Studienarbeit

Gesichtsdetektion mittels Detektorkaskaden aus mehreren Kameraperspektiven

Tanya Garneva
Juni 2006

Universität Karlsruhe (TH)
Fakultät für Informatik
Institut für Theoretische Informatik
Prof. Dr. A. Waibel

Betreuer: Dipl. Inform. K. Nickel
Dr. R. Stiefelhagen

Inhaltverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Einleitung

Um ein besseres Verständnis vom Sachverhalt dieser Arbeit zu gewährleisten, muss noch am Anfang der Unterschied zwischen zwei im Bereich des maschinellen Sehens oft auftretenden Begriffen festgelegt werden: die Lokalisation (engl. *detection*) und die Identifikation (engl. *recognition*).

Es ist zu unterscheiden zwischen der Lokalisation eines Gesichts im Bild und der Zuordnung des Gesichts zu einer bestimmten Person. Im ersten Fall wird geprüft, ob und wo ein Gesicht zu sehen ist (das eigentliche Thema dieser Arbeit), und im zweiten Fall wird bestimmt, wer da zu sehen ist, quasi die Erkennung einer Person (die als ein Anwendungsbeispiel der Detektion anzusehen ist).

Die meisten bekannten Anwendungen sind unter dem Begriff Gesichtserkennung bekannt. Dabei muss beachtet werden, dass, damit ein Gesicht zugeordnet werden kann, es zuerst als solches detektiert werden muss. Das bedeutet, dass in den meisten Fällen vor der Erkennung eine Detektion der Gesichter stattfinden muss.

Die Gesichtsdetektion kann zum Beispiel bei Videokonferenzen oder Tagungen eingesetzt werden. Mit ihrer Hilfe kann die Anzahl der Teilnehmer festgestellt werden. Wird der Prozess der biometrischen Gesichtserkennung daran angeknüpft, kann der momentan aktive Teilnehmer festgestellt werden. Aufgrund dieser Daten können dann Besprechungsprotokolle automatisiert erstellt werden.

Ein anderer Begriff, der oft mit Detektion zusammen vorkommt, ist Tracking. Ziel des Trackings ist die Bewegung eines Objektes zu verfolgen. Um die Effektivität des Trackingsprozesses zu erhöhen, erweist sich oft als sinnvoll, diese mit einer Objektdetektion zu verknüpfen.

In der vorliegenden Arbeit wurde der bereits von Viola und Jones [1], [2] entwickelte Objektdetektionsalgorithmus beim speziellen Anwendungsbeispiel der Gesichtsdetektion in einer Multi – Kamera – Umgebung eingesetzt. Ziel dabei ist es, die Effektivität und Robustheit des Algorithmus zu überprüfen sowie geeignete Trainingskaskaden zu entwickeln. Auf Basis dieser Kaskaden wurde dann ein Programm entwickelt, mit dem die eigentliche Gesichtsdetektion durchgeführt und als Ergebnis die Anzahl der sich im Raum befindenden Menschen bestimmt wird.

Dabei wurde speziell auf die Schwierigkeiten, die bei dieser konkreten Anwendung im Bereich der Gesichtsdetektion auftreten, geachtet. Einige Beispiele werden im folgen Kapitel aufgelistet und kurz erörtert.

1.1. Probleme bei der Gesichtsdetektion

Probleme die sich bei der Detektion aus dem Inhalt dieser Bilder ergeben, lassen sich in verschiedene Kategorien einteilen.

Die erste Kategorie ist durch die sogenannte Rotation des Gesichtes charakterisiert. Ein nach vorn gerichtetes Gesicht kann geneigt oder gedreht sein. Dadurch ist es möglich, dass verschiedene Gesichtsmerkmale wie Augen, Ohren oder Gesichtskonturen nicht mehr sichtbar sind und so nicht mehr zu Detektion herangezogen werden können.

Ein wichtiger Faktor in der Kommunikation zwischen Menschen ist der Gesichtsausdruck. Der Ausdruck beeinflusst dabei die Erscheinung des Gesichts zum Teil sehr stark (zum Beispiel um die Mund und Augenpartien). Das kann zu erheblichen Problemen während der Gesichtsdetektion führen.

Das gleiche gilt auch für die Gesichtsmerkmale. Typische Gesichtsmerkmale können durch Hüte, Bärte, Brillen, etc. verdeckt sein. Gleichzeitig wiederum kann man Brille oder Bart als zusätzliche Gesichtsmerkmale betrachten.

Eine weitere Schwierigkeit kann durch eine eventuelle Verdeckung des Gesichtes entstehen. Durch größere Objekte im Vordergrund kann das Gesicht teilweise verdeckt sein, so dass man es trotz frontaler Gesichtsorientierung mit Problemen, wie bei unterschiedlicher Pose zu tun haben kann.

Entscheidend für die Effektivität des Gesichtdetektionsprozesses sind auch die Lichtbedingungen, die während der Aufnahme der Bilder herrschen. Zu analysierende Bilder können bei Lichtbedingungen aufgenommen worden sein, für die das System nicht trainiert wurde. Solche Bedingungen umfassen Anzahl, Art und Richtung der Lichtquellen (Schatten), Lichttemperatur, zusätzliche Reflektionen der Umgebung, oder Ähnliches.

Auch die Kameraparameter der aufnehmenden Kamera, wie Weißabgleich, Fokussierung oder Farbtemperatur, können ins Gewicht fallen. Die Art und Intensität der Strukturierung eines Bildhintergrundes sind ebenfalls von Bedeutung.

1.2. Verfahren zur Gesichtsdetektion

In der Literatur ist eine Vielzahl von Arbeiten zu finden, die sich mit dem Thema Gesichtsdetektion befassen. Yang et al. [3] sprechen in ihrem Artikel von über 150 Verfahren. Diese lassen sich in verschiedene Kategorien einteilen, wobei die Übergänge aufgrund von
Kombinationen verschiedener Methoden manchmal fließend sind. Im Folgenden werden die wesentlichen Verfahren zusammenfassend aufgelistet:

1. Wissensbasierte Verfahren: Diese Methode basiert auf das menschliche Wissen über die typische Gesichtsstruktur. Üblich werden Regel aufgebaut, die Zusammenhänge zwischen unterschiedliche Gesichtsmerkmale erfassen.
2. Verfahren basierend auf invarianten Merkmalen: Es gibt Verfahren, die sich auf das Bestimmen von Merkmalen konzentrieren, welche von unterschiedlichen Kopfhaltungen oder anderen Problemen unbeeinflusst sind. Dieser Ansatz basiert auf der Idee, dass Menschen ohne Schwierigkeit, Gesichter unter unterschiedlichen Verhältnissen erkennen können und deshalb invariante Merkmale existieren müssen. Diese Merkmale werden extrahiert und ein Modell ihrer Verhältnisse zueinander aufgebaut.
3. Mit Vorlagen vergleichenden Verfahren: Anstatt menschlichem Wissen über das Verhältnis zwischen Gesichtsmerkmalen in Regeln zu fassen, kann auch eine Standardvorlage für ein Gesicht definiert oder durch Parameter beschrieben werden. Soll ein Bild analysiert werden, wird dieses mit der Vorlage in Relation gesetzt. Diese Relation werden dann bewertet, um eine Aussage darüber zu treffen, ob es sich um ein Gesicht handelt oder nicht.
4. Ansichtsbasierte Verfahren: Die Vielfältigkeit menschlicher Gesichter in einer Vorlage zu definieren gestaltet sich schwierig. Ein Ansatz ist deshalb, die Eigenheiten eines Gesichts aus repräsentativen Beispielen zu lernen. Verfahren, die in diese Kategorie fallen, basieren meist auf Statistik und Maschinenlernen. Die Variabilität drückt sich dann in den Parametern aus, die das entstehende Modell beschreiben.

Eine Übersicht über die unterschiedlichen Methoden wird in der folgenden Tabelle 1 gegeben.

Verfahren
Wissensbasierte Verfahren
Verfahren basierend auf invarianten Merkmalen - Gesichtsmerkmale - Farbe - Textur - Kombinationen
Mit Vorlagen vergleichenden Verfahren - Vordefinierte Gesicht - Templates - Verformbare Templates
Ansichtsbasierte Verfahren - Eigenfaces - Verteilungsbasierte Methoden - Neuronale Netze - Support Vector Machine (SVM) - Naïve Bayes Classifier - Hidden Markov Model (HMM) - Informations-Theoretische Vorgehensweise

Tabelle 1: Kategorisierung der Methoden zur Gesichtsdetektion

Die vorliegende Arbeit befasst sich mit Verfahren basierend auf invarianten Merkmalen. Deshalb wird im nächsten Kapitel näher auf die verwandten Arbeiten in diesem Bereich angegangen.

1.3. Verwandte Arbeiten

Der in der Arbeit von Campadelli, Cusmai und Lanzarotti [4] vorgeschlagene Detektionsalgorithmus lokalisiert Hautregionen und Augen auf 2D Bilder, basierend auf Farbinformationen. Die Methode funktioniert unabhängig von der Position und Größe des Gesichtes sowie mit einzelnen oder Gruppen von Gesichtern. Die Detektion erfolgt in zwei Schritten. Im ersten werden aufgrund der Hautfarbe Regionen ausgesucht, in denen sich ein Gesicht befinden könnte. Aufgrund der schon beschriebenen Beleuchtungsproblematik oder der durch Rassenunterschiede abweichenden Hautfarbe bietet diese Methode keine besonders hohe Zuverlässigkeit. Deshalb wird im Rahmen eines zweiten Validierungsschrittes nach mindestens einem Auge im Bereich des lokalisierten Gebietes gesucht. Dafür werden für das Auge spezifische chromatische Charakteristiken eingesetzt. Wird ein Auge erkannt, so handelt es sich bei der detektierten Region tatsächlich um ein Gesicht. Es sind keine zusätzlichen Handeinstellungen und Operationen notwendig. Die Methode wurde bereits als erster Schritt eines Personenidentifizierungssystems eingesetzt.

Ähnlich wie bei dem später in dieser Arbeit ausführlich beschrieben Viola-Jones Algorithmus benutzt auch Schneidermann [9], [14] eine Kaskadenmethode zur Objektdetektion. Seine Kombination von zuerst einfacheren Merkmalauswertungen mit solcher von steigender Komplexität erreicht sowohl eine hohe Genauigkeit, als auch eine gute Recheneffizienz. Die Methode wurde erfolgreich zur Detektion von Menschen und Fahrzeugen eingesetzt.

Da Beleuchtungsvarianz das größte Problem während der Detektion darstellt, bieten Song und Nevatia [5] eine Methode an, bei der die Detektion nicht auf der Gesichtshautfarbe basiert, sondern auf den biometrischen Zusammenhängen zwischen Menschenkopf und -körper aufbaut. Wie in der vorliegenden Arbeit wird für die Detektion der AdaBoost Algorithmus von Viola und Jones eingesetzt. Die Methode betrachtet Kopf und Körper sowohl separat, als auch in Zusammenhang. Dadurch wird die Robustheit der Methode erhöht, weil zuverlässige Ergebnisse auch selbst dann zu erzielen sind, wenn eines der beiden Detektionsmerkmale für die Kamera unsichtbar ist.

Eine Vielzahl an Literaturverweisen zu Arbeiten, die sich mit den unterschiedlichen Verfahren zur Gesichtsdetektion befassen, kann im Artikel von Jang et al. [3] gefunden werden.

2. Theoretische Grundlagen

2.1. Objektdetektion

Die in diesem Kapitel beschriebenen Grundlagen der Objektdetektion basieren auf den Papers von Viola und Jones *„Robust Real-time Object Detection"* [1] und *„Fast Multi - view Face Detection"* [2]. All die Zahlen, Abbildungen und Tabellen, die zur Verdeutlichung der dargestellten Thematik verwendet wurden, stammen von den oben genannten Veröffentlichungen.

Der in dieser Arbeit untersuchte Objektdetektionsalgorithmus stellt einen Detektor von visuellen Objekten dar, der Bilder extrem schnell verarbeiten lässt beim Erzielen einer sehr hohen Detektionsrate. Dabei sind drei Schlüsselpunkte wichtig: Der erste ist die Einführung einer neuen Bilddarstellung, die Integralbild *(integral image)* genannt wird und eine sehr schnelle Auswertung der Bilder mittels der sogenannten *features* (Merkmale) erlaubt. Der zweite ist ein lernender Algorithmus, basierend auf AdaBoost, der eine kleine Anzahl von kritischen Visualeigenschaften vorwählt und extrem leistungsfähige Klassifikatoren [6] erbringt. Der dritte Schlüsselpunkt ist eine Methode zur Kombinierung von Klassifikatoren in einer "Kaskade", die erlaubt, Hintergrundregionen des Bildes schnell wegzuwerfen, um mehr Rechenressourcen für die vielversprechenderen und objektähnlichen Regionen freizustellen. In der Literatur sind eine Reihe von Experimenten bekannt, die die Anwendung dieses Algorithmus im Gebiet der Gesichtsdetektion beschreiben und vergleichbare Ergebnisse zu den besten existierenden Systemen vorzeigen [8, 12, 13, 14, 16].

2.1.1. Merkmale

Wie es im Bereich der Gesichtdetektion typisch ist, wird das Eingangsbild mit einem Suchfenster durchlaufen. An jeder Position wird eine unabhängige Entscheidung betreffend des Vorhandenseins eines Gesichtes getroffen. Dieses führt zu einer sehr großen Anzahl von Klassifikatorsauswertungen; ungefähr 50000 in einem 320 x 240 Bild.

Viola und Jones verwenden eine *geboostete* Ansammlung von Merkmalen, um die Bildfenster zu klassifizieren. Nach dem AdaBoost Algorithmus von Freund und von Schapire [8] wird ein Satz binärer Klassifikatoren von einem Trainingssatz erlernt. Jeder Klassifikator ist eine einfache Funktion, die von den Summen aller Pixel in einem Rechteck gebildet wird, welche von einer Schwelle gefolgt werden. Da sie interpretiert und sichtbar gemacht werden können, werden diese Klassifikatoren von Viola und Jones als Merkmale bezeichnet.

Das Detektionsverfahren von Viola und Jones stuft die Bilder ein, basierend auf dem Wert der einfachen Merkmale. Es gibt viele Beweggründe für den Einsatz von Merkmalen, anstatt direkt die Pixel zu verwenden. Ein auf Merkmale basierendes System funktioniert viel schneller als ein auf Pixel basierendes System.

