Grosser Beleg

Kamerabasierte Personenortung

Michael Zitzmann

Technische Universität Dresden
Fakultät Informatik

10. November 2008

Inhaltsverzeichnis

1 Einleitung

Das automatisierte Verfolgen von Personen in Kamerabildern ist ein häufig auftretendes Problem in Anwendungen wie Videokonferenzen, Videoüberwachung und der Interaktion mit multimedialen Systemen.

Um eine Person erfolgreich über die Zeit hinweg verfolgen zu können, muss sie zunächst einmal im Kamerabild gefunden werden. Im Wesentlichen gibt es zwei verschiedene Ansätze, um dies zu erreichen: Die Segmentierung der vollständigen Körper von Personen und das Klassifizieren der Gesichtsregionen. In vielen Anwendungsfällen interessiert jedoch insbesondere die Position des Kopfes. Zum einen befinden sich im Kopfbereich mit den Augen und Ohren zentrale Sinnesorgane, die es dem Menschen erst ermöglichen, sich im Raum zu orientieren. Diese Tatsache macht Gesichtstracking für Anwendungsbereiche der Mensch-Maschine-Interaktion relevant, da Signale wie z.B. Bild und Ton an die Position des Gesichts des Betrachters angepasst werden können und dieser durch Veränderung seines Standpunkts aktiv auf das Verhalten eines Systems einwirken kann. Zum anderen ist das Auffinden eines Gesichts der wesentliche erste Schritt, um in sicherheitsrelevanten Anwendungen die Identifizierung der zugehörigen Person vornehmen zu können. Zusätzlich findet die Gesichtsverfolgung auch durch die stetig steigenden Rechenkapazitäten Anwendung in der Analyse von Videosequenzen, z.B. *video structuring*, *video coding* und *content-based image retrieval*.

Eine vom Gesichtsdetektor als Gesicht klassifizierte Pixelregion wird schließlich von einem Filteralgorithmus verfolgt, um zum einen Fehler in der Positionsbestimmung zu glätten, und zum anderen die Robustheit des Systems gegen die temporäre Abwesenheit von Messungen zu erhöhen. Dadurch werden abrupte Sprünge der verfolgten Position verhindert. Zwei Verfahren, die am häufigsten zur Objektverfolgung eingesetzt werden, stellen das Kalmanfilter und das Partikelfilter dar.

Zunächst wird die konkrete Aufgabenstellung vorgestellt. In den weiteren Kapiteln wird auf die Gesichtsdetektion als zentraler Bestandteil des Trackingsystems eingegangen. Darauf folgend wird die Funktionsweise des Partikelfilters beschrieben und im Anschluss daran die Modellierung, in der wichtige Entscheidungen getroffen wurden, welche die Echtzeitfähigkeit des Trackers beeinflussen. Abschließend werden einzelne Aspekte des Protoypen analysiert, die wesentlich für den Erfolg des Systems sind.

1.1 Problemstellung

Die konkrete Problemstellung der vorliegenden Arbeit ergab sich im Rahmen einer Arbeit am Institut für Akustik und Sprachkommunikation der Fakultät der Elektrotechnik und Informationstechnik. In dieser sollte die Tatsache untersucht werden, dass der räumliche Klangeindruck in einem multimedialen stereophonen Raum nur in einem sehr begrenzten Bereich mit dem Bildeindruck übereinstimmt. Tritt man aus dem so genannten „*sweet spot*“ heraus, so verschiebt sich die wahrgenommene Position der Schallquelle relativ zum Bild. Um dem entgegen zu wirken, sollte der *sweet spot* durch Manipulation der Audiosignale der einzelnen Lautsprecher mit der sich im Raum bewegenden Person mitgeführt werden. Die Positionsbestimmung durch am Kopf der Person befestigte Sensoren wurde prinzipiell ausgeschlossen, um eine natürliche Interaktion zu ermöglichen. Statt dessen wurde eine kamerabasierte Lösung bevorzugt, die es erlaubt, ohne zusätzliche Hilfsmittel unmittelbar nach dem Eintreten in den Raum als Person erkannt und verfolgt zu werden, und so mit dem System zu interagieren.

Nach einer ersten Analyse wurden zunächst folgende Anforderungen festgelegt: Um die Echtzeitfähigkeit des Gesamtsystems gewährleisten zu können, wurde eine maximale Bearbeitungszeit pro Zeitschritt von 100 ms gefordert. Unter natürlichen Bedingungen sollten auch vor inhomogenen Hintergründen Gesichter zuverlässig verfolgt werden können mit einer möglichst großen Toleranz gegen unterschiedliche Beleuchtungssituationen und Rotationswinkel in der Bildebene (seitliche Neigung des Kopfes) bzw. aus der Bildebene heraus (Drehung der Person um die eigene Längsachse). Dabei sollte die Ungenauigkeit in der Positionsbestimmung maximal ± 10 cm betragen.

Die ermittelte Position wird nach jedem Zeitschritt an das System, das den Raumklang steuert, weitergegeben. Zudem wird ein Qualitätsmerkmal übergeben, das Auskunft darüber erteilt, wie verlässlich die Berechnung der übermittelten Position tatsächlich ist.

In Grafik 1 ist der Aufbau der Szene in dargestellt.

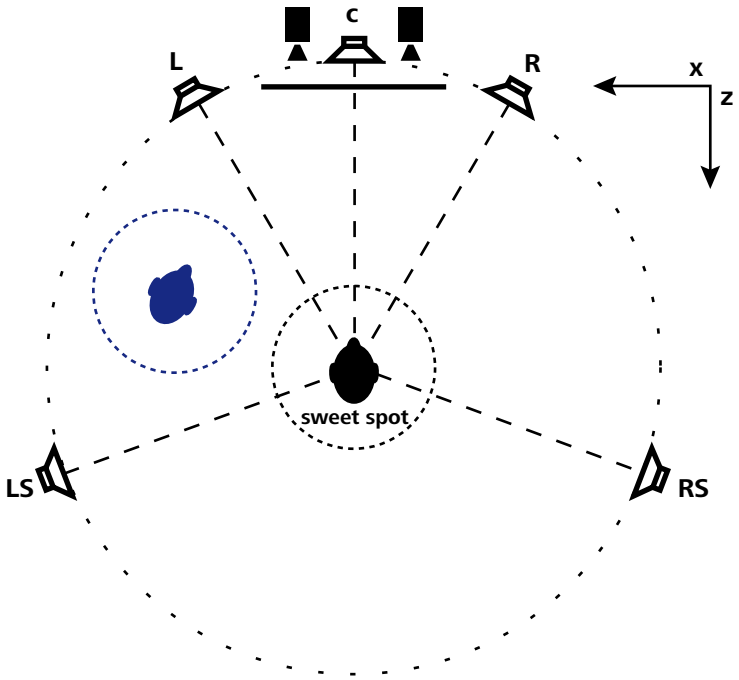


Abbildung 1: Diese Skizze verdeutlicht das geplante Einsatzszenario des Objektverfolgungssystems in einem multimedialen Raum, in dem der *sweet spot* des Lautsprechersystems mit der Position des Rezipienten geführt werden soll. Der optimale Raumklang ist nur dann zu hören, wenn die Schallwellen die gleiche Strecke bis zur Person zurücklegen müssen, in der Skizze ist dies die schwarz dargestellte Position. Vom der blauen Standpunkt aus wären akustische und visuelle Wahrnehmungen einer Schallquelle nicht deckungsgleich, wodurch die Präsentation für den Betrachter an Glaubwürdigkeit verliert. Mithilfe einer Stereokamera, deren Bildebene parallel zur Projektionsleinwand ist, werden auf Basis eines Gesichtsdetektors und eines Partikelfilters die Bewegungen einer Person im Raum verfolgt. Die ermittelten Positionen werden an das System übergeben, das den Raumklang steuert.

2 Detektion von Gesichtern in Bildern

Viele unterschiedliche Motivationen für das maschinelle Suchen von Gesichtern in Kamerabildern lassen sich ausmachen. Einfache Algorithmen zur Gesichtsdetektion finden sich inzwischen auch in eingebetteten Systemen, wie zum Beispiel Digitalkameras, um Belichtung und Schärfe an die sich im Bild befindenden Gesichter anzupassen. In den letzten Jahren wurden echtzeitfähige Algorithmen entwickelt, was für viele Anwendungen eine wichtige Voraussetzung ist, wie z.B. für multimediale Systeme und Installationen sowie für die Videoüberwachung.

Prinzipiell sollte ein idealer Gesichtsdetektor in einem gegebenen Bild alle Gesichter lokalisieren können, unabhängig von deren Position, Größe und Rotation. Zusätzlich können Beleuchtung, Gesichtsausdruck, Bärte, Brillen und Hüte den Algorithmen Schwierigkeiten bei der Erkennung von Gesichtern bereiten.

2.1 Überblick über Techniken zur Gesichtsdetektion

Zahlreiche Methoden zur Lokalisierung von Gesichtern in Bildern wurden entwickelt, sie reichen von Anwendungen einfacher Techniken der Bildverarbeitung bis hin zu komplexen statistischen Methoden der Mustererkennung und des maschinellen Lernens. Diese Techniken unterscheiden sich sowohl hinsichtlich der Erfolgsquote auch bei idealen Bedingungen – wenn die Person frontal und aufrecht bei guter Ausleuchtung des Gesichts in die Kamera blickt – als auch in der Toleranz gegen nicht optimale Bedingungen sowie des Rechenaufwands. Um sich einen Überblick über die verschiedenen existierenden Techniken zu verschaffen, ist eine Kategorisierung unerlässlich [HL01, JL05, YKA02]. Eine besonders detaillierte Kategorisierung nehmen Hjelmås und Low in [HL01] vor (siehe Abb. 2). Grundsätzlich wird unterschieden in merkmals- und bildbasierte Ansätze. Merkmalsbasierte Techniken nutzen explizites Wissen über das Aussehen von Gesichtern, wie z.B. Informationen über Hautfarbe und die Geometrie von Gesichtern. Bildbasierte Techniken untersuchen dagegen Bilder mit Hilfe von implizitem Wissen über Gesichter, meist unter Verwendung von Lernalgorithmen.

