positives.

Optimale Parameterwahl (s. Abb. 6.12 und Tabellen 6.2, 6.3)

Nachdem die möglichen Parameter einzeln evaluiert worden sind, lassen sich aus den jeweiligen Diagrammen die optimalen Parameterwerte ablesen. Durch Einstellung der Parameter auf ihren optimalen Wert führt ein erneuter Detektionslauf auf den Testbildern zu einem insgesamt besseren Detektionsergebnis. Die einzelnen Verbesserungen, die sich durch optimierte Einstellungen ergeben, summieren sich nun auf und es wird eine stärkere Aufwertung erreicht als durch Variation einzelner Parameter.

Die insgesamt beste Kaskade ist die zweite alternative OpenCV-Kaskade mit 151 True Positives ($= 59,7\,\%$) und nur 4 False Positives. Auf den zweiten Platz kommt bei einem Verhältnis von 143:3 ($= 56,5\,\%$ True Positives) zwischen False Positives und True Positives die Standardkaskade von OpenCV. An dritter und vierter Stelle folgen unsere Kaskaden *viper_2* mit 131 korrekten Detektionen ($= 51,8\,\%$) und 21 fälschlicherweise detektierten Gesichtern und *viper_1*, welche bei 119 True Positives ($= 47\,\%$) 24 False Positives liefert.

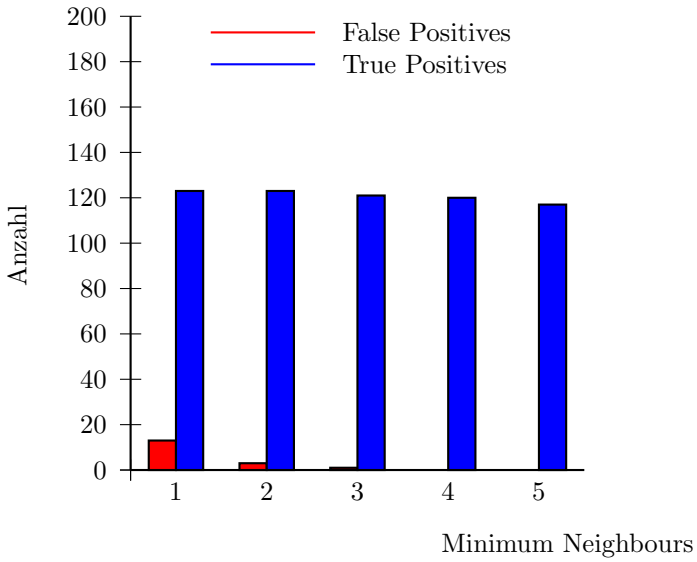


Abbildung 6.8: Evaluation des Minimum Neighbours Parameters
Kaskade: frontalface_default

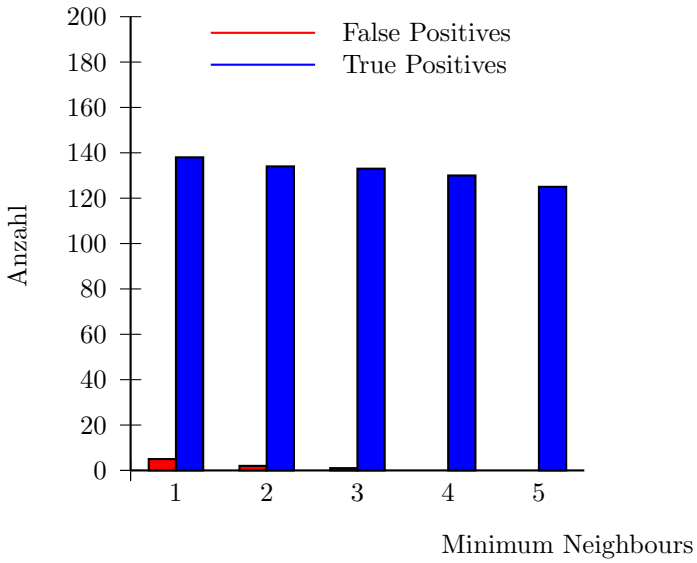


Abbildung 6.9: Evaluation des Minimum Neighbours Parameters
Kaskade: frontalface_alt2

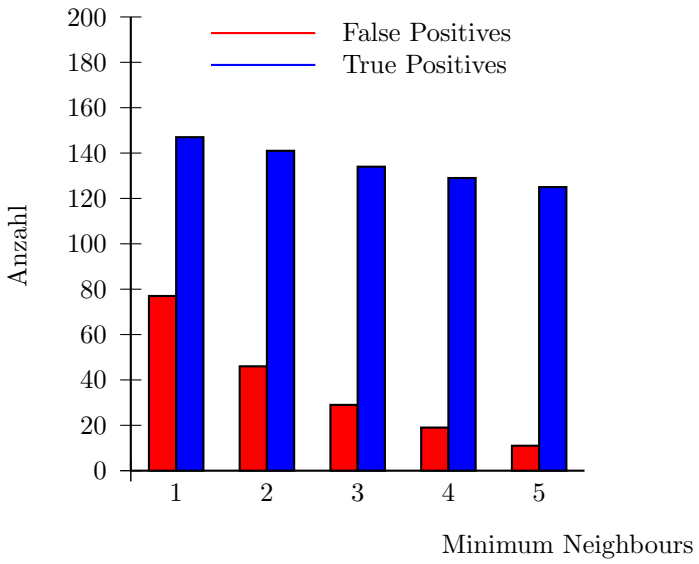


Abbildung 6.10: Evaluation des Minimum Neighbours ParametersKaskade: viper_1

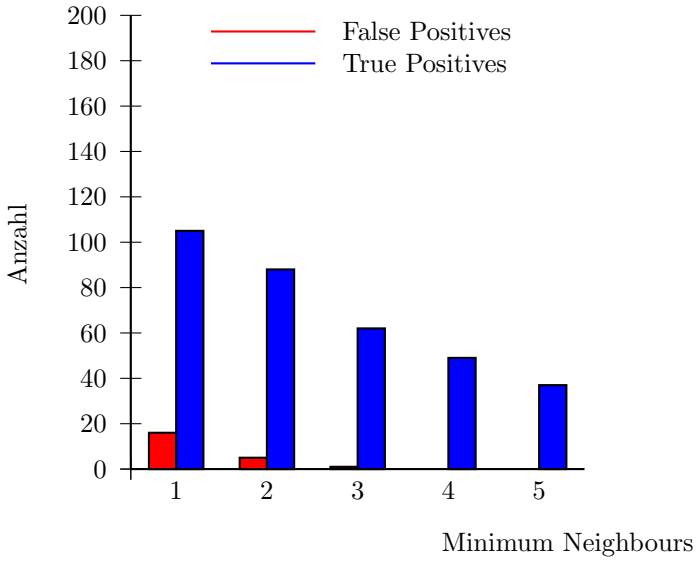


Abbildung 6.11: Evaluation des Minimum Neighbours ParametersKaskade: viper_2

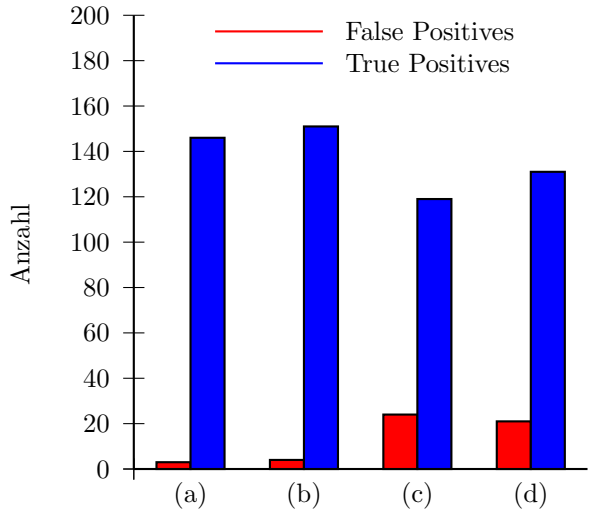


Abbildung 6.12: Evaluation bei optimaler Parameterwahl: (a) *frontalface_default*, (b) *frontalface_alt2*, (c) *viper_1*, (d) *viper_2*

	CannyPruning	SkinColor	ScaleFactor	MinimumNeighbours
frontalface_default	an	0,6	1,95	4
frontalface_alt2	an	0,65	1,3	4
viper_1	an	0,95	1,05	3
viper_2	an	0,75	1,05	1

Tabelle 6.2: Abgelesene optimale Parameterwerte

	True Positives	False Positives	Detektionsrate	False Positive Rate
frontalface_default	146	3	0,58	0,012
frontalface_alt2	151	4	0,6	0,016
viper_1	119	24	0,47	0,095
viper_2	131	21	0,52	0,083

Tabelle 6.3: Ergebnisse der Detektion mit optimal gewählten Parametern (Die False Positive Rate bezieht sich hier auf die Gesamtzahl möglicher Gesichtsdetektionen.)

6.1.5 Gesamtinterpretation der Ergebnisse

Nachdem wir im vorherigen Kapitel unsere Detektionsergebnisse präsentiert und deren Bedeutung für die einzelnen Parameter interpretiert haben, fassen wir sie in diesem Abschnitt zusammen und stellen noch einmal die OpenCV-Kaskaden unseren selbst trainierten Kaskaden gegenüber.

Bei der Evaluation ist deutlich geworden, dass der Canny-Kantenfilter keinen Einfluss auf Detektionsquantität und -qualität ausübt. Im Gegensatz zu den anderen möglichen Parametereinstellungen kann dieser also, um eine möglichst hohe Detektionsgeschwindigkeit zu erzielen, immer eingeschaltet werden.

Den größten Einfluss hat die Wahl des Skalierungsfaktors. Dieser muss immer individuell passend zur Trainingsmenge und dem Detektionsszenario (z. B. Kameratyp, Abstand zu den zu detektierenden Gesichtern) für jede Kaskade einzeln eingestellt werden, um die bestmöglichen Resultate zu erzielen. Wird der Skalierungsfaktor falsch gewählt, so kann auch eine Kaskade mit hohem Detektionspotential keine guten Ergebnisse liefern.

Die Anzahl der Mindestdetektionen in der Nachbarschaft ist nicht für jede Kaskade gleich wichtig. Detektiert eine Kaskade überwiegend mehr Gesichter als eigentlich im Bild zu sehen sind, hilft eine höhere Mindestanzahl, um wirkliche Gesichter von False Positives abzugrenzen (s. Abb. 6.10). Reduzieren sich dabei jedoch False Positives und True Positives um annähernd den gleichen Wert, ist die Kaskade falsch trainiert worden und erkennt Gesichter generell nur unzuverlässig. Für Kaskaden, die dazu neigen, weniger Gesichter zu erkennen als tatsächlich in einem Bild vorhanden sind, ist es nicht ratsam, die Mindestanzahl der Detektionen in der Nachbarschaft hoch anzusetzen (vgl. Abb. 6.11). Dadurch würden nur weitere True Positives irrtümlicherweise ausgeschlossen.

Abgesehen von den Parametereinstellungen des Detektors selbst bringt die korrekte Justierung des nachträglich angewandten Hautfarbenfilters eine enorme Verbesserung des Detektionsresultates. Bis zu einem gewissen Wert können für jede Kaskade ohne Einbuße von True Positives die False Positives drastisch eingeschränkt werden. Als besonders nützlich stellt sich auch diese Kenngröße – ähnlich wie die Anzahl der Mindestanzahl Nachbarschaftsdetektionen – bei Kaskaden heraus, die potentiell eher zu viele Gesichter finden. Tatsächlich trägt die Hautfarbenklassifikation stark dazu bei, dass unsere eher schwächere Kaskade *viper_1* immer noch passable Ergebnisse liefert. Ohne Unterscheidung zwischen Haut und Nicht-Haut, ist die False Positive-Rate um ein Vielfaches höher.

Damit widmen wir uns der Beurteilung der Kaskaden im Gesamtkontext und dem Vergleich unserer Kaskaden mit denen der OpenCV-Bildverarbeitungsbibliothek. Auffällig ist, dass sowohl die Standard- als auch die zweite alternative OpenCV-Kaskade besser als unsere selbst trainierten Kaskaden sind. Da ein Kaskadentraining sehr viel Zeit – in der Regel mehrere Wochen – in Anspruch nimmt, haben wir in der begrenzten Zeit der Projektgruppe nur eine geringe Anzahl an Kaskaden trainieren können. Bei jeder neuen Kaskade haben wir die Erfahrungen mit den vorangegangen Trainingsläufen genutzt, um die Trainingssets zu verfeinern und somit das Detektionsverhalten zu verbessern. Die Qualität der Kaskade hängt stark von der Parameterwahl für das OpenCV-Kaskadentrainingsprogramm ab. Die möglichen Einstellungen sind vielfältig und so konnten nicht

alle Wertekombinationen getestet werden, jedoch haben wir uns bei den Grundeinstellungen stark an den zwei in diesem Kapitel evaluierten OpenCV-Kaskaden orientiert. Damit nähert sich die Qualität der selbsttrainierten Kaskaden an die der OpenCV-Kaskaden. Sie können sicher noch verbessert werden, vor allem aber Kaskade *viper_2* liefert schon jetzt ein sehr gutes Detektionsresultat.

6.2 Evaluation des Trackers

In diesem Kapitel wird die Evaluierung des Trackers als einzelnes Modul beschrieben. Zu diesem Zweck haben wir dem Einsatzbereich des Trackers entsprechende Videosequenzen erstellt. Ziel der Evaluation ist, die Qualität unserer Erweiterung gegenüber dem originalen CAMShift-Algorithmus zu bewerten, sowie Stärken und Schwächen des Trackers aufzuzeigen. Zunächst wird im Abschnitt 6.2.1 erläutert, wie die zugrunde liegenden Evaluationsdaten aussehen, auf denen die beiden Algorithmen verglichen und bewertet werden. Danach folgen die Abschnitte 6.2.2, in dem die Ergebnisse der einzelnen Evaluationen vorgestellt werden und 6.2.3, in dem die Ergebnisse in ihrer Gesamtheit erläutert und bewertet werden.

6.2.1 Evaluationsdaten

Die zur Evaluierung des Trackermoduls verwendeten Videosequenzen decken verschiedene Problematiken ab. Ziel ist es, alle Schwierigkeiten zu erfassen, mit denen das ViPer-System konfrontiert wird und die vom Tracker erfolgreich zu bewältigen sind. Für die Erläuterung der Evaluationsdaten und Ergebnisse werden lediglich drei Sequenzen herangezogen. Tests auf weiteren Sequenzen wurden zwar durchgeführt, führten jedoch hinsichtlich der Trackingqualität in Problemsituationen zu keinen neuen Erkenntnissen. Die Evaluierungssequenzen wurden innerhalb des intelligenten Raumes und mit der dort vorhandenen Hardware erstellt, um repräsentative Ergebnisse für das in der Projektbeschreibung vorgegebene Einsatzgebiet des Gesamtsystems zu erlangen. Auch die Initialisierung des Trackermoduls orientiert sich an diesen Anforderungen und geschieht auf Basis der vom Detektormodul gelieferten Detektionsdaten zu den Videosequenzen.

Um eine Bewertung der Ergebnisse des Trackermoduls zu ermöglichen, wurden die Einzelbilder der Sequenzen manuell annotiert. Jedes Gesicht in jedem Einzelbild einer jeden Sequenz ist in seiner Position und Ausdehnung bekannt. Auf dieser Basis werden die vom Tracker erstellten Trajektorien mit denen der Annotationsdaten verglichen und bewertet.

Evaluationssequenzen

In diesem Abschnitt werden die Evaluationssequenzen erläutert. Des Weiteren wird dargestellt, welche Annotationsdaten zu den Einzelbildern der Sequenzen vorliegen und mit welchen Detektoreinstellungen diese gewonnen wurden.

Wie bereits in Abschnitt 6.2.1 erwähnt, wird die Evaluation für drei Sequenzen in diesem Kapitel näher beschrieben.

- Sequenz 1: Eine Sequenz, in der zu jeder Zeit das Gesicht zu sehen ist.
- Sequenz 2: Eine Sequenz, in der volle Kopfdrehungen vorkommen, das Gesicht ist somit nicht immer im Blickfeld der Kamera.
- Sequenz 3: Eine Sequenz, in der eine Person mit rotem Oberteil zu sehen ist. Zusätzlich sind in dieser Sequenz volle Kopfdrehungen enthalten.

Diese Sequenzen wurden so gewählt, um die verschiedenen angesprochenen Problematiken in diesem Szenario abzudecken. Zudem haben die Sequenzen eine Länge von mindestens 60 Sekunden, um zu prüfen, ob das Trackermodul auch längerer Zeit das Objekt zuverlässig trackt.

Der Standard-CAMShift Algorithmus ist für das Tracken von Gesichtern konzipiert, insbesondere solange hautfarbene Bereiche des Objektes im Bild vorhanden sind. Dies ist jedoch im Einsatzfeld von ViPer nicht immer gewährleistet. Insbesondere Situationen, in denen sich das Gesicht von der Kamera abwendet, sind für das Tracking problematisch. Aus diesem Grund wurden Sequenzen mit Kopfdrehungen für die Evaluation des Trackers verwendet. Das Tracken von Personen, deren Gesicht von der Kamera abgewendet ist, soll durch unsere Erweiterungen des CAMShift-Algorithmus hin zu einem dynamischen Objektmodell ermöglicht werden.

Sequenzen mit einer Person, die ein rotes Oberteil trägt, wurden in die Evaluation aufgenommen, weil die Farbrepräsentation solcher Kleidungsstücke der Repräsentation von Hautfarben ähnlich ist und das Oberteil zudem direkt an den Gesichtsbereich angrenzt. Der Standard-CAMShift ist nicht in der Lage, diese Situationen erfolgreich zu bewältigen. Ähnliche Probleme existieren bei gelben oder orangefarbenen Kleidungsstücken. Es soll geprüft werden, ob die höhere Auflösung des zweidimensionalen HS-Histogramms unserer Erweiterung (gegenüber dem eindimensionalen H-Histogramm des Standard-CAMShifts) in der Lage ist, diese Situationen zu meistern.

Annotierte Videobilder

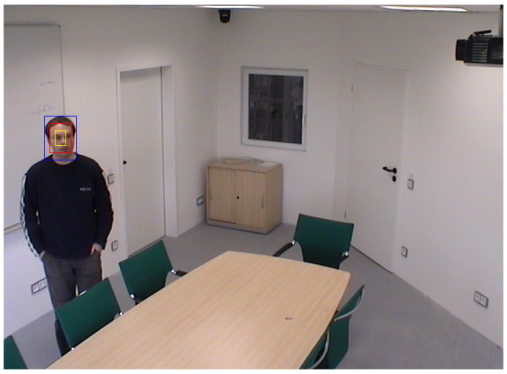
In Abbildung 6.13 sind annotierte Einzelbilder zu sehen, bestehend aus einem äußeren und einem inneren Rechteck. Bei der Evaluation Trackers zählt das Objekt als erfolgreich getrackt, wenn das Trackingergebnis zwischen diesen beiden Rechtecken liegt.