Die einfachen Funktionen, die benutzt werden, sind von den Haar Basis Funktionen abgeleitet, die von Papageorgiou [10] verwendet worden sind. Es werden drei Arten von Merkmalen verwendet. Der Wert eines Zwei-Rechteck-Merkmals ist die Differenz zwischen der Summe der Pixel innerhalb zweier rechteckiger Regionen. Die Regionen haben die gleiche Größe und Form und sind horizontal oder vertikal angrenzend (siehe Abb.1). Ein Drei-Rechteck-Merkmal berechnet die Summe innerhalb zweier äußerer Vierecke, die von der Summe in einem Mittelviereck subtrahiert werden. Schließlich berechnet ein Vier-Rechteck-Merkmal den Unterschied zwischen diagonalen Paaren-Vierecken.

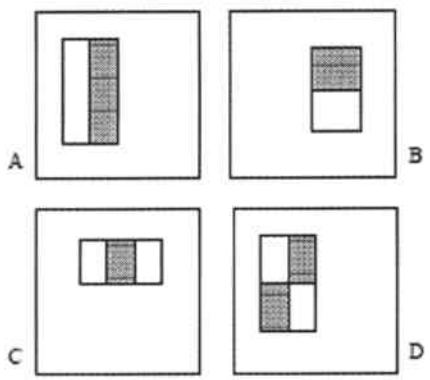


Abbildung 1: Rechteck-Merkmale im Verhältnis zu dem umgebenden Detektion-Fenster

2.1.1.1. Integralbild

Rechteck-Merkmale können mit einer Zwischendarstellung für das Bild, das von Viola und Jones Integralbild genannt wird, sehr schnell berechnet werden. Das Integralbild an der Position x,y enthält die Summe der Pixel über und links von der x,y – Position:

$$ii(x,y) = \sum_{x' \le x, y' \le y} i(x',y'),$$

wobei $ii(x, y)$ das Integralbild und $i(x, y)$ das Originalbild darstellt (siehe Abb.2). Mit den folgenden Ausdrucken:

$$s(x,y) = s(x,y-1) + i(x,y)$$
$$ii(x,y) = ii(x-1,y) + s(x,y)$$

(wo $s(x,y)$ die kumulative Summe ist, $s(x,-1) = 0$ und $ii(-1,y) = 0$) kann das Integralbild in einem Durchlauf über dem ursprünglichen Bild berechnet werden.

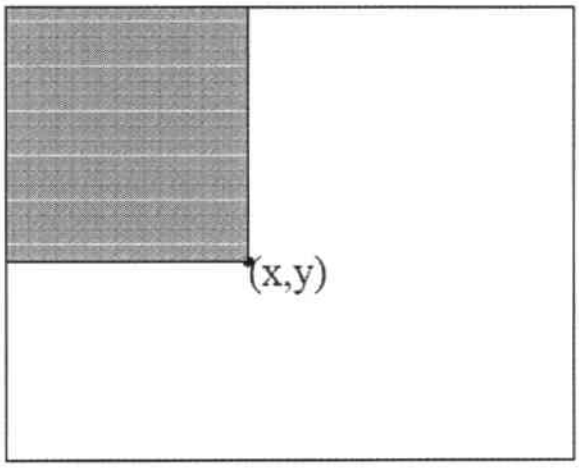


Abbildung 2: Der Wert des Integralbildes am Punkt (x, y) ist die Summe aller Pixel im dargestellten Rechteck

Mit dem Integralbild kann jede Rechteck-Summe mit vier Feldzugriffen berechnet werden (siehe Abb.3). Offenbar kann der Unterschied zwischen zwei Rechteck-Summen mit acht Feldzugriffen berechnet werden. Da die Zwei-Rechteck-Merkmale, die oben definiert wurden, angrenzende Rechteck-Summen miteinbeziehen, können sie mit sechs Feldzugriffen berechnet werden, in acht im Fall des Drei-Rechteck-Merkmale und in neun für Vier-Rechteck-Merkmale.

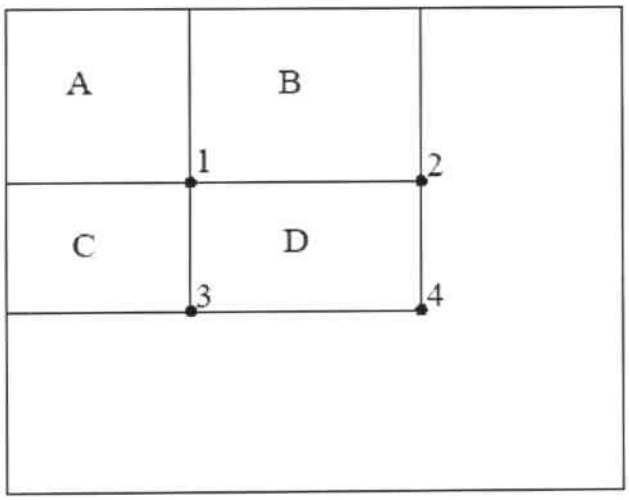


Abbildung 3: Die Summe der Pixels innerhalb des Rechteckes D kann mit vier Feldzugriffen berechnet werden. Der Wert des Integralbildes am Punkt 1 ist die Summe der Pixel in Rechteck A. Der Wert am Punkt 2 ist A + B, am Punkt 3 ist A + C, am Punkt 4 ist A + B + C + D. Die Summe innerhalb D kann als 4 + 1- (2 + 3) berechnet werden.

Ein alternativer Anlass für das Benutzen des Integralbildes stellt die Arbeit von Simard [15] dar. Die Autoren unterstreichen, dass im Falle linearer Operationen (z.B. $f \bullet g$), jede umkehrbare lineare Operation an f und g angewendet werden kann, wenn ihre Inverse am Ergebnis angewendet werden kann. Z.B. im Falle der Faltung:

$$f * g = \iint (f' * g').$$

Die Autoren haben gezeigt, dass die Faltung wesentlich beschleunigt werden kann, wenn die Ableitungen von f und g klein sind. Ähnlich: es kann eine umkehrbare lineare Operation an f angewendet werden, wenn ihre Inverse an g angewendet wird:

$$(f'') * (\iint g) = f * g.$$

Bezogen auf den hier betrachteten Anwendungsfall bedeutet dies, dass die Rechteck-Summe als das Produkt $i \bullet r$ dargestellt werden kann, wobei i das Bild und r das *„box car"* Bild (mit dem Wert 1 innerhalb des interessierenden Rechtecks und 0 außerhalb) ist. Die Operation kann, wie folgt, beschrieben werden:

$$i \cdot r = (\iint i) \cdot r''.$$

Das integrale Bild ist tatsächlich das doppelte Integral des Bildes (zuerst entlang der Reihen und dann der Spalten). Die zweite Ableitung des Rechteckes (zuerst in der Reihe und dann in der Spalte) erbringt vier Deltafunktionen an den Ecken des Rechteckes. Die Auswertung des zweiten Produktes wird mit vier Feldzugriffen vollendet.

2.1.1.2. Diagonale Filter

Experimente zeigen, dass die drei Arten von Merkmalen, die oben vorgestellt wurden, nicht genügend waren, nichtaufrechte Gesichter und nichtfrontale Gesichter mit genug hoher Genauigkeit zu ermitteln. Um dieses zu vermeiden, wurde von Jones und Viola [2] eine vierte Art von Rechteck-Merkmalen entwickelt, die auf diagonalen Strukturen im Bildfenster fokussiert ist. Diese diagonalen Filter werden in Abbildung 4 veranschaulicht. Sie bestehen aus vier sich überlappenden Rechtecken, die rechts in der Abbildung gezeigt werden. Sie funktionieren genauso wie die vorhergehenden Merkmale. Die Summe der Pixel in der dunkelgrauen schattierten Region wird von der Summe in der hellgrauen schattierten Region subtrahiert. Mit einem Integralbild kann der diagonale Filter berechnet werden, indem man nur die 16 Eckpixel betrachtet.

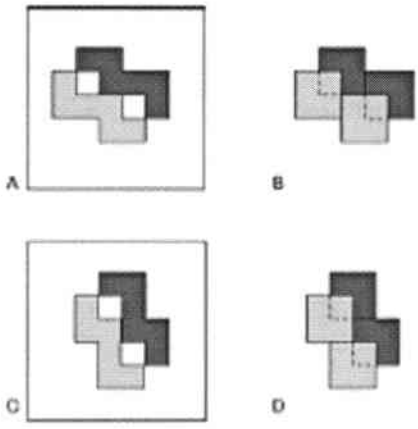


Abbildung 4: Diagonale Filter

2.1.1.3. Skalierung

Um den Berechnungsvorteil der integralen Bildtechnik abzuschätzen, wird eine konventionelle Annäherung betrachtet, in der eine Pyramide von Bildern berechnet wird. Wie die meisten Objektdetektionssysteme tastet der Detektor das Inputbild an vielen Skalen ab; beginnend an der Basisskala, in der Objekte mit einer Größe von 24x24 Pixel detektiert werden, wird das Bild in 11 Skalen gescannt, die um ein Faktor von 1.25 größer sind als die letzte. Die konventionelle Annäherung soll eine Pyramide von 11 Bildern berechnen, jede 1.25 mal kleiner als das vorhergehende Bild. Ein örtlich festgelegter Skaladetektor wird dann über jedem dieser Bilder gescannt. Die Berechnung der Pyramide erfordert einen erheblichen Zeitaufwand. Ausgeführt auf herkömmlicher Hardware ist es extrem schwierig, eine Pyramide bei 15 Frames pro Sekunde zu berechnen.

Demgegenüber haben Viola und Jones einen sinnvollen Satz von Merkmalen definiert, die die Eigenschaft haben, dass ein einzelnes Merkmal an jeder möglichen Skala und Position in einigen Operationen ausgewertet werden kann. Es wird in Abschnitt 2.1.3. gezeigt, dass wirkungsvolle Gesichtdetektore mit so wenigen wie Zwei-Rechteck-Merkmalen konstruiert werden können. Die Berechnungseffizienz dieser Merkmale kann dadurch verdeutlicht werden, dass der komplette Gesichtsdetektionsprozess für ein gesamtes Bild an jeder Skala und bei 15 Frames pro Sekunde in weniger Zeit durchgeführt wird, als eine einzige Bildpyramide mit 11 Niveaus. Jedes Verfahren, das eine Pyramide dieser Art laufen lässt, wird langsamer sein als dieser Detektor.

2.1.2. Der AdaBoost Algorithmus

Es sei darauf hingewiesen, dass mit jedem Bildfenster 45.396 Rechteck-Merkmale assoziiert sind, eine Zahl weit größer als die Anzahl der Pixel. Obwohl jedes Merkmal sehr leistungsfähig berechnet werden kann, ist der komplette Satz aufwendig zu berechnen. Eine Hypothese ist, dass es eine sehr kleine Anzahl von diesen Merkmalen kombiniert werden kann, um einen wirkungsvollen Klassifikator zu bilden. Die Hauptherausforderung ist, diese Merkmale zu finden.

In dem System wird eine Variante von AdaBoost verwendet, um die Merkmale vorzuwählen und den Klassifikator [6] zu trainieren. In seiner ursprünglichen Form wird der AdaBoost - Erlernensalgorithmus verwendet, um die Klassifikationsleistung eines einfachen Erlernensalgorithmus zu *„boosten"* (z.B. konnte er verwendet werden, um die Leistung eines einfachen *perceptron* zu steigern). Er tut dies, indem er eine Ansammlung schwacher Klassifikationsfunktionen kombiniert, um einen stärkeren Klassifikator zu bilden. In der Sprache des *„Boosting"* wird der einfache Erlernensalgorithmus ein schwacher *Learner* genannt. So z.B. sucht der *perceptron* - Erlernensalgorithmus einen Satz der möglichen *perceptrons* und bringt das *perceptron* mit dem niedrigsten Fehler zurück. Der *Learner* wird schwach genannt, weil es nicht erwartet wird, dass sogar die beste Klassifikationsfunktion die Trainingsdaten gut klassifizieren wird (d.h. für ein gegebenes Problem kann das beste *perceptron* die Trainingsdaten nur in 51% der Zeit richtig einstufen). Damit der schwache *Learner „geboostet"* wird, wird gefordert, eine Reihe von Erlernensproblemen zu lösen. Nach dem ersten Umlauf des Lernens werden die Trainingsdaten erneut gewichtet, um die hervorzuheben, die falsch durch den vorhergehenden schwachen Klassifikator eingestuft wurden. Der abschließende starke Klassifikator nimmt die Gestalt eines perceptron an, eine gewichtete Kombination der schwachen Klassifikatoren, die von einer Schwelle gefolgt werden.

Die formalen Garantien, die vom AdaBoost - Erlernensverfahren bereitgestellt werden, sind ziemlich stark. Es wird geprüft, dass die Trainingsfehler des starken Klassifikators sich exponential mit der Zahl der Umläufe zu null nähert.

Das konventionelle AdaBoost Verfahren kann leicht als gieriger Merkmalselektionsprozess bezeichnet werden. Das allgemeine Problem des *„Boosting"* wird betrachtet, indem ein großer Satz Klassifikationsfunktionen mit einer großen Gewichtung kombiniert werden. Die Herausforderung soll eine große Gewichtung mit jeder guten Klassifikationsfunktion und eine kleinere Gewichtung mit schlechten Funktionen verbinden. AdaBoost ist ein aggressiver Mechanismus für das Vorwählen eines kleinen Satzes guter Klassifikationsfunktionen, die dennoch eine bedeutende Vielzahl haben. Betrachtet man eine Analogie zwischen schwachen Klassifikatoren und Merkmalen, ist AdaBoost ein wirkungsvolles Verfahren für

das Heraussuchen einer kleinen Anzahl von guten Merkmalen, die dennoch eine bedeutende Differenziertheit aufweisen.

Eine praktische Methode für das Durchführen dieser Analogie besteht darin, den schwachen *Learner* auf den Satz der Klassifikationsfunktionen einzuschränken, von denen jede von einem einzelnen Merkmal abhängt. Zur Unterstützung dieses Ziels wurde der schwache Erlernensalgorithmus entworfen, um das einzelne Rechteck-Merkmal vorzuwählen, das gut die positiven und negativen Trainingsdaten trennt. Für jedes Merkmal stellt der schwache *Learner* die optimale Schwellklassifikationsfunktion so fest, dass eine Mindestzahl von Trainingsdaten nicht klassifiziert wird. Ein schwacher Klassifikator ($h_j(x)$) besteht folglich aus einem Merkmal (f_j), einem Schwellwert (Θ_j) und einer Parität (P_j), welche die Richtung des Verschiedenheitszeichens anzeigt:

$$h_j(x) = \begin{cases} 1 & wenn \quad p_j f_j(x) < p_j \Theta_j \\ 0 & sonst \end{cases}$$

Hier ist x ein Fenster eines Bildes mit 24x24 Pixels.