2.1.1 Merkmalsbasierte Methoden

Der Merkmalsbasierte Ansatz lässt sich weiter in drei Kategorien gliedern: Low-Level-Analyse, Merkmalsanalyse und Active Shape Models. Low-Level-Ansätze verwenden Kanten-,

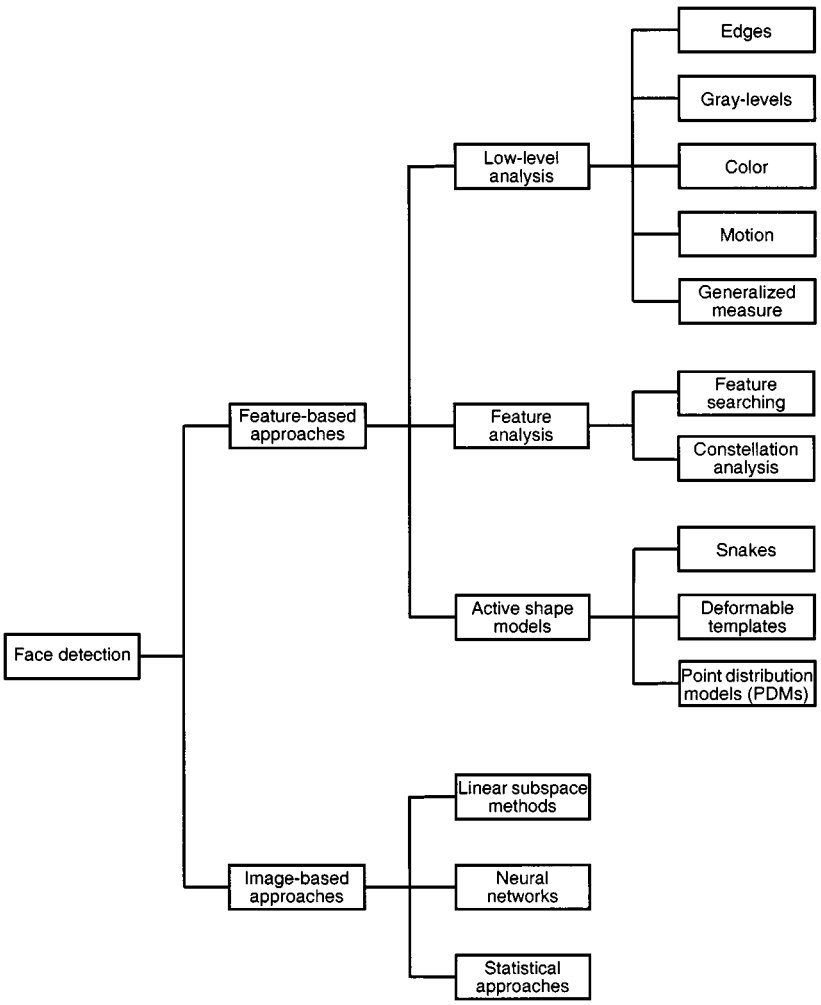


Abbildung 2: Kategorisierung von Techniken für das Detektieren von Gesichtern in Bildern nach HJELMÅS [HL01]

Grauwert-, Farb- und Bewegungsinformationen, um festzustellen, ob ein Bildbereich ein Gesicht enthält. Allerdings liefern diese Techniken keine stabilen Ergebnisse. Beispielsweise erkennt ein Detektor, der auf einer Hautfarben-Analyse basiert, auch Hintergrundobjekte als Gesichter, wenn deren Farbe in das gesuchte Spektrum fällt. Dieses und andere Probleme der Low-Level-Techniken können durch eine genauere Analyse behoben werden, indem die Positionen wichtiger Gesichtsmerkmale bestimmt werden, wie beispielsweise die Lage der Augen oder die Hauptachse des Gesichts. Merkmalsanalysen kann man wiederum in zwei Herangehensweisen unterscheiden. *Merkmalssuche* basiert auf Heuristiken der relativen Position von Gesichtsmerkmalen. Daraus ergibt sich auch die Problematik dieser Techniken: Ihre Flexibilität, sich an neue Situationen, wie variable Posen anzupassen, ist gering. Um dieses Problem zu umgehen, arbeitet die *Konstellationsanalyse* mit flexibleren Gesichtsmodellen, wie zweidimensionalen Gesichtsschablonen oder variablen dreidimensionalen Netzen, für deren Erstellung statistische Analyseverfahren angewandt werden. *Active Shape Models* beschreiben im Gegensatz zu den bisher genannten Methoden die tatsächliche physische Beschaffenheit der Merkmale. Zu ihnen zählen *Snakes*, *deformable templates* und *point distribution models*.

2.1.2 Bildbasierte Methoden

Da sich das explizite Modellieren von Gesichtsmerkmalen als wenig flexibel und dadurch störanfällig erwiesen hat, wurden in den letzten Jahren verstärkt Ansätze auf Basis der Mustererkennung entwickelt. Der Einsatz von Lernalgorithmen macht die explizite Beschreibung von Gesichtsmerkmalen überflüssig, wodurch es vermieden wird, durch ungenaue Beschreibungen fehlerhafte Modelle zu erstellen. Stattdessen werden mit Hilfe von Datensätzen aus positiven und negativen Beispielen Klassifikatoren angelernt. Grundlegend dafür, dass ein Klassifikator möglichst alle positiven Beispiele erkennt und alle negativen Beispiele ablehnt sind repräsentative Trainigsdaten [PCD08].

Weiterhin lassen sich die bildbasierten Methoden in *Linear Subspace Models*, *Neuronale Netze* und *statistische Modelle* untergliedern. Linear Subspace Models umfassen die *Principal Component Analysis*, die *Linear Discriminant Analysis* und die *Faktorenanalyse*. Unter den statistischen Modellen werden sonstige statistische Ansätze zusammengefasst wie z.B. *Support Vector Machines* und *Boosting*.

Derzeit sind Gesichtsklassifikatoren auf Basis von AdaBoost hinsichtlich der Genauigkeit und der Geschwindigkeit führend [JL05]. Im folgenden Kapitel werden die Grundlagen und die Implementierung des in dieser Arbeit eingesetzten Gesichtsdetektors beschrieben.

Boosting basiert auf der Idee, aus einfachen, schwachen Basisklassifikatoren – oft auch *weak learners* genannt – durch geschickte Kombination mächtige Klassifikatoren zu bilden. Die Basisklassifikatoren können dabei sehr einfach aufgebaut sein, und müssen einzeln betrachtet lediglich besser als eine zufällige Entscheidung sein. Eine optimale Kombination unterschiedlicher Basisklassifikatoren kann dadurch zu einem genaueren Klassifikator führen, als es durch das Anlernen eines einzelnen Klassifikators möglich ist [Alp08].

2.2 AdaBoost nach Freund und Schapire

Freund und Schapire entwickelten 1996 in [FS95] einen adaptiven Boosting-Algorithmus namens *AdaBoost*. Durch AdaBoost werden die Basisklassifikatoren sequentiell trainiert, wobei jeder Klassifikator in der Sequenz mit einer gewichteten Variante des ursprünglichen Datensatzes angelernt wird, und die Gewichtungen jedes Datenpunktes vom Erfolg der vorhergehenden Klassifikatoren abhängen. Falsch klassifizierte Daten erhalten dadurch beim Anlernen des folgenden Klassifikators ein stärkeres, richtig klassifizierte dagegen ein geringeres Gewicht. Schließlich werden die Vorhersagen der Basisklassifikatoren durch ein Gewichtungsverfahren *(weighted majority voting scheme)* kombiniert. Im Folgenden wird der Algorithmus nach Bishop [Bis06] im Hinblick auf den Anwendungsfall der Gesichtsdetektion als Zwei-Klassen-Klassifikationsproblem beschrieben.

Als Eingabe erhält der Algorithmus die bereits den Klassen zugeordneten N Trainingsdaten $(\mathrm{x}_1, y_1), \ldots, (\mathrm{x}_N, y_N)$, wobei x_n die Daten bezeichnen und $y_n \in \{ -1, 1 \}$ deren Zugehörigkeit zur jeweiligen Klasse. Ein Trainingsdatum x_n besteht beispielsweise aus den Pixelwerten eines zu klassifizierenden Bildausschnitts. Außerdem wird jedem Datenpunkt ein Gewicht $w_n \in \mathbb{R}^+$ zugeordnet. Zusätzlich ist eine Funktion erforderlich, die einen Basisklassifikator $h(\mathrm{x}_n) \rightarrow \{-1, 1\}$ anlernt. Dem Algorithmus steht eine Menge dieser schwachen Klassifikatoren zu Verfügung (siehe auch 2.3.4), aus der zu jedem Iterationsschritt derjenige gewählt wird, der den geringsten Fehler auf den Trainingsdaten generiert. Schließlich wird die Anzahl der Iterationen benötigt, mit denen der endgültige Klassifikator angelernt wird. Diese Anzahl entspricht der Zahl der Basisklassifikatoren, aus deren *Konvexkombination* sich der anzulernende Klassifikator zusammensetzt.