Detektordaten

Die Initialisierung des Trackers erfolgt auf Basis der Ergebnisse des Detektors, um der Abhängigkeit der Module im Gesamtsystem Rechnung zu tragen. Für das Detektormodul werden dabei folgende Einstellungen verwendet:

- SkinThreshold $= 0,7$
- scaleSaveDetectedFacesX $= 0,4$
- scaleSaveDetectedFacesY $= 0,4$

(a) Annotiertes Einzelbild 329 aus Sequenz 1



(b) Annotiertes Einzelbild 253 aus Sequenz 1

Abbildung 6.13: Beispielhafte Darstellung der annotierten Videodaten, bestehend aus dem äußeren blauen und inneren gelben Rechteck. Diese Koordinaten werden während der Evaluation mit den Ergebnissen des Trackings – rotes Rechteck – verglichen. Das Tracking gilt als erfolgreich, wenn kein Schnitt zwischen dem Rechtecken besteht.

- scalingImage $= 0,5$
- Die übrigen Einstellungen des Detektors entsprechen den Standardwerten.

Die Skalierung des Bildes ist auf 0.5 festgesetzt, um die Gesichtsdetektion auf den Videosequenzen zu beschleunigen. Bei der Initialisierung des Trackers, der auf der vollen Bildgröße arbeitet, werden die vom Detektor ermittelten Koordinaten auf die volle Bildgröße umgerechnet. Die detektierten Gesichtsrechtecke werden zudem mit einem Wert von 0.4 auf der X- und Y-Achse skaliert, um die Bildausschnitte auf den Kopfbereich zu beschränken und Hintergrundbereiche abzuschneiden. Dies verhindert, dass Farben des Hintergrunds in die Histogramme des Objektmodells einfließen, und das Objekt bereits zum Zeitpunkt der Initialisierung nicht zuverlässig getrackt wird. Auch das ViPer-Gesamtsystem führt im Betrieb diese Skalierung der Detektionsrechtecke zur Initialisierung eines Trackers durch. In Abbildung 6.14 ist anhand der Sequenz 1 beispielhaft dargestellt, wie die Detektionen auf den Sequenzen aussehen. Für alle Evaluierungssequenzen werden solche Detektionsergebnisse ermittelt und diese Ausschnitte für die Initialisierung des Trackers genutzt.

6.2.2 Ergebnisse

In diesem Abschnitt werden die Evaluationsergebnisse der beiden Trackingalgorithmen (CAMShift mit und ohne Erweiterungen) vorgestellt. In 6.2.3 folgt ein Fazit der einzelnen Ergebnisse aus diesem Abschnitt.

(a) 15 (b) 141 (c) 246 (d) 328 (e) 477

Abbildung 6.14: Beispielhaft sind hier die Detektionsergebnisse, in original Größe, für die Einzelbilder 15, 141, 246, 328 und 477 der Sequenz 1 dargestellt. Diese Ausschnitte werden zur Initialisierung des Trackers während der Evaluation genutzt.

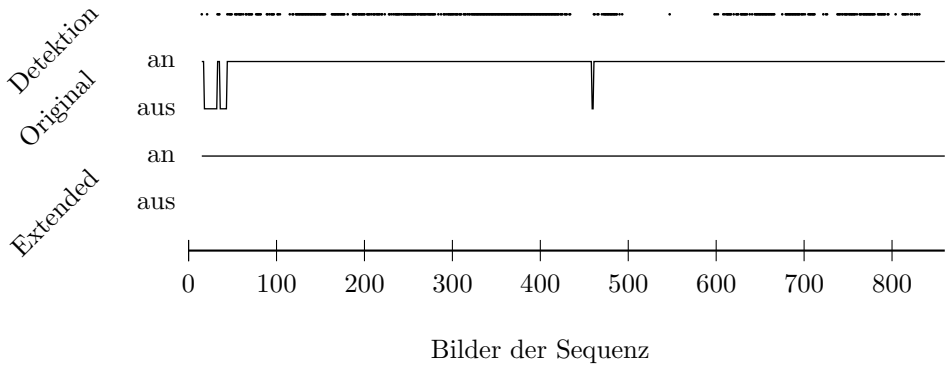


Abbildung 6.15: Der erweiterte CAMShift trackt von der ersten Detektion, bis zum Ende der Sequenz durch. Der ursprüngliche Algorithmus trackt erst ab der Detektion in Bild 44. Kurzzeitig verliert er das Objekt in Bild 459, kann aber mit der Detektion aus Bild 461 weiter tracken

Sequenz 1

In der ersten untersuchten Sequenz kommen keine vollen Kopfdrehungen vor, so dass stets ein Teil des Gesichtes von der Kamera erfasst ist. Die Sequenz besteht aus 861 Einzelbildern und hat eine Länge von 172 Sekunden. Beide Tracker sollten auf dieser Sequenz ein robustes Tracking gewährleisten. In Diagramm 6.15 ist das Ergebnis der beiden Trackingverfahren auf dieser Sequenz zu sehen.

Es fällt auf, dass der erweiterte CAMShift-Algorithmus das Objekt ab der ersten Detektion verfolgt, während der originale CAMShift-Algorithmus erst ab der vierten Detektion mit dem Tracking beginnt. Der erweiterte Algorithmus ist durch sein dynamisches und höher aufgelöstes HS-Histogramm bei der Initialisierung sehr robust. Der Standard-CAMShift auf Basis des H-Histogramms hingegen hat Schwierigkeiten das Objekt bei der Initialisierung zu erfassen, sobald sich ähnlichfarbene, sehr helle oder dunkle Regionen (d.h. Bereiche mit ähnlichen H-Werten und starkem Rauschen im H-Kanal) in der Nähe des Detektionsrechtecks befinden.

Des Weiteren verliert der Standard-CAMShift das Objekt in Bild 459, wird aber mit der Detektion in Bild 461 erneut initialisiert. Der CAMShift-Algorithmus ohne Erweiterungen erreicht somit auf dieser Sequenz eine Objektverfolgung über 96, 8 Prozent der

(a) 3 (b) 144 (c) 175

(d) 269 (e) 301

Abbildung 6.16: Beispielhafte Darstellung der zweiten Sequenz. Es sind die Einzelbilder 3, 144, 175, 269 und 301 zu sehen. Die volle Kopfdrehungen zeigt Bild 175.

Zeit, wohingegen der erweiterte Algorithmus dieses Sequenz zu 100 Prozent erfolgreich bewältigt.

Sequenz 2

Dies ist eine lange Videosequenz, in der volle Kopfdrehungen vorkommen. Sie besteht aus 536 Einzelbildern, einige davon sind in Abbildung 6.16 beispielhaft dargestellt. Zu erwarten ist, dass der originale CAMShift-Algorithmus das Objekt verliert, sobald kein Gesicht und somit keine Hautfarbe im Bild vorhanden ist. Der von uns erweiterte Algorithmus hingegen sollte das Objekt nicht verlieren, weil sich das dynamische Histogramm der Farbänderungen des Objekts anpasst. Das Ergebnis der Evaluation ist in Diagramm 6.17 zu sehen.

Wie erwartet, trackt der erweiterte Algorithmus das Objekt in dieser Sequenz deutlich besser als der ursprüngliche CAMShift. Bei der ersten vollen Kopfdrehung (Bild 175) verliert der Originalalgorithmus das Objekt und kann dies auch zu einem späteren Zeitpunkt nicht weiterverfolgen, da keine Detektion für eine robuste Neuinitialisierung ausreicht: Das Objekt wird stets sofort erneut verloren. Der erweiterte CAMShift verliert das Objekt erst in Bild 402, als die Person sich erneut dreht. Der Tracker verfolgt ab hier das Oberteil der Person, da diesem in der Projektion des dynamischen Histogramms hohe Wahrscheinlichkeiten zugeordnet werden. Im weiteren Verlauf wird der Tracker nicht erneut initialisiert, weil keine Detektion mehr vorliegt. Der erweiterte CAMShift trackt 75 Prozent der Sequenz, gemessen ab der ersten Detektion, der originale CAMShift lediglich 32, 46 Prozent.

Abbildung 6.17: Der originale CAMShift kann auf Basis der Gesichtsdetektion in Bild 1 nicht initialisiert werden. Erst mit der zweiten Detektion in Bild 3 beginnt er, das Objekt bis Bild 175 zu tracken. An dieser Stelle hat sich die Person gedreht. Es sind keine hautfarbenen Bereiche des Gesichtes im Bild und das Objekt wird verloren. Der erweiterter CAMShift hingegen trackt ab der ersten Detektion bis Bild 402. Die Person dreht sich zu diesem Zeitpunkt erneut und der Tracker springt auf das Oberteil der Person

Sequenz 3

In Sequenz 3 trägt die zu trackende Person ein rotes Oberteil. Der H-Wert dieser Farbe liegt im Bereich der für Hautfarbe typischen Werte. Das Tracking auf einem statischem Farbhistogramm ist infolgedessen schwierig, da in der Projektion auf Basis des H-Histogramms das rote Oberteil nicht von dem zu trackenden Gesicht zu unterscheiden ist. Darüber hinaus enthält die Sequenz erneut eine volle Kopfdrehung. Die Sequenz hat 333 Einzelbilder und ist 66 Sekunden lang. Diagramm 6.18 zeigt die Ergebnisse.

Der Standard-CAMShift ist nicht in der Lage, die Person in dieser Sequenz zu verfolgen. Eine Unterscheidung zwischen dem roten Oberteil und dem Gesicht der Person auf Basis des statischen Farbhistogramms ist aufgrund der ähnlichen H-Werte nicht möglich. Der erweiterte CAMShift kann hingegen aufgrund des zweidimensionalen dynamischen Histogramms feingranularer zwischen Gesicht und rotem Oberteil differenzieren und ist in der Lage, das Objekt zu tracken. In Abbildung 6.19 sind die Wahrscheinlichkeitsverteilungen in der Projektion für das dynamische und statische Histogramm abgebildet. Es ist zu erkennen, dass das dynamische Histogramm den Bereichen des Oberkörpers in der Projektion lediglich sehr niedrige Wahrscheinlichkeitswerte zuordnet, das statische hingegen sehr hohe.

Der erweiterte CAMShift trackt von der ersten Detektion in Bild 31 bis Bild 310. Dort wird die Person verloren, als sie sich herumdreht. In diesem Augenblick wird das dynamische Histogramm zunächst den Farbwerten des Hinterkopfes entsprechend angepasst, das Trackingfenster springt dann jedoch auf das rote Oberteils über. Die Häufigkeitsverteilung im dynamischen Histogramm verschiebt sich ab diesem Zeitpunkt zunehmend hin zu den Farbwerten des Oberteils, und der Kopf der zu verfolgenden Person ist in

Abbildung 6.18: Der ursprüngliche CAMShift ist auf dieser Sequenz nicht in der Lage, das Objekt zu verfolgen. Der erweiterte CAMShift trackt das Objekt von der ersten Detektion in Bild 31 bis Bild 310.

der Projektion nur noch unzureichend repräsentiert. Abbildung 6.20 zeigt die Wahrscheinlichkeitprojektion des dynamischen Farbhistogramms, nachdem sich die Person herumgedreht hat.

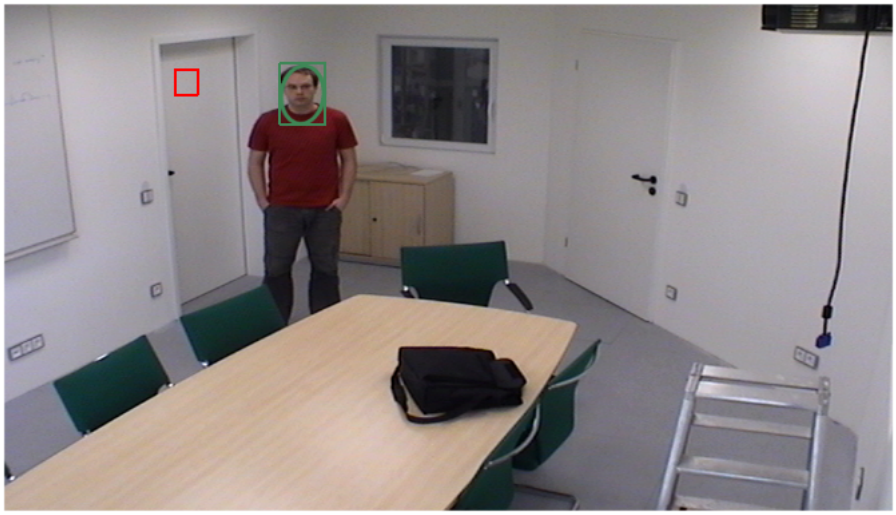
Obwohl die Person letztendlich verloren wird, trackt der erweiterte CAMShift 92, 4 Prozent der Sequenz, gemessen ab der ersten Detektion.

6.2.3 Interpretation

Die Ergebnisse zeigen auf, dass der von uns erweiterte CAMShift-Algorithmus in unserem Einsatzfeld messbar bessere Ergebnisse liefert als der originale CAMShift-Algorithmus. Es gibt Situationen in diesem Szenario, in denen der ursprüngliche Algorithmus vollständig versagt. Dies ist bei dem erweiterten Algorithmus nicht gegeben. Die Tests zeigen, dass für unser Einsatzfeld der Tracker robust genug ist, um Personen im Gesamtsystem ViPer zu verfolgen. Zudem hat der Tracker trotz der zusätzlichen Laufzeit, die zur Berechnung des dynamischen Histogramms, der zusätzlichen Projektion sowie deren Gewichtung benötigt wird, nicht seine Echtzeitfähigkeit eingebüßt. Die Initialisierung der zu verfolgenden Objekte auf Basis der Gesichtsdetektionen hat sich darüber hinaus als geeignetes Verfahren bewiesen und ermöglicht eine einfache Integration des Trackingmoduls in das ViPer-Gesamtsystem.

6.3 Evaluation des Identifizierers

In diesem Kapitel folgt die Beschreibung der Evaluierung des Identifizierers als Einzelmodul. In Abschnitt 6.3.1 wird beschrieben, welche Parameter der Identifizierung wir beispielhaft für die Evaluierung ausgewählt haben. Es folgt eine Beschreibung der Zusammensetzung der Evaluierungsdaten in Abschnitt 6.3.2 und schließlich die Darstellung der Evaluierungsergebnisse in Abschnitt 6.3.3. Abschließend werden die Ergebnisse in Abschnitt 6.3.4 interpretiert.

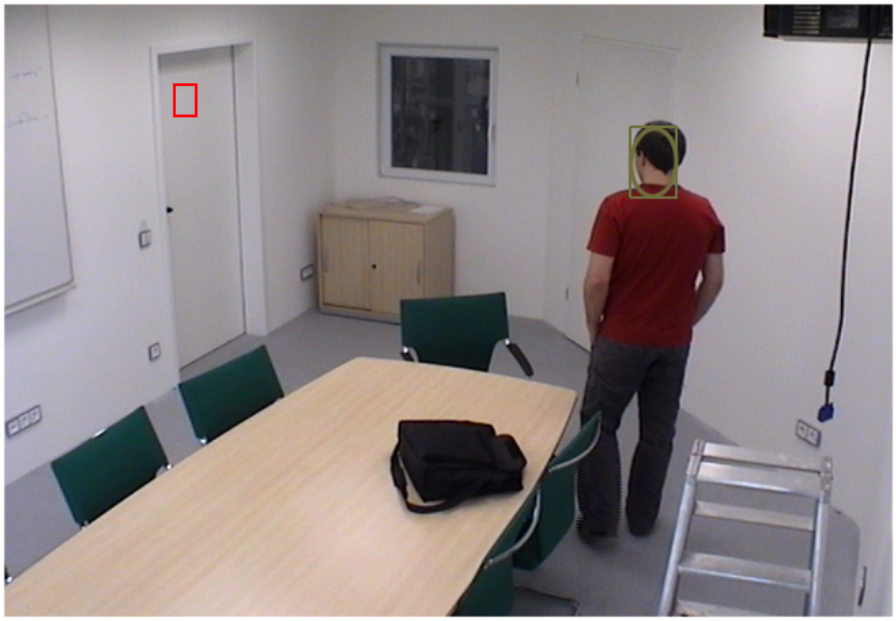


(a) Einzelbild aus einer Sequenz: Das grüne Rechteck und die Ellipse zeigen das Trackingergebnis des erweiterten Algorithmus

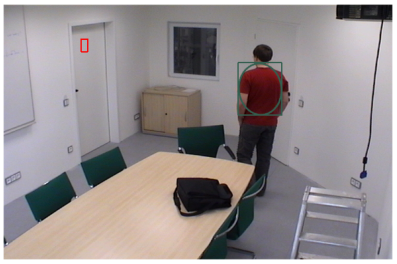
(b) Wahrscheinlichkeitsverteilung auf Basis des statischen Histogramms. Es ist zu sehen, wie der ganze Oberkörper der Person hohe Wahrscheinlichkeitswerte hat. Ein Tracking nur auf dieser Verteilung ist nicht möglich

(c) Wahrscheinlichkeitsverteilung auf Basis des dynamischen Histogramms. Der Oberkörper hat deutlich niedrigere Wahrscheinlichkeitswerte im Vergleich zum Gesicht der zu trackenden Person. Auf dieser Verteilung ist das Tracken möglich

Abbildung 6.19: Das erste Bild zeigt, das Ergebnis des Trackings des erweiterten CAMShiftAlgorithmus. Im zweiten Bild ist die Projektion des statischen, im dritten die des dynamischen Histogramms dargestellt



(a) Das grüne Rechteck und die Ellipse zeigen, wie der erweiterte CAMShift die umgedrehte Person trackt



(b) Der erweiterte CAMShift hat die umgedrehte Person verloren und verfolgt den Oberkörper

(c) Wahrscheinlichkeitsverteilung in der Projektion des dynamischen Histogramms. Der Oberkörper hat, nachdem sich das dynamische Histogramm an das rote Oberteil angepasst hat, deutlich höhere Wahrscheinlichkeitswerte als der Kopf der Person (siehe 6.20(c))

Abbildung 6.20: Das Bild zeigt das Tracken des Objektes (grüne Ellipse). Im zweiten Teil hat der erweiterte CAMShift die Person verloren und trackt den Oberkörper (olivfarbene Ellipse). Der dritte Teil zeigt die Projektion des dynamischen Histogramms, nachdem es sich den Farbwerten des Kleidungsstücks der Person angepasst hat

6.3.1 Evaluierungsparameter

Bei der Evaluierung des Identifizierers unterscheiden wir grundsätzlich zwei Arten von Parametern. Zum einen betrachten wir die *Einstellungsparameter* des EBGM-Algorithmus, zum anderen *mögliche Variationen der Eingabebilder* wie Bildqualität, Kopfdrehung und Gesichtsgröße. Diese Unterscheidung ist sinnvoll, da es sich um grundsätzlich verschiedene Parameterarten handelt. Insbesondere kann bei den Einstellungsparametern eine Evaluierung zu besseren Ergebnissen führen und somit eine Gesamtverbesserung des ViPer-Systems mit sich bringen. Auf der anderen Seite führt die Evaluierung der Erkennungsraten auf unterschiedlichen Eingaben nicht zu einer Verbesserung des Systems, sondern nur zu einer besseren Einschätzbarkeit der Stärken und Schwächen des Identifizierers. Im Folgenden sind die einzelnen Parameter im Detail aufgeführt.