In der Praxis kann kein Merkmal die Klassifikationsaufgabe mit einem niedrigen Fehler durchführen. Merkmale, die früh in dem Prozess vorgewählt werden, ergeben eine Fehlerrate zwischen 0.1 und 0.3. Merkmale, die in den späteren Umläufen gewählt werden, wenn die Aufgabe schwieriger wird, liefern Fehlerhäufigkeit zwischen 0.4 und 0.5. Tabelle 2 zeigt den Erlernensalgorithmus.

- Seien $(x_1, y_1), \ldots, (x_n, y_n)$ Trainingssamples mit y_i = 0,1 für negative bzw. positive Samples.
- Gewichtinitialisierung $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ für y_i = 0,1, wobei m und l die Zahl der *negatives* bzw. *positives* sind.
- Für $t = 1, \ldots, T$:
    1. Gewichtnormalisierung,

$$w_{t,i} \longleftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

so dass w_t die Wahrscheinlichkeitsverteilung ist.

2. Für jede Markmal j trainiere einen Klassifikator h_j, das beschränkt ist, einzelne Merkmale zu benutzen. Der Fehler ist bezüglich w_t ausgewertet.

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

3. Wähle der Klassifikator h_t mit dem kleinsten Fehler ε_t.

4. Aktualisiere die Gewichtungen:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i},$$

wo e_i = 0 wenn das Sample x_i korrekt klassifiziert ist, sonst e_i = 1, und

$$\beta_t = \frac{\varepsilon_t}{1-e_t} .$$

- Der starke End – Klassifikator ist:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & sonst \end{cases}$$

wobei $\alpha_t = \log \frac{1}{\beta_t}$.

Tabelle 2: Ablauf des *Boosting* Algorithmus

Für die Aufgabe der Gesichtsdetektion sind die von AdaBoost gewählten Initial-Rechteck-Merkmale sinnvoll. Das erste Merkmal, das vorgewählt wird, basiert auf der Eigenschaft, dass die Region der Augen häufig dunkler als die Region der Nase und der Backen ist (siehe Abb. 5). Dieses Merkmal ist im Vergleich mit dem Detektionfenster verhältnismäßig groß und sollte zur Größe und zur Position des Gesichtes ein wenig unempfindlich sein. Das zweite Merkmal, das vorgewählt wird, beruht auf der Eigenschaft, dass die Augen dunkler als die Brücke der Nase sind.

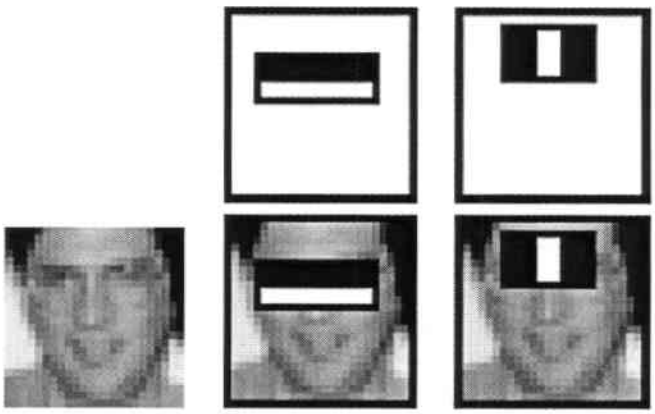


Abbildung 5: Das erste und das zweite Merkmal, das von AdaBoost gewählt wird.

2.1.3. Die Detektorkaskade

Dieser Abschnitt beschreibt einen Algorithmus für das Konstruieren einer Kaskade aus Klassifikatoren, die eine erhöhte Detektionsleistung bei verringerter Berechnungszeit erzielt. Der Schlüsselpunkt ist, dass kleinere und folglich leistungsfähigere *„boosted"* Klassifikatoren konstruiert werden können, die viele der negativen Bildfenster aussortieren, während sie fast alle positiven Samples erkennen. Einfachere Klassifikatoren werden verwendet, um die Mehrheit der Bildfenster auszusortieren, bevor kompliziertere Klassifikatoren aufgefordert werden, um eine niedrige Fehlerrate zu erzielen.

An der Stelle ist die Definition der folgenden Begriffe notwendig:

- *Positives*: richtig detektierte Objekte im Bild
- *False positives*: falsch detektierte Objekte
- *Negatives*: richtig detektierte non – Objekte (Hintergrund)
- *False negatives*: nicht detektierte Objekte

Die Stufen in der Kaskade werden durch Trainieren der Klassifikatoren mit AdaBoost konstruiert. Beginnend mit einem starken Zwei-Markmal-Klassifikator kann ein effektvoller Gesichtsfilter bekommen werden, indem die Schwelle des starken Klassifikators so angepasst wird, dass die Rate der *false negatives* minimiert wird. Die initialisierte AdaBoost Schwelle,

$$\frac{1}{2}\sum_{t=1}^{T}\alpha_t ,$$

ist so entworfen, dass sie eine niedrige Fehlerrate auf den Trainingdaten ergibt. Eine untere Schwelle erbringt eine höhere Detektionsrate und eine höhere Rate der *false positives*. Basierend auf Performancemessungen mit Validierungstrainingsdaten kann behauptet werden, dass der Zwei-Merkmal-Klassifikator 100% der Gesichter mit einer Rate der *false positives* von 40% detektiert. Siehe Abb. 5 für eine Beschreibung der zwei Merkmale, die in diesem Klassifikator benutzt werden.

Die Detektionsleistung des Zwei-Merkmal-Klassifikators ist weiterhin unakzeptabel als Objektdetektionssystem. Dennoch kann der Klassifikator die Zahl der zur Weiterverarbeitung benötigten Bildfenster mit Hilfe weniger Operationen erheblich reduzieren:

1. Evaluieren der Rechteck-Merkmale (zwischen 6 und 9 Felder pro Merkmal sind erforderlich).
2. Berechnung der schwachen Klassifikatoren für jedes Merkmal (eine Schwelleoperation pro Merkmal ist erforderlich).
3. Kombination der schwachen Klassifikatoren (ein Multiplizieren pro Merkmal, eine Addition und schließlich eine Schwelle sind erforderlich).

Ein Zwei-Merkmal-Klassifikator ergibt ungefähr 60 Mikroprozessorinstruktionen. Es ist schwierig, sich vorzustellen, dass jeder einfachere Filter eine höhere Aussortierungsrate erzielen könnte. Zum Vergleich würde das Scannen einer einfachen Bildmaske oder des *single-layer-perceptron* mindestens 20 mal mehrere Operationen pro Bildfenster erfordern.

Die gesamte Form des Detektionsprozesses ist ein degenerierter Entscheidungsbaum, der "Kaskade" genannt wird [11] (siehe Abb. 6). Ein positives Resultat vom ersten Klassifikator löst die Auswertung eines zweiten Klassifikators aus, der auch angepasst worden ist, um eine sehr hohe Detektionsrate zu erzielen. Ein positives Resultat vom zweiten Klassifikator löst einen dritten Klassifikator und so weiter. Ein negatives Resultat an irgendeinem Punkt führt zur sofortigen Aussortierung des Bildfensters.

Die Struktur der Kaskade reflektiert die Tatsache, dass innerhalb jedes einzelnen Bildes eine Mehrzahl an Fenstern *negatives* sind. So versucht die Kaskade möglichst viele *negatives* im frühesten Stadium auszusortieren. Liegt ein *positives* vor, wird eine Auswertung aller Klassifikatoren in der Kaskade ausgelöst, was in der Tat ein außerordentlich seltener Fall ist.

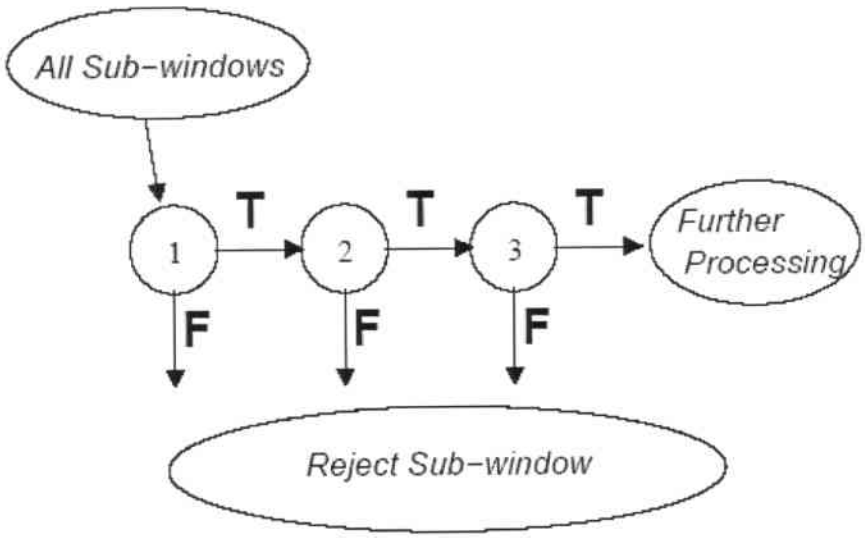


Abbildung 6: Schematische Beschreibung von der Detektionskaskade

Ganz wie im Entscheidungsbaum werden nachfolgende Klassifikatoren nur mit jenen Samples trainiert, die alle vorhergehenden Stufen durchlaufen haben. Infolgedessen ist die Aufgabe des zweiten Klassifikators schwieriger als die des ersten. Die Samples, die es durch die erste Stufe geschafft haben, sind "härter" als die typischen Samples. Mit einer gegebenen Detektionsrate haben spätere Klassifikatoren eine entsprechend höhere Rate von *false positives*.

2.1.3.1. Trainieren einer Kaskade von Klassifikatoren

Der Entwicklungsprozess der Kaskade wird von einer Gruppe von Detektions- und Performancezielen bestimmt. Bei der Gesichtdetektionsaufgabe haben frühere Systeme gute Detektionsraten (zwischen 85 und 95 Prozent) und eine extrem niedrige Rate der *false positives* erzielt (10^{-5} oder 10^{-6}). Die Zahl von den Kaskadestufen und die Größe jeder Stufe müssen ausreichend sein, um eine ähnliche Detektionsperformance bei einer Minderung des Rechenaufwandes zu erzielen.

Die Rate der *false positives* in einer trainierten Kaskade ist:

$$F = \prod_{i=1}^{K} f_i,$$

wobei F die Rate der *false positives* vom Kaskaden-Klassifikator, K die Nummer des Klassifikators und f_i die falsche Positivrate vom i - ten Klassifikator ist, durch das die Samples durchgelaufen sind. Die Detektionsrate ist:

$$D = \prod_{i=1}^{K} d_i,$$

wobei D die Detektionsrate vom Kaskade-Klassifikator, K die Nummer des Klassifikators und d_i die Detektionsrate vom i - ten Klassifikator ist, durch das die Samples durchgelaufen sind.

Beim Starten des Trainingsprozesses werden konkrete Ziele für die Rate der *false positives* und die Erekennungsrate für jede Stufe des Kaskadenprozesses festgelegt. Als Beispiel kann eine Detektionsrate von 0,9 von einem zehnstufigen Klassifikator erreicht werden, in dem jeder Klassifikator eine Detektionsrate von 0,99 aufweist ($0,9 \approx 0,99^{10}$). Diese Aufgabe mag ziemlich schwierig klingen, wird aber wesentlich erleichtert durch die Tatsache, dass jede Stufe eine Rate der *false positives* von rund 30 % erreichen muss ($0,3^{10} \approx 6 * 10^{-6}$).

Die Anzahl der ausgewerteten Merkmale beim Scannen eines realen Bildes ist ein Wahrscheinlichkeitsprozess. Jedes gegebene Bildfenster schreitet durch die Kaskade einen Klassifikator weiter vor, bis es entschieden ist, dass das Bildfenster ein *negatives* ist oder, bei seltenen Umständen, das Fenster mit jedem Test Erfolg hat und positiv *„gelabelled"* wird. Das erwartete Verhalten dieses Prozesses wird durch die Verteilung der Bildfenster in einem typischen Testsatz festgestellt. Das Schlüsselmaß jedes Klassifikatoren ist seine Rate der *positives*, der Anteil der Fenster, die als möglicherweise interessierende Objekte *„gelabelled"* werden. Die erwartete Anzahl an Merkmalen, die ausgewertet wird, ist:

$$N = n_0 + \sum_{i=1}^{K} \left(n_i \prod_{j<i} p_j \right),$$

wobei K die Nummer von Klassifikatoren, p_i die Positivrate des i - ten Klassifikators und n_i die Merkmalanzahl im i-ten Klassifikator ist.

Der Prozess, mit dem jedes Element der Kaskade trainiert wird, erfordert etwas Achtsamkeit. Das AdaBoost Erlernensverfahren, präsentiert in Abschnitts 2.1.2, versucht nur Fehler zu minimieren und ist nicht spezifisch entworfen, um eine hohe Detektionsrate auf Kosten von einer großen Rate der *false positives* zu erzielen. Eine einfache und sehr konventionelle Art und Weise diesen Fehler auszugleichen, ist das Anpassen der Schwelle des *perceptrons*, das von AdaBoost produziert wird. Höhere Schwellen ergeben Klassifikatoren mit einer niedrigeren Rate der *false positives* und einer niedrigeren Detektionsrate. Tiefere Schwellen ergeben Klassifikatoren mit einer höheren Rate der *false positives* und einer höheren Detektionsrate. Es ist an diesem Punkt nicht klar, ob die Anpassung der Schwelle auf diese Weise das Training und die Verallgemeinerungsgarantien, die von AdaBoost zur Verfügung gestellt werden, erhalten.