Algorithmus 1 AdaBoost

Eingabe: N markierte Trainingsdaten (x_n, y_n), Anzahl der Iterationen M, Menge von Basisklassifikatoren

1: $w_n^1 = \frac{1}{N}$

2: **for** $m = 1$ to M **do**

3: wähle Basisklassifikator $h_m(\mathrm{x})$ für den gilt: $\min \sum_{n=1}^{N} w_n^m \, \delta(h_m(\mathrm{x}_n) \neq y_n)$

4: $\epsilon_m = \frac{\sum_{n=1}^{N} w_n^m \, \delta(h_m(\mathrm{x}_n) \neq y_n)}{\sum_{n=1}^{N} w_n^m}$

5: $\alpha_m = ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right)$

6: $w_n^{m+1} = w_n^m \exp\{\, \alpha_m \delta(h_m(\mathrm{x}_n) \neq y_n) \,\}$

7: **end for**

8: $H_M(\mathrm{x}) = sign\left(\sum_{m=1}^{M} \alpha_m h_m(\mathrm{x}_n)\right)$

Zu Beginn des Algorithmus werden die Gewichte der Dateneinheiten auf $\frac{1}{N}$ initialisiert. Dann wird in einer Schleife über die Anzahl der Iterationen zunächst der beste Basisklassifikator $h_m(\mathrm{x}_n)$ ermittelt. Dies geschieht in Zeile 3 des Algorithmus 1 durch Minimierung der gewichteten Fehlerfunktion zu der Verteilung über dem gewichteten Datensatz. Dadurch wird derjenige Basisklassifikator gewählt, der die meisten Daten des Trainingssatzes richtig klassifiziert. Die Funktion $\delta(h_m(\mathrm{x}_n) \neq y_n)$ liefert hierbei 1, wenn $h_m(\mathrm{x}_n) \neq y_n$, anderenfalls 0. Anschließend wird der relative Fehler ϵ_m zu den gewichteten Daten ermittelt, aus dem sich α_m berechnet. Am Ende jeder Iteration werden die Gewichte der Daten neu berechnet, wobei falsch klassifizierte Daten ein stärkeres Gewicht zugewiesen bekommen, richtig klassifizierte dagegen ein geringeres. Der erste Klassifikator in der Sequenz wird mit gleich stark gewichteten Dateneinheiten trainiert. Durch die Neuberechnung der Gewichtungskoeffizienten nach jedem Iterationsschritt legt der nachfolgende Klassifikator stärkeres Gewicht auf falsch klassifizierte und geringeres Gewicht auf richtig klassifizierte Daten. Je öfter ein Datum falsch klassifiziert wird, desto stärker wird das Gewicht, das die

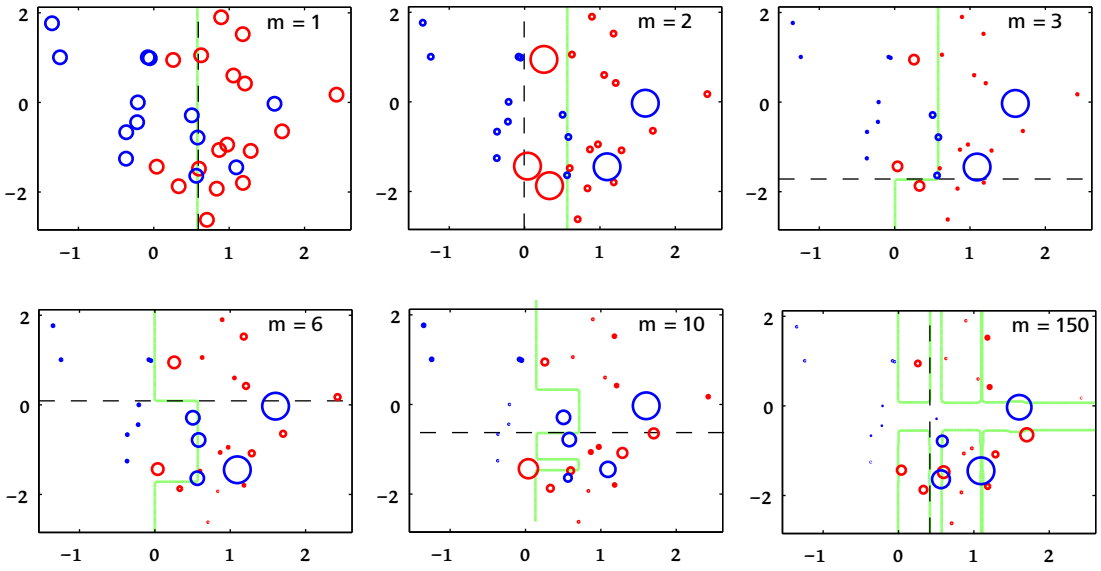


Abbildung 3: In der Grafik wird illustriert, wie AdaBoost *weak learners* zu einem mächtigen Klassifikator anlernt. Die einfachen Klassifikatoren bestehen aus Schwellwerten, welche sich auf die horizontale bzw. vertikale Achse beziehen. Jede Illustration zeigt den Zustand, nachdem m Basisklassifikatoren angelernt wurden. Die schwarz gestrichelte Linie stellt dabei den Schwellwert des *weak learners* dar und der kombinierte Klassifikator wird durch die grüne Linie veranschaulicht. Durch den Radius des Kreises wird das Gewicht jedes Datenpunktes angegeben, das bei falscher Klassifizierung im nächsten Schritt beim Anlernen des folgenden Klassifikators erhöht wird.

folgenden Klassifikatoren auf dieses Datum legen. In die Berechnung des endgültigen Klassifikators $H_M(x)$ gehen die einzelnen Basisklassifikatoren mit einer Gewichtung von α_m ein, wodurch genauere Klassifikatoren stärker berücksichtigt werden als ungenaue.

Die Leistung von AdaBoost hängt stark von den Trainingsdaten und dem schwachen Lernalgorithmus ab. Kritikpunkte an AdaBoost sind besonders die Anfälligkeit für Rauschen und Ausreißer [Alpo8]. Im Folgenden wird der Algorithmus vorgestellt, auf dem der in dieser Arbeit verwendete Gesichtsdetektor basiert.

2.3 AdaBoost mit Haar-Wavelet-ähnlichen Merkmalen

Viola und Jones stellen in [VJo1] einen Algorithmus zu Objekterkennung vor, der verschiedene Techniken nutzt, um eine hohe Verarbeitungsgeschwindigkeit zu erreichen. Die Grundlage bilden einfache, aus mehreren Rechtecken zusammengesetzte, Haar-Wavelet-

ähnliche Merkmale. Sie repräsentieren Intensitätsunterschiede zwischen benachbarten Pixelregionen und lassen sich durch *Integralbilder* effizient berechnen. Mittels AdaBoost werden diese Merkmale zu stärkeren Klassifikatoren trainiert. Die stärkeren Klassifikatoren werden wiederum in einer Kaskade mit steigender Komplexität zu einem mächtigen Klassifikator zusammengefügt. Unter bestimmten Voraussetzungen – wie die Beschränkung der Auflösung und *Downsampling* – ermöglicht die Kombination dieser Techniken echtzeitfähige Objekterkennung.

Unterschiedliche Beleuchtungssituationen erschweren die Objekterkennung erheblich, da der Trainingsdatensatz diese Informationen enthalten müsste. Dadurch wären wesentlich mehr Trainigsdaten erforderlich, als wenn man von einer natürlichen Beleuchtung ausgehen kann. Normalisiert man die Bilder, bevor man einen Klassifikator auf sie anwendet, so wirkt man diesem Problem entgegen. Durch die Repräsentation der Bilder mithilfe von Integralbildern kann eine Normalisierung mit geringem Aufwand zur Laufzeit geschehen.

2.3.1 Haar Wavelets

Durch AdaBoost wird ein starker Klassifikator angelernt, indem mehrere schwache Basisklassifikatoren in einer Linearkombination zusammengefügt werden. Diese *weak learners* müssen isoliert voneinander betrachtet nur besser als eine zufällige Entscheidung sein. In [VJ01] werden als Basisklassifikatoren Rechteck-Merkmale verwendet, die verglichen mit den natürlichen Formen, wie sie in Gesichtern vorkommen, zwar sehr grob sind, und nur einfache horizontale, vertikale und diagonale Strukturen wie Ecken und Balken repräsentieren können. Sie lassen sich aber mithilfe von *Integralbildern* sehr effizient berechnen und bilden eine gute Basis für Lernalgorithmen.

Die in [VJ01] vorgestellten Merkmale sind je aus zwei oder drei Rechtecken zusammengesetzt. Berechnet wird die Differenz der Summe der Pixel im schwarzen Rechteck von der Summe der Pixelwerte im weißen Rechteck. Im Fall des Drei-Rechteck-Merkmals wird die Summe der Pixelwerte in den weißen Rechtecken gebildet und davon die Summe der Pixelwerte im schwarzen Rechteck abgezogen, beim Vier-Rechteck-Merkmal wird die Differenz zwischen den diagonalen Rechteckpaaren berechnet.

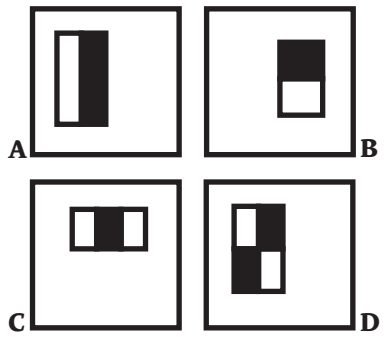


Abbildung 4: Durch Translation, Skalierung und Rotation relativ zu dem zu klassifizierenden Bildausschnitt ist der Merkmalsraum sehr groß. Diese Abbildung zeigt Beispiele für Zwei-Rechteck-Merkmale (A und B), sowie für Drei-Rechteck-Merkmale (C) und Vier-Rechteck-Merkmale (D). Sie berechnen sich aus der jeweiligen Differenz zwischen den Summmen der Pixelwerte der schwarzen und weißen Rechtecke.

2.3.2 Integralbilder

Um die Differenzen von den Rechtecken der Haar-Merkmale effizient zu berechnen, werden in [VJ01] *Integralbilder* verwendet. Ein Integralbild – in der Computergrafik als *Summed Area Table* bekannt – ist eine Repräsentation eines Bildes. An einer Position (x_i, y_i) wird die Summe aller Pixel abgespeichert, die oberhalb und links von ihr liegen:

$$II(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Dabei bezeichnet $I(x, y)$ das Originalbild und $II(x, y)$ das Integralbild. Durch das Vorberechnen des Integralbildes kann die Summe jedes beliebigen Rechtecks in einem Bild durch vier Array-Zugriffe berechnet werden, siehe Abbildung 5.

2.3.3 Normalisierung

Beleuchtungsschwankungen erschweren die Objekterkennung, da einem Klassifikator die benötigten Informationen während des Lernprozesses zur Verfügung stehen müssten. Dies würde aber einen um ein Vielfaches größeren Trainingsdatensatz erfordern. Stattdessen kann ein Klassifikator mit Beispielen, die eine begrenzte Varianz in der Beleuchtung aufweisen, trainiert werden. Die zu klassifizerenden Bilder werden dann in einem Vorverarbeitungsschritt normalisiert, um Schwankungen in der Beleuchtung zu verringern. Auf

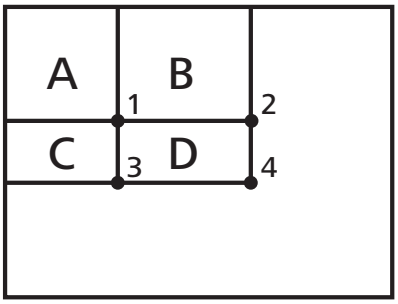


Abbildung 5: An der Position 4 ist die Summe aller Pixelwerte in A + B + C + D gespeichert. Die Summe der Werte in *D* lässt sich durch 4 + 1 - 2 - 3 berechnen.