Einstellungsparameter

- Auswahl der Gesichtsgraphen in der Datenbank

  Die Auswahl der Gesichtsgraphen in der Datenbank spielt eine entscheidende Rolle für die Qualität des Identifizierers. Um eine möglichst gute Erkennungsrate zu erzielen, ist es nötig, eine homogene Auswahl an Bildern für die Datenbank auszuwählen. Denn wenn nur eine Person auf dem Bild in der Datenbank deutlich lächelt, so tendieren alle zu identifizierenden Gesichtsbilder mit lächelnden Personen stärker zu dieser Person in der Datenbank. Gleiches gilt beispielsweise für die Kopfdrehung oder die Bildqualität. Informelle Tests haben ergeben, dass auch bei möglichst gleicher Bildkonfiguration einzelne Personen in der Datenbank dominant sind, d. h. dass diese Personen bei Fehlidentifikationen häufiger vorkommen. Ein Ziel der Auswahl der Gesichtsgraphen war es somit, dominante Gesichter zu vermeiden und eine möglichst gleichmäßige Erkennungsrate über alle Personen zu erzielen. Die Auswahl der Gesichtsgraphen wird im Folgenden nicht weiter evaluiert, da zum einen nicht genügend verschiedene Bilder für die Datenbank vorliegen (da diese zu einem anderen Zeitpunkt aufgenommen worden sein müssen als die Evaluierungsbilder) und zum anderen die Optimierungsziele nicht klar umrissen sind. So stellt sich beispielsweise die Frage, ob es besser ist, eine möglichst gute Erkennungsrate zu erzielen oder eine möglichst gleich gute Erkennung aller Personen auch auf Kosten einer absolut schlechteren Erkennungsrate zu bevorzugen ist. Alle Evaluierungen in diesem Kapitel sind daher mit den gleichen Gesichtsgraphen erstellt worden.

- Benutzung eines zusätzlichen Lokalisierungs-Bunchgraphen

  Wie in Kapitel 4.3 beschrieben, bietet der Identifizierer die Möglichkeit, einen Lokalisierungs-Bunchgraphen zu verwenden. Da im ViPer-Szenario die Bilder nicht manuell kontrolliert an den Identifizierer übergeben werden, besteht die Hoffnung, dass ein Lokalisierungs-Bunchgraph die Erkennungsrate verbessert. Da die Verwendung eines Lokalisierungs-Bunchgraphen eine Erweiterung des einfachen EBGM-Algorithmus ist, haben wir diesen Parameter für die Evaluierung ausgewählt.

- Parameter des Lokalisierungsalgorithmus

  Der Lokalisierungsalgorithmus ist an vielen Stellen frei einstellbar. Wir untersuchen den Einfluss einiger dieser Einstellungen auf die Erkennungsrate. Im ersten Schritt des Lokalisierungsalgorithmus, der »Approximation der Gesichtsposition«, wird der Schwerpunkt des Gesichtsgraphen innerhalb eines mittig liegenden Quadrats vorgegebener Größe mit einer festen Schrittweite bewegt. Die Parameter `S1_Square` und `S1_Increment` geben diese Platzierungsmöglichkeiten für den Durchschnittsgraphen vor. Im zweiten Schritt, der »Anpassung der Position und Größe«, wird die Skalierung des Graphen über den Parameter `S2_ScaleFactor` bestimmt. Schließlich ermöglicht der Parameter `SR_UseRotation` die Anwendung der Rotation des Graphen in der Bildebene um einen Winkel von bis zu `SR_MaxAngle` und Schritten von `SR_Increment`. Die Möglichkeit der Rotation wurde im Kapitel 4.3 beschrieben. Die hier beschriebenen und untersuchten Parameter stellen nur eine Auswahl der Einstellungsmöglichkeiten des EBGM-Algorithmus dar. Wir untersuchen den Einfluss der Parameter auf die Erkennungsrate sowie auf die Laufzeit des Algorithmus.

- Abbruchschwellenwerte

  Der Identifizierer muss, um das Gesamtsystem fehlertolerant zu machen, auch mit Bildern umgehen können, die keine Gesichter enthalten. Es ist möglich diese Bilder schon in den frühen Schritten des Lokalisierungs-Algorithmus zu erkennen und zu verwerfen, um Rechenzeit zu sparen. Dazu werden Schwellenwerte für die verschiedenen Schritte des Lokalisierungs-Algorithmus festgelegt (siehe Kapitel 4.3). Wir untersuchen die Wirksamkeit dieser Abbruchschwellenwerte in einem unabhängigen Evaluierungsschritt.

- Rückweisungsschwellenwerte

  Statt jedes erhaltene Gesicht einer Person aus der Datenbank zuzuordnen, muss der Identifizierer auch in der Lage sein, unbekannte Personen als solche zu klassifizieren. Natürlich kann es dabei auch passieren, dass eine Person aus der Datenbank zu Unrecht als unbekannt klassifiziert wird. Das genaue Vorgehen zur Klassifizierung als unbekannte Person wird im Abschnitt »Identifizierung« in Kapitel 4.2.2 erläutert. Wir beschreiben durch die Evaluierung der Rückweisungsschwellenwerte, wie diese auf sinnvolle Weise festgelegt werden.

Mögliche Variationen der Eingabebilder

- Gesichtsgröße

  Da die in den Bildern enthaltenen Informationen direkt von der Bildgröße abhängig sind, erwarten wir eine Abhängigkeit der Erkennungsrate von der Größe der Bilder. Diese Information ist gerade für das ViPer-Szenario von außerordentlichem Interesse, da es wünschenswert ist, die Personen direkt beim Betreten des Raumes zu identifizieren. Da die Kamera jedoch in der gegenüberliegenden Ecke des Raumes angebracht ist und wir keine aktive Kamerasteuerung implementiert haben, sind

Abbildung 6.21: Die vier verschiedenen Gesichtsgrößen, die in der Evaluierung betrachtet werden. Die Bilder sind in Abständen von 1,7 m, 2,9 m, 3,1 m und 4,3 m von der Kamera aufgenommen.

Abbildung 6.22: Die drei verschiedenen Kopfdrehungen, die in der Evaluierung betrachtet werden. Diese Bilder sind im Abstand von 1,7 m von der Kamera aufgenommen und stellen ein frontales Gesicht, ein leicht gedrehtes Gesicht (zwischen 10 und 25 Grad) und ein etwas stärker gedrehtes Gesicht (zwischen 25 und 40 Grad) dar.

Abbildung 6.23: Verschiedene alternative Gesichtsausdrücke einer Testperson. Diese Bilder sind im Abstand von 1,7 m von der Kamera aufgenommen.

gerade diese Bilder recht klein. Aufgrund der Passivität der Kameras sind bei ViPer die Größe der Bilder und die Entfernung der Personen von der Kamera korreliert. Wir interessieren uns also für den Schwellenwert der Entfernung, ab dem eine gute Identifikation möglich ist. In Abbildung 6.21 sind die verschiedenen Gesichtsgrößen dargestellt, die wir in der Evaluierung betrachten.

- Kopfdrehung

  Eine Anforderung an die Interaktion mit dem intelligenten Raum war die Vermeidung unnötiger Restriktionen für die anwesenden Personen. Dazu gehört auch, dass nicht verlangt werden kann, dass die Personen zur Identifizierung direkt in die Kamera blicken. Für den Detektor stellen leichte Kopfdrehung keine große Schwierigkeit dar, so dass das Weltmodell auch gedrehte Köpfe erhält und zum Identifizierer weiterleitet. Uns interessiert daher die Abhängigkeit der Erkennungsrate von der Kopfdrehung. In Abbildung 6.22 sind die verschiedenen Kopfdrehungen dargestellt, die wir in der Evaluierung betrachten.

-

- Gesichtsausdruck

  Auch der Gesichtsausdruck der Personen im Raum kann starken Schwankungen unterliegen. Für das Identifizierer-Modul bedeutet dies, dass es beispielsweise mit offenen und geschlossenen Mündern und Augen zurechtkommen muss. Wir betrachten die Erkennungsrate von neutralen Gesichtsausdrücken und die Erkennungsrate von alternativen Gesichtsausdrücken. In Abbildung 6.23 sind verschiedene Gesichtsausdrücke dargestellt, die wir in der Evaluierung betrachten. Für die Aufnahme der Gesichtsausdrücke gab es keine genaue Vorgabe, so dass sich die Art der alternativen Gesichtsausdrücke von Person zu Person unterscheidet.

- Bildqualität (Interlaced-Effekt)

  Aufgrund der Konfiguration der Kameras im intelligenten Raum schwankt die Bildqualität zum Teil sehr stark. Die Hauptschwierigkeit dabei sind Bilder mit einem

Abbildung 6.24: Verschiedene Interlaced-Bilder einer Testperson. Diese Bilder sind im Abstand von 1,7 m von der Kamera aufgenommen und stellen ein frontales Bild und ein leicht gedrehtes Bild mit leichten Interlaced-Effekten dar. Es sind nach Möglichkeit Bilder mit geringem Interlaced-Effekt ausgewählt worden, um die Identifizierung grundsätzlich zu ermöglichen.

Interlaced-Effekt. Dieser Effekt beruht darauf, dass jedes Bild aus zwei Halbbildern zusammengesetzt wird. Starke Bewegungen führen dann dazu, dass es in den zusammengesetzten Bilder sichtbare Interlaced-Streifen gibt. Wir untersuchen die Abhängigkeit der Identifikation von diesem Effekt. In Abbildung 6.24 sind die verschiedenen Interlaced-Bilder dargestellt, die wir in der Evaluierung betrachten.

6.3.2 Evaluierungsdaten

Für die Evaluierung der *Einstellungsparameter* haben wir ein umfangreiches Evaluierungsset zusammengestellt, das die verschiedenen Problematiken unseres Szenarios abdeckt. Im Folgenden wird die genaue Zusammensetzung der Bilder in unserem Evaluierungsset beschrieben.

Es gibt vier Arten von Parametern, die in den Bildern variieren: Die *Entfernung* der Person zur Kamera (und damit die Größe des Bildes), die *Drehung* des Kopfes, der *Gesichtsausdruck* und die *Bildqualität*. Es erscheint nicht sinnvoll, für die Zusammensetzung des Evaluierungssets alle Parameter mit gleichem Anteil zu berücksichtigen, da dann eine eben so große Gewichtung auf sehr schwierigen (klein, nicht frontal, schlechte Qualität) wie auf einfachen Bildern (groß, frontal, gute Qualität) liegt. Für die Einzeluntersuchung des Identifizierers halten wir es daher für sinnvoller, diejenigen Bilder stärker zu gewichten, die der Identifizierer in einem reinen Identifizierungsszenario erhalten würde. Dies sind insbesondere die einfacheren Bilder. Um jedoch unser ViPer-Szenario nicht aus den Augen zu verlieren, wird ein geringer Anteil schwieriger Bilder hinzugefügt. Insgesamt befinden sich 14 Personen im Evaluierungsset, von denen jeweils 25 Bilder enthalten sind. Somit umfasst das Set 350 Gesichtsbilder. Die Zusammensetzung ist in Tabelle 6.4 darge-

	frontal	leicht gedreht	etwas stärker gedreht
1,7 m	5 neutraler Gesichtsausdruck 4 alternativer Gesichtsausdruck 1 interlaced	3 neutral 1 interlaced	3 neutral
2,9 m	2 neutraler Gesichtsausdruck	1 neutral	1 neutral
4,1 m	1 neutraler Gesichtsausdruck	1 neutral	
5,3 m	1 neutraler Gesichtsausdruck	1 neutral	

Tabelle 6.4: Zusammensetzung des Evaluierungssets bezogen auf die Entfernung zur Kamera und die Kopfdrehung. Leicht gedrehte Köpfe haben einen Winkel zwischen 10 und 25 Grad zur Kamera, etwas stärker gedrehte Köpfe einen Winkel von 25 bis 40 Grad.

Entfernung zur Kamera		Kopfdrehung		Gesichtsausdruck		Bildqualität	
1,7 m	68 %	frontal	56 %	neutral	84 %	normal	92 %
2,9 m	16 %	leicht gedreht	28 %	alternativ	16 %	Interlaced	8 %
4,1 m	8 %	stärker gedreht	16 %				
5,3 m	8 %						

Tabelle 6.5: Verteilung der Bilder auf die Ausprägungen der Parameter.

stellt. Dabei ergibt sich eine prozentuale Verteilung der Bilder auf die Ausprägungen der Parameter entsprechend Tabelle 6.5. Damit haben wir unter kontrollierten Bedingungen sowohl einen großen Umfang der möglichen Bilder abgebildet, als auch eine Gewichtung im Sinne einer Einzelmodul-Evaluierung vorgenommen. Die Darstellung der Ergebnisse findet sich in Abschnitt 6.3.3.

Für die Evaluierung der verschiedenen *Variationen der Eingabedaten* haben wir für jede untersuchte Variation ein eigenes Evaluierungsset zusammengestellt. So werden beispielsweise im Evaluierungsset für die Gesichtsgröße ausschließlich Frontalbilder mit normalem Gesichtsausdruck berücksichtigt, die in verschiedenen Entfernungen zur Kamera aufgenommen wurden. Durch diese Vorgehensweise versuchen wir zu verhindern, dass andere als der untersuchte Parameter einen zu großen Einfluss auf die Erkennungsrate haben. Jedes Set besteht aus drei Bildern pro Person und Parameterausprägung. Die betrachteten Parameterausprägungen entsprechen denen in Tabelle 6.5.

Schließlich werden die *Abbruchschwellenwerte* und *Rückweisungsschwellenwerte* in unabhängigen Untersuchungen festgelegt. Zur Festlegung der Abbruchschwellenwerte benutzen wir zusätzlich zum umfangreichen Evaluierungsset mit unseren Testpersonen ein Set mit 350 Bildern auf denen sich keine Gesichter befinden. Diese Bilder stellen allesamt Fehldetektionen des Detektors dar und sind somit in unserem Szenario von Bedeutung. Zur Festlegung der Rückweisungsschwellenwerte verwenden wir schließlich ein

	ohne Lokalisierungs-BG	mit Lokalisierungs-BG
Erkennungsrate	165 von 350 Bildern erkannt 47 %	168 von 350 Bildern erkannt 48 %
Durchschnittliche Rechenzeit	2,6 sek	5,2 sek

Tabelle 6.6: Vergleich des EBGM-Algorithmus mit und ohne Lokalisierungs-Bunchgraph.

zusätzliches Evaluierungsset, bestehend aus 214 Bildern von Personen, die nicht in der Datenbank enthalten sind. Das Ziel der Evaluierung ist es, Rückweisungsschwellenwerte zu bestimmen, die diese unbekannten Personen zurückweisen und gleichzeitig weiterhin eine Identifizierung der bekannten Personen erlauben.

Bei allen Evaluierungen wird, ausgehend von einem festen Parametersatz, nur der Evaluierungsparameter variiert. Im festen Parametersatz verwenden wir keinen Lokalisierungs-Bunchgraph und es wird keine Rückweisung und kein Abbruch vorgenommen, um die Ergebnisse nicht zu verfälschen. Der Lokalisierungs-Algorithmus ist außerdem auf die Standardparameter eingestellt. In der Datenbank sind die Gesichtsgraphen der 14 Personen abgelegt, die auch im Evaluierungsset enthalten sind. Dabei wurden die Graphen an Gesichtsbildern erstellt, die etwa drei Monate vor den Evaluierungsbildern aufgenommen wurden. Der Bunchgraph selbst wurde ebenfalls mit Bildern trainiert, die etwa drei Monate vor den Evaluierungsbildern aufgenommen wurden. Insgesamt beruht das Training des Bunchgraphen auf den Bildern von 52 Personen.

Alle Rechenzeiten beziehen sich auf durchschnittliche Rechenzeiten für die Identifikation eines einzelnen Bildes. Die Berechnungen wurden auf einem Rechner mit einem 1,83 Ghz Intel Dual Core Prozessor mit 1 GB Hauptspeicher durchgeführt.

6.3.3 Ergebnisse

In Tabelle 6.6 ist die Erweiterung um den Lokalisierungs-Bunchgraphen gegenüber dem ursprünglichen EBGM-Algorithms ohne Lokalisierungs-Bunchgraphen anhand des Evaluierungssets aus Abschnitt 6.3.2 dargestellt. Die deutlich erhöhte Laufzeit des EBGM mit Lokalisierungs-Bunchgraphen ist nicht verwunderlich, da in diesem Fall der Lokalisierungs-Algorithmus für jedes Bild zweimal ausgeführt werden muss. Insgesamt ist keine deutliche Verbesserung gegenüber dem einfachen Lokalisierungs-Algorithmus feststellbar, was daran liegt, dass die Lokalisierung bereits in der einfachen Variante sehr gut funktioniert.

In Tabelle 6.7 ist die Evaluierung der Parameter `S1_Square` und `S1_Increment` anhand des Hauptevaluierungssets aus Abschnitt 6.3.2 dargestellt. In dieser und den nachfolgenden Tabellen sind die Standardeinstellungen des Lokalisierungs-Algorithmus mit einem $*$ gekennzeichnet. Die Laufzeit für eine Identifizierung steigt wie erwartet mit größerem Suchquadrat und mit kleinerer Schrittweite. Überraschenderweise stellt sich die beste Erkennungsrate beim kleinsten Suchquadrat ein. Allerdings sind die Unterschiede in der Erkennungsrate nicht signifikant.

		S1_Square				
		2	**4***	**6**	**8**	**10**
S1_I	**1***	167 48 % 2,2 sek	165 47 % 2,6 sek	163 47 % 3,3 sek	161 46 % 4,3 sek	158 45 % 5,2 sek
	2	164 47 % 2,2 sek	163 47 % 2,3 sek	158 45 % 2,5 sek	157 45 % 2,6 sek	156 45 % 3,0 sek
	3	162 46 % 2,1 sek	165 47 % 2,1 sek	160 46 % 2,3 sek	159 45 % 2,3 sek	157 45 % 2,4 sek

Tabelle 6.7: Untersuchung der Parameter des ersten Schrittes des Lokalisierungsalgorithmus. Es sind jeweils die Anzahl der erkannten Bilder sowie die Erkennungsrate und die Rechenzeit angegeben. Die Größe des Suchfensters S1_Square ist auf der horizontalen und die Schrittweite S1_Increment auf der vertikalen Achse aufgetragen.