Der gesamte Trainingsprozess bezieht zwei Arten von Kompromissen mitein. In den meisten Fällen erzielen Klassifikatoren mit mehreren Merkmalen eine höhere Detektionsrate und senken die Rate der *false positives*. Gleichzeitig erfordern Klassifikatoren mit mehreren Merkmalen mehr Zeit zum Rechnen. Prinzipiell könnte man eine Optimierung durchführen, indem man:

1. die Zahl der Klassifikator - Stufen
2. die Zahl der Merkmale, n_i, in jeder Stufe
3. die Schwelle in jeder Stufe

optimiert, mit dem Ziel, bei gegebenen F und D die erwartete Merkmalanzahl N zu minimieren. Das Optimum zu finden, ist leider eine enorm schwierige Aufgabe.

In der Praxis wird ein sehr einfaches System benutzt, um einen wirkungsvollen Klassifikator zu produzieren. Der Benutzer wählt die minimale akzeptable Rate für f_i und d_i vor. Jede Schicht der Kaskade wird von AdaBoost (wie in der Tabelle 2 beschrieben ist) mit der Merkmalanzahl trainiert, bis die Zieldetektion erhöht wird und die Rate der *false positives* für diese Stufe getroffen wird. Die Rate wird festgestellt, indem man den gegenwärtigen Detektor mit einem Validierungssatz prüft. Werden die Anforderungen an der Rate der *false positives* nicht getroffen, wird eine andere Schicht der Kaskade hinzugefügt. Der negative Satz für nacheinanderfolgende Trainingschichten wird durch eine Sammlung aller falschen Detektionen erhalten, die der gegenwärtige Detektor in einem Satz von Bildern gefunden hat, die keine Detektionsobjekte beinhalten. Dieser Algorithmus wird genau in Tabelle 3 gegeben.

- Der Benutzer bestimmt die Werte von f, die maximale akzeptable Rate der *false positives*, und d, die minimale akzeptable Detektionsrate pro Stufe.
- Der Benutzer bestimmt die Zielrate der *false* Positiven, F_{target}.
- P = Satz von positiven Samples
- N = Satz von negativen Samples
- $F_0 = 1.0$; $D_0 = 1.0$
- $i = 0$
- while $F_i > F_{target}$
    - $i \leftarrow i + 1$
    - $n_i = 0$; $F_i = F_{i-1}$

- while $F_i > f \times F_{i-1}$
  - ❖ $n_i \leftarrow n_i + 1$
  - ❖ Benutze P und N zum Trainieren eines Klassifikators mit n_i Merkmale mittels AdaBoost.
  - ❖ Auswerte der aktuell Kaskade-Klassifikator vom Validierungssatz um F_i und D_i zu bestimmen.
  - ❖ Setze die Schwelle für den i-ten Klassifikator herunter, bis die Detektionsrate des aktuellen Kaskade-Klassifikators mindestens den Wert von $d * D_{i-1}$ erreicht (das beeinflusst auch F_i).
- $N \leftarrow \varnothing$
- Wenn $F_i > F_{target}$ dann werte den aktuell Kaskade-Detektor auf dem Satz von non-face Bildern aus und verschiebe jede falsche Detektion im Satz N.

Tabelle 3: Der Trainingsalgorithmus für das Bilden eines Kaskade-Detektors

2.2. Triangulation

In der vorliegenden Arbeit wurde auch eine Methode zum Bestimmen der 3-D Koordinaten in einem Raum und der dazu gehörenden Software benutzt. Dank dieser Methode kann festgestellt werden, ob zwei Punkte von verschiedenen Kameraperspektiven miteinander korrespondieren und schließlich das gleiche Objekt darstellen. [17, 18]

Mehrere Kameras stellen einige Perspektiven auf einer beobachteten Szene zur Verfügung. Wenn das räumliche Verhältnis zwischen den Kameras und ihren internen optischen Bedingungen bekannt ist, können Objekte in der Szene in den 3-D Koordinaten lokalisiert werden.

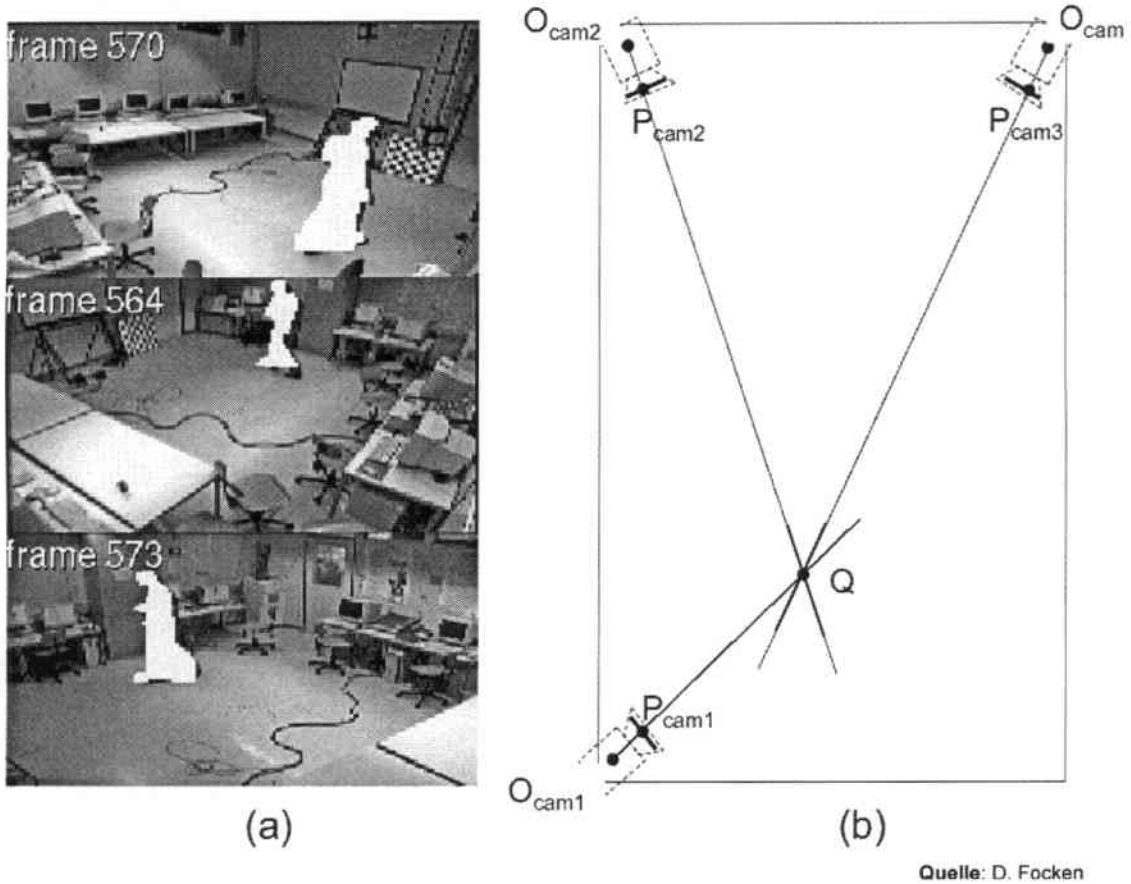
Zuerst, erscheint ein Objekt als Punktemenge in einem Bild. Um seine 3-D Position zu berechnen, muss ein einzelner Punkt auf dem Objekt vorgewählt werden, der seine Position darstellt. Eine mögliche Wahl ist der Schwerpunkt des Objekts. Im Falle der vorliegenden Arbeit wird ein Punkt aus dem Kopf des Menschen gewählt. Die Hauptschwierigkeit für einen Lokalisationsalgorithmus ist es, den entsprechenden Punkt des Objekts in jedem der unterschiedlichen Kamerabilder zu finden.

Wurden erst einmal die korrespondierenden Punkte eines Objektes in den unterschiedlichen Perspektiven identifiziert, werden dann geometrische Verhältnisse zwischen den Perspektiven eingesetzt, um die Position des Objektes zu berechnen. Diese geometrischen Verhältnisse werden durch die Perspektivekameramodelle jedes Visual-Sensors beschrieben. Da die Kameras während der Experimente nicht verschoben werden, bleiben die Kameramodelle in einem vorgehenden Kalibrierungsschritt erhalten. Außer den notwendigen vorzugehenden Schritten zur Berechnung der Position eines Objektes wird im Folgenden auch die Positionsberechnung selbst beschrieben, bekannt auch als „Triangulation".

Um den Algorithmus zu beschreiben, wird erst einmal auf ein allgemeines Beispiel eingegangen, in dem eine Positionsbestimmung mit Hilfe von drei Kameras durchgeführt wird, wie sie in der Diplomarbeit von D. Focken [7] vorgestellt wird. Die Abbildung 7 (a) zeigt eine Person in einem Innenraum durch drei Kameraperspektiven. Die Berechnung der Position der Person in den 3-D Koordinaten bedeutet erstens, einen Punkt aus dem Körper vorzuwählen, der seine Position darstellt. In der Abbildung 7 (a) wird die Oberseite des Kopfes als solcher Punkt gekennzeichnet.

Es wird angenommen, dass die geometrischen Verhältnisse zwischen und in den Kameras, d.h. die Kameramodelle, bekannt sind, so können die Positionen der Kameras auf ein Diagramm des Raumes gezeichnet werden (siehe Abb. 7 (b)). Außerdem können die Bildkoordinaten des Kopfes in jeder Kamera verwenden werden, um einen Strahl zu

zeichnen, der von der optischen Mitte der Kamera durch die Projektion des Kopfes durchgeht. Da der Strahl von der Oberseite des Kopfes entsteht, geht er auch durch die Position des Kopfes in der Szene durch. Zum Beispiel für Kamera ' cam1 ' geht der Strahl von der optischen Mitte O_{cam1} durch die Projektion des Kopfes auf dem Bild P_{cam1} durch die Oberseite des Kopfes in der Szene Q durch.



Quelle: D. Focken

Abbildung 7: Triangulieren die 3-D Position des Kopfes mittels dreier Kameras

Offensichtlich ergibt der Durchschnitt dieser Strahlen die Position des Kopfes Q. Um die 3-D Koordinaten von Q zu erhalten, werden die Strahlen im 3-D Koordinatensystem parametrisiert, das durch die Kameramodelle hergestellt wird:

$$x_{ray} = t * u + P, \text{ wobei } t \in R$$

Der Strahl für Kamera ' cam1 ' wird dargestellt, indem man P zur Koordinate von O_{cam1} einstellt und indem man die Richtung von Vektor u mit dem Kameramodell von ' cam1 ' und den Bildkoordinaten von P_{cam1} berechnet.

Die Berechnung des Durchschnitts der Strahlen wird erzielt, indem man jeden Strahl in eine lineare Gleichung umwandelt, deren Lösungsraum die Punkte auf dem Strahl in den 3-D Koordinaten ist:

$$A_i * x_{ray\,i} = b_i \text{, wobei } A_i \in R^{2x3},\ b_i \in R^2$$

Die lineare Gleichung für einen Strahl ist unterbestimmt, aber ein Schnitt zwischen zwei oder mehren Strahlen ergibt eine überbestimmte lineare Gleichung, die leicht zum Beispiel mit der Pseudo-inversen Technik gelöst werden kann:

$$\begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} x_{intersect} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Die Lösung dieser Gleichung ist $x_{intersect}$, ein 3-D Punkt, in dem die drei Strahlen in der Gleichung sich schneiden. In der Praxis schneiden sich die drei Strahlen nicht fehlerlos wegen Kalibrierung und Korrespondenzfehlern. Da der Lösungsraum in einer strengen mathematischen Richtung leer ist, berechnen wir $x_{intersect}$ mit der kleinsten Quadratlösung der überbestimmten linearen Gleichung. Um die Ungenauigkeit des Durchschnitts festzusetzen, bietet die Norm des Rests der Gleichung eine gute Möglichkeit:

$$r = \left\| \begin{pmatrix} A_1 \\ A_2 \\ A_3 \end{pmatrix} x_{intersect} - \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \right\|$$

Das Maß r ist besonders nützlich zur Berechnung der Korrespondenzvermutungen oder zum Überprüfen die Qualität eines Kalibrierungsprozesses.

Es wurde angenommen, dass Kameramodelle vorhanden sind, die das Verhältnis zwischen absoluten Bildkoordinaten und realen Weltkoordinaten darstellen. Die Kameramodelle ermöglichen die Berechnung der Strahlparametrisierung von der optischen Mitte der Kamera zum Objekt.

3. Vorgehensweise

Ziel der vorliegenden Arbeit ist es, den bereits von Viola und Jones entwickelten Objektdetektionsalgorithmus auf dem speziellen Anwendungsbeispiel der Gesichtsdetektion in einer Multi – Kamera - Umgebung anzuwenden. Als besonders hilfreich erweist sich dabei das Benutzen von Programmen und Funktionen, die bereits auf diesem Algorithmus basieren und somit als Werkzeuge für die Realisierung der gestellten Aufgabe dienen können. Ein Satz von solchen Programmen und Funktionen bietet die weitverbreitete Bibliothek OpenCV [http://sourceforge.net/projects/opencvlibrary/].

Die Open Source Computer Vision Library (OpenCV) ist eine auf C-Plattform entwickelte Bibliothek, die sehr nützliche Funktionen für Objektdetektion bietet. OpenCV stellt eine große Anzahl von Funktionen zur Verfügung, die bei der Programmierung von Anwendungen aus dem Bereich des Maschinensehens hilfreich sind. In der vorliegenden Arbeit finden die folgenden Funktionsgruppen aus OpenCV Verwendung: elementare Bildverarbeitungsoperationen, Objektdetektionsfunktionen und Programme.

Mit Hilfe der in OpenCV zur Verfügung stehenden Programme wurden zwei Kaskaden für Gesichtsdetektion trainiert, jeweils eine für frontale Gesichter und eine für Profile. Die auf Basis dieser Kaskaden nachfolgende Gesichtsdetektion wird mittels eines im Rahmen dieser Arbeit entwickelten Programms verwirklicht.

Der erste Schritt besteht allerdings in der Vorbereitung der Trainingsdaten. Dieser Prozess wird im folgenden Kapitel beschrieben.

3.1. Erstellen der Kaskaden

3.1.1. Trainingsdaten

Als Input für die Programme, die zum Trainieren des Gesichtsdetektionsalgorithmus eingesetzt werden, werden möglichst umfangreiche Datensätze mit Bildern benötigt. Im Internet stehen eine Vielzahl an Objektdatenbanken, die von verschiedenen Hochschulen und Forschungsinstituten zur Verfügung gestellt worden sind. Im Rahmen dieser Arbeit wurden Daten aus ISL Seminaren vom Jahr 2003 benutzt. Dabei handelt sich um Filme von sieben unterschiedlichen Seminaren, die alle im so genannten *smart room* durchgeführt worden sind. Aus jedem Seminar wurden je vier Segmente, jedes mit einer Dauer von ca. 5 Minuten, aufgenommen. Die Bildfrequenz beträgt 15 Frames pro Sekunde und die Bildgröße 640 x 480 Pixel. Für die Auswertung wurde aus jedem der sieben Seminare je ein Segment benutzt.