Grundlage der Integralbilder kann eine Normalisierung der Bilder relativ einfach durchgeführt werden, indem die Abweichung der Intensität jedes Pixels vom Durchschnittswert durch die Standardabweichung dividiert wird:

$$I'(u,v) = \frac{I(u,v) - \mu}{c\sigma},\ c \in \mathbb{R}^+ \tag{2.1}$$

In [LKP03] wurde die Konstante c, die eine Streckung bzw. Stauchung des Verhältnisses bewirkt, auf 2 gesetzt. Der Mittelwert μ lässt sich direkt aus dem Integralbild $I(u,v)$ berechnen. Für die Berechnung der Standardabweichung σ benötigt man jedoch die Summe der quadrierten Pixelwerte $I^2(u,v)$. Zugunsten der besseren Lesbarkeit werden die Pixelwerte des Bildes im Folgenden als eindimensionales Array behandelt: $x := I(u,v)$. In einem Bild mit N Pixel berechnet sich dann die Varianz σ^2 wie folgt:

$$\sigma^2 = \frac{1}{N}\sum_i^N (x_i - \mu)^2 \tag{2.2}$$

$$= \frac{1}{N}\sum_i^N x_i^2 - \frac{2 * \mu}{N}\sum_i x_i + \mu^2 \tag{2.3}$$

$$= \frac{1}{N}\sum_i^N x_i^2 - \mu^2 \tag{2.4}$$

Für eine robustere und dennoch schnelle Objekterkennung wird daher für jedes Kamerabild sowohl ein einfaches als auch ein quadriertes Integralbild berechnet.

Abbildung 6: In der oberen Reihe sind die ersten beiden Basisklassifikatoren dargestellt, die von AdaBoost in [VJ01] gewählt wurden. Zur Veranschaulichung wurden sie über ein typisches Bild aus dem Trainingsdatensatz gelegt. Der erste Klassifikator misst den Unterschied der Intensitäten zwischen der Augen- und Wangenpartie. Im zweiten Fall wird der Kontrast zwischen Nasenrücken und Augenhöhlen gemessen.

2.3.4 Merkmalsselektion

Die Anzahl von möglichen Merkmalen, die durch Translation und Skalierung (und Rotation um 45º in [LKP03]) der Haar-Merkmale in einer untersuchten Maske entstehen, ist sehr groß. Um die Geschwindigkeit sowohl beim Anlernen des Klassifikators als auch bei der Anwendung zu erhöhen, müssen Redundanzen und irrelevante Informationen beseitigt werden. Dies geschieht mit Hilfe der Merkmalsselektion *(feature selection)* und wird ausführlich in Kapitel 9 in [Web02] beschrieben. Häufig findet dabei die *Principal Component Analysis* (PCA) Anwendung. VIOLA und JONES verwendeten jedoch wiederum AdaBoost, um die Dimension des Merkmalsraums zu reduzieren.

2.3.5 Weak Learners

Mit Hilfe eines markierten Tainingsdatensatzes mit positiven und negativen Proben bestimmt der *weak learner* das Rechteck-Merkmal, welches die positiven und negativen Beispiele am besten unterscheidet. Für jedes Merkmal bestimmt der *weak learner* die optimale Schwellwertfunktion, so dass die Anzahl der falsch klassifizierten Daten minimiert wird. Ein Basisklassifikator $h_j(x)$ besteht aus einem Merkmal f_j, einem Schwellwert θ_j und einer Parität p_j, welche die Richtung des Ungleichheitszeichens angibt. Der untersuchte Bildaus-

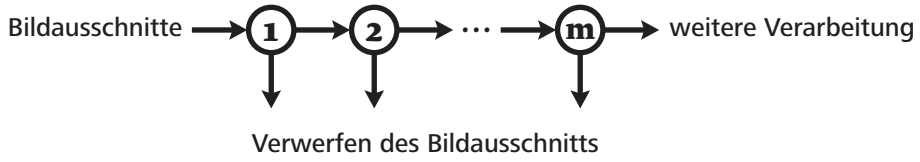


Abbildung 7: Degenerierter Entscheidungsbaum: Der Kaskade von Klassifikatoren werden die zu untersuchenden Bildausschnitte übergeben. Diese werden in der Kaskade nur dann weiter gereicht, wenn sie als Gesichtsregionen klassifiziert wurden, anderenfalls werden sie verworfen. Nur wenn ein Bildausschnitt von allen m Klassifikatoren der Kaskade positiv klassifiziert wurde, enthält er mit Wahrscheinlichkeit $p(x \mid z) = 0,98$ ein Gesicht (Gleichung 2.5).

schnitt wird mit x bezeichnet:

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

2.3.6 Kaskadierung einfacher Klassifikatoren

Um die Anwendung des Klassifikators zu beschleunigen, wurden weniger komplexe und dadurch effizienter berechenbare Klassifikatoren in einer Kaskade zu einem mächtigen Klassifikator zusammengefasst. Am Anfang der Kaskade stehen einfachere Klassifikatoren, die einen geringen Rechenaufwand erfordern, aber dennoch einen Großteil der Regionen, die kein Gesicht enthalten, aussondern. Positiv klassifizierte Beispiele werden in der Kaskade weiter gereicht an immer komplexere und rechenaufwändigere Klassifikatoren, die sukzessive die Zahl der fälschlicherweise als Gesicht klassifizierten Regionen (*false positives*) verringern, bei negativer Klassifikation wird der Durchlauf abgebrochen. Daher lässt sich die Klassifikator-Kaskade in Form eines degenerierten Entscheidungsbaums beschreiben (siehe Abbildung 7).

Der implizite Fehler des Detektors ist unmittelbar durch seine Struktur und den Vorgang des Anlernens gegeben. In der Implementierung von Lienhart et al. in OpenCV [LKP03] wurden die einzelnen Klassifikatoren jeder Stufe einer Kaskade so angelernt, dass eine Trefferquote auf den positiven Beispielen von 99,9% eingehalten wurde und die Rate der *false positives* bei maximal 50% lag. Bezeichnet $x \in$ {Gesicht vorhanden = 1, kein Gesicht vorhanden = 0} und $z \in$ {als Gesicht klassifiziert = 1, als Nicht-Gesicht klassifiziert = 0}, dann ergeben sich nach Durchlaufen der 20 Stufen des Klassifikators folgende Wahr-

scheinlichkeiten:

$$p(z = 1 \mid x = 1) = 0,999^{20} = 0,98 \tag{2.5}$$

$$p(z = 1 \mid x = 0) = 0,5^{20} = 9,6 \cdot 10^{-7} \tag{2.6}$$

$$p(z = 0 \mid x = 1) = 0,02 \tag{2.7}$$

$$p(z = 0 \mid x = 0) = 0,999999 \tag{2.8}$$

Am Anfang des Traversierens der Klassifikator-Kaskade ist die Wahrscheinlichkeit also noch relativ hoch, dass eine Region des Bildes fälschlicherweise als Gesicht klassifiziert wird, am Ende dagegen nahezu Null.

3 Objektverfolgung

Um ein Objekt in Videosequenzen verfolgen zu können, werden Informationen über seine Eigenschaften benötigt, wie sein Aussehen und Informationen über sein Verhalten. Prinzipiell sind Algorithmen zur Objektverfolgung nach einem *Vorhersage-Korrektur-Schema* aufgebaut: Zunächst wird aufgrund von Informationen über das Bewegungsverhalten des Objekts, dem *Bewegungsmodell*, eine Vorhersage über seinen Folgezustand getroffen. Aufgrund der anschließenden Messung des Objektzustands – die im Allgemeinen verrauscht ist – wird der Korrekturschritt durchgeführt und die Repräsentation des Objektzustands aktualisiert.

Im Folgenden wird das Prinzip der Zustandsschätzung anhand des *Bayes-Filters* erläutert, worauf eine kurze Analyse der Eigenschaften von *Kalmanfiltern* und *Partikelfiltern* als Anwendungen des Prinzips Bayesscher Zustandsschätzung folgt. Am Ende dieses Kapitels wird die Funktionsweise des Partikelfilters erläutert.

3.1 Grundlagen Bayesscher Zustandsschätzung

Um die Zustände solcher Prozesse zu schätzen, kommen meist Bayes-Filter zum Einsatz. Allgemein lässt sich der Zustand x_t eines Objekts zum Zeitpunkt t beschreiben durch $p(x_t \mid x_{0:t-1}, z_{1:t-1}, u_{1:t-1})$, wobei z_t die Messung bezeichnet und u_t die Kontrollaktion. Im betrachteten System finden keine aktiven Veränderungen des Zustands statt, daher wird u_t als Aktion „tue nichts“ behandelt und in allen weiteren Betrachtungen ignoriert. Außerdem wird davon ausgegangen, dass die einzelnen Messungen z_t voneinander stochastisch unabhängig sind:

$$p(z_t \mid x_{0:t}, z_{0:t-1}) = p(z_t \mid x_t). \tag{3.1}$$

Die Bewegung eines Objekts im Raum kann als *Markow-Prozess* betrachtet werden, so dass der aktuelle Zustand x_t nur vom unmittelbar vorhergehenden Zustand x_{t-1} abhängt und unabhängig von allen vorhergehenden Zuständen $x_{0:t-2}$ ist. Damit kann die Zustandsübergangswahrscheinlichkeitsdichte vom Zeitpunkt $t-1$ zum Zeitpunkt t ausgedrückt werden als

$$p(x_t \mid x_{0:t-1}) = p(x_t \mid x_{t-1}).$$

Aufgrund der Gültigkeit der Markow-Eigenschaft repräsentiert diese Wahrscheinlichkeitsdichte das gesamte Wissen über den Prozess. Daher wird sie häufig auch *Prozessmodell* ge-

nannt. Kennt man den Zustand x_t zum Zeitpunkt t und will eine Hypothese für die Messung z_t aufstellen, so kann man wiederum die Markow-Eigenschaft ausnutzen und das *Beobachtungsmodell* ergibt sich folgendermaßen:

$$p(z_t \mid x_t, z_{1:t-1}) = p(z_t \mid x_t). \tag{3.2}$$

Zu jedem Zeitpunkt $t \geq 0$ lässt sich die a-posteriori-Wahrscheinlichkeit $bel(x_t) = p(x_t \mid z_{1:t})$ durch das Theorem von Bayes beschreiben [TBF05]:

$$\begin{aligned} p(x_t \mid z_{1:t}) &= \frac{p(z_t \mid x_t, z_{1:t-1})\; p(x_t \mid z_{1:t-1})}{\int p(z_t \mid x_t, z_{1:t-1})\; p(x_t)\; dx_t} \\ &= \eta\; p(z_t \mid x_t, z_{1:t-1})\; p(x_t \mid z_{1:t-1}) \\ &= \eta\, p(z_t \mid x_t)\; p(x_t \mid z_{1:t-1}) \end{aligned} \tag{3.3}$$