S2_ScaleFactor				
1,1	**1,2**	**1,3***	**1,4**	**1,5**
164 47 %	166 47 %	165 47 %	165 47 %	168 48 %

Tabelle 6.8: Untersuchung der Parameter des zweiten Schrittes des Lokalisierungsalgorithmus. Es sind jeweils die Anzahl der erkannten Bilder und die Erkennungsrate angegeben. Die Rechenzeiten unterscheiden sich bei diesem Parameter nicht.

In den Tabellen 6.8 und 6.9 ist die Untersuchung der Skalierung und der Rotation dargestellt. Auch bei diesen Parametern verursachen die verschiedenen Einstellungen keinen signifikanten Unterschied in der Erkennungsrate. Bemerkenswert ist, dass die Erkennungsrate sogar ohne Anwendung der Rotation ein vergleichbares Niveau erreicht. Dies hat wohl die Ursache darin, dass kaum in der Bildebene rotierte Gesichter im Evaluierungsset enthalten sind. Da diese Bilder jedoch im ViPer-Szenario auftreten können, ist die Verwendung der Rotation bei der Lokalisierung dennoch sinnvoll.

Zur Evaluierung der Abbruchparameter haben wir einen Vergleich der Anzahl der verworfenen Gesichter mit der Anzahl der verworfenen Nichtgesichter vorgenommen. Die Ergebnisse sind in Abbildung 6.25 dargestellt. Es zeigt sich, dass eine komplette Trennung der Gesichts- von den Nichtgesichtsbildern nicht möglich ist. Weitere Experimente mit Kombinationen der Abbruchschwellenwerte haben jedoch eine recht gute Trennung erbracht. Bei der Wahl von 0,87 als Abbruchschwellenwert nach dem ersten Lokalisierungsschritt, sowie 0,55 und 0,56 für die Schwellenwerte nach dem zweiten und dritten Schritt ergibt sich auf unserem Evaluierungsset eine Anzahl von 153 richtig erkannten

			SR_MaxAngle		
		ohne Rotation	**6**	**10***	**14**
SR_I	**1**		165 47 % 2,9 sek	165 47 % 3,8 sek	165 47 % 4,7 sek
	2*	164 47 % 1,6 sek	165 47 % 2,3 sek	165 47 % 2,7 sek	165 47 % 3,1 sek
	3		163 47 % 2,1 sek	165 47 % 2,3 sek	165 47 % 2,7 sek

Tabelle 6.9: Untersuchung der Parameter des Rotationsschrittes des Lokalisierungsalgorithmus. Es sind jeweils die Anzahl der erkannten Bilder sowie die Erkennungsrate und die Rechenzeit angegeben. Der maximale Rotationswinkel SR_MaxAngle ist auf der horizontalen und die Schrittweite SR_Increment auf der vertikalen Achse aufgetragen.

Gesichtern. Insgesamt 60 Gesichter wurden als Nichtgesichter verworfen, davon wären ohne Abbruch weitere 12 Bilder richtig identifiziert worden. Da jedoch deutlich mehr Fehlidentifikationen verworfen werden, steigt die Erkennungsrate auf 53 %. Aus dem Set der 350 Nichtgesichter werden nur 28 zu Unrecht als Gesichter akzeptiert.

Zur Bestimmung der Rückweisungsschwellenwerte, also der Schwellenwerte, die bestimmen ob ein Gesicht als bekannt oder als unbekannt klassifiziert wird, betrachten wir die höchste Gesichtsähnlichkeit jedes Bildes sowie den Abstand zwischen der höchsten und zweithöchsten Gesichtsähnlichkeit. In Abbildung 6.26 sind diese beiden Werte für unser Evaluierungssets, getrennt nach richtigen Identifikationen und falschen Identifikationen, dargestellt. Zusätzlich sind die entsprechenden Werte von unserem Evaluierungsset unbekannter Personen dargestellt. Es zeigt sich, dass eine komplette Trennung der Mengen nicht möglich ist. Je nach Wahl der Rückweisungsschwellenwerte kann der Trade-off zwischen Erkennungsrate und Gesamtanzahl richtig erkannter Bilder beeinflusst werden. Da wir in unserem Szenario an einer hohen Zuverlässigkeit des Identifizierers interessiert sind, legen wir die Rückweisungsschwellenwerte wie folgt fest: Die höchste Graphenähnlichkeit $\mathcal{S}'_{\mathcal{G}}$ muss mindestens 0,88 und der Abstand zwischen der höchsten und der zweithöchsten Graphenähnlichkeit muss mindestens 0,01 betragen. Außerdem gibt es eine lineare Trennung mit einer Steigung von -0,7, die durch den Punkt $(0{,}91; 0)$ geht. Diese Schwellenwerte sind ebenfalls in Abbildung 6.26 eingetragen und verdeutlichen dort die gewünschte Trennung der Klassen. Auf unsere Evaluierungssets wirken sich die gewählten Schwellenwerte wie folgt aus: von den 350 Bildern aus dem Set mit bekannten Personen werden 247 Bilder zurückgewiesen, 23 Personen falsch und 80 Personen richtig identifiziert. Das entspricht einer Erkennungsrate von 78 % auf den akzeptierten Bildern. Der Anteil der Bilder, die akzeptiert werden liegt bei 23 %. Vom Evaluierungsset mit 214 Bildern von unbekannten Personen werden 178 Bilder als unbekannt zurückgewiesen und 36 Bilder einer falschen Person aus der Datenbank zugewiesen.

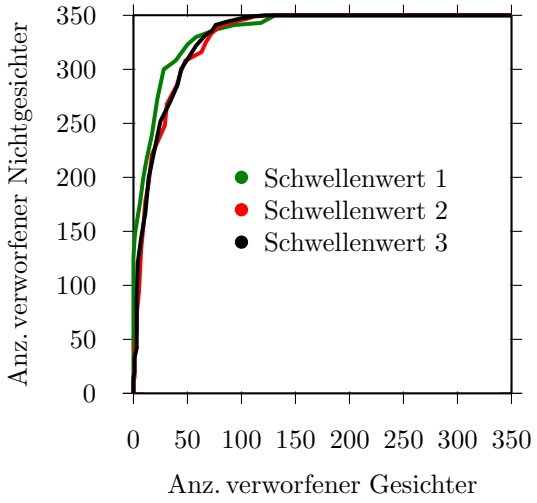


Abbildung 6.25: Analyse der drei Abbruchschwellenwerte. Es ist die Anzahl der verworfenen Gesichter gegenüber der Anzahl der verworfenen Nichtgesichter aufgeführt. Beide Sets bestehen aus 350 Bildern, insgesamt wurden also 700 Bilder berücksichtigt. Die grüne Kurve zeigt den ersten, die rote Kurve den zweiten und die schwarze Kurve den dritten Abbruchschwellenwert.

Dies entspricht einer Erfolgsquote von 83 %.

Die Untersuchung der Abhängigkeit der Erkennungsrate von der Variation der Eingabebilder ist in Tabelle 6.10 dargestellt. Es zeigt sich im Wesentlichen das erwartete Verhalten: Die Erkennungsrate sinkt mit abnehmender Bildgröße, mit zunehmender Kopfdrehung, mit verändertem Gesichtsausdruck und mit abnehmender Bildqualität. Abweichungen von dieser pauschalen Beobachtung finden sich bei einer Entfernung von 4,1 m zur Kamera. Allerdings ist diese erneute Zunahme der Erkennungsrate nicht sonderlich hoch, so dass man eher davon ausgehen kann, dass andere Parameter (wie Beleuchtung und Bildqualität) das Ergebnis beeinflusst haben. Das überraschend gute Ergebnis bei den Interlaced-Bildern lässt sich dadurch erklären, dass die Stärke des Interlaced-Effekts bei den Bildern sehr verschieden war. Bei etlichen Bildern trat nur ein sehr schwacher Interlaced-Effekt auf, der die Identifizierung offenbar nicht sonderlich stark beeinflusst.

6.3.4 Interpretation

Informelle Tests haben ergeben, dass der Einfluss der Wahl der Gesichtsgraphen auf die Erkennungsrate deutlich stärker ist, als beispielsweise die Einstellungsparameter des Lokalisierungsalgorithmus. Aufgrund der ausgesprochenen Komplexität und des hohen Aufwands, der für eine Evaluierung der Wahl der Gesichtsgraphen nötig wäre, haben wir darauf verzichtet, diesen Aspekt genauer zu untersuchen. Unter Berücksichtigung dieser Tatsache kann aus den Ergebnissen für die anderen Einstellungsparameter nur eine Tendenz bezüglich des Einflusses auf die Erkennungsrate ausgemacht werden. Insbeson-

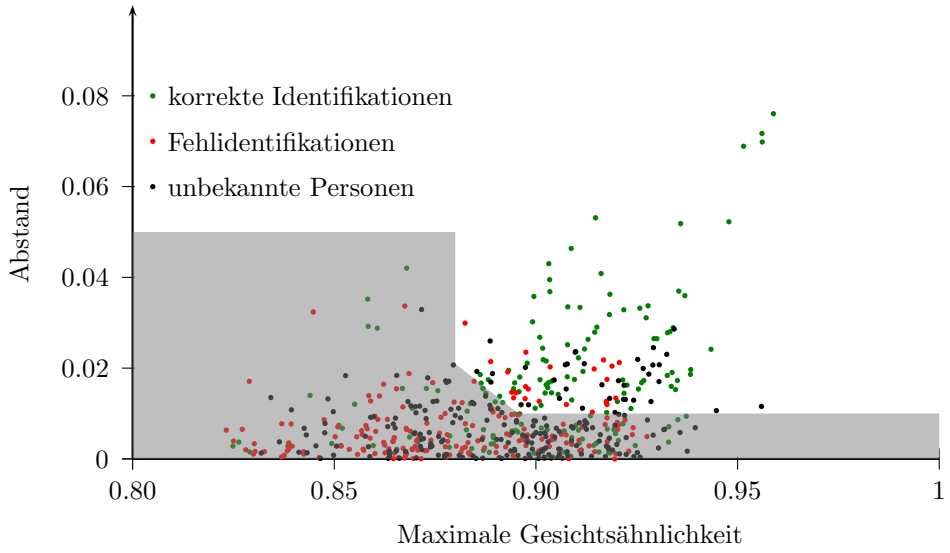


Abbildung 6.26: Festlegung der Rückweisungsparameter. Die maximale Gesichtsähnlichkeit ist gegenüber dem Abstand zwischen höchster und zweithöchster Gesichtsähnlichkeit aufgetragen. Richtige Identifikationen sind in grün, falsche Identifikationen in rot und unbekannte Personen in schwarz dargestellt. Die Rückweisungsschwellenwerte sind durch die Grenzlinien des grauen Bereichs dargestellt. Alle Bilder im grauen Bereich werden zurückgewiesen.

Entfernung zur Kamera		Kopfdrehung		Gesichtsausdruck		Bildqualität	
1,7 m	67 %	frontal	67 %	neutral	67 %	normal	67 %
2,9 m	57 %	leicht gedreht	38 %	alternativ	22 %	Interlaced	40 %
4,1 m	62 %	stärker gedreht	26 %				
5,3 m	17 %						

Tabelle 6.10: Untersuchung der Abhängigkeit der Erkennungsrate von der Variation der Eingabebilder.

dere zeigt sich aber, dass der Lokalisierungsalgorithmus durch geeignete Parameterwahl beschleunigt werden kann, ohne die Erkennungsrate wesentlich zu verschlechtern.

Aussagekräftiger sind die Untersuchungen der Variationen der Eingabedaten. Wie nicht anders zu erwarten war, korreliert die Erkennungsrate deutlich mit Parametern wie der Kopfdrehung und der Bildqualität. Eingeschränkt auf die besten Bedingungen in unserem Szenario (frontale Bilder aus geringer Entfernung mit neutralem Gesichtsausdruck) erreicht der Identifizierer eine Erkennungsrate von 67 %. Im Vergleich zu Ergebnissen aus der Literatur [TP91, WFKM96] wirkt dieses Ergebnis auf den ersten Blick

relativ schlecht. Es sollte bei der Bewertung jedoch nicht vergessen werden, dass die Bilder mit analogen Videokameras aufgenommen wurden und in der Bildqualität, was Schärfe und Auflösung angeht, mit in der Literatur verwendeten Bilddatenbanken nicht mithalten können. Vergleichbarkeit unserer Implementierung könnte dadurch erreicht werden, die Evaluierung anhand einer bekannten Bilddatenbank durchzuführen. Hierfür müsste jedoch ein komplett neuer Bunchgraph trainiert werden, da dieser natürlich vollständig auf unseren Trainingsbildern beruht.

Im laufenden Betrieb von ViPer sollten Bilddaten in einer mit dem Evaluierungsset vergleichbaren Qualität und Quantität zum Identifizierer gesendet werden. Eingestellt mit den oben genannten Abbruchs- und Rückweisungsschwellenwerten würde der Identifizierer somit etwa jedes vierte Bild zur Identifikation akzeptieren und mit einer Erkennungsrate von 78 % verarbeiten. Die Chancen einer erfolgreichen Identifizierung steigen natürlich, wenn sich die Person in eine für den Identifizierer günstige Position begibt und einen kleinen Moment darin verharrt. Allerdings widerspricht es den Anforderungen an das Projekt, den Benutzern derartige Restriktionen aufzuerlegen.

6.4 Evaluation des Gesamtsystems

Aufgabe des Gesamtsystems ist es, festzustellen, ob sich Personen in dem intelligenten Raum befinden und welche Personen das sind. Dies ist nur durch ein Zusammenwirken der einzelnen Module möglich. Mit Hilfe des Detektors wird festgestellt, ob Personen anwesend sind. Der Identifizierer liefert die Identitäten anwesender Personen. Um verschiedene Bilder der gleichen Person einander zuordnen zu können, verfolgt der Tracker Personen über die Zeit. Möchte man zum Beispiel wissen, welche Person soeben den Raum betreten hat, muss ein entsprechendes Detektorergebnis der Auslöser für eine Anweisung an den Identifizierer sein. Solche Koordinationsaufgaben übernimmt im ViPer-System das Weltmodell.

Die Evaluation der einzelnen Module sagt nur bedingt etwas über die Leistungsfähigkeit des Systems aus. Laufen alle Module im Gesamtsystem zusammen, entstehen neue Abhängigkeiten unter den Modulen. So ist zum Beispiel aus der Architektur ableitbar, dass nur dann Identifizierer und Tracker überhaupt gestartet werden, wenn der Detektor ein Gesicht gefunden hat. Ohne diese Information gibt es auch keinen Bildausschnitt, der an Tracker oder Identifizierer als Argument übergeben werden kann. Des Weiteren liefert der Identifizierer abhängig von der Qualität der Gesichtsbilder, die ihm zugesandt werden, Fehlidentifikationen. Wird die falsch erkannte Person dazu auch noch vom Tracker verloren und anschließend neu detektiert, kann die Fehlidentifikation nicht korrigiert werden. Dies zeigt, dass eine Evaluation der Integration aller Module notwendig ist.

6.4.1 Evaluationskriterien

Es wird im Folgenden also das gesamte ViPer-System als Black Box hinsichtlich der Anforderung, anwesende Personen innerhalb eines intelligenten Raumes zu erkennen, bewertet. Ein *Testfall* ist eine Person, die alleine oder gleichzeitig mit anderen Personen den Raum betritt, einige Sekunden oder Minuten darin verweilt und den Raum wieder

Abbildung 6.27: Evaluationsdaten: Person betritt intelligenten Raum

verlässt. Für jeden Testfall interessiert uns nun, ob das System einerseits erkennt, dass eine Person den Raum betritt und andererseits, ob diese Person korrekt identifiziert wird. Außerdem wäre es wünschenswert, dass die Positionen der anwesenden Personen dem System zu jeder Zeit bekannt sind.

6.4.2 Evaluationsdaten

Zum Zweck der Evaluation haben wir mehrere Videosequenzen mit den in dem Raum installierten Kameras aufgenommen. Ziel dabei war es, möglichst authentische, dem oben genannten Anwendungsgebiet entsprechende Situationen zu erzeugen. Die Sequenzen zeigen also meist mehrere Personen, die den Raum kurz aufeinander folgend betreten, eventuell Platz nehmen, sich kurz unterhalten und den Raum wieder verlassen. Um Fehler der einzelnen Module zu provozieren, haben wir Sequenzen erzeugt, die sich schnell bewegende Personen zeigen, sowie Personen in unterschiedlichen Abständen zur Kamera. Auszüge aus einer durchschnittlichen Sequenz zeigen die Abbildungen 6.27, 6.28 und 6.29.

Zur Evaluation wird ViPer also nicht »live« betrieben, sondern der Kameramanager wird angewiesen, Bilder aus einem Verzeichnis, anstatt von einer Kamera zu laden.

6.4.3 Ergebnisse

In Tabelle 6.11 werden alle Testfälle nach Erfüllung der Evaluationskriterien gezählt. In der ersten Zeile ist abzulesen, bei wie vielen Testfällen eine Detektion stattgefunden hat und wie viele Testfälle 50, 100, 150, 200 Frames bzw. bis zum Verlassen des Raumes verfolgt werden. Ein Eintrag einer Spalte kann nicht größer sein, als sein Vorgängerwert.

Abbildung 6.28: Evaluationsdaten: Weitere Personen sind anwesend bzw. betreten den Raum

Abbildung 6.29: Evaluationsdaten: Personen verlassen den Raum

	Detektiert	Verfolgt bis Frame: 50	100	150	200	Ende
18 Testfälle	17	16	15	12	12	10
korrekt identifiziert nach:	2	3	3	3	4	4

Tabelle 6.11: Jeweils Anzahl Testfälle, die detektiert und 50 bis 200 Frames (bzw. bis zum Verlassen des Raumes) verfolgt oder identifiziert werden. Von 18 Testfällen werden 17 detektiert, 10 bis zum Verlassen des Raumes verfolgt und insgesamt vier korrekt identifiziert. Zwei werden sofort beim Betreten des Raumes identifiziert, ein weiterer kann nach 50 und noch ein weiterer nach 200 Frames identifiziert werden.

Je mehr große Werte weiter rechts stehen, umso besser bzgl. der Kriterien verhält sich das Gesamtsystem. Die zweite Zeile gibt an, wie viele Testfälle nach wie vielen verstrichenen Frames korrekt identifiziert werden. Hier können die Werte nicht abfallen. Testfälle, die schon kurz nach der Detektion korrekt identifiziert werden, sind dies nach 50, 100 usw. Frames immer noch.

6.4.4 Interpretation

Es werden fast alle Testfälle detektiert und über viele Frames, meist bis zum Verlassen des Raumes, verfolgt. Werden Personen von dem Tracker verloren, liegt dies meist an einer kurzfristigen Verdeckung von einer anderen Person oder an dem sich hinter einer Person befindenden Tisch, dessen Oberfläche hautfarbenähnlich ist.