Für das Erstellen der Kaskaden werden drei Sätze von Bilder gebraucht. Zwei Sätze, die die sogenannten positiven Samples beinhalten, und ein mit den sogenannten negativen Samples. Die positiven Samples sind die Bilder, die das gewünschte Objekt, in dem Fall die Gesichter, enthalten und die negativen, die Bilder, die kein Objekt enthalten und somit den Hintergrund darstellen.

Bei den positiven Samples wird weiterhin zwischen frontalen Gesichtern und Profilen unterschieden. Daher wurden auch zwei unterschiedliche Datensätze gebildet, die als Input zur Erstellung von zwei unterschiedlichen Kaskaden dienten. Bei den Trainingsbildern mit den frontalen Gesichtern müssen zwei Augen sichtbar sein und mindestens 2/3 vom Gesicht, ebenso können auch Gesichter, bei denen ein Teil vom Gesicht mit der Hand bedeckt ist, brauchbar sein. Die frontalen Gesichter können auch bis 45° gedreht werden. Als Profile dienen dann Bilder von Gesichtern, die um 45° bis 100° von der frontalen Position gedreht sind. In den beiden Fällen wurden auch leicht geneigte Gesichter in die Trainingssätze aufgenommen. (Abb. 8 und 9)

Die interessierenden Objekte wurden mit der Unterstützung eines Hilfsrogramms „gelabeled". Dabei werden die Gesichter von Hand so annotiert, dass das ganze Gesicht in einen Rahmen passt und ein Abstand von 2 – 3 Pixel bis zum Rande bleibt. Das Hilfsprogramm gibt die Koordinaten der so erstellten Rechtecke aus. So wurden von den sieben Filmsegmenten insgesamt Bilder von 2034 frontalen Gesichtern und 2482 Profilen erstellt.

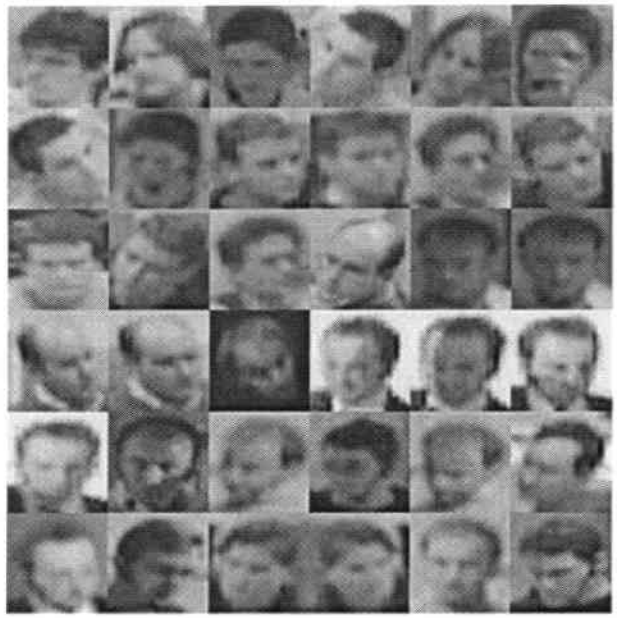


Abbildung 8: Beispiele von den Trainingsdaten, die für die Erstellung der Frontalgesichterkaskade gebraucht wurden.

Abbildung 9: Beispiele von den Trainingsdaten, die für die Erstellung der Profilgesichterkaskade gebraucht wurden.

Anschließend wurden die Bereiche der gelabelten Gesichter schwarz eingefärbt. So konnten die ganzen Bilder für das Erstellen der negativen Samples benutzt werden. Es wurden insgesamt 2696 Bilder als Input für die negativen Samples erstellt. (Abb. 10)

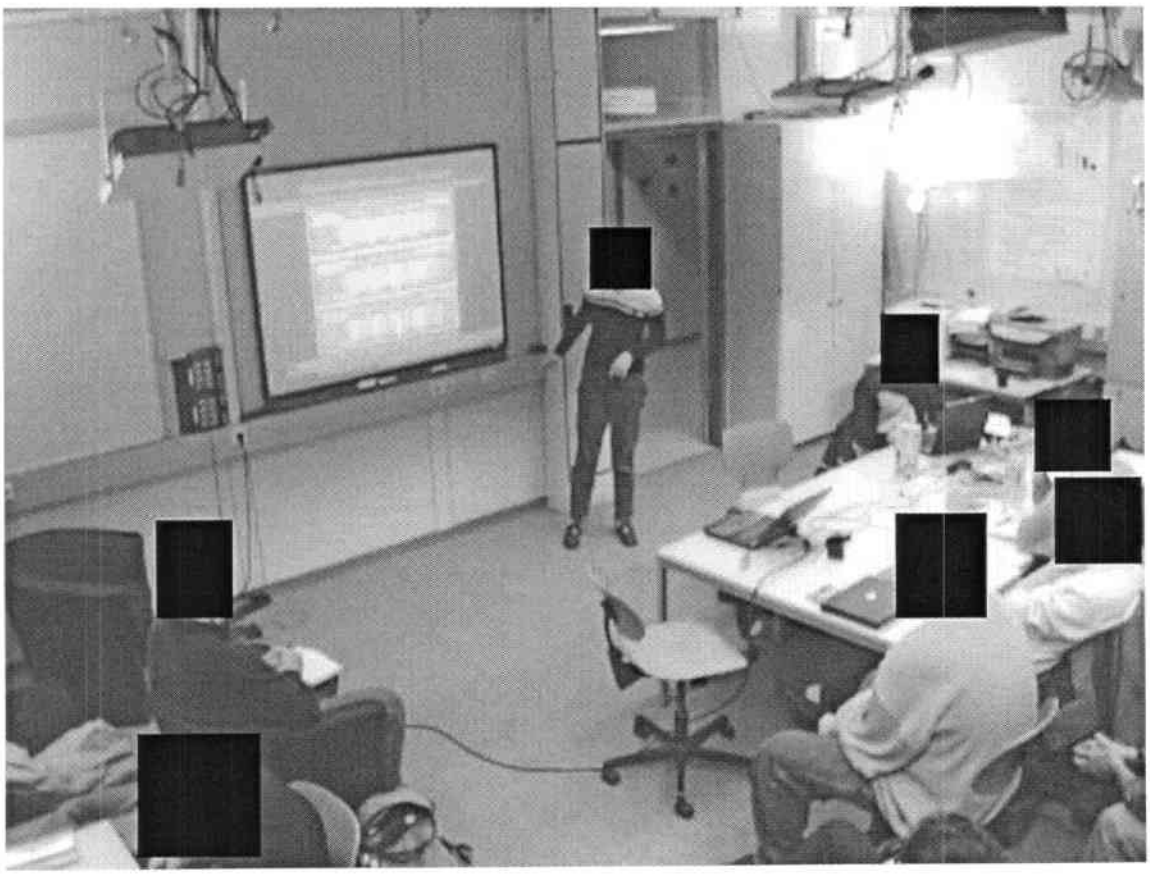


Abbildung 10: Negative Samples

Mit Hilfe eines Programms wurde dann eine Liste erstellt, in der der Name des jeweiligen Ursprungsbildes, die Anzahl der gelabelten Gesichter in diesem Bild sowie ihre Position beschrieben sind. Diese Liste dient als Inputdatei für das OpenCV Programm „Createsamples", mit dessen Hilfe die eigentlichen positiven und negativen Samples erzeut werden. Dabei werden die unterschiedlichen gelabelten Gesichter auf eine Einheitsgröße von 20 x 20 Pixel skaliert.

Createsamples erzeugt seinerseits eine sogenannte .vec Datei, die der eigentliche Trainingsdatensatz ist. Somit ist der erste, aber arbeitsaufwendigste Schritt - die Trainingsdatenaufbereitung - abgeschlossen.

3.1.2. Trainingsprozess

Der eigentliche Trainingsprozess oder die Erstellung der Kaskaden wird mit Hilfe des OpenCV Haartrainig Programm durchgeführt. Dieses Programm basiert seinerseits auf dem in Kapitel 2.1 ausführlich beschriebenen Viola – Jones Algorithmus. Als Input für Haartraining Programm dient die vom Createsamples erzeugte .vec Datei, die den kompletten Trainingsdatensatz, bestehend aus den positiven Samples, beinhaltet.

Wie bereits in Kapitel 2.1.3.1 erläutert wurde, besteht einer der wesentlichen Herausforderungen beim Trainingsprozess in der Einstellung der Anzahl der sogenannten Stufen. Werden zu wenig Stufen als Eingangsparameter bei der Erstellung der Kaskade eingegeben, so werden viele Objekte als Gesichter detektiert und die Fehlerquote liegt entsprechend hoch. Werden umgekehrt zu viele Stufen eingestellt, so besteht die Gefahr, dass nicht alle Gesichter auch als solche detektiert werden. Also besteht die Aufgabe darin, ein Optimum zu finden, bei dem die Kaskaden nach dem Training einen effektiven und zugleich zuverlässigen Gesichtsdetektionsprozess gewährleisten. Daher wurde die Anzahl der Stufen zwischen 20 und 50 variiert. Eine Lösung mit einer Stufenanzahl zwischen 28 und 30 scheint dabei als bester Kompromiss zu dienen.

Beim Starten des „Haartainig" Programms muss weiterhin auch die Größe der Samples, die in „Createsamples" Programm benutzt wurde, eingegeben werden, nämlich 20 x 20 Pixel. Die minimale verlangte Detektionsrate für jede Stufe wurde auf 0,995 gesetzt. Die maximale *false alarm* Rate für jede Stufe wurde auf 0,5 gesetzt. Somit könnte die *false alarm* Rate der gesamten Kaskade einen minimalen Wert von 9.10^{-10} (max_false_alarm ^ $anzahl\_stufen$) erreichen. Alle andere Parameter wurden auf deren Defaultwerte eingestellt. Die Anzahl der benutzten Merkmale war 43483. Die nach dem Training tatsächlich erreichten Werte sind für die Detektionsrate 0.875123 und für die *false alarm* Rate 0.000056. Der Trainingsprozess hatte 4,5 Stunden auf einem Rechner mit 800MB Speicher gedauert.

3.2. Benutzen der Kaskaden

Als Abschluss dieser Arbeit wurde ein Programm entwickelt, das den Prozess der Gesichtdetektion auf Basis der Kaskaden, die nach dem im vorigen Kapitel beschriebenen Trainingsprozess gebildet wurden, im sogenannten *smart room* demonstrieren muss.

Im nächsten Kapitel folgt eine kurze Beschreibung dieses „intelligenten" Raumes.

3.2.1. *Smart room*

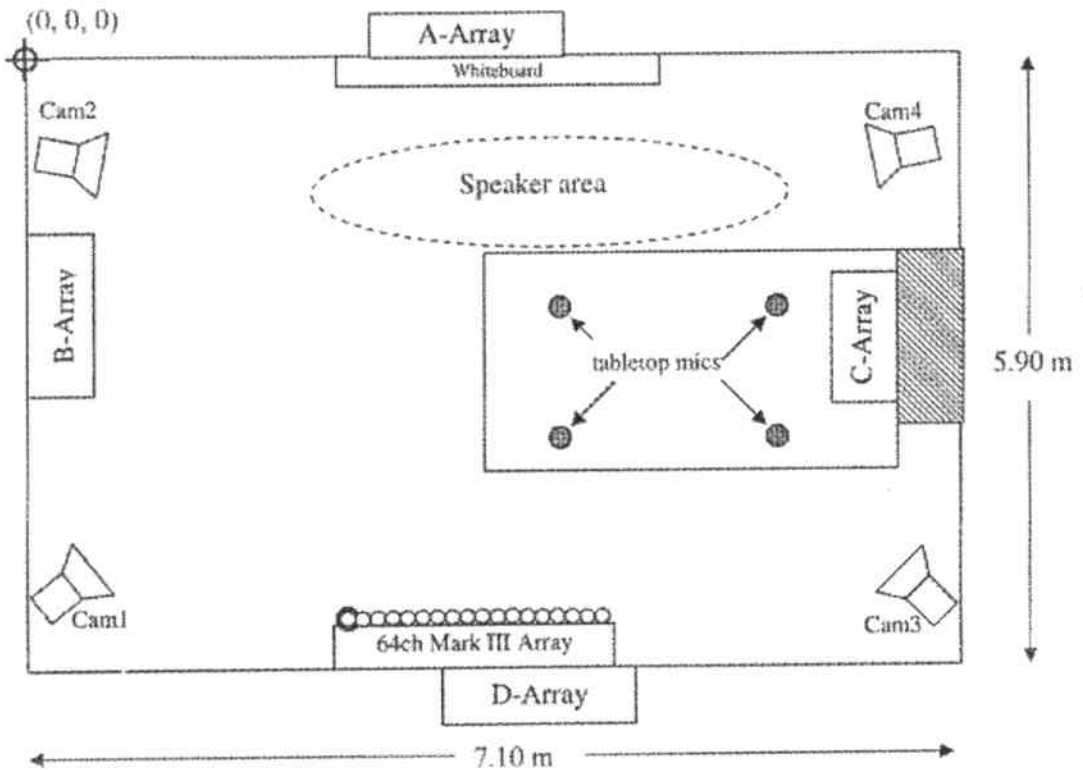


Abbildung 11: *Smart room*

Der in der Abbildung 11 dargestellte *smart room* von ISL ist ein 7,10 mal 5,90 Meter großer Konferenzraum. Ein Tisch in der Mitte des Raumes, ein *whiteboard* und ein Videoprojektor erlauben das Durchführen von Vorträgen oder Sitzungen. Die sensorische Ausrüstung besteht aus vier stationären Kameras, einigen Mikrophonen und einer Mikrophonreihe für Spracherkennung und Sprecherlokalisierung sowie weiteren nicht stationären Geräten. Die vier Kameras, die im Rahmen dieser Arbeit benutzt wurden, sind 2,70 m Meter über dem Fußboden an den vier Ecken des Raumes positioniert. Deren Position ist fixiert und wird für

diese Arbeit als gegeben betrachtet. Dieses ergibt besondere Randbedingungen, mit denen beim Entwickeln des Programms zu rechnen ist.