Die Wahrscheinlichkeitsverteilung $\overline{bel}(x_t) = p(x_t \mid z_{1:t-1})$ lässt sich weiter umformen zu

$$p(x_t \mid z_{1:t-1}) = \int p(x_t \mid x_{t-1})\, p(x_{t-1} \mid z_{1:t-1})\, dx_{t-1}\,. \tag{3.4}$$

Gleichung 3.4 stellt den *Vorhersage-Schritt* dar: Die Wahrscheinlichkeitsverteilung, die den geschätzten Zustand repräsentiert und nur auf den Messungen bis zum vorhergehenden Zeitpunkt beruht, berechnet sich aus dem Produkt des Prozessmodells mit der Verteilung, die den vorhergehenden Zustand repräsentiert. Im anschließenden *Korrektur-Schritt* (Gleichung 3.3) wird die Wahrscheinlichkeitsverteilung des Zustands proportional zur Wahrscheinlichkeit einer gültigen Messung und der zuvor bereits berechneten Vorhersage aktualisiert.

Sowohl *Kalman Filter* als auch *Partikelfilter* basieren auf dem Prinzip der Bayesschen Zustandsschätzung. Jedoch stellt das Kalman-Filter strenge Anforderungen an das Prozess- und das Beobachtungsmodell, zudem müssen die Störgrößen normalverteilt sein. Unter diesen Voraussetzungen liefert das Kalman-Filter aber die optimale Lösung für das Problem der Zustandsschätzung. Im Gegensatz dazu stellen Partikelfilter kaum Anforderungen an die Modellierung, generieren aber nicht die optimale Zustandsschätzung und sind nichtdeterministisch. Dennoch können sie durchaus robustere Ergebnisse als Kalman-Filter liefern und ermöglichen die Objektverfolgung auch in schwierigen Situationen. Abrupte Bewe-

gungsänderungen, wie sie in realen Situationen auftreten können, lassen sich aufgrund ihrer Nichtlinearität nicht durch einfache Gaußverteilungen modellieren [TBF05]. Für Partikelfilter stellt sich dies durch die nichtparametrische Modellierung der Wahrscheinlichkeitsdichteverteilung über dem Zustand weniger problematisch dar [FHL+03].

Beim Kalmanfilter stellt sich das Problem der Initialisierung bzw. der Reinitialisierung von Gesichtern, im Fall, dass ein verfolgtes Gesicht verloren wurde, oder die Person das Blickfeld der Kamera verlassen hat. Dies gestaltet sich beim Partikelfilter einfacher. Aus diesen Gründen fiel die Wahl bei der Implementierung des Systems auf das Partikelfilter. Im folgenden Kapitel wird zunächst der grundlegende Algorithmus vorgestellt und anschließend auf dessen Eigenschaften eingegangen.

3.2 Partikelfilter

Das Partikelfilter – auch bekannt unter den Namen *condensation filter, bootstrap filter, survival of the fittest* – ist eine sequentielle *Monte-Carlo-Methode* zur Zustandsschätzung. Zu jedem Zeitpunkt t wird die a-posteriori-Wahrscheinlichkeitsverteilung $p(x_t \mid z_{1:t})$ durch eine Menge X_t von N gewichteten Partikeln x_t^n approximiert, mit $1 \leq n \leq N$. Aufgrund ihrer nichtparametrischen Natur kann sie beliebige Wahrscheinlichkeitsverteilungen repräsentieren, insbesondere solche mit mehreren Maxima. Dies ermöglicht einem Partikelfilter, mehrere Hypothesen eines Zustands gleichzeitig zu betrachten. Anhand der Beschreibung des Algorithmus in [TBF05] wird nun die Funktionsweise des Partikelfilters vorgestellt. Anschließend wird kurz auf das *resampling* eingegangen, *low variance sampling* als eine effiziente Variante des *resampling* beschrieben und schließlich eine mathematische Herleitung des Algorithmus gegeben.

Zu jedem Zeitpunkt t wird eine Hypothese x_t^n aus den Informationen über die Partikel X_{t-1} generiert, die im Idealfall proportional zur a-posteriori-Wahrscheinlichkeit ist:

$$x_t^n \propto p(x_t \mid z_{1:t}) . \tag{3.5}$$

Als Eingaben werden dem Algorithmus (2) die Partikelmenge X_{t-1} und die aktuelle Messung z_t übergeben. Von Zeile 2 bis 6 wird in einer Schleife über alle Partikel x_{t-1}^n eine temporäre Partikelmenge X_t' erzeugt, welche die a-priori-Wahrscheinlichkeitsverteilung $\overline{bel}(x_t) = p(x_t \mid z_{1:t-1})$ repräsentiert. In Zeile 3 wird für jedes Partikel eine neue Hypothese generiert, indem das Prozessmodell auf dieses angewandt wird. Im Anschluss wird

Algorithmus 2 ParticleFilter

Eingabe: Menge der Partikel mit ihren Gewichten X_{t-1}, Messung z_t
Ausgabe: neue Partikelmenge X_t

1: $X'_t = X_t = \emptyset$
2: **for** $i = 1$ to N **do**
3: Generieren einer neuen Hypothese $x_t^n \sim p(x_t \mid x_{t-1}^n)$
4: $w_t^n = p(z_t \mid x_t^n)$
5: $X'_t = X'_t \cup \{\langle x_t^n, w_t^n \rangle\}$
6: **end for**
7: **for** $i = 1$ to N **do**
8: Ziehen von i mit $p(i) \propto w_t^i$
9: $X_t = X_t \cup \{x_t^i\}$
10: **end for**

das Gewicht w_t^n jedes Partikels berechnet durch die bedingte Wahrscheinlichkeit, die angibt, wie groß die Wahrscheinlichkeit ist, dass die gegebene Messung vorliegt, unter der Voraussetzung, dass der Zustand, den das Partikel repräsentiert, eingetreten ist. Bereits nach diesem Schritt wird die gesuchte Wahrscheinlichkeitsverteilung $bel(x_t)$ durch die Menge der gewichteten Partikel approximiert. Jedoch sind in ihr viele Partikel mit geringem Gewicht enthalten, die sich in Regionen des Zustandsraums mit geringer a-posteriori-Wahrscheinlichkeit befinden, wodurch das Partikelfilter wesentlich mehr Partikel benötigen würde, um über die Zeit hinweg zuverlässig die gesuchte Verteilung $bel(x_t)$ darstellen zu können. Aus diesem Grund wird in den Zeilen 7 bis 10 das *resampling* durchgeführt. Dabei wird aus der generierten Menge $\mathcal{X}'_t$ entsprechend der Gewichte der Partikel mit Zurücklegen gezogen. Partikel mit hohem Gewicht werden so mit höherer Wahrscheinlichkeit häufiger gezogen als solche mit geringem Gewicht. In der resultierenden Partikelmenge finden sich daher viele Duplikate von stark gewichteten Partikeln, andere Partikel sind dagegen aufgrund ihres geringen Gewichtes nicht enthalten. Über ihre Verteilung im Zustandsraum approximieren die nun wieder gleich gewichteten Partikel der neuen Partikelmenge $\mathcal{X}_t$ die a-posteriori-Wahrscheinlichkeitsverteilung $bel(x_t) = \eta\, p(z_t \mid x_t^n)\, \overline{bel}(x_t)$.

3.2.1 *Importance Sampling*

Resampling benötigt Informationen über die Wahrscheinlichkeitsverteilung, aus der gezogen werden soll. Diese Verteilung liegt aber nur als Approximation in Form der Partikelmenge vor und lässt das effektive Ziehen von Stichproben nicht ohne weiteres zu. Ein einfacher Ansatz ist das *rejection sampling*, das jedoch gerade aus dem Grund fehlender Informationen über die Verteilung sehr ineffizient ist, da häufig gezogen werden muss, um eine gültige Probe zu erhalten [Biso6]. *Importance sampling* ist eine Technik, die es ermöglicht, aus einer beliebigen Verteilung zu ziehen, die häufig auch *proposal distribution* genannt wird. Die zugrundeliegende Idee ist hierbei, die Proben nicht von der unbekannten Verteilung f – oft auch als *target distribution* bezeichnet – zu nehmen. Stattdessen wird an deren Stelle eine ähnliche bekannte Verteilung g verwendet, aus der man auf einfache Weise Proben generieren kann, wie es beispielsweise bei Gauß-Verteilungen möglich ist [TBF05].