Auffällig ist die geringe Zahl der korrekt identifizierten Testfälle. Hierfür sind die Eigenschaften der für den Identifizierer ausgewählten Bildausschnitte verantwortlich. Zum einen sind dies nicht immer Bilder von frontal aufgenommenen Gesichtern und zum anderen sind oft gedrehte oder geneigte Gesichter zu sehen. Häufig sind die Ausschnitte auf Grund der Entfernung der Person zur Kamera sehr klein. Auffallend sind die Artefakte des Zeilensprungverfahrens, das von den im intelligenten Raum verwendeten PAL-Kameras eingesetzt wird. Nahezu jedes vom Identifizierer empfangene Bild ist mit ausgeprägten Treppenstufen versehen. Die wenigen korrekt identifizierten Fälle sind auf Aufnahmen zurückzuführen, die zufällig keine Darstellungsfehler enthalten, weil sich die Person nahe der Kamera für einen kurzen Augenblick nicht bewegt. In 6.30 sind alle Bildausschnitte einer Person zu sehen, die während einer Testsequenz dem Identifizierer als Eingabe dienten.

Zuletzt ist die Quantität der an den Identifizierer gesendeten Bildausschnitte zu erwähnen. Bei den von uns evaluierten Sequenzen werden relativ wenige Ausschnitte an den Identifizierer gesendet (zwischen einem und fünf pro Testfall). Dies liegt daran, dass das Weltmodell nach Kriterien wie der Größe des Bildausschnitts oder der vergangenen Zeit seit der letzten Identifikation dieser Person, Bildausschnitte auswählt bzw. verwirft. Eine Variation dieser Kriterien und ihrer Parameter ist nicht Bestandteil dieser Evaluation. Es ist jedoch denkbar, dass durch eine geschickte Wahl der Identifizierer mit mehr brauchbaren Bildern versorgt werden kann.

Abbildung 6.30: Alle einer Person zugehörigen Bildausschnitte, die während einer Testsequenz an den Identifizierer gesandt wurden. Zu beobachten sind unter anderem der Treppenstufeneffekt, der von unserer Sensorik erzeugt wird, sowie die geringe Auflösung einiger Bilder.

7 Zusammenfassung

Mit dem ViPer-System soll die Grundlage für eine restriktionslose Interaktion von Personen mit einem intelligenten Raum geschaffen werden. Voraussetzung für die Interaktion ist das Wissen um die Anwesenheit von Personen. Ist zusätzlich die Identität und Position einzelner Personen bekannt, so können verschiedene, im Kontext dieser Informationen angebotene Dienste, durch das intelligente System realisiert werden. Einfache Beispiele sind die automatische Begrüßung oder eine Anwesenheitsprotokollierung.

Die Beschaffung des für diese Dienste nötigen Wissens auf Grundlage von Videoinformation ist Aufgabe des ViPer-Systems. Für die Detektion, das Tracking und die Identifikation von Personen wurden jeweils einzelne Module entwickelt. Geeignete Verfahren aus der aktuellen Forschung wurden ausgewählt, für das Szenario angepasst und von uns erweitert.

Bei der Implementierung des Detektors konnte durch die Erweiterung des Viola & Jones Verfahrens um die Auswertung von Farbinformationen eine starke Verbesserung hinsichtlich des Auftretens von Fehldetektionen erzielt werden.

Der CAMshift-Tracker profitiert in unserem Szenario von der Ergänzung um ein dynamisches Histogramm, mit der auch Kopfdrehungen verfolgt werden können. Die Tests zeigen, dass der CAMshift-Tracker mit statischem Histogramm in vielen Sequenzen die Person bereits nach kurzer Zeit verliert, während der erweiterte Tracker die Person über die gesamte Sequenz verfolgt.

Die Erkennungsrate des zur Identifikation eingesetzten EBGM-Algorithmus ist zu einem großen Teil von guten Eingangsbildern abhängig. Die im intelligenten Raum aufgenommenen Bilder haben lediglich eine maximale Größe von 756×556 Pixeln, die Teilbilder auf denen identifiziert wird, haben eine Maximalgröße von etwa 250×250 Pixeln. Es konnte gezeigt werden, dass unsere Implementierung des EBGM-Verfahrens auch bei solchen Bildern noch über zwei Drittel der Personen korrekt identifiziert.

Die Tests des Gesamtsystems zeigen, dass zu der Detektion, dem Tracking und der Identifikation noch die Schwierigkeit der Koordination der Module hinzukommt. Insbesondere die Auswahl von Bildern zur Identifikation ist entscheidend für das Erkennen der Identität der detektierten Personen. Hier erweist sich das Auftreten des Interlaced-Effekts bei zusammengesetzten Halbbildern als große Schwierigkeit. Um eine korrekte Identifikation zu ermöglichen, ist es notwendig, dass die Person mit Blick in die Kamera kurz verharrt. Es steht zu vermuten, dass sich hier durch die Benutzung einer anderen Kamera noch wesentliche Verbesserungen erzielen lassen.

Eine weitere Schwierigkeit bei der Konstruktion des Gesamtsystems ergibt sich durch die begrenzte Übertragungskapazität eines 100 MBit-Netzwerks. Da alle Module auf unkomprimierten, farbigen Einzelbildern mit einer Auflösung von 756×556 arbeiten, liegt das theoretische Maximum bei etwa 10 Hz, in der Praxis jedoch eher bei 5–7 Hz. Im Livebetrieb können schnelle Bewegungen der Personen daher vom Tracker nicht immer

verfolgt werden. Inwieweit ein Gigabit-Ethernet die Qualität des Trackings verbessern kann, bleibt zu untersuchen.

Insgesamt konnte jedoch gezeigt werden, dass sich mit entsprechenden Verfahren ein System zur Detektion, Verfolgung und Identifikation von Personen auf aktueller Desktophardware realisieren lässt. Bei der Identifikation kommt es aufgrund der niedrigen Auflösung der Gesichtsbilder bei großem Abstand zwischen Kamera und Person und dem Interlacing-Effekt der Kameras häufig zu Problemen. Eine mögliche Lösung wäre der Einsatz einer separaten schwenk- und zoombaren Kamera für die Identifikation, deren Steuerung dann vom ViPer-System übernommen würde.

Wie die Evaluation des Gesamtsystems zeigt, arbeiten die Detektion und das Tracking auch in realen Szenarien recht zuverlässig. Insbesondere werden beim Betreten des Raumes nahezu alle Personen detektiert, so dass auf Basis dieser Information bereits eine restriktionslose Interaktion mit Personen implementiert werden kann.

A Anhang

A.1 Kameramanager und Zubehör

A.1.1 Einleitung und Problemstellung

Der Kameramanager wird verwendet, um das gesamte System mit Bildern zu versorgen. Die Bilder, die an die weiteren Komponenten des ViPer-Systems gesendet werden, können auf drei verschiedene Arten geladen werden. So kann der Kameramanager Bilder entweder aus einem vorhandenen Verzeichnis, von einer lokal per USB oder FireWire angeschlossenen Kamera oder mittels einer Netzwerkverbindung von einer analogen Kamera eines Alpha-Terminals laden. Wenn die Bilder aus einem Verzeichnis geladen oder von einer lokal angeschlossenen Kamera geholt werden sollen, muss sichergestellt sein, dass das Dateiformat bzw. die Kamera von der OpenCV-Bibliothek[1] unterstützt wird.

Wenn aktuelle Bilder aus dem intelligenten Raum verwendet werden sollen, bietet es sich an, die Bilder von einem der Alpha-Terminals zu holen, dessen Kamera Bilder aus dem Raum liefert. Die Terminals sind mit einer analogen Kamera sowie einer Framegrabberkarte versehen. Auf dem Terminal kann ein Sende-Modul namens *ACamPro* gestartet werden. Das Modul ermöglicht es, Rohbilder über eine TCP-Verbindung zu empfangen.

Damit der Bildempfang durch den Kameramanager auch ohne Alpha-Workstations getestet werden kann, existiert eine weitere Anwendung namens *AlphaSender*, die den Versand der Bilder durch eine Alpha-Workstation emuliert. Als Bildquelle dient entweder eine über die OpenCV-Bibliothek angebundenen Kamera oder beliebiges Verzeichnis mit Bildern.

Zur Erzeugung von Bildsequenzen existiert zudem noch die Anwendung *AlphaDumpImage*. Dieses Programm tritt bei der Kommunikation mit einem Alpha-Terminal an die Stelle des Kameramanagers und erlaubt das lokale Abspeichern der von den Alpha-Stationen gelieferten Einzelbilder.

Da sich im Laufe des Projektes ergeben hat, dass die Geschwindigkeit des lokalen Netzwerks nicht ausreicht, um Bilder fließend anzuzeigen, existiert zudem noch eine Anwendung namens *ACamProDumper*, die es ermöglicht, direkt auf einem Alpha-Terminal die Bilder abzuspeichern.

A.1.2 Kameramanager

Funktionalität

Der Kameramanager als zentrale Bildversorgungskomponente kann Verbindungen mit dem Tracker und dem Detektor als Bildkonsumenten unterhalten.

1 http://www.intel.com/technology/computing/opencv/

Bei Verbindungen mit einem Tracker oder Detektor agiert die Kamera je nach Wahl des Communicator-Objekts als Client oder Server. In der Rolle als Bildempfänger für Bilder der Alpha-Stationen handelt der Kameramanager immer als Client, der sich mit dem Alpha-Terminal verbinden muss.

Der Kameramanager selbst kann wahlweise als reine Konsolenanwendung oder als Programm mit graphischer Benutzeroberfläche gestartet werden.

Bildübertragung

Wenn der Kameramanager eine Verbindung mit den Alpha-Terminals aufbauen soll, so geschieht dies durch Verwendung eines TCP/IP-Sockets, über den die Bilder der analogen Kameras geliefert werden. Zu Beginn einer Verbindung sendet das Alpha-Terminal einmalig die Bildinformationen wie Auflösung (756x, 556 oder 378x278), Anzahl der Bildkanäle, Bildformat (RGB, YUV 4:4:4, YUV:4:2:2, YUV:4:2:0) und Kompression zum Kameramanager. Im Normalfall haben die Bilder eine Auflösung von 756x556 Pixeln bei einer Farbtiefe von 24 Bit. Damit die bestmögliche Übertragungsrate ermöglicht werden kann, werden pro Bild anschließend lediglich die Farbinformationen pro Pixel in einem fortlaufenden Bitstrom übermittelt, da sich die übrigen Informationen wie Auflösung oder Farbtiefe nicht ändern. Wenn der Kameramanager ein Bild empfangen hat, wird der Versand des nächsten Bildes durch den Versand eines Bestätigungs-Bytes vom Kameramanager aus angestoßen.

Sobald die Bilder empfangen worden sind, werden sie in jeweils das OpenCV-Bildformat umgewandelt, da die übrigen Systemkomponenten dieses Bildformat verlangen.

Konsolenanwendung

Wenn der Kameramanager als Konsolenanwendung gestartet wird, erfolgt die Steuerung des Programms durch Angabe entsprechender Parameter beim Programmaufruf. Die einzelnen Parameter werden im Kapitel A.3.5 erklärt.

GUI-Anwendung

Im Betriebsmodus mit graphischer Oberfläche (GUI) erscheint der Kameramanager so, wie es in Abbildung A.1 dargestellt ist. Die Oberfläche lässt sich in die vier Bereiche Menüleiste, Netzwerk-Steuerleiste, Bildbereich und Detail- bzw. Quellensteuerungsbereich einteilen.

Menüleiste
Über die Menüleiste lässt sich der Kameramanager beenden.

Bildbereich
Der Bildbereich ist in zwei nebeneinander liegende Teilbereiche eingeteilt. Im linken Teil wird bei Bedarf das zuletzt an die angeschlossenen Bildkonsumenten übermittelte Bild angezeigt. Die Anzeige erfolgt, sobald die entsprechende Checkbox über dem Bildbereich ausgewählt wird.

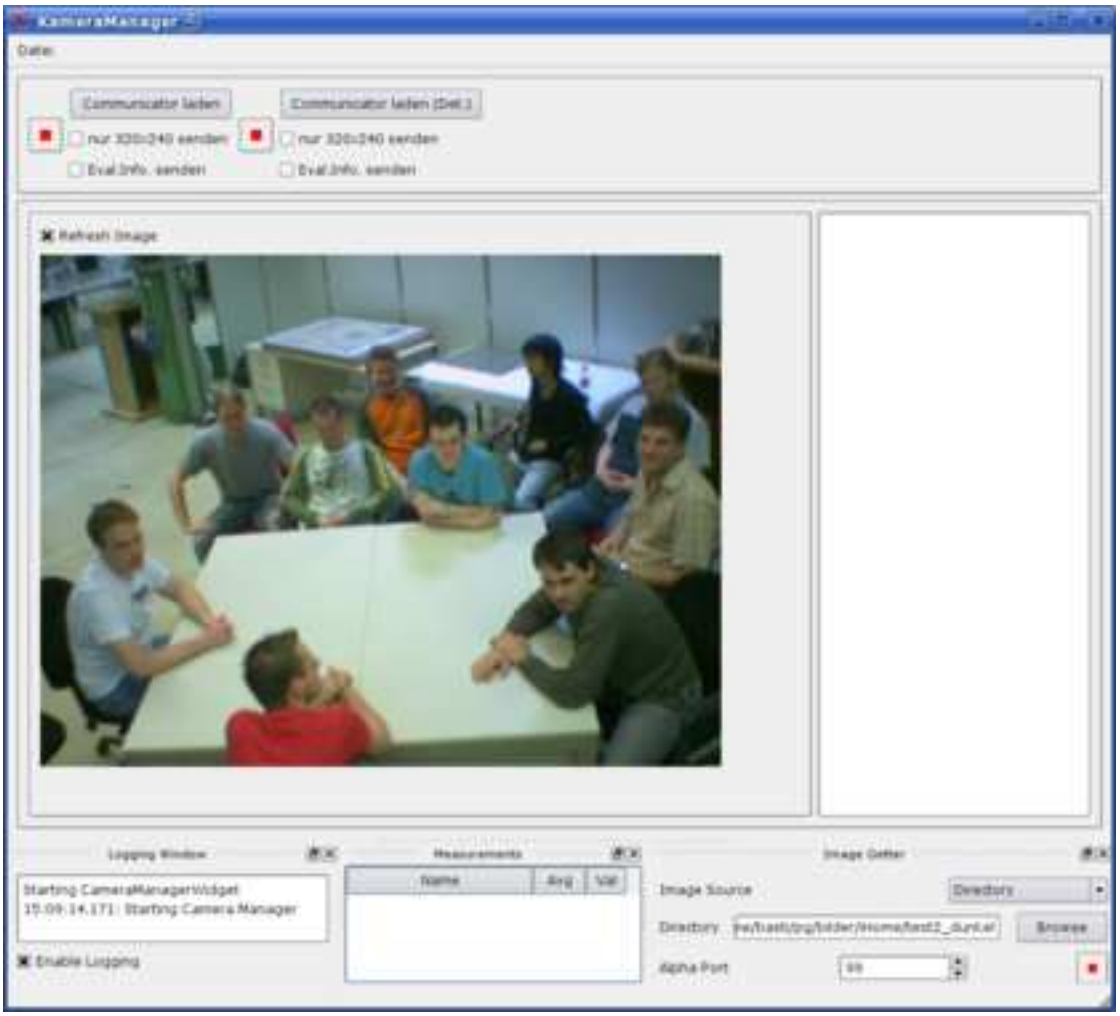


Abbildung A.1: GUI des Kameramanagers

Sofern dem Kameramanager ein Verzeichnis als Bildquelle dienen soll, enthält der rechte Teil des Bildbereichs eine Liste der im ausgewählten Bildverzeichnis vorhandenen Dateien.

Netzwerk-Steuerleiste
Die Netzwerk-Steuerleiste enthält die notwendigen Kontrollobjekte zur Verwaltung der Verbindungen mit den Bildkonsumenten. Es befindet sich dort eine Schaltfläche zur Erzeugung einer neuen Verbindung, ein Widget zur Anzeige des Verbindungszustandes sowie jeweils eine Checkbox zur Auswahl der zu übertragenden Informationen bzw. der zu übertragenden Bildgröße.

Detail-/Quellensteuerungsbereich
Im dreigeteilten Detail-/Quellensteuerungsbereich werden im linken Teil auf Wunsch aktuelle Systemmeldungen ausgegeben. Im mittleren Teil erfolgt bei bestehenden Verbindungen die Ausgabe der gemessenen Übertragungszeiten zwischen den einzelnen Komponenten.

Auf der rechten Seite kann die Auswahl der Bildquelle vorgenommen werden. Wenn die Bilder aus einem Verzeichnis geladen werden, kann hier das entsprechende Verzeichnis ausgewählt werden. Sofern eine lokale Kamera benutzt werden soll, sind außer der Auswahl der Kamera keine weiteren Angaben erforderlich. Um den Kameramanager mit einem Alpha-Terminal oder -Sender zu verbinden, müssen der Computername bzw. die IP-Adresse des Senders sowie der entsprechende Port angegeben werden.

A.1.3 AlphaSender

Nicht in jeder Testumgebung wird man als Anwender die Möglichkeit haben, auf eine analoge Kamera einer Alpha-Workstation zuzugreifen. Daher dient in einem solchen Fall der AlphaSender dazu, das Verhalten einer solchen Workstation zu emulieren und über einen TCP/IP-Socket den Kameramanager mit Bildern zu versorgen.

A.1.4 AlphaDumpImage, ACamProDumper und ImageConverter

Die Anwendungen AlphaDumpImage und ACamProDumper dienen dazu, Bilder der analogen Kameras der Alpha-Terminals abzuspeichern, so dass beispielsweise Testsequenzen aufgenommen werden können.

Die Anwendung ACamProDumper wird direkt auf einem Alpha-Terminal ausgeführt und speichert die Bilder in einem beliebigen, erreichbaren Verzeichnis in dem Rohformat, das auch bei der Bildübertragung zum Kameramanager verwendet wird. Damit diese Bilder später wieder vom Kameramanager benutzt werden können, müssen sie zuvor in eines der von der OpenCV-Bibliothek lesbaren Bildformate umgewandelt werden. Das Programm ImageConverter übernimmt genau diese Aufgabe: Nach Übergabe eines Verzeichnisnamens werden alle dort enthaltenen Rohbilder in das vom Benutzer angegebene Bildformat umgewandelt.

Wenn die Bilder des Alpha-Terminals per Netzwerk empfangen werden, kann das Programm AlphaDumpImage anstelle des normalerweise verwendeten Kameramanagers

die Bilder ebenfalls direkt in einem Verzeichnis ablegen. Als Format kann dort jedes von der OpenCV-Bibliothek unterstützte Format gewählt werden.

A.2 Annotationswerkzeug

A.2.1 Einleitung, Problemstellung

Wir wollen die Güte der Detektionen beurteilen und mit manuell annotierten Bildern vergleichen. Fragen, die sich dabei stellen, sind z.B.:

- »Werden automatisch alle Gesichter erkannt?«
- »Werden zuviele Gesichter erkannt, also fälschlicherweise Nicht-Gesichter als Gesichter markiert?«
- »Sind die Detektionen exakt oder eher ungenau?«

Wir haben uns entschieden, Gesichter durch ein Rechteck auszuwählen, das durch die Augenmittelpunkte und durch den Mundmittelpunkt läuft. Eine Detektion ist erfolgreich, wenn sie zwischen dem so erstellten und einem, um festgelegte Faktoren in x- und y-Richtung, größeren Rechteck liegt, das automatisch nach der Markierung der drei Gesichtskoordinaten erzeugt wird.