Durch die Position der Tafel ist der Raum, in dem sich der Sprecher befinden könnte, weitgehend festgelegt. Die Zuhörer dagegen können sich beliebig im Raum befinden. Weiterhin befinden sich nicht alle Zuhörer in Ruhe, sondern betreten und verlassen den Raum während des Vortrages. Durch die fixierte Position der vier Kameras und die willkürliche Verteilung der Personen im Raum kommt es zu Situationen, bei denen ein Gesicht auf mehrere Kameras zu detektieren ist. Es ist aber auch möglich, dass es sich Personen im Raum befinden, deren Gesichter auf keiner der vier Kameras zu sehen sind. D.h., die vier Kameras decken den *smart room* nicht vollständig ab. All dies erschwert dann z.B. die Antwort auf die Frage: Wie viele Personen befinden sich gerade im Raum?

Besondere Anforderungen gelten für die Beleuchtung des Raumes. Wechselnde Lichtbedingungen sowie Reflektionen wie z.B. vom *whiteboard*, beeinflussen deutlich die Ergebnisse des Gesichtdetektionsprozesses. Daher müssen Segmente, bei denen andere Lichtbedingungen herrschen, als bei jenen, die bei den Trainingssegmenten vorhanden waren, vorsichtig ausgewertet werden.

3.2.2. Gesichtdetektionsprogramm

Die eigentliche Gesichtsdetektion innerhalb eines Eingabebildes wird durch die OpenCV Funktion *cvHaarDetectObject* durchgeführt. Diese Funktion wurde als Basis für ein Programm benutzt, das eine robuste und effektive Gesichtsdetektion, ausgehend von einer Bildfolge, durchführen kann. Die *cvHaarDetectObject* Funktion braucht zwei Parameter. Zum einen die Trainingskaskaden und zum anderen die Bilder, auf denen die Gesichter detektiert werden müssen. Durch die Funktion *detect_and_draw* werden in einer Schleife die Bildfolgen als Input für die Funktion *cvHaarDetectObject* zur Verfügung gestellt. Aus Gründen, die später in diesem Kapitel diskutiert werden, kann die maximale Anzahl an Bildern in einer Bildfolge maximal 127 betragen. Weiterhin können die Frequenz, mit der Bilder aus einem Segment für die Auswertung ausgewählt werden, und die Dauer und die Lage der Sequenz, aus der die Bilder stammen, im Programm definiert werden. Ein Segment dauert ca. 5 Minuten, und es stehen 15 Bilder pro Sekunde zur Verfügung.

Die wesentliche Aufgabe des Gesichtsdetektionsprogramms besteht darin, die Ergebnisse von *cvHaarDetectObject* auszuwerten und dabei sicherzustellen, ob es sich beim detektierten Objekt tatsächlich um ein Gesicht handelt. Die Funktion *cvHaarDetectObject* findet rechteckige Regionen im gegebenen Bild, die ähnlich aussehen wie die Objekte, mit denen die Kaskade trainiert. Die Funktion durchläuft das Bild mehrmals in verschiedenen

Skalierungen. Jedes Mal werden die überlappenden Regionen im Bild berücksichtigt. Auf diese Regionen werden die Klassifikatoren mittels der Funktion *cvRunHaarClassifierCascade* angewendet. Alle Regionen, die als ein Objekt detektiert werden, werden als Ergebnis von *cvHaarDetectObject* als Sequenz aus Rechtecken geliefert.

Als erster Filter wird dann vom Gesichtsdetektionsprogramm die Größe dieses Rechtecks überprüft. Damit es sich beim detektierten Objekt um ein Gesicht handelt, dürfen die Kantenlängen des Rechteckes nicht 60 Pixel überschreiten. Das ist die maximale Größe, mit der ein Gesicht auf dem Kamerabild bei der gegebenen Auflösung und Position der Kameras erscheinen kann.

Wird diese erste Prüfung bestanden, wird die Position des Mittelpunktes dieses Rechtecks in einem sogenannten Histogramm gespeichert. Ein Histogramm ist eine graphische Darstellung der Häufigkeitsverteilung von Messwerten innerhalb einer Messreihe. In unserem Fall stellt es die Position des Mittelpunktes aller detektierten Gesichter innerhalb der ausgewerteten Bildreihe dar. Technisch wird das Histogramm erstellt, indem der Wert w jedes Pixels, auf dem sich ein Mittelpunkt eines detektierten Objektes befindet, um einen bestimmten Wert i verringert wird. Am Anfang haben alle Pixel im Histogramm den Wert 255. Es liegt also ein weißes Bild vor. Der Mittelpunkt jedes detektierten Objektes wird dann als grauer Punkt auf dem Histogramm sichtbar (Abb. 12 und Abb. 13). Sollte ein Objekt in allen Bilder einer Bildreihe detektiert werden und seine Position nicht geändert haben, so sollte der Pixel, der sein Mittelpunkt darstellt, den Wert 0 haben und somit schwarz sein. Daher gilt für i folgende Beziehung:

$$255 - n*i = 0$$

wobei n die Anzahl der Bilder in der ausgewerteten Bildreihe darstellt.

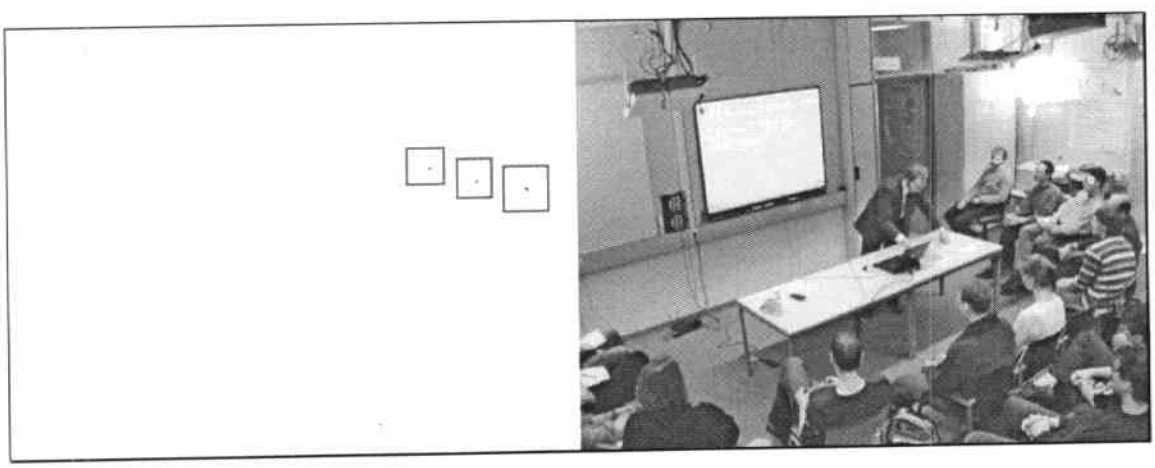


Abbildung 12: Histogramm und das dazugehörige Bild

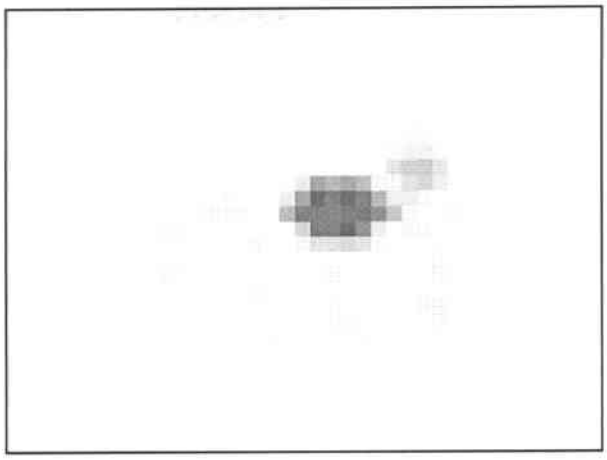


Abbildung 13: Vergrößerter Bereich eines Histogramms

Im Rahmen einer Voruntersuchung wurde ein Programm geschrieben, das die Ergebnisse aus der *cvHaarDetectObject* Funktion für jedes Bild der Bildreihe visualisiert. Dabei werden die Objekte, die durch die Kaskade für frontale Gesichter detektiert wurden, mit blauen Rechtecken und die Objekte, die durch die Kaskade für Profile detektiert wurden, mit roten Vierecken umrandet (Abb. 14). Dabei fällt auf, dass relativ viele Gesichter von den beiden Kaskaden gleichzeitig detektiert wurden. Daher muss die oben definierte Beziehung für den Wert i modifiziert werden, indem die zwei Kaskaden berücksichtigt werden:

$$255 - 2*n*i = 1.$$

Daher ergibt sich eine Begrenzung für die maximale Anzahl der Bilder in einer Bildreihe n. Die darf nicht größer als 127 sein, damit der Wert des Pixels w nicht negativ werden kann.

Abbildung 14: Ergebnisse aus den Kaskaden für Profile und frontale Gesichter

In der Realität liegen natürlich nicht alle Mittelpunkte der detektierten Objekte auf allen Bildern auf der gleichen Position. Der Grund dafür ist, dass sich die Personen im Raum bewegen. Da es sich bei den ausgewerteten Bildersequenzen um Seminaraufnahmen handelt, sind die Bewegungen der Zuhörer nicht allzu stark. Daher ist es möglich, einen Raum zu definieren, in der sich die Position des Gesichtes im Sitzen durch Nicken oder eine andere Sitzposition ändern kann. Auf dem Histogramm wird diese Umgebung U als ein Rechteck mit den Maßen 30 x 30 Pixel definiert.

Durch die Bewegung der Zuhörer im *smart room* während des Seminars folgt, dass nicht jeder Punkt auf dem Histogramm ein Gesicht darstellt. Nebeneinander liegende Punkte stellen einfach die unterschiedlichen Positionen des Gesichtes auf den unterschiedlichen Bildern der Bildreihe dar. Um festzustellen, wie viele Gesichter in einem Segment tatsächlich zu detektieren sind, wird eine Suche im Histogramm durchgeführt. Es wird zunächst das erste Pixel j mit einem Wert unter 255 gesucht, also ein graues Pixel, das möglicherweise ein detektiertes Gesicht darstellt. Von diesem Pixel ausgehend wird eine Summe der Abweichungen der Werte aller Pixel vom Wert 255 (weiß) in der oben beschriebenen 30 x 30 Umgebung U_j gebildet. So wird der so genannte Klassifikationsschwellwert T gebildet:

$$Tj \equiv \sum_{i=1}^{900} (255 - W_i) \; ; \; i \in Uj \; ; \; Uj = \text{ 30 x 30 Umgebung um Pixel } j$$

Dabei ist W_i der Wert des i-ten Pixel in dem 30 x 30 Raum im Histogramm. Überschreitet der Klassifikationsschwellwert T eine definierte Grenze, wird ein Gesicht markiert und die Werte aller Pixel in der Umgebung U_j auf 255 zurückgesetzt. So kann man mit der Suche fortfahren, bis das Histogramm vollständig durchgesucht ist.

Als Grenze für den Klassifikationsschwellwert T wurde in dieser Arbeit ein Wert von 30 als vorteilhaft gefunden. Dies bedeutet, dass bei einer Bildreihe von z.B. 100 Bildern auf mindestens 30 Bildern ein Gesicht in einem 30 x 30 Raum detektiert wurde.

Anschließend wird noch eine Filterung der detektierten Objekte durchgeführt. Dafür werden Bedingungen für die mögliche Raumverteilung der Gesichter eingeführt. Es dürfen sich keine Gesichter auf dem Boden oder auf der Decke befinden. Dafür wird für jede Kamera eine Grenze bestimmt, in der sich Gesichter befinden dürfen. Bei allen Objekten, die diesen Kriterien nicht entsprechen, handelt es sich um keine Gesichter (Abb 15).

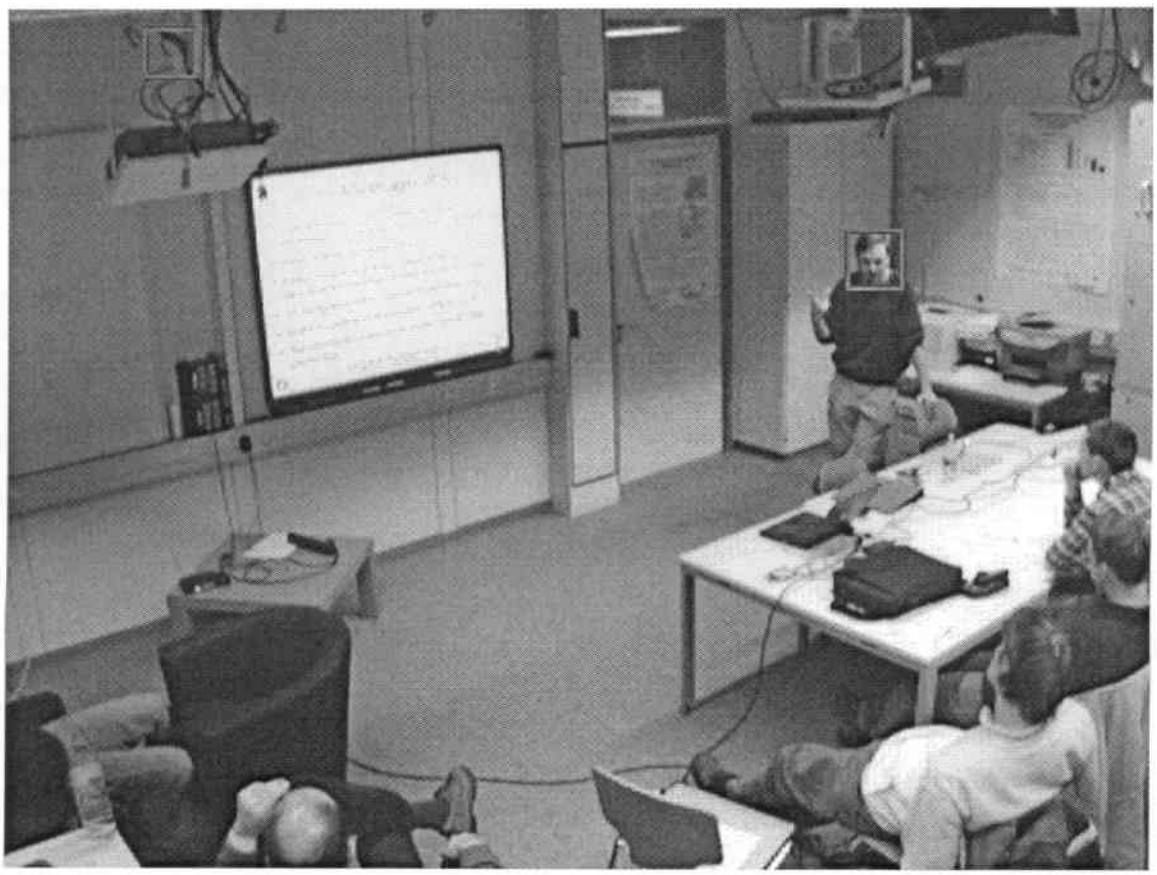


Abbildung 15: Falsch detektiertes Objekt (links oben), später wird dieses Objekt herausgefiltert und nicht zu den gefundenen Gesichter gezählt.