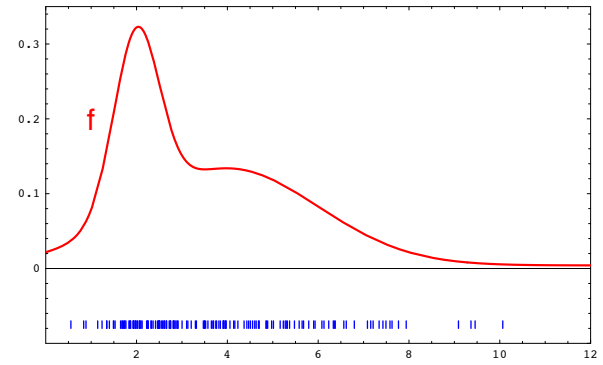
$$\begin{aligned} E_f[I(\mathrm{x} \in A)] &= \int f(\mathrm{x})\, I(\mathrm{x} \in A)\, d\mathrm{x} \\ &= \int \frac{f(\mathrm{x})}{g(\mathrm{x})}\, g(\mathrm{x})\, I(\mathrm{x} \in A)\, d\mathrm{x} \\ &= E_g\left[w(\mathrm{x})\, I(\mathrm{x} \in A)\right] \end{aligned} \tag{3.6}$$

I bezeichnet hierbei die Indikator-Funktion, die 1 liefert, wenn $\mathrm{x} \in A$ zutrifft und anderenfalls zu 0 wird, A bezeichnet ein beliebiges Intervall auf der Verteilung $p(\mathrm{x}_t \mid \mathrm{z}_t)$. Der Quotient

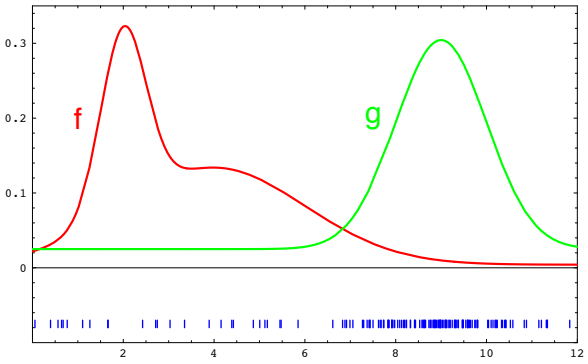
$$w(\mathrm{x}) = \frac{f(\mathrm{x})}{g(\mathrm{x})} \tag{3.7}$$

wird häufig *importance weight* genannt, und ist ein Maß für die Abweichung von f von g. Für jedes Partikel aus f soll gelten, dass die Wahrscheinlichkeit, dieses Partikel über der Verteilung von g zu ziehen, nicht Null ist: $f(\mathrm{x}) > 0 \Rightarrow g(\mathrm{x}) > 0$. Ist dies gegeben, dann konvergiert die Anzahl der gezogenen Partikel, die in A enthalten sind, gegen das Integral von g über A:

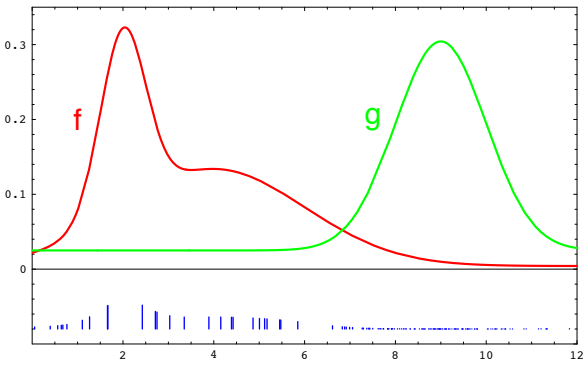
$$\frac{1}{N} \sum_{i=1}^{N} I(\mathrm{x}^n \in A) \rightarrow \int_A g(\mathrm{x})\, d\mathrm{x} \tag{3.8}$$



(a)



(b)



(c)

Abbildung 8: Grafik 8(a) zeigt die gesuchte Verteilung f, die durch die blau dargestellten Partikel approximiert wird. Da man aus f nicht effizient Stichproben ziehen kann, nimmt man diese stattdessen von einer ihr ähnlichen Verteilung g. In 8(b) wird die so generierte Partikelmenge dargestellt, die entsprechend g verteilt ist. Durch das Gewichten jedes Partikels x mit $f(x)/g(x)$ erhält man eine Approximation von f, wie in 8(c) illustriert. Bei Partikelfiltern entspricht die Verteilung g der a-priori-Verteilung $p(x_t \mid z_{1:t-1})$ und f der a-posteriori-Verteilung $p(x_t \mid z_{1:t})$.

3.2.2 Eigenschaften des Partikelfilters

Wesentlich für den Erfolg und die Robustheit der Zustandsschätzung eines Partikelfilters ist die Anzahl N der Partikel, welche die Wahrscheinlichkeitsverteilung $bel(x_t)$ über dem Zustandsraum repräsentieren. Gilt $N \to \infty$, so konvergiert diese Approximation gegen $bel(x_t)$. Es existiert allerdings keine offensichtliche Vorgehensweise zur Ermittlung der für eine robuste Verfolgung notwendigen Größe der Partikelmenge [KF00]. Zum einen ist N von der Dimensionalität des Zustands abhängig, aber auch von der Form von $bel(x_t)$. Denn wenn die Unsicherheit über den Zustand maximal ist und die Partikelmenge beispielsweise eine gleichverteilte Wahrscheinlichkeitsdichtefunktion approximiert, muss N größer sein, um eine zuverlässige Zustandsvorhersage zu ermöglichen, als im Fall einer Verteilung mit einem Maximum mit geringer Varianz [TBF05]. Bei einer konstanten Anzahl T von Bildern existiert eine Anzahl N von Partikeln, mit denen die tatsächliche Wahrscheinlichkeitsverteilung über der Bildsequenz von $t = 1, \ldots, T$ mit einer bestimmten Genauigkeit approximiert werden kann [KF00]. Für $t \to \infty$ können jedoch keine Aussagen getroffen werden, wie ISARD und BLAKE bemerken. Möglicherweise werde mit zunehmender Zeit eine größere Anzahl von Partikeln notwendig [IB98]. KING und FORSYTH stellen in [KF00] folgende Eigenschaften des Partikelfilters fest:

- Nichtdeterminismus: Unter gleichen Bedingungen kann das Partikelfilter stark abweichende Ergebnisse liefern.
- Einzelne Durchläufe generieren Erwartungswerte mit geringer Varianz und scheinen stabile Ergebnisse zu liefern.
- In einer Zeitspanne, die in etwa linear abhängig von der Anzahl der Partikel ist, kann der Zustand des Filters zu einem einzigen Maximum kollabieren.
- Dieses Maximum muss nicht in Zusammenhang mit dem Prozessmodell stehen.
- Selbst wenn keine sinnvolle Messung vorliegt, kann der Eindruck entstehen, als ob das Partikelfilter eine wahrscheinliche Hypothese verfolgt.

Der Fehler, der durch das Ziehen von Stichproben aus einer Approximation der eigentlichen Verteilung entsteht, wird auch *sampling variance* genannt. Problematisch ist, dass mit jedem Resampling-Schritt dieser Fehler verstärkt wird, und die Schätzung eines Zustands immer stärker vom eigentlichen Zustand abweicht. In einem Gedankenexperiment wird das Problem anhand eines Extrembeispiels deutlich [TBF05].

Ein Roboter, der sich nicht bewegt und keine Sensoren besitzt, versucht sich mit kartografischem Wissen über seine Umgebung zu lokalisieren. Geht man außerdem von einem deterministischen Prozessmodell aus, führt das extensive Resampling dazu, dass im Laufe der Zeit alle Partikel genau einen Zustand repräsentieren. Die Varianz innerhalb der Partikelmenge ist zwar minimal und es scheint, als ob das Partikelfilter den Zustand des Roboters mit sehr hoher Wahrscheinlichkeit bestimmen kann. Jedoch ist die Varianz zur tatsächlichen Verteilung über dem Zustand aufgrund des Approximationsfehlers sehr groß.

3.2.3 *Low Variance Sampling*

Um die Varianz von der Approximation zur tatsächlichen Verteilung zu reduzieren, werden Ansätze wie *variance reduction* (z.B. *sampling importance resampling* und *sequential importance sampling*), *low variance sampling* oder *stratified sampling* angewandt. In dieser Arbeit wurde das *low variance sampling* nach [TBF05] verwendet, wie es in Algorithmus 3 beschrieben ist.

Algorithmus 3 Low Variance Sampling

Eingabe: Partikelmenge X_t, Menge der Gewichte W_t

Ausgabe: X'_t

1: $X'_t = \emptyset$
2: $r = rand(0; \frac{1}{N})$
3: $c = w_t^1$
4: $i = 1$
5: **for** $j = 0$ to $N - 1$ **do**
6: $\quad U = r + \frac{j}{N}$
7: $\quad$ **while** $U > c$ **do**
8: $\quad\quad i = i + 1$
9: $\quad\quad c = c + w_t^i$
10: $\quad$ **end while**
11: $\quad X'_t = X'_t \cup \{x_t^i\}$
12: **end for**

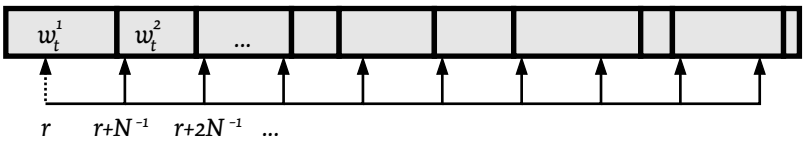


Abbildung 9: Prinzip des *low variance sampling*. Durch Ziehen einer Zufallszahl r werden N Partikel gemäß $u = r + \frac{(n-1)}{N}$ gezogen, wobei $1 \leq n \leq N$

Mittels *Low variance sampling* (bzw. *stochastic universal sampling*) werden N Stichproben aufgrund einer einzigen Zufallszahl gezogen. Dabei ist die Wahrscheinlichkeit, ein bestimmtes Partikel zu ziehen proportional zu seinem Gewicht. Auf einem Intervall $[0; \frac{1}{N}]$ wird eine Zufallszahl r generiert. Jeder zufällig gezogenen Zahl kann eindeutig ein Partikel x_t^i zugewiesen werden. In der while-Schleife ab Zeile 7 wird sukzessive das kumulierte Gewicht c der Partikel berechnet und ein Partikel entsprechend (3.9) gewählt:

$$i = \arg\min_j \sum_{j=1}^{n} w_t^j \geq U \tag{3.9}$$

Dieses wird schließlich in Zeile 11 der neuen Partikelmenge hinzugefügt. Für alle weiteren Stichproben muss lediglich sukzessive $\frac{1}{N}$ zu r addiert werden. Wenn nun alle Partikel das gleiche Gewicht besitzen, wie es auch im einführenden Beispiel mit dem stillstehenden Roboter der Fall wäre, finden sich in der neu generierten Menge alle Partikel wieder. Somit gehen in diesem Fall im Gegensatz zum einfachen *Resampling* keine Informationen verloren. Dadurch, dass das Ziehen aller Partikel abhängig von nur einer einzigen Zufallszahl und der Anzahl der Partikel geschieht, ist auch die Komplexität des *low variance sampling* linear bezüglich der Partikelanzahl, sie beträgt also $O(N)$ statt wie bei *rejection sampling* $O(N\,log N)$ [TBF05].