Um die Kaskade für den Viola-Jones-Gesichtsdetektor trainieren zu können, ist es wichtig, mit dem Annotationsprogramm genau das Gesicht extrahieren zu können. Die Daten werden für das Trainingsprogramm benötigt.

Auch der Identifizierer soll vom Annotationstool profitieren, indem zu den zwei bereits erwähnten Rechtecken noch ein drittes hinzugefügt wird, das ebenfalls relativ zum Gesichtsrahmen skaliert wird. Dieses gibt den Ausschnitt im Bild wieder, der im Optimalfall nur das Gesicht umgibt. So ausgewählte 'Unterbilder' müssen komfortabel aus dem Ursprungsbild extrahiert werden können. Die Gesichter müssen jeweils die Möglichkeit bieten, eine ID, anhand der man sie eindeutig der richtigen Person zuordnen kann, mit angeben zu können. Praktisch ist auch eine Funktion, die die ausgeschnittenen Gesichter, die auf den Originalbildern oft in der Kameraebene verdreht sind sind, so ausrichtet, dass die Augen waagerecht liegen. Damit werden die zum Training des Identifizierers genutzten Gesichter aneinander angepasst.

Außerdem benötigen wir ein Programm, mit dem man per Hand Bild für Bild durchgehen und für den Tracker den zu trackenden Bereich markieren kann. Dazu ist es wichtig, dass in einer Serie von Bildern ein im ersten annotierten Bild ausgewählter Bereich für das jeweils folgende Bild übernommen werden kann und nur noch an die richtige Position geschoben werden muss.

A.2.2 Lösungsansatz

Um die gesamte geforderte Funktionalität zu implementieren, ist - der Anzahl der verfügbaren Bibliotheken und der Entwicklungsgeschwindigkeit wegen - Java die Programmiersprache der Wahl. Dies steht im Kontrast zum Rest von ViPer (der in C++ geschrieben

ist), da es dort auf Geschwindigkeit zur Laufzeit ankommt. Das Annotationstool, wegen des großen Funktionsumfangs 'Annotate Ultra Plus' genannt, muss im Betrieb keinen besonderen Performance-Anforderungen gerecht werden, sondern einfach und schnell bedienbar sein. Das Programm bietet sämtliche Funktionen direkt im Hauptfenster an, damit der schnellen Annotation hunderter Bilder keine Suche in Menüs im Wege stehen. Nur die Auswahl der Gesichtskoordinaten bzw. die manuelle Selektion eines Bildausschnittes müssen über die Maus getätigt werden.

Natürlich gehört zu einer einfachen Bedienbarkeit eine - trotz der Funktionsvielfalt - übersichtliche Anordnung der Bedienelemente. Diese sind deshalb sinnvoll gruppiert und teilen das Fenster des Annotationstools grob in 5 Bereiche (A.2) ein:

1. Den Annotations-/Bildbereich (A.3): hier wird das aktuell geladene Bild inklusive aller Selektionen angezeigt; hier können auch neue Selektionen getätigt werden

2. Den Bildlistenbereich (A.4): hier werden alle zur Annotation ausgewählten Bilder aufgelistet; Bilder können hinzugefügt oder entfernt werden

3. Den Einstellungsbereich (A.5): hier kann eingestellt werden, ob Selektionen in das nächste Bild übertragen werden sollen, ob die ausgeschnittenen Gesichter beim Speichern anhand der Augen ausgerichtet werden sollen und ob die nächste Auswahl anstatt durch die Augen und den Mund durch die Augen und das Seitenverhältnis der letzten Selektion definiert werden soll

4. Den Eingabebereich (A.6): hier kann einem ausgewählten Rahmen eine ID zugewiesen werden und die Seitenverhältnisse für die automatisch generierten Rahmen eingestellt werden

5. Den Buttonbereich (A.7): hier können die Selektionen und extrahierten Rechtecke gespeichert, sowie das Programm beendet werden

A.2.3 Funktionalität

Annotations-/Bildbereich A.3

Der Annotationsbereich gliedert sich in 3 Unterbereiche: den eigentlichen Annotationsbereich, dem Annotations-Button und der Annotations-Statusanzeige.

1. Mit dem Annotations-Button wird der Annotationsmodus ein- bzw. ausgeschaltet. Um die Bedienung zu beschleunigen ist hier kein Klicken nötig. Die Betätigung der Leertaste (de-)aktiviert den Button.

2. In der Annotations-Statusanzeige wird angezeigt in welcher Phase der Annotation man sich zurzeit befindet und welches Merkmal durch die nächste Auswahl selektiert wird (linkes Auge, rechtes Auge, Mundmitte oder automatisches Seitenverhältnis). So behält man den Überblick, welchen Punkt im Bild man als nächstes anklicken muss.

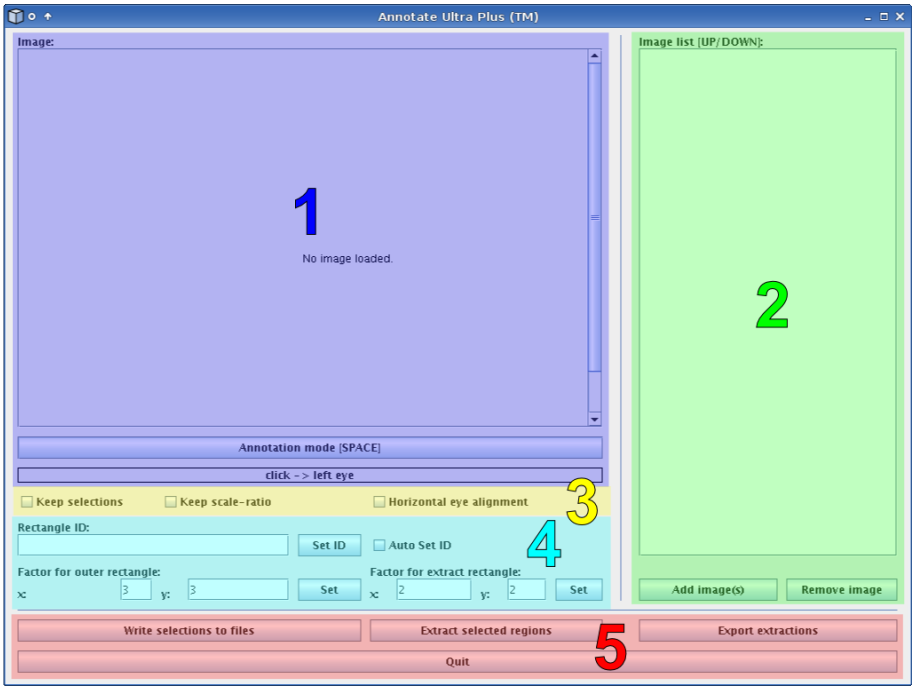


Abbildung A.2: Annotate Ultra Plus Fensterbereiche

3. Der eigentliche Annotationsbereich nimmt den größten Teil des Fensters ein. In ihm werden das aktuell ausgewählte Bild und die bereits erfolgten Annotationen angezeigt. Ist der Annotationsmodus eingeschaltet, so kann durch Klicken mit der Maus der in der Statusanzeige angezeigte nächste Punkt markiert werden. Ist der Mund markiert, so erscheinen drei Rechtecke: in gelb das Rechteck durch die Augen und den Mund, in rot das auszuschneidende Gesicht und in blau der äußere Toleranzrahmen für Gesichtsdetektion. Zusätzlich zur Gesichtsauswahl durch drei Klicks besteht die Möglichkeit, durch Klicken, halten und ziehen mit der Maus einen Rahmen auszuwählen. Ist dieser Rahmen ausgewählt, so werden insgesamt zwei Rahmen angezeigt: in gelb der ausgewählte Bereich und in rot der auszuschneidende Bereich. Alle Selektionen können per Drag and Drop, beginnend im innersten Rahmen, verschoben werden. Ein Rechtsklick in einen inneren Rahmen löscht die entsprechende Auswahl. Die Backspace-Taste löscht den zuletzt markierten Punkt und die Escape-Taste löscht die gesamte aktuelle Markierung.

Bildlistenbereich A.4

Der Bildlistenbereich besteht aus drei Elementen: der Bildliste, dem Add-Button und dem Remove-Button. In der Bildliste werden die Dateinamen der Bilder angezeigt, die zur Annotation ausgewählt werden.

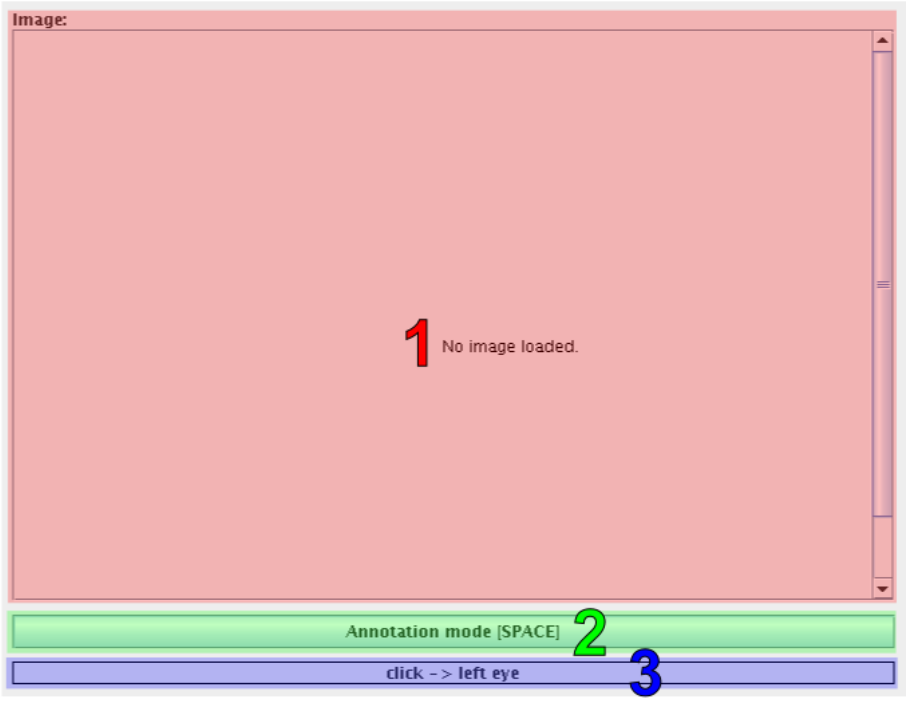


Abbildung A.3: Annotations-/Bildbereich

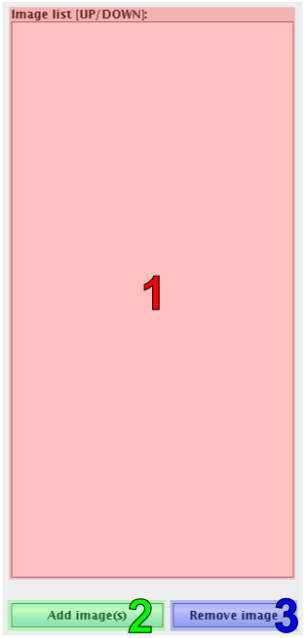


Abbildung A.4: Bildlistenbereich

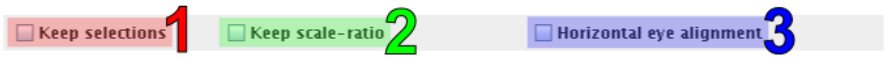


Abbildung A.5: Einstellungsbereich

1. Die Bildliste ermöglicht die schnelle Auswahl des nächsten/vorherigen Bildes durch Drücken der Cursortasten 'hoch/runter' auf der Tastatur. Eine Wahl per Mausklick ist natürlich auch möglich. Wird die Auswahl geändert, so zeigt der Annotationsbereich das neu gewählte Bild inkl. Selektionen an.

2. Über den Add-Button wird ein Dateiauswahldialog geöffnet, der Einzel- oder Mehrfachauswahl neuer Bilder erlaubt (es werden gängige Bildformate unterstützt). Durch betätigen des Öffnen-Buttons im Dialogfenster werden die ausgewählten Bilder an die Bildliste angehängt.

3. Betätigt man den Remove-Button, so wird das in der Bildliste markierte Bild aus der Liste entfernt.

Einstellungsbereich A.5

Im Einstellungsbereich gibt es drei Checkboxen, über die weitere Funktionen zugeschaltet werden können.

1. Die Checkbox 'Keep selections' aktiviert die Übernahme der aktuellen Selektionen in das nächste ausgewählte Bild. So werden alle ausgewählten Rechtecke in das Bild übertragen, das man als nächstes annotieren möchte.

2. Mit der Checkbox 'Keep scale-ratio' wird der Größenverhältnismodus für die Gesichtsannotation aktiviert. Ist das Häkchen gesetzt, müssen im Annotationsbereich nur noch die Augen ausgewählt werden. Danach erscheint sofort das Auswahlrechteck, das das gleiche Seitenverhältnis hat, wie das zuletzt markierte Gesicht. Natürlich funktioniert diese Option nur, wenn bereits mindestens ein Gesicht auf dem 'normalen' Weg annotiert wurde.

3. Die dritte Checkbox hat unmittelbare Auswirkungen auf die ausgeschnittenen und in einzelne Dateien gespeicherten Gesichter. Ist die Checkbox aktiviert, so dauert das Ausschneiden erheblich länger, da vor dem Speichervorgang noch eine Ausrichtung der Augen in der Waagerechten vorgenommen wird. Dies ist für das Training des Identifizierers wichtig.

Eingabebereich A.6

Der Eingabebereich ist in zwei Unterbereiche aufgeteilt: im oberen Teil befinden sich das Eingabefeld für die Rahmen-ID sowie eine Checkbox zur automatischen Übernahme der

Abbildung A.6: Eingabebereich

Abbildung A.7: Buttonbereich

eingegebenen ID für alle weiteren Annotationen, im unteren Teil sind mehrere Eingabefelder für die Faktoren der Extraktions- und Toleranzrahmen untergebracht. Ist der Annotationsmodus deaktiviert, kann man im Annotationsbereich per Mausklick (links) ein Rechteck auswählen und in das ID-Feld eine ID eintragen. Über den Button rechts vom Feld kann diese ID übernommen werden. Auch wenn das Eingabefeld den Fokus verliert, wird die ID für das ausgewählte Rechteck übernommen. In die Faktorfelder eingetragene Werte für die Rahmen werden auch durch den jeweiligen Button rechts daneben übernommen. Verlieren die Felder den Fokus, werden die Werte ebenfalls gespeichert. Die Faktoren gelten jeweils für eine laufende oder die nächste Selektion.

Buttonbereich A.7

Der Buttonbereich setzt sich aus vier Buttons zusammen: dem 'Write selections to files'-, dem 'Extract selected regions'-, dem 'Export extractions'- und dem Quit-Button.

1. Mit dem 'Write selections to files'-Button werden zu allen in der Bildliste befindlichen Bildern die entsprechenden Selektionen abgespeichert. Dazu werden Dateien angelegt, deren Dateiname mit dem des jeweiligen Bildes bis auf die Endung übereinstimmt. Die Endung lautet '.selected'.

2. Drückt man den 'Extract selected regions'-Button, so öffnet sich ein Verzeichnisauswahldialog, über den man ein Zielverzeichnis für die zu extrahierenden Bereiche auswählen muss. Hat man eines ausgewählt, so werden in dieses Verzeichnis die Bildausschnitte der Bilder als '$BILDNAME_$AUSSCHNITT.png' gespeichert. Zusätzlich dazu werden '$BILDNAME_$AUSSCHNITT.face'-Dateien erzeugt, die die Gesichtskoordinaten enthalten (falls vorhanden).

3. Durch den 'Export extractions'-Button wird man über einen Dateiauswahldialog nach einer Zieldatei gefragt. In diese werden nach Bestätigung der Auswahl die Selektionen aller Bilder im CSU-Format für das Detektortraining gespeichert.

4. Über den Quit-Button kann man das Programm beenden.

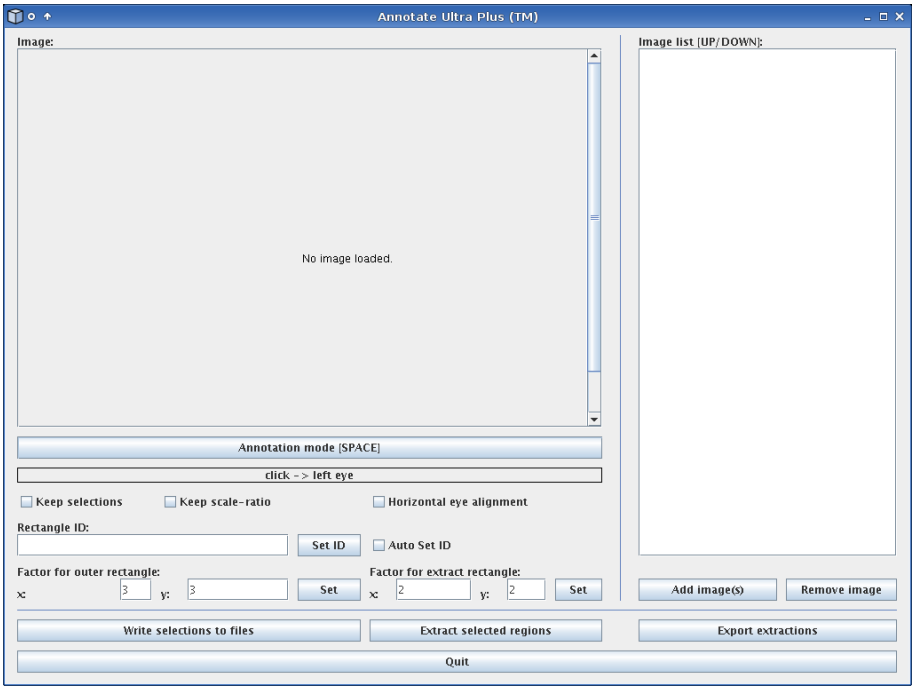


Abbildung A.8: Annotate Ultra Plus direkt nach dem Programmstart

A.2.4 Eine Beispielanwendung

Um zu illustrieren, wie eine beispielhafte Benutzung von »Annotate Ultra Plus« aussieht, beschreibt dieser Abschnitt eine mögliche Annotation zweier Bilder und die nachfolgende Speicherung der Annotationsdaten sowie die Extraktion der selektierten Gesichter.

Programm starten

Der erste Schritt ist das Starten des Programms. Danach erscheint das Fenster von »Annote Ultra Plus« und ist direkt bereit zur Annotation.

Dateien zur Annotation laden

Um Bilder annotieren zu können, müssen sie zunächst geladen werden. Dazu klickt man auf den »Add images«-Button und ruft somit den Dateiauswahldialog auf. Dieser filtert automatisch alle nicht-lesbaren Bild- und Dateiformate aus der Ansicht heraus.