Da der *smart room* von 4 Kameras erfasst wird, werden insgesamt 4 Histogramme für jedes Segment gebildet und ausgewertet. Dies bedeutet allerdings, dass Objekte, die von mehreren Kameras erfasst wurden, auch in den entsprechenden Histogrammen detektiert werden. Um eine Doppelzählung der Gesichter zu vermeiden, wird mittels Triangulation (siehe Abschnitt 2.2) bestimmt, welches Gesicht mit welchem in der anderen Kameraperspektive korrespondiert.

Die Koordinaten der detektierten Objekte aus allen vier Histogrammen werden in einer Matrix gespeichert. So ist die Anzahl und die Position der von den vier Kameras detektierten Objekte beschrieben. Um die mehrfach detektierten Gesichter aus der Gesamtanzahl auszuschließen, werden alle Gesichter, die von einer Kamera erfasst wurden, mit allen Gesichtern, die von einer anderen Kamera detektiert wurden, untereinander auf Korrespondenz mittels Triangulation überprüft. Als Ergebnis dieser Prüfung wird das Residuum zwischen den geprüften Objekten geliefert. Bei genug kleinen Residuen kann behauptet werden, dass es um dasselbe Objekt geht. Insgesamt bilden die vier Kameras sechs mögliche Kamerapaare. Zwei von denen, die aus zwei gegenüberliegenden Kameras bestehen, werden aber nicht verwendet, weil es wegen der spezifischen Kameraposition unmöglich ist, dass auf zwei solchen Kameras gleichzeitig dasselbe Gesicht erscheint.

Dies bedeutet, dass vier solche Korrespondenzprüfungen für jede Bildfolge durchgeführt werden müssen, da vier Kameranachbarpaare vorhanden sind. Jedes Mal, wenn eine Korrespondenz festgestellt wird, wird die Gesamtanzahl der detektierten Personen um eins reduziert.

Anschließend werden mittels Triangulation auch Objekte ausgefiltert, die von mindestens zwei Kameras detektiert wurden, sich aber unter- oder oberhalb einer bestimmten Höhe befinden (kleiner als 0.7m und höher als 2m). Das ist leider nicht bei allen Objekten möglich, da mindestens zwei Kameraperspektiven notwendig sind, um die Lage des Objektes im Raum zu bestimmen.

Nach diesem Schritt gibt das Programm die Anzahl der detektierten Gesichter an sowie die graphische Darstellung der ungefähren Position der detektierten Gesichter aufgrund des letzten Bildes aus der Bildreihe. Diese grafische Darstellung erfolgt allerdings, bevor die doppelt erfassten Gesichter mittels Triangulation identifiziert wurden. Es werden also alle detektierten Gesichter aus den vier Kameraperspektiven markiert.

4. Ergebnisse

Ziel der vorliegenden Arbeit war es, die Effektivität und Robustheit des Viola-Jones Algorithmus beim speziellen Anwendungsbeispiel der Gesichtsdetektion in einer Multi – Kamera – Umgebung zu überprüfen sowie geeignete Trainingskaskaden zu entwickeln, auf deren Basis ein Programm erstellt wurde, das die Anzahl der sich in einem Konferenzraum befindenden Personen bestimmen kann. Um richtig zu beurteilen, inwieweit diese Ziele erreicht wurden, ist ein systematisches Testen verbunden mit einer geeigneten Auswertung der einzelnen Komponenten des entwickelten Programms notwendig. Sinnvoll ist es, diese Auswertung in drei Schritten durchzuführen. Im ersten wird die Qualität des eigentlichen Detektors anhand einzelner Bilder untersucht. Besonders wird dabei auf die Anzahl der Kaskadenstufen geachtet. Im zweiten Schritt wird an ganzen Bildfolgen, die von den einzelnen Kameras geliefert wurden, überprüft, inwieweit die Detektionsergebnisse durch Akkumulation verbessert werden können. Anschließend wird im letzten Schritt die endgültige, mittels Triangulation ermittelte Gesichteranzahl ausgewertet.

4.1. Qualität des Detektors

Um die Qualität des Detektors überprüfen zu können, werden Datensätze von Bildern benötigt, auf denen die Gesichter bereits gelabeled sind. So können die Ergebnisse aus der Gesichtsdetektion für jedes einzelnes Bild mittels eines einfachen Programms überprüft werden. Daraus werden dann die Detektionsrate und die Rate der *false positives* als Mittelwert der Ergebnisse aus allen Bildern im Datensatz ermittelt.

Es wurden zwei Datensätze verwendet. Der erste besteht aus 1543 Bildern, die für das Trainieren der Kaskaden benutzt wurden. Ein weiterer Datensatz, der aus rund 250 Bildern besteht, wurde speziell für die Auswertung gelabeled und nicht für den Trainingsprozess benutzt.

Entscheidend für einen erfolgreichen Kompromiss zwischen Detektionsrate und die Rate der *false positives* ist, wie bereits in Kapitel 3.1.2. erörtert, die Anzahl der Stufen in jeder Kaskade. Betreffend der Anzahl der Stufen werden hier zwei unterschiedliche Varianten betrachtet: Kaskaden mit 28 Stufen und Kaskaden mit 30 Stufen.

Weiterhin wird hier die Qualität der Kaskaden für frontale Gesichter und für Profile getrennt ausgewertet. Im Gesichtdetektionsprogramm werden jedoch beide Kaskaden gleichzeitig angewendet (siehe auch Kapitel 3.2.2).

Die letzte untersuchte Einflussgröße ist die sogenannte maximal zulässige Positionstoleranz (Abb.16). Durch diese Größe wird bestimmt, inwieweit die Position des erkannten Gesichts von der tatsächlichen Position des Gesichts auf dem geprüften Bild abweichen darf. Bei einer Durchschnittskantenlänge der gelabelten Gesichter von ca. 30 Pixel wurden zwei Varianten untersucht – mit Toleranzen von jeweils 5 oder 10 Pixel.

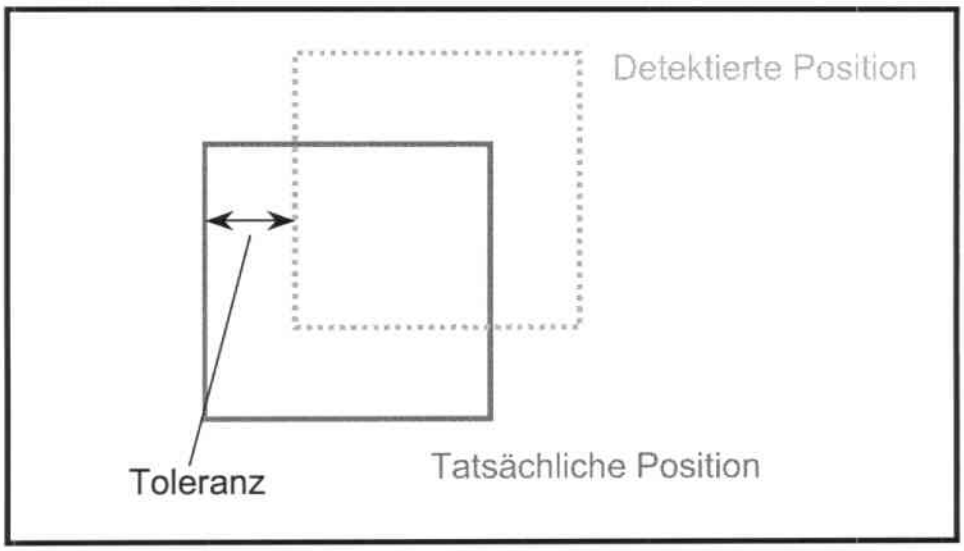


Abbildung 16: Positionstoleranz des detektierten Gesichts

Die Auswertung der Ergebnisse mit dem Datensatz, bestehend aus den Trainingsbildern, ist in der folgenden Tabelle 4 zusammengefasst.

Stufen	Kaskade	Toleranz	Erkennungsrate	false positive
28 Stufen	frontal	10 pix	62%	56%
		5 pix	55%	59%
	profile	10 pix	69%	62%
		5 pix	59%	67%
30 Stufen	frontal	10 pix	58%	44%
		5 pix	52%	47%
	profile	10 pix	65%	55%
		5 pix	55%	60%

Tabelle 4: Ergebnisse mit dem Trainingsdatensatz

Wie erwartet, führt eine niedrigere Stufenzahl zu einer höheren Detektionsrate, aber gleichzeitig auch zu einer höheren Rate der *false positives.* Die Gründe dafür wurden bereits im Kapitel 2.1.3. erläutert. Für die weiteren Untersuchungen wurde die Variante mit der höheren Detektionrate ausgewählt, weil durch die Akkumulation mehrerer nacheinander folgenden Bilder eine Reduzierung der Rate der *false positives* zu erwarten ist.

Als logisch lässt sich auch der Einfluss der Toleranz auf den Ergebnissen interpretieren. Eine weniger strenge Positionstoleranz von 10 Pixel führt zu besseren Ergebnissen, was sowohl die Detektionsrate, als auch die Rate der *false positives* betrifft. Eine größere Abweichung, 10 Pixel sind bezogen auf die Samplekantenlänge immerhin ca. 30%, könnte die Interpretation der Ergebnisse unrealistisch machen. Bei der Auswertung mit dem Testdatensatz wurde allerdings nur die größere Toleranz in Betracht gezogen, da beim Labeln der Gesichter andere Kriterien als die bei den Trainingsbildern verwendet wurden. Deshalb ist eine höhere Ungenauigkeit bei der Position der detektierten Gesichter zu erwarten.

Stufen	Kaskade	Toleranz	Erkennungsrate	*False positives*
28 Stufen	frontal	10 pix	54%	57%
	profile	10 pix	62%	65%
30 Stufen	frontal	10 pix	47%	45%
	profile	10 pix	57%	60%

Tabelle 5: Ergebnisse mit dem Testdatensatz

Die gleichen Tendenzen wie beim Datensatz mit den Trainingsbildern sind auch bei der Auswertung der Ergebnisse zu beobachten, die mit dem Datensatz mit den Testbildern erzielt wurden. Allerdings sind die Werte der Detektionsrate und der Rate der *false positives* etwas schlechter als jene mit den Trainingsdaten, was auch zu erwarten war. Trotzdem befinden sich die Ergebnisse mit diesem Testdatensatz in der gleichen Größenordnung wie diejenige, die mit den Trainingsdaten erzielt wurden.

4.2. Verbesserung durch Akkumulation

Eine Möglichkeit, die Zuverlässigkeit der Gesichtsdetektionsmethode zu steigern, besteht darin, nicht nur einzelnen Bilder, wie das im vorigen Kapitel der Fall war, auszuwerten, sondern ganze Bildfolgen. Dadurch könnte die Auswirkung von Umgebungseinflüssen wie zum Beispiel Reflexionen, wechselnde Lichtbedingungen usw. minimiert werden. Andererseits kann bei einer Situation, bei der Objekte ständig falsch als Gesichter detektiert wurden, vorkommen, dass als Gesamtergebnis eine höhere Rate der *false positives* entsteht im Vergleich zur Detektion anhand einzelner Bilder. Eine weiter Einschränkung entsteht dadurch, dass die Personen im Raum ihre Position während des betrachteten Zeitraumes nicht wesentlich ändern dürfen. Das bedeutet, dass entweder sehr kurze Zeiträume oder nur statische Szenen, wie das z. B. in dem Fall eines Seminars ist, betrachtet werden können. Für die hier untersuchte Situation bedeutet das, dass nur die Zuschauer betrachtet werden, aber nicht der Sprecher. Das kommt durch die Tatsache, dass er sich ständig bewegt, daher wird auch nicht erwartet, dass er durch Akkumulation von Bildern überhaupt detektiert wird. Sollte das trotzdem passieren, so werden diese Detektionen nicht als Ergebnis gezählt.

Beim Akkumulieren von mehreren Bildern sind, wie im Kapitel 3.2.2 ausführlich erläutert wurde, drei wichtige Inputgrößen zu beachten:

1. Die Frequenz, mit der Bilder aus einer Bildfolge zur Auswertung entnommen werden.
2. Die Anzahl der Bilder, die aus der Bildfolge entnommen werden.
3. Die T - Variable, die besagt, auf wieviel Prozent der Bilder ein Objekt erkannt werden muss, damit es als ein Gesicht vom Programm wahrgenommen wird.

Da eine Untersuchung der Einflüsse aller drei signifikanten Parameter zu einem erheblichen Auswertungsaufwand führen würde, wurde beschlossen, die Anzahl der Bilder auf 20 festzulegen, da dieser Wert gut genug zum Durchführen von weiteren Experimenten mit Akkumulation zu sein scheint. Als erstes wird dann der Einfluss der Frequenz auf die Detektionsrate untersucht. Für diesen Zweck wird die T-Variable auf eins gesetzt. Das bedeutet, dass jedes detektierte Objekt vom Programm als ein Gesicht akzeptiert wird. Bei der Auswertung der Ergebnisse wird manuell verglichen, wieviele der von der Kamera erfassten Gesichter erkannt wurden. In diesem Stadium werden die *false positives* nicht ausgewertet. Die folgende Tabelle 6 fasst die Ergebnisse zusammen:

Frequenz	Detektionsrate
1	65%
15	84%
50	83%
100	89%

Tabelle 6: Einfluss der Frequenz auf die Detektionsrate

Eine Frequenz von 1 bedeutet, dass 20 unmittelbar nacheinanderstehende Bilder aus einer Reihenfolge mit mehreren Bildern als Eingabe für das Detektionsprogramm dienen, und eine Frequenz von 100, dass jedes hundertste Bild aus dieser Reihenfolge genommen wird. Das bedeutet, dass bei einer Frequenz von 100 einen Zeitraum, der 100 Fach länger ist als jener bei einer Frequenz von 1, betrachtet wird. Dies macht möglich, dass Gesichter, die bei einem kurzen Zeitraum möglicherweise verdeckt, schlecht beleuchtet sind oder sich in einer ungünstigen Position befinden, auch erkannt werden. Diese Tatsache wird auch von den Untersuchungsergebnissen bestätigt – die höchste Frequenz liefert die höchste Erkennungsrate.