3.2.4 Mathematische Beschreibung des Partikelfilters

Partikel können als Abfolge von Zuständen $x_{0:t}^n = x_0^n, x_1^n, \ldots, x_t^n$ betrachtet werden [TBF05]. Dann lässt sich die gesuchte a-posteriori-Verteilung wie folgt beschreiben:

$$\begin{aligned} p(x_{0:t} \mid z_{1:t}) &\overset{Bayes}{=} \eta\, p(z_t \mid x_{0:t}, z_{1:t-1})\, p(x_{0:t} \mid z_{1:t-1}) \\ &\overset{Markow}{=} \eta\, p(z_t \mid x_t)\, p(x_{0:t} \mid z_{1:t-1}) \\ &= \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{0:t-1}, z_{1:t-1})\, p(x_{0:t-1} \mid z_{1:t-1}) \\ &\overset{Markow}{=} \eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1})\, p(x_{0:t-1} \mid z_{1:t-1})\,. \end{aligned} \tag{3.10}$$

Im Gegensatz zur Herleitung des Bayes Filters in 3.1 werden hier alle vorhergehenden Zustände in der a-posteriori-Wahrscheinlichkeitsverteilung betrachtet, wodurch die Integralzeichen wegfallen. Die Hypothese x_t^n wird rekursiv berechnet aus dem vorhergehenden Zustand x_{t-1}^n und dem Prozessmodell:

$$p(x_t \mid x_{t-1})\, bel(x_{0:t-1}) = p(x_t \mid x_{t-1})\, p(x_{0:t-1} \mid z_{1:t-1})\,. \tag{3.11}$$

Mittels *Importance Sampling* berechnet sich das Gewicht des Partikels x_t^n (s. Gleichung 3.7) zum Zeitpunkt t aus

$$\begin{aligned} w_t^n &= \frac{\text{target distribution}}{\text{proposal distribution}} \\ &= \frac{\eta\, p(z_t \mid x_t)\, p(x_t \mid x_{t-1})\, p(x_{0:t-1} \mid z_{1:t-1})}{p(x_t \mid x_{t-1})\, p(x_{0:t-1} \mid z_{1:t-1})} \\ &= \eta\, p(z_t \mid x_t) \end{aligned} \tag{3.12}$$

Da das *Resampling* mit proportionaler Wahrscheinlichkeit zu w_t^n geschieht, ist die resultierende Verteilung über den Partikeln proportional zum Produkt der *proposal distribution* mit den Gewichten w_t^n:

$$bel(x_{0:t}) = \eta\, w_t^n\, p(x_t \mid x_{t-1})\, p(x_{0:t-1} \mid z_{1:t-1}) \tag{3.13}$$

Wenn $x_{0:t}^n$ proportional zu $bel(x_{0:t})$ verteilt ist, so folgt daraus, dass entsprechend die Verteilung von x_t^n proportional zu $bel(x_t)$ ist.

4 Modellierung und Implementierung

In diesem Kapitel werden auf Grundlage der verwendeten Techniken des Partikelfilters und des kaskadierten, Haar-Wavelet-basierten Klassifikators die Modellierung der Bewegung von verfolgten Gesichtern, sowie die Modellierung der Messungen erläutert. Wie in Kapitel 3 ausführlich beschrieben wurde, sind Prozess- und Beobachtungsmodell ausschlaggebend für die erfolgreiche Zustandsschätzung des Partikelfilters. Durch sie erhält der Tracker die notwendigen Informationen über das zu verfolgende Objekt und dessen Position im Raum. In dieser Arbeit wurde besonderer Wert darauf gelegt, die Modellierung einfach zu halten, damit die Echtzeitfähigkeit gewährleistet werden kann.

Einleitend wird zunächst noch einmal der Kontext des Umfelds und der Technik beschrieben, in dem der Prototyp eingesetzt werden soll. Anschließend an die Ablaufbeschreibung eines einzelnen Verarbeitungsschrittes des Systems wird der Zustand definiert, auf dessen Grundlage Bewegungs- und Messmodell entwickelt werden.

In einem multimedialen Raum sollen Inhalte durch mit dreidimensionalem Klang verbundenen Videoprojektionen präsentiert werden. Jedoch ist der optimale audiovisuelle Eindruck nur in einem begrenzten Bereich in gleichem Abstand des Betrachters von den Lautsprechern überzeugend. Tritt der Besucher aus diesem *sweet spot* heraus, dann müssen die Schallwellen der näher gelegenen Lautsprecher so verzögert werden, dass sie bei ihm zeitgleich mit denen der weiter entfernten Lautsprecher ankommen. Werden die Schallwellen nicht an die Position des Betrachters angepasst, so scheint sich für ihn die Position der akustisch empfundenen Schallquelle relativ zur visuell wahrgenommenen zu verschieben, und er wird an der Immersion in die ihm präsentierten Inhalte gestört. Die beschriebene Situation ist in Abbildung 1 skizziert.

Da eine Projektionsfläche vorhanden ist, zu der sich der Besucher hinwendet, wurde die kamerabasierte Lösung eines Gesichtstrackers auf Basis der in 2.3 und 3.2 vorgestellten Techniken realisiert. Um eine zuverlässige dreidimensionale Ortung im Raum vorzunehmen, wurde eine Strereokamera des Modells *„Bumblebee“* von Point Grey Research [Reso8] verwendet. Diese muss vor dem Einsatz im beschriebenen Szenario mit den Lautsprechern kalibriert werden.

4.1 Implementierung des Gesichtsklassifikators in OpenCV

Auf Grundlage des leicht angepassten Gesichtsdetektors aus der OpenCV-Bibliothek wurde das Messmodell zu realisiert. In OpenCV stehen mehrere bereits angelernte Klassifikatoren zur Verfügung, die zunächst in einer XML-Repräsentation des kaskadierten Klassifikators vorliegen. Für die Anwendung eines Klassifikators wird aus der XML-Datei durch einen Aufruf aus der OpenCV-API eine eine baumartige Struktur aus C-Strukturen generiert. Die Implementierung in OpenCV ist sehr allgemein gehalten, um komplexere Klassifikatoren zu ermöglichen. Eine einfache Klassifikator-Kaskade ist folgendermaßen strukturiert:

In einem *kaskadierten Klassifikator* sind generelle Informationen des Klassifikators gespeichert, wie die Anzahl der Stufen der Kaskade, die originale Maskengröße der Trainingsbeispiele, mit denen der Klassifikator angelernt wurde, sowie die Skalierung, in der die Klassifikationen durchgeführt werden sollen. Dieser Skalierungswert bezieht sich auf die Originalmaskengröße. Außerdem sind je ein Zeiger auf den ersten Stufenklassifikator und auf eine von der API gekapselte Struktur gespeichert. Diese Struktur repräsentiert die gesamte Kaskade nochmals und wird im Anschluss kurz beschrieben.

Stufenklassifikatoren enthalten die Anzahl der einfachen Klassifikatoren, aus denen sie bestehen. Zudem enthalten sie je einen Schwellwert und Zeiger auf den folgenden sowie auf den vorhergehenden Stufenklassifikator der Kaskade. Im Fall, dass auch baumartig angeordnete Stufenklassifikatoren zugelassen sind, ist in ihnen zudem ein Zeiger auf den Elternknoten gespeichert. Wenn die Summe der Antworten der einfachen Klassifikatoren den Schwellwert übersteigen, wird der klassifizierte Bildausschnitt vom Stufenklassifikator als Gesicht klassifiziert, anderenfalls als Nicht-Gesicht. Schließlich enthält ein Stufenklassifikator einen Zeiger auf den ersten einfachen Klassifikator seiner Stufe.

Einfache Klassifikatoren besitzen wiederum einen Schwellwert, Zeiger auf benachbarte einfache Klassifikatoren und einen Zeiger auf eine Haar-Merkmal-Struktur. Anhand des Schwellwerts und des durch das Haar-Merkmal berechneten Wertes wird der Rückgabewert bestimmt.

Ein *Haar-Merkmal* repräsentiert schließlich ein aus zwei oder drei Rechtecken zusammengesetztes Merkmal, wie es in 2.3.1 beschrieben wurde. Jedes der Rechtecke ist mit einem Gewicht versehen, das angibt, ob die Summe seiner Pixel zur Gesamtsumme des Merkmals addiert oder subtrahiert wird. Die Gewichte eines Merkmals summieren sich dabei zu

Null, so dass im Fall des Drei-Rechteck-Merkmals (siehe Abb. 4 C) das schwarze Rechteck das doppelte Gewicht eines der weißen Rechtecke erhält.

Zu jeder der genannten Strukturen existieren außerdem zugehörige, dem Nutzer von der API versteckte Strukturen, in denen die genannten Informationen dupliziert werden. Im Folgenden werden sie mit ihren Bezeichnungen aus der OpenCV-Bibliothek benannt. In der Struktur `CvHidHaarClassifierCascade`, die dem kaskadierten Klassifikator entspricht, werden zusätzlich Zeiger auf die beiden (quadrierten) Integralbilder gespeichert. Für jedes Integralbild existieren außerdem vier Zeiger, die auf die Eckpunkte der Integralbilder in der entsprechenden Skalierung verweisen. Die Repräsentation des einfachen Klassifikators `CvHidHaarClassifier` enthält einen Zeiger auf einen `CvHidHaarTreeNode`, der wiederum Zeiger auf ein `CvHidHaarFeature` besitzt. In diesem sind letztendlich Zeiger auf die Eckpunkte der einzelnen Rechtecke im Integralbild gespeichert, aus denen das Merkmal zusammengesetzt ist. Ein Überblick über die Strukturen und deren Verbindungen ist in Abbildung 10 dargestellt.

Um die kaskadierten Strukturen zu generieren, und die Integralbilder zu berechnen, stellt OpenCV mehrere Funktionen bereit. Das Umwandeln der XML-Repräsentation eines Klassifikators geschieht durch den Aufruf von `cvLoad()`, das intern wiederum die gekapselte Funktion `icvCreateHaarClassifier()` nutzt. Für das Generieren der versteckten Strukturen wird die ebenfalls gekapselte Funktion `icvCreateHidHaarClassifier()` verwendet. Sie reserviert für eine gegebene Klassifikatorkaskade den für ihre gekapselte Repräsentation benötigten Speicherplatz. Das Berechnen der Integralbilder erledigt ein Aufruf von `cvIntegral()`. Schließlich werden die Zeiger der Haar-Wavelet-Merkmale auf die ihnen in der jeweiligen Skalierung entsprechenden Positionen in den Integralbildern durch die Funktion `cvSetImagesForHaarClassifierCascade()` berechnet.