Bestätigt man die Bildauswahl mit dem »Öffnen«-Button des Dialogs, so wird der Auswahldialog geschlossen und die ausgewählten Dateien werden in die Bildliste übernommen. Das erste Bild in der Liste wird direkt im Annotationsbereich angezeigt und kann annotiert werden.

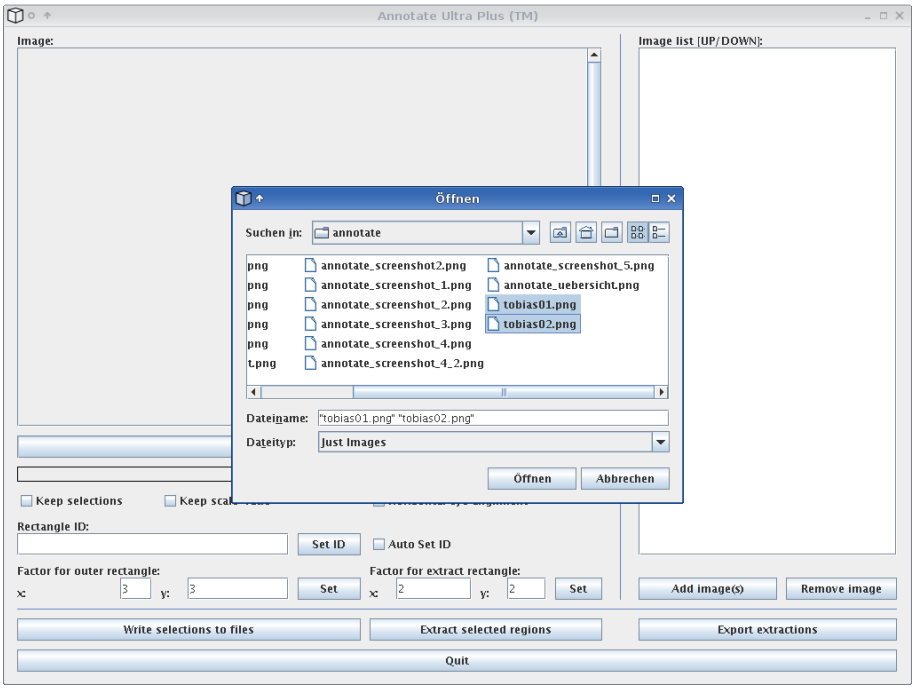


Abbildung A.9: Dialog zum Laden von Bildern

Gesichtsannotation

Um die Gesichtsannotation zu beginnen, muss der Annotationsmodus (entweder per Druck auf die Leertaste oder per Klick auf den Annotationsbutton) aktiviert werden. Die Annotation erfolgt dann über drei Klicks, mit denen man zuerst den Mittelpunkt des linken (s. A.11), danach den Mittelpunkt des rechten Auges (s. A.12) und zum Schluss den Mundmittelpunkt selektiert.

Der Klick auf den Mundmittelpunkt schließt die Annotation eines Gesichtes ab (s. A.13). Er löst die Berechnung der einzelnen Rahmen für die obere und untere Detektionsschranke und das zu extrahierende Gesicht aus. Dazu werden die im Eingabebereich eingestellten Faktoren eingelesen und für die Berechnung der oberen Detektionsschranke und des Extraktionsbereichs herangezogen.

Jede weitere Gesichtsannotation erfolgt analog. Um zum jeweils nächsten geladenen Bild zu gelangen, kann man die Auswahl per Maus im Bildlistenbereich durchführen oder die Cursortasten benutzen. Bei bereits (teil-)annotierten Bildern werden auch die bereits vorhandenen Annotationen mitgeladen und angezeigt.

Annotationen speichern

Der Button »Write selections to files« speichert zu allen annotierten Bildern eine ».selected«-Datei, die alle Selektionen (Augen- und Mundkoordinaten und die drei Rechtecke) ent-

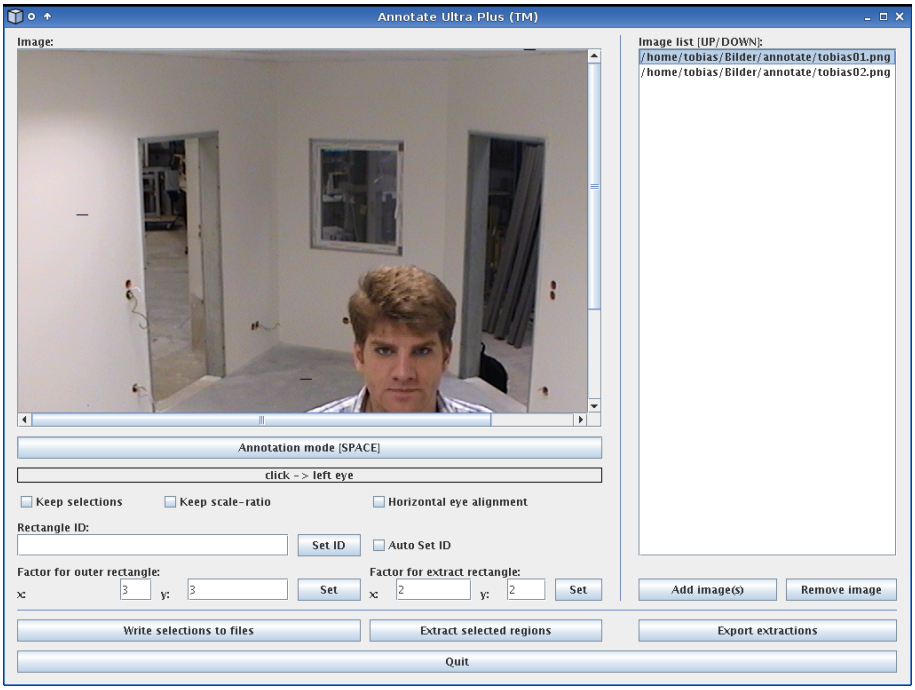


Abbildung A.10: Nach Laden der Bilder

hält. Diese hat bis auf die Endung - ».selected« - denselben Dateinamen wie das jeweilige Bild.

Annotierte Gesichter extrahieren

Durch Klick auf »Extract selected regions« öffnet sich ein Verzeichnisauswahldialog. Hier wählt man das Zielverzeichnis für die Ablage der zu extrahierenden Bildbereiche. Nach Bestätigung der Auswahl wird der Dialog geschlossen, die Teilbilder gespeichert und zum Hauptprogramm zurückgekehrt. Die Extraktionen liegen nun zur weiteren Verarbeitung (z.B. als Grundlage für ein Kaskadentraining für den Detektor) vor. Als hilfreich hat sich herausgestellt, zu den extrahierten Gesichtern auch die Gesichtskoordinaten abzuspeichern. Deshalb wird zu jeder Extraktion eine ».face«-Datei erzeugt, die eben diese Koordinaten enthält.

Annotationen im CSU-Format speichern

Das OpenCV-Trainingsprogramm für den Detektor benötigt als zusätzliche Eingabe zu den Positivbeispielen eine Datei, die alle zu benutzenden Bilddateien und die in ihnen enthaltenen Gesichter enthält. Pro Zeile dieser Datei steht am Anfang der Dateiname der Bilddatei, dahinter die Anzahl der Gesichter und im letzten Abschnitt für jedes Gesicht die vier Rahmenkoordinaten (x, y, Breite, Höhe). Der Button »Export selections« muss

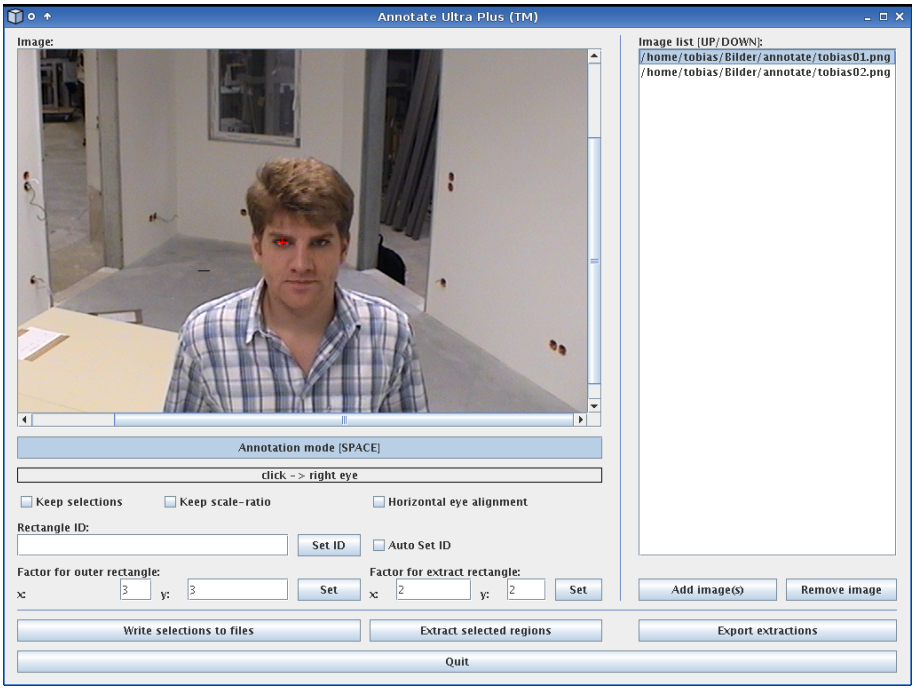


Abbildung A.11: Linkes Auge selektiert

angeklickt werden, um diese Datei zu erzeugen.

A.2.5 Erzeugte Dateien

Das Annotationstool erzeugt drei verschiedene Dateitypen: ».selected«-, ».face«- und CSU-Dateien. Die ersteren enthalten jeweils alle Annotationen eines bestimmten Bildes, die zweiteren nur die Gesichtsannotationen eines extrahierten Gesichtes und die letzteren alle Annotationen für alle Bilder.

- ».selected«
  1. `(412.0,339.0,454.0,340.0,433.0,386.0;412.0,339.0,42.0,47.0;370.0,292.0,126.0,141.0;391.0,315.5,84.0,94.0;)`
  2. `(419.0,338.0,466.0,339.0,441.0,380.0;419.0,338.0,47.0,42.0;372.0,296.0,141.0,126.0;395.5,317.0,94.0,84.0;)`
- ».face«
  1. `(412.0,339.0,454.0,340.0,433.0,386.0)`
  2. `(419.0,338.0,466.0,339.0,441.0,380.0)`
- CSU

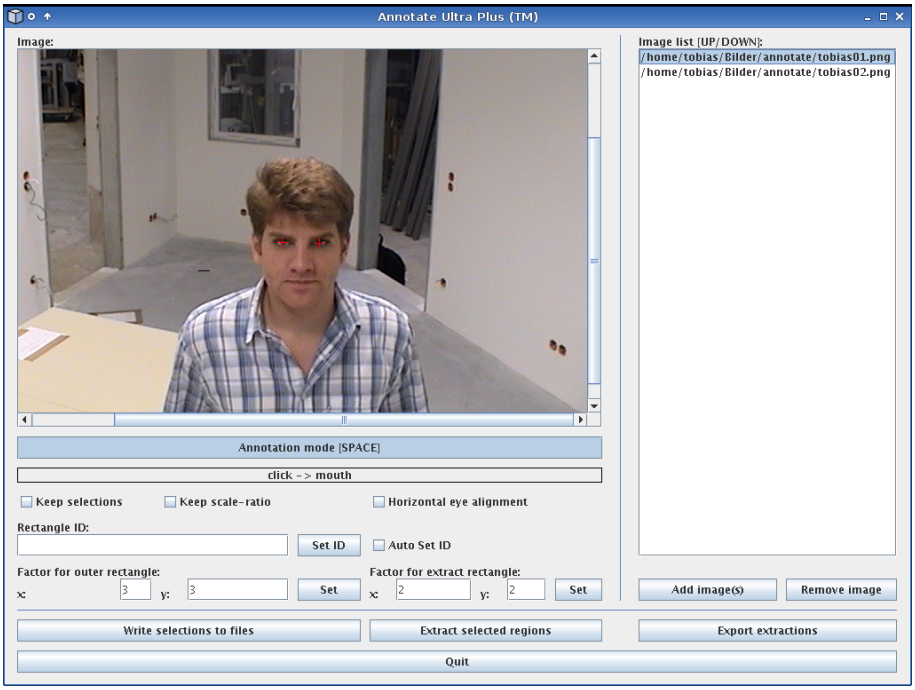


Abbildung A.12: Linkes und rechtes Auge selektiert

```
tobias01.png        1        391 315 84 94
tobias02.png        1        395 317 94 84
```

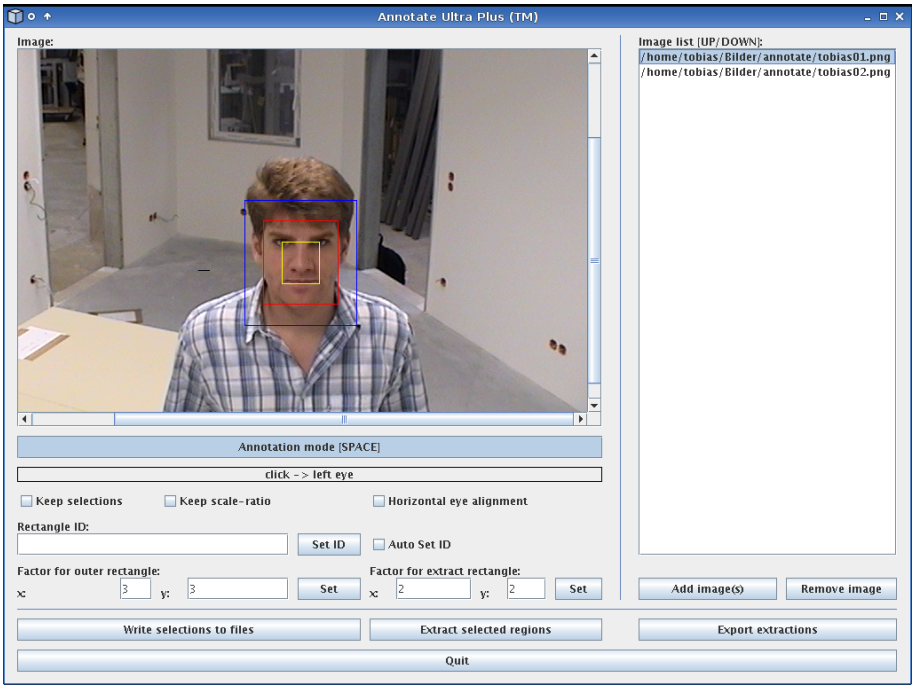


Abbildung A.13: Nach Selektion des Mundmittelpunktes ist das Gesicht fertig annotiert

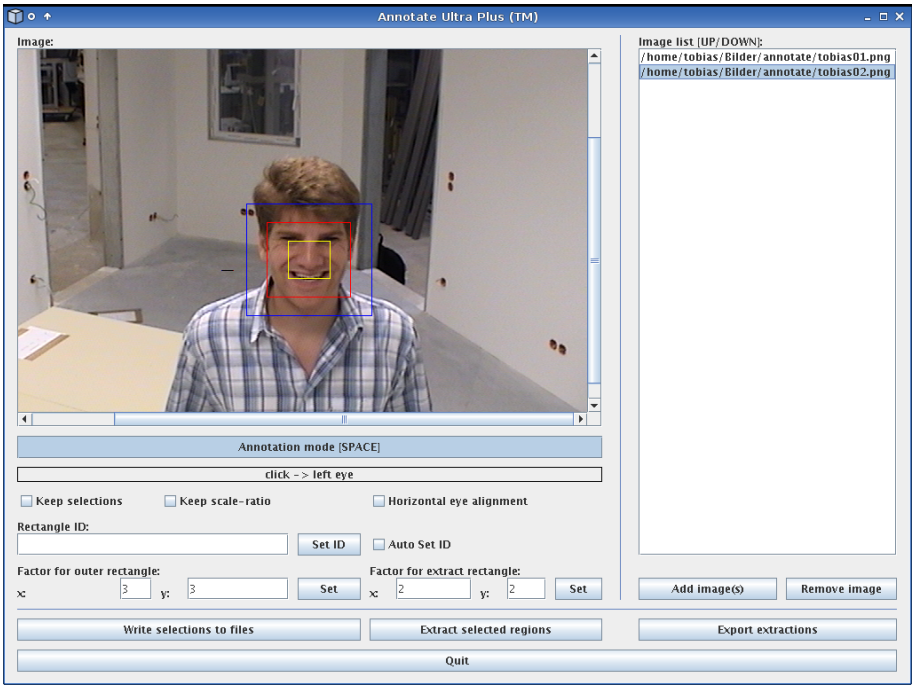


Abbildung A.14: Selektion des zweiten Gesichtes

Abbildung A.15: Extrahiertes Gesicht des ersten Bildes

Abbildung A.16: Extrahiertes Gesicht des zweiten Bildes

A.2.6 Tastaturbelegung

Tastenkombination	Funktion
Leertaste	(De-)aktiviert den Annotationsmodus
Backspace	Löscht den zuletzt markierten Punkt
Escape	Löscht alle markierten Punkte der aktuellen Annotation
Cursortaste Hoch	Wählt das vorherige Bild aus der Bildliste zur Annotation aus
Cursortaste Runter	Wählt das nächste Bild aus der Bildliste zur Annotation aus

Tabelle A.1: Mögliche Tastenkürzel mit verknüpften Funktionen des Annotationstools

A.3 Dokumentation der Kommandozeilenparameter

A.3.1 Detektor

Name des Parameters	Beschreibung	Default Wert
haarcascade,H	haarcascade Datei	haarcascades/ haarcascade_ frontalface_alt2. xml
logfile	Logfile Pfad	./Detector.log
scalefactor,S	Skalierungsfaktor für Suchfenster	1.2
minneigh,N	Minimale Anzahl Nachbardetektionen	1
cannyprun,C	Zuschaltung der Canny Edge Detection	1
minSize	Minimum von Höhe und Breite der Gesichter in Bezug auf Bildbreite	0.053
autoconnect,A	Der Detektor versucht, am Anfang die Default Module auf den Default Ports zu verbinden	1
autorun,R	Der Detektor ruft direkt Bilder auf (wenn autorun true ist, wird auch autoconnect auf true gesetzt); wenn im Standalone Modus verwendet, fängt der Detektor an, Bilder sofort nach dem Start zu detektieren	0
portWC,A	Default Port für die Verbindung zum Worldcoordinator	4322

Fortsetzung...