Bei der Betrachtung dieser Ergebnisse muss beachtet werden, dass die Festlegung der T– Variable auf dem Wert 1 zu einer sehr hohen Rate der *false positives* führt. Die Auswirkung der Frequenz auf die Rate der *false positives* wurde nicht extra untersucht. Durch die sich in einem längeren Zeitraum stärker ändernden Umgebungsbedingungen wird die Möglichkeit, Objekte an der gleichen Position falsch als Gesichter zu erkennen, geringer. Daher ist zu erwarten, dass die höchste Frequenz die niedrigste Rate der *false positives* liefert.

Als nächstes wird bei der bereits ermittelten Frequenz von 100 der Einfluss der T– Variable auf die Detektionsergebnisse überprüft. Die Detektionsrate bzw. Rate der *false positives* werden als Anzahl der richtig erkannten bzw. falsch erkannten Gesichter durch die Anzahl der tatsächlich im Bild befindenden Gesichter definiert.

T-Variable	Mind. enthalten in % Bilder	Detektionssrate	Rate der *false positives*
1	1	0,89	2,93
20	16	0,68	1,46
40	31	0,67	0,78
60	47	0,50	0,42
80	63	0,34	0,38
100	79	0,26	0,25
130	100	0	0

Tabelle 7: Einfluss der T- Variable auf die Erkennungsergebnisse

Den in der Tabelle 7 dargestellten Ergebnissen ist zu entnehmen, dass sowohl die Rate der *false positives* als auch die Detektionsrate mit steigendem Wert der T– Variable abnehmen. Wie auch auf der in Abb.17 dargestellten ROC – Kurve zu erkennen ist, wird ein gutes Verhältnis zwischen Detektionsrate und Rate der *false positives* bei einem Wert der T– Variable von ca. 60 erreicht.

Generell fällt aber als ein Grund für die insgesamt sehr hohe Rate der *false positives* die Tatsache auf, dass Objekte wie zum Beispiel Steckdosen und Hinterkopfbereiche sehr zuverlässig als Gesichter detektiert werden. Wahrscheinlich weisen diese gesichtsähnliche Merkmale auf, die vom Detektor einstudiert worden sind. Daher müssen anschließend zum Detektionsprozess weitere Filter eingebaut werden, die dieses Problem eliminieren können.

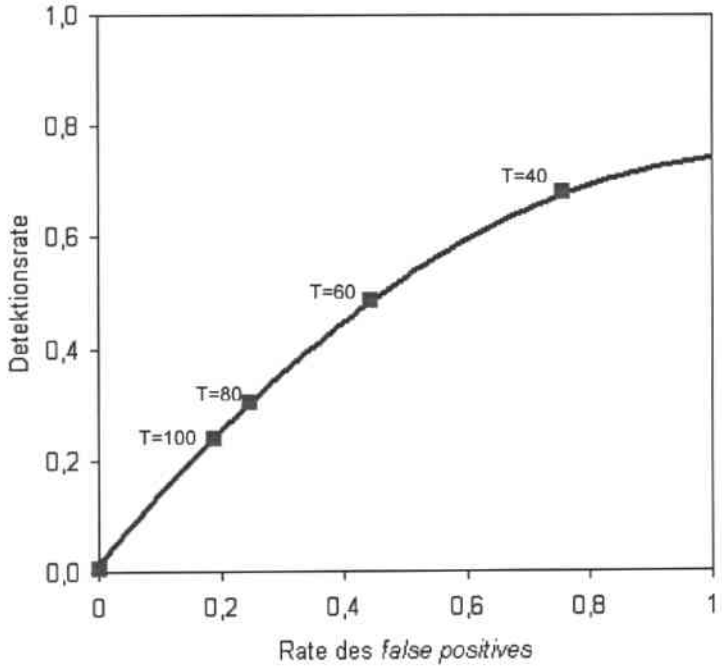


Abbildung 17: ROC - Kurve

4.3. Triangulation

Bereits im Kapitel 3.2.1. wurde die Anordnung im sogenannten *smart room* ausführlich erläutert. Durch die Tatsache, dass der Raum von insgesamt vier Kameras erfasst wird, kommt es dazu, dass die Gesichter von Teilnehmern auf zwei Kameras gleichzeitig so erscheinen, dass sie vom Detektor auch erkannt werden können. Ein doppeltes Zählen dieser Gesichter bei der Auswertung der Detektionsergebnisse wird im Programm durch Triangulation vermieden. Das Prinzip der Triangulation wurde bereits im Kapitel 2.2. vorgestellt.

Abbildung 18: Output-Daten von den vier Kameras mit allen detektierten Gesichtern

Die verbessernde Auswirkung der Triangulation auf das Endergebnis ist schnell und einfach anhand des auf Abb. 18 dargestellten Beispiels erklärt. Insgesamt wurden von den vier Kameras 24 Gesichter detektiert. Sechs von ihnen sind aber von zwei unterschiedlichen Perspektiven gleichzeitig erkannt, was auch vom Programm durch die Triangulationmethode erkannt wurde. Somit wird als Endergebnis die korrigierte Personenanzahl von 18 geliefert. Tatsächlich befinden sich 19 Personen im Raum, leider wurden aber auch ein paar falsche Objekte vom Detektor aufgenommen.

5. Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurde der bereits von Viola und Jones entwickelte Objektdetektionsalgorithmus vorgestellt. Weiterhin wurde die Anwendung dieses Algorithmus im Bereich der Gesichtsdetektion geprüft, wobei speziell auf seine Effektivität und Robustheit geachtet wurde. Der Objektdetektionsalgorithmus stellt einen Detektor von visuellen Objekten dar, der Bilder extrem schnell verarbeiten kann beim Erzielen einer sehr hohen Detektionsrate. Drei Schlüsselpunkte sind als besonders wichtig zu betrachten: Der erste ist die Einführung einer neuen Bilddarstellung, die Integralbild genannt wird und eine sehr schnelle Auswertung der Bilder mittels der Merkmale erlaubt. Der zweite ist ein lernender Algorithmus, basierend auf AdaBoost, der eine kleine Anzahl von kritischen Visualeigenschaften vorwählt und extrem leistungsfähige Klassifikatoren erbringt. Der dritte Schlüsselpunkt ist eine Methode zur Kombinierung von Klassifikatoren in einer "Kaskade", die erlaubt, Hintergrundregionen des Bildes schnell wegzuwerfen, um mehr Rechenressourcen für die versprechenderen und objektähnlichen Regionen freizustellen.

Als besonders hilfreich erwies sich bei der Arbeit das Benutzen von Programmen und Funktionen, die bereits auf diesem Algorithmus basieren und somit als Werkzeuge für die Realisierung der gestellten Aufgabe dienen konnten. Ein Satz von solchen Programmen und Funktionen bietet die weitverbreitete Bibliothek OpenCV. Mit Hilfe der in OpenCV zur Verfügung stehenden Programme „Createsamples" und „Haartraining" wurden zwei Kaskaden für Gesichtsdetektion trainiert, jeweils eine für frontale Gesichter und eine für Profile.

Als Input für die Programme, die zum Trainieren des Gesichtsdetektionsalgorithmus eingesetzt werden, wurden möglichst umfangreichere Datensätze mit Bildern benötigt. Im Rahmen dieser Arbeit wurden Daten aus ISL Seminaren vom Jahr 2003 benutzt. Dabei handelt sich um Filme von sieben unterschiedlichen Seminaren, die alle im sogenannten *smart room* durchgeführt worden sind. Für das Erstellen der Kaskaden wurden drei Sätze von Bildern gebraucht. Zwei Sätze, die die sogenannten positiven Samples beinhalten (ein Satz für frontale Gesichter und ein für Profile), und ein Satz mit den sogenannten negativen Samples.

Die eigentliche Gesichtsdetektion innerhalb eines Inputbildes wird durch die OpenCV Funktion *cvHaarDetectObject* durchgeführt. Diese Funktion wurde als Basis für ein Programm benutzt, das eine robuste und effektive Gesichtsdetektion, ausgehend von einer Bildfolge, durchführen kann. Hauptaufgabe dieses Programms ist es die Anzahl der sich in einem Raum befindenden Personen anhand einer Detektion ihrer Gesichter zu bestimmen. Die

Verwirklichung dieser Aufgabe wurde mittels Akkumulation von mehreren Bildern und Triangulation gewährleistet.

Anschließend wurden die einzelnen Komponenten des Detektionsprogramms getestet, um die Auswirkung verschiedener Parameter auf die Performance des Detektors zu untersuchen. Dabei war ständig festzustellen, dass ein Kompromiss zwischen Detektionsrate und Rate der *false positives* zu suchen ist. War das Erreichen einer relativ hohen Detektionsrate mit den in der Arbeit untersuchten Maßnahmen möglich gewesen, so wurde das Problem mit den zu hohen Raten der *fase positives* nicht vollständig gelöst und bietet somit Platz für weitere an diese Arbeit anknüpfende Untersuchungen.

Es haben sich drei Hauptmöglichkeiten gezeigt, wie die Zuverlässigkeit des Gesichtdetektors weiter erhöht werden kann. Zum einen müssen nach dem Moto von Bob Mercer „There is no data like more data!" weitere Trainingsdaten aufbereitet werden und im Training des Algorithmus eingesetzt werden. Zum zweiten kann durch Erweiterung des Programms mit einer Hautfarbendetektion die Rate der *false positive* reduziert werden, indem ein detektiertes Objekt erst nach dem Vorweisen dieses Merkmales als Gesicht vom Programm akzeptiert wird. So werden zum Beispiel die sehr oft als Gesichter detektierten Steckdosen zuverlässig aus dem Endergebnis ausgeschlossen werden. Und zum dritten ist während der Auswertung aufgefallen, dass beträchtlich viele Hinterköpfe vom Detektor fälschlicherweise als Gesichter detektiert wurden. Daher liegt die Vermutung nahe, dass der Detektor nicht nur die typischen Gesichtsmerkmale, sondern auch die Kopfform erkennt. Daher bietet sich die Möglichkeit an, durch Erweiterung des Programms mit einer Kaskade, die auf das Detektieren von Hinterkopfbereiche speziell trainiert wird, den Detektionsprozess auf den Menschenkopf zu erweitern und damit die Robustheit und Effizienz des Detektors zu verbessern. In diesem Fall wird die Rolle der anknüpfenden Triangulation noch größer, da jeder Kopf auf mindestens zwei Kameras erscheinen muss. Daher werden Objekte, die keine Korrespondenzpartner in einer der anderen Perspektiven finden, auch nicht als Köpfe vom Programm detektiert. Diese Vorgehensweise wird dann zu einer wesentlichen Verminderung der Rate der *false positives* führen.

6. Literaturverzeichnis

[1] P. Viola and M. Jones, *Robust Real-time Object Detection*, Second International Workshop On Statistical And Computational Theories Of Vision – Modeling, Learning, Computing And Sampling, Vancouver, Canada, July 13, 2001.

[2] M. Jones and P. Viola , *Fast Multi - view Face Detection*, Mitsubishi Electric Research Labs, demonstration at the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), June 2003.

[3] M. Yang, D. Kriegman, N. Ahuja, *Detecting Faces in Images: A Survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence vol.24 no.1: 24–58, January 2002.

[4] P. Campadelli, F. Cusmai, R. Lanzarotti, *A Color-Based Method For Face Detection*, Dipartimento di Scienze dell'Informazione Universit_a degli Studi di Milano, Via Comelico, 39/41 20135 Milano, Italy.

[5] X. Song and R. Nevatia, *Combined Face-body Tracking in Indoor Environment*, University of Southern California, Institute for Robotics and Intelligent Systems, Los Angeles, CA 90089-0273, 2004.

[6] Y. Freund and R. E. Schapire, *A decision-theoretic generalization of on-line learning and an application to boosting*, Computational Learning Theory: Eurocolt '95, pages 23–37. Springer-Verlag, 1995.

[7] D. Focken, *Vision-based 3-D Tracking of People in a Smart Room Environment*, Diplomarbeit, Interactive Systems Laboratories Universität Karlsruhe (TH), Germany, 2002.

[8] Y. Amit, D. Geman, and K. Wilder, *Joint induction of shape features and tree classifiers*, 1997.

[9] H. Schneiderman, *Feature-Centric Evaluation for Efficient Cascaded Object Detection*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04), 2004.

[10] C. Papageorgiou, M. Oren, and T. Poggio, *A general framework for object detection*, International Conference on Computer Vision, 1998.

[11] J. Quinlan, *Induction of decision trees*, Machine Learning, 1:81–106, 1986.

[12] D. Roth, M. Yang, and N. Ahuja, *A snowbased face detector*, Neural Information Processing 12, 2000.

[13] H. Rowley, S. Baluja, and T. Kanade, *Neural network-based face detection*, IEEE Patt. Anal. Mach. Intell., volume 20, pages 22–38, 1998.

[14] H. Schneiderman and T. Kanade, *A statistical method for 3D object detection applied to faces and cars*, International Conference on Computer Vision, 2000.

[15] Patrice Y. Simard, Lon Bottou, Patrick Haffner, and Yann Le Cun, *Boxlets: a fast convolution algorithm for signal processing and neural networks*, In M. Kearns, S. Solla, and D. Cohn, editors, *Advances in Neural Information Processing Systems*, volume 11, pages 571–577, 1999.

[16] K. Sung and T. Poggio, *Example-based learning for view-based face detection*, IEEE Patt. Anal. Mach. Intell., volume 20, pages 39–51, 1998.

[17] Q. Cai and J. Aggarwal, *Tracking human motion using multiple cameras*, Proceedings of International Conference on Pattern Recognition, Vienna, Austria, 1996.

[18] I. Mikic, S. Santini, and R. Jain, *Tracking objects in 3-d using multiple camera views*, Technical report, Computer Vision and Robotics Research Laboratory, University of California at San Diego, San Diego, USA, 2000.