Name des Parameters	Beschreibung	Default Wert
portCM	Default Port für die Verbindung zum Cameramanager	4331
HostWC	Default Host für die Verbindung zum Worldcoordinator	localhost
HostCM	Default Host für die Verbindung zum Cameramanager	localhost
gui,G	GUI wird am Anfang gezeigt (Wenn keine GUI gezeigt wird, lauscht der Detektor auf dem Default Port)	1
scalingGui,X	Skalierungsfaktor für die Anzeige des Bildes in der GUI	1
scalingImage	Skalierungsfaktor für die Detektion	1.0
smoketest,T	Evaluationsinformationen für den Smoketest schreiben	0
saveDetectedFaces,D	Detektierte Gesichter werden in Datei gespeichert	0
saveDetectedFacesPath	Pfad, in dem die detektierten Gesichter gespeichert werden	
saveDetectedFacesName	Jedem gespeicherten Gesicht wird ein eigener Name gegeben(*.pgm)	
scaleSaveDetectedFacesX,F	Detektionsbereich horizontal um diesen Faktor skalieren, bevor er gespeichert wird	1.5
scaleSaveDetectedFacesY,Y	Detektionsbereich vertikal um diesen Faktor skalieren, bevor er gespeichert wird	1.5
standalone	Startet den Detektor direkt mit ImageGetter (wenn autorun true ist, fängt der Detektor an, Bilder sofort nach dem Start zu detektieren)	0
ImageDirectory,D	Pfad zum Ordner, der die Bilder enthält	/vol/viperdata/ IdentifierData/ Selections/ SelectionBastian/
skincolor	Gesichtsfarbenhistogramm wird benutzt, um Detektionen zu verifizieren	1
facepath	Pfad zum Vergleichsgesicht für Gesichtsfarbendetektion	face.jpg

Fortsetzung...

Name des Parameters	Beschreibung	Default Wert
skinthreshold	Schwellwerte für Gesichtsfarbenverifikation (zwischen 0 und 1)	0.9
saveBildWidgetToFile	Die Resultate, die im BildWidget angezeigt werden, werden in einer .png Datei gespeichert	0

A.3.2 Tracker

Name des Parameters	Beschreibung	Default Wert
logfile	Logfile path	./Tracker.log
vmin,V	V-min	10
vmax,M	V-max	256
smin,S	S-min	30
maxTrackerWindowSize	Maximale Größe des Trackersfensters (relativ zu den Bilddimensionen)	0.5
maxTrackers	Maximale Anzahl Tracking-Threads	4
portCM	Default Port für die Verbindung zum Cameramanager	4321
portWC	Default Port für die Verbindung zum Worldcoordinator	4332
portID	Default Port für die Verbindung zum Identifier	4323
HostCM	Default Host für die Verbindung zum Cameramanager	localhost
HostWC	Default Host für die Verbindung zum Worldcoordinator	localhost
HostID	Default Host für die Verbindung zum Identfier	localhost
ModeCM	Default Mode für die Verbindung zum Cameramanager	mClient
ModeWC	Default Mode für die Verbindung zum WorldCoordinator	mClient
ModeID	Default Mode für die Verbindung zum Identidier	mClient
autoconnect,A	Der Tracker versucht, am Anfang zu den Default Module auf den Default Ports zu verbinden	1

Fortsetzung...

Name des Parameters	Beschreibung	Default Wert
guiShow	GUI wird am Anfang gezeigt (Wenn keine GUI gezeigt wird, wird autoconnect auf true gesetzt)	1
autorun,R	Der Tracker lädt direkt Bilder zum Tracking, wenn zum Cameramanager verbunden wurde (wenn autorun true ist, wird autoconnect auf true gesetzt)	1
guiScaling,X	Skaliert Bild vor der Anzeige in der GUI	1
method	Tracking Methode (0=Alle Features, 1=Standard Camshift)	0
Image Directory	Pfad zum Ordner, der die Bilddateien enthält	/vol/viperdata/ fuerTrackerTobias
x1	x-Wert der oberen linken Ecke des initialen Vierecks	0.0
y1	y-Wert der oberen linken Ecke des initialen Vierecks	0.0
x2	x-Wert der unteren rechten Ecke des initialen Vierecks	0.0
y2	y-Wert der unteren rechten Ecke des initialen Vierecks	0.0
saveBildWidgetToFile	Die Resultate, die im BildWidget angezeigt werden, werden in einer .png Datei gespeichert	0
standaloneAutorun	Der Tracker fängt automatisch an nach dem Start eine gegebene Auswahl zu tracken	0
standaloneShutdown	Der Tracker beendet sich, wenn er das getrackte Objekt verliert	0
evaluate	Die getrackte Trajektorie wird in eine XML-Datei gespeichert, mit der Trajektorie der Eval Infos verglichen und überprüft, ob das getrackte Objekt verloren geht	1
Trajectory File	XML-Datei, in der die getrackte Trajektorie gespeichert wird	trackedTrajectories.xml
Evaluation Trajectory File	XML-Datei, in der die getrackte Trajektorie gespeichert ist, die durch die Evaluationsinformationen gegeben ist	evalTrajectories.xml

A.3.3 Identifier

Name des Parameters	Beschreibung	Default Wert
SubspacePath	Wenn kein Pfad spezifiziert wird, wird der Subspace aus den Bildern in ImageDir aufgebaut	
ImageDir	Absoluter Bildpfad für Training	/tmp/testbilder/colin/
PortTR	Default Port für die Verbindung zum Tracker	4323
PortWC	Default Port für die Verbindung zum WorldCoordinator	4333
HostTR	Default Host für die Verbindung zum Tracker	localhost
HostWC	Default Host für die Verbindung zum WorldCoordinator	localhost
ModeTR	Default Modus für die Verbindung zum Tracker	0
ModeWC	Default Modus für die Verbindung zum WorldCoordinator	1
autoconnect,A	Der Identifier versucht, am Anfang zu den Default Modulen auf den Default Ports zu verbinden	1
gui,G	GUI wird am Anfang angezeigt (Wenn keine GUI gezeigt wird, werden autoconnect und autorun auf true ein gesetzt)	1
autoEval	Die Evalution des ausgewählten Algorithmus startet automatisch	0
autoEvalTest	Absoluter Pfad zu den zu evaluierenden Bildern	/vol/viperdata2/IdentifierData/Eval/BeispielEval/
IdentificationAlgorithm,I	Algorithmus, der zur Identifikation eingesetzt wird. Verfügbare Werte: Eigenface | EBGM	EBGM
EBGMBunchgraph	Der Pfad zur Viper-Graph-Datei, die als Bunchgraph verwendet werden soll	training/graphs/identify/faceBunchGraph.graph
EBGMFacegraphDir	Der Pfad, der die Viper-Graph-Daten für bekannte Personen enthält	training/graphs/identify/

Fortsetzung...

Name des Parameters	Beschreibung	Default Wert
showEBGMTrainingOnly	EBGM Training beginnt und andere Optionen werden ignoriert	0
saveReceivedFaces	Alle empfangenen Gesichter werden gespeichert und später evaluiert	0
saveNormalizedFaces	Alle normalisierten Gesichter werden gespeichert und später evaluiert	0
autorun,R	Der Identifier lädt direkt Bilder (wenn autorun true ist, wird autoconnect auf true gesetzt)	1
Minimal_Accepted_Similarity	Mindestgesichtsähnlichkeit damit das Bild akzeptiert wird	0.88
Minimal_Accepted_Distance	Mindestabstand der besten Gesichtsähnlichkeiten damit das Bild akzeptiert wird	0.01
Separation_Steigung	Parameter für lineare Trennung damit das Bild akzeptiert wird	-0.7
Separation_Offset	Parameter für lineare Trennung damit das Bild akzeptiert wird	0.91

A.3.4 Worldcoodinator

Name des Parameters	Beschreibung	Default Wert
Name des Parameters	Beschreibung	Default Wert
moduleListFile,M	*.vim Datei enthält Liste der Module	
backgroundImage	Pfad zum Bild, das im Hintergrund des Worldcoordinators angezeigt wird	/vol/viperdata/ hintergrundBilderWC/ raum20061213.png
autoconnect,A	Der Worldcoordinator verbindet am Anfang automatisch alle Module der Moduleliste	1
gui,G	GUI wird am Anfang angezeigt (wenn keine GUI gezeigt wird, wird autoconnect auf true gesetzt)	1
minFrames	Minimale Anzahl der aufeinanderfolgenden Frames eines Gesichtes, das als neues Objekt(1-3) detektiert wird	3
SMTPEvents	Das Ereignis wird per SMTP gesendet (als E-Mail)	0

Fortsetzung...

Name des Parameters	Beschreibung	Default Wert
SMTPServer	SMTP Server für Versand der Ereignismails	localhost
SMTPMailTo	E-Mail-Adresse für Versand der Ereignismails	event@viper
SMTPMailFrom	Absender-Adresse für E-Mails	wc@viper
SMTPSubject	Subject zum Einsatz in E-Mails	Viper Event

A.3.5 Cameramanager

Name des Parameters	Beschreibung	Default Wert
Name des Parameters	Beschreibung	Default Wert
listen,L	Der CameraManager lauscht auf dem Default Port	1
port,P	Default Port	4321
port2,Z	Zweiter Default Port	4331
alphaPort,a	Empfangs-Port für Bilder von ACamPro	9999
alphaHost	Hostname für ACamPro	curtiz
directory	Default Bildverzeichnis oder Bild-Dateiname	
source	Bildquelle: (d)irectory, (c)amera or (a)lpha	d
gui,G	GUI wird am Anfang angezeigt (wenn keine GUI angezeigt wird, lauscht der CameraManager auf dem Default Port)	1
onlySmallTracker,O	Die Größe der Bilder wird um 50 Prozent herabgesetzt, bevor sie zum Tracker gesendet werden	0
onlySmallDetector,D	Die Größe der Bilder wird um 50 Procent herabgesetzt, bevor sie zum Detektor gesendet werden	0
smokeTest,T	EvalInfo wird automatisch für Smoketest ausgegeben	0

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[BJ81] Baron ; J., Robert: Mechanisms of Human Facial Recognition. In: *International Journal of Man-Machine Studies* 15 (1981), Nr. 2, S. 137–178

[Bol03] Bolme, David S.: *Elastic Bunch Graph Matching*, Colorade State University, Diplomarbeit, 2003

[BP93] Brunelli, R. ; Poggio, T.: Face Recognition: Features versus Templates. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993), Nr. 10, S. 1042–1052

[Bra98] Bradski, Gary R.: Computer Vision Face Tracking For Use in a Perceptual User Interface. In: *Intel Technology Journal* (1998), Nr. Q2, S. 15

[Com05] Committee, The C++ S.: *Draft Technical Report on C++ Library Extensions*. Version: 2005. http://www.open-std.org/JTC1/SC22/WG21/docs/papers/2005/n1836.pdf, Abruf: 17.03.2007

[CRM03] Comaniciu, Dorin ; Ramesh, Visvanathan ; Meer, Peter: Kernel-Based Object Tracking. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (2003), Nr. 5, S. 564–577

[CTB92] Craw, I. ; Tock, D. ; Bennett, A.: Finding Face Features. In: *Proc. 2nd European Conf. Computer Vision*, 1992, S. 92–96

[Dau88] Daugman, J. G.: Complete discrete 2-D Gabor transform by neural networks for image analysis and compression. In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 36 (1988), Nr. 7, S. 1169–1179

[Faw04] Fawcett, Tom: *ROC Graphs: Notes and Practical Considerations for Researchers*. Version: 2004. http://home.comcast.net/~tom.fawcett/public_html/papers/ROC101.pdf, Abruf: 19.03.2007

[FHT00] Friedman, J. ; Hastie, T. ; Tibshirani, R.: Special Invited Paper. Additive Logistic Regression: A Statistical View of Boosting. In: *The Annals of Statistics* 28 (2000), Nr. 2, S. 337–374

[Fre95] Freund, Y.: Boosting a Weak Learning Algorithm by Majority. In: *Inform. Comput.* 121 (1995), September, Nr. 2, S. 256–285. – Also appeared in COLT90

[FS95] Freund, Y. ; Schapire, R. E.: A decision-theoretic generalization of on-line learning and an application to boosting. In: *European Conference on Computational Learning Theory*, 1995, S. 23–37

[FT97] Fieguth, P. ; Terzopoulos, D.: Color-based tracking of heads and other mobile objects at video frame rates. In: *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1997, S. 21–27

[HHD98] Haritaoglu, Ismail ; Harwood, David ; Davis, Larry S.: W4: Who, When, Where, What: A Real Time System for Detecting and Tracking People. In: *Proc. Third Conf. Face and Gesture Recognition*, 1998, 222–227

[HW94] Hunke, M. ; Waibel, A.: Face locating and tracking for human-computer interaction. In: *Proc. 28th Asimolar Conf. On Signals, Sys. and Comp.*, 1994, S. 1277–1281

[IB98] Isard, Michael ; Blake, Andrew: Condensation – conditional density propagation for visual tracking. In: *International Journal of Computer Vision* 29 (1998), Nr. 1, S. 5–28

[IM01] Isard, M. ; MacCormick, J.: BraMBle: a Bayesian multiple-blob tracker. In: *Proccedings Eighth International Conference on Computer Vision*, 2001, S. 34–41

[KK96] Kjeldsen, R. ; Kender, J.: Finding Skin in Color Images. In: *Proc. Second Int'l Conf. Automatic Face and Gesture Recognition*, 1996, S. 312–317

[Kol96] Koller, Dieter: *Model-Based Object Tracking in Road Traffic Scenes*. http://www.vision.caltech.edu/koller/ModelTracking.html. Version: 1996, Abruf: 17.03.2007

[KST+06] Katsarakis, Nikos ; Souretis, George ; Talantzis, Fotios ; Pnevmatikakis, Aristodemos ; Polymenakos, Lazaros: *3D Audiovisual Person Tracking Using Kalman Filtering and Information Theory*. http://www.ait.edu.gr/research/RG1/files/CLEAR_06b.pdf. Version: 2006, Abruf: 17.03.2007

[LBP95] Leung, T.K. ; Burl, M.C. ; Perona, P.: Finding Faces in Cluttered Scenes Using Random Labeled Graph Matching. In: *Proc. Fifth IEEE Int'l Conf. Computer Vision*, 1995, S. 637–644

[LKP03] Lienhart, R. ; Kuranov, A. ; Pisarevsky, V.: Empirical Analysis of Detection Cascades of Boosted Classifiers for Rapid Object Detection. In: *German Pattern Recognition Symposium*, 2003, S. 297–304

[LTC95] Lanitis, A. ; Taylor, C.J. ; Cootes, T.F.: An Automatic Face Identification System Using Flexible Appearance Models. In: *Image and Vision Computing* 13 (1995), Nr. 5, S. 393–401

[McC03] McConnell, Steve: *Code Complete*. Microsoft Press, 2003

[MP01] Moon, H. ; Phillips, P.J.: Computational and Performance aspects of PCA-based Face Recognition Algorithms. In: *Perception* 30 (2001), S. 303–321

[MPP01] Mohan, A. ; Papageorgiou, C. ; Poggio, T.: Example-based object detection in images by components. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23 (2001), April, Nr. 4, S. 349–361

[OJAD00] Orr ; J., Robert ; Abowd ; D., Gregory: The smart floor: a mechanism for natural user identification and tracking. In: *Proceedings of ACM CHI 2000 Conference on Human Factors in Computing Systems* Bd. 2, 2000 (Short talks: multimodal interaction), S. 275–276

[Pet06] Peters, Victor: *Effizientes Training ansichtsbasierter Gesichtsdetektoren.* Universität Bielefeld, January 2006

[POP98] Papageorgiou, C. ; Oren, M. ; Poggio., T.: A general framework for Object Detection. In: *International Conference on Computer Vision*, 1998, S. 555–562

[RBK98] Rowley, H. ; Baluja, S. ; Kanade, T.: Neural Network-Based Face Detection. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (1998), Nr. 1, S. 38–44

[RKK+98] Rajagopalan, A. ; Kumar, K. ; Karlekar, J. ; Manivasakan, R. ; Patil, M. ; Desai, U. ; Poonacha, P. ; Chaudhuri, S.: Finding Faces in Photographs. In: *Proceedings of the Sixth IEEE International Conference on Computer Vision*, 1998, S. 640–645

[SB91] Swain, M. ; Ballard, D.: Color indexing. In: *International Journal of Computer Vision* 7 (1991), November, Nr. 1, S. 11–32

[SBB03] Sim, T. ; Baker, S. ; Bsat, M.: The CMU Pose, Illumination, and Expression Database. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25 (2003), Nr. 12, S. 1615–1618

[Sch90] Schapire, R.E.: The strength of weak learnability. In: *Machine Learning* 5 (1990), Nr. 2, S. 197–227

[SM04] Shakhnarovich, Gregory ; Moghaddam, Baback: Face Recognition in Subspaces / Mitsubishi Electric Research Laboratories. 2004 (TR2004-041). – Forschungsbericht

[SP96] Sobottka, Karin ; Pitas, Ioannis: Segmentation and tracking of faces in color images. In: *Proceedings of the Second International Conference on Automatic Face and Gesture Recognition*, 1996, S. 236–241

[SPP05] Stergiou, A. ; Pnevmatikakis, A. ; Polymenakos, L.: Audio-Visual Person Identification. In: *2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*, 2005

[TP91] Turk, M. ; Pentland, A.: Eigenfaces for Recognition. In: *Journal of Cognitive Neuroscience* 3 (1991), Nr. 1, S. 71–86

[VJ01] Viola, Paul ; Jones, Michael: Rapid Object Detection using a Boosted Cascade of Simple Features. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* Bd. 01, 2001, S. 511–518

[WADP97] Wren, Christopher ; Azarbayejani, ALi ; Darrell, Trevor ; Pentland, Alex: Pfinder: Real-Time Tracking of the Human Body. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997), July, Nr. 7, S. 780–785

[WFKM96] Wiskott, Laurenz ; Fellous, Jean-Marc ; Krüger, Norbert ; Malsburg, Christoph von d.: Face Recognition by Elastic Bunch Graph Matching / Institut für Neuroinformatik. Ruhr-Universität Bochum, D-44780 Bochum, 1996 (IR-INI 96-08). – Forschungsbericht

[WMR01] Wallhoff, Frank ; Müller, Stefan ; Rigoll, Gerhard: Hybrid Face Recognition Systems for Profile Views Using The MUGSHOT Database. In: *IEEE ICCV Workshop on Recognition, Analysis, and Tracking of Faces and Gestures in Real-Time Systems (RATFG-RTS'01).* Vancouver, Canada : IEEE Computer Society Washington, DC, USA, 2001, S. 0149

[YC97] Yow, K.C. ; Cipolla, R.: Feature-Based Human Face Detection. In: *Image and Vision Computing* 15 (1997), Nr. 9, S. 713–735

[YH94] Yang, G. ; Huang, T. S.: Human Face Detection in Complex Background. In: *Pattern Recognition* 27 (1994), S. 53–63

[YKA02] Yang, Ming-Hsuan ; Kriegman, David J. ; Ahuja, Narendra: Detecting Faces in Images: A Survey. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24 (2002), Nr. 1, S. 34–58

[ZCPR03] Zhao, W. ; Chellappa, R. ; Phillips, P. J. ; Rosenfeld, A.: Face recognition: A literature survey. In: *ACM Computing Surveys* 35 (2003), Nr. 4, S. 399–458