In dieser Arbeit wird ein Klassifikator verwendet, der in 20 Stufen angelernt wurde. Weder die Stufenklassifikatoren noch die einfachen Klassifikatoren sind in einer eigenen Baumstruktur organisiert, es handelt sich also um die einfachste Form des Klassifikators (siehe Abb. 10). Die Größe der Bildausschnitte in den Trainingsdaten betrug 20 Pixel.

4.2 Ablauf des Programms

Zu Beginn dieses Abschnitts wird der grundlegende Ablauf eines Iterationsschritts des Programms beschrieben und anschließend auf die Parameter des Systems näher eingegangen.

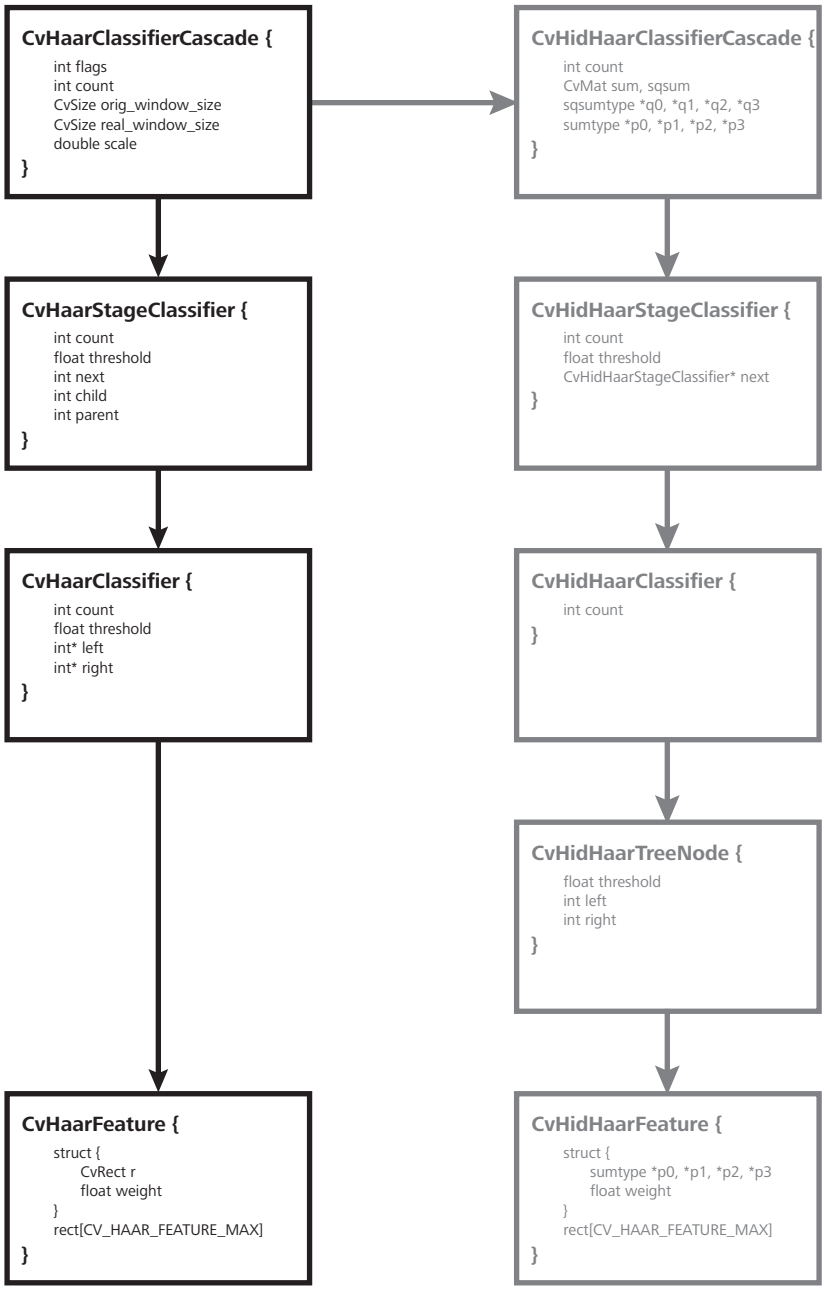


Abbildung 10: Diese Abbildung skizziert die Struktur der Implementierung des Klassifikators in OpenCV. Die Pfeile entsprechen Zeigern auf die jeweilige Datenstruktur. Vor der öffentlichen API versteckte Strukturen sind grau dargestellte, dokumentierte Strukturen schwarz.

Zum Programmstart werden die Partikel des Partikelfilters zufällig im Sichtkörper der Stereokamera initialisiert. Die Betrachtung des Zustands der Partikel geschieht in Kapitel 4.3. In jedem Iterationsschritt zum Zeitpunkt t werden nach der Akquisition des Kamerabildpaars die Partikel nach dem Prozessmodell (Kapitel 4.4) bewegt. Für jede Kamera existiert je ein Gesichtsdetektor, der die Integralbilder und für jede Skalierungsstufe eine Repräsentation des Klassifikators berechnet. Das Berechnen dieser Repräsentationen ist sehr zeitaufwändig, da die Zeiger in den zugehörigen versteckten Strukturen für alle Klassifikatoren der Kaskade auf die ihnen entsprechenden Positionen in der jeweiligen Skalierungsstufe in den Integralbildern bestimmt werden. Daher können diese Strukturen nicht zur Laufzeit für jedes Partikel gesondert berechnet werden. Stattdessen wird für jedes Bild eine bestimmte Anzahl von Repräsentationen des Klassifikators vorberechnet, die diskreten Skalierungsstufen entsprechen. Zum Programmstart werden die dafür benötigten Parameter wie der maximale Abstand von der Bildebene und die Anzahl der Stufen übergeben. Der minimale Abstand ist implizit durch die maximale Maskengröße, die Auflösung der Kamerabilder und die Gesichtsgröße in den Trainingsdaten gegeben (siehe Gleichung 4.7). Jedes Partikel wird nun zunächst entsprechend Kapitel 4.5.1 bezüglich beider Brennpunkte in die Bildebene projiziert, so dass ein Pixelpaar $(u_l, v_l), (u_r, v_r)$ für das linke – respektive rechte – Kamerabild entsteht. Anschließend werden die Gewichte der Partikel berechnet (Kapitel 4.5.3). Zuerst wird der Klassifikator im linken Kamerabild an Position (u_l, v_l) in der am nächsten gelegenen Skalierungsstufe (siehe Gleichung 4.5) angewandt, und nur im Fall, dass ein Gesicht klassifiziert wurde, wird auch im rechten Bild eine Klassifikation an der korrespondierenden Position (u_r, v_r) durchgeführt. Um diesen rechenaufwändigen Schritt zu beschleunigen, werden für jeden Zeitpunkt bereits berechnete Werte in einem dreidimensionalen Tensor gespeichert, der die Pixelpositionen in den jeweiligen Skalierungsstufen repräsentiert. Eine Pixelregion wird daher nur dann explizit klassifiziert, wenn in diesem Tensor an entsprechender Stelle noch kein Wert gespeichert wurde. Besonders bei einer großen Anzahl von Partikeln beschleunigt dies die Verarbeitungszeit deutlich, da durch das *resampling* viele Partikel auf die gleichen Pixel in derselben Skalierung projiziert werden.

Ist ein Partikel durch das Prozessmodell außerhalb des Sichtkörpers der Stereokamera bewegt worden, so wird sein Gewicht auf Null gesetzt und es wird im folgenden Iterationsschritt $t + 1$ neu initialisiert. Durch den Resampling-Schritt des Partikelfilters wird

die Partikelwolke in Bereichen im Raum konzentriert, in denen $bel(x_t)$ Maxima aufweist. Anschließend beginnt ein neuer Iterationsschritt.

4.3 Definition des Zustands

Um ein Gesicht im Raum zu verfolgen, sind als minimale Information dessen Koordinaten im dreidimensionalen Raum notwendig. Aus Gründen der Echtzeitfähigkeit wurde zunächst auf die Berücksichtigung der Geschwindigkeit verzichtet, da sich die Dimension des Zustands im *worst-case* exponentiell in der Komplexität des Partikelfilters niederschlägt [FHL+03]. Zusätzliche Informationen etwa über die Orientierung eines Gesichts wären zwar wünschenswert, können jedoch nicht alleine mit dem verwendeten Gesichtsdetektor aus den Bildern gewonnen werden, und werden daher in dieser Arbeit nicht berücksichtigt. Der Zustand wird somit durch folgenden Vektor beschrieben:

$$x_t = \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix}.$$

4.4 Prozessmodell

Vorhersagen über die Bewegung von Personen im Raum sind in der Regel schwer zu treffen. Gerade wenn diese sich in einer interaktiven multimedialen Ausstellungssituation befinden, können häufige abrupte Wechsel zwischen Bewegungs- und Ruhephasen sowie augenblickliche Richtungsänderungen zu erwarten sein. Das Einbeziehen von Geschwindigkeit und Beschleunigung in das Prozessmodell würde einen höher dimensionalen Zustand erforderlich machen, wodurch ein erhöhter Verarbeitungsaufwand entstünde (siehe 4.3). Dabei ist zu vermuten, dass eine physikalische Modellierung der Bewegung keinen entscheidenden Vorteil im Bezug auf die Genauigkeit der Objektverfolgung bringen würde. Daher wurde das Prozessmodell als ein Rauschen auf der Position modelliert. Unter diesen Bedingungen lässt sich das Prozessmodell folgendermaßen beschreiben:

$$x_t = x_{t-1} + \begin{pmatrix} \sigma_x \\ \sigma_y \\ \sigma_z \end{pmatrix}.$$

4.5 Beobachtungsmodell

Aufgrund des Fehlens einer genauen Beschreibung der Bewegung von Personen im Raum ist die Modellierung des Beobachtungsmodells besonders wichtig für die Genauigkeit des Trackers, da es ausschlaggebend für die Qualität des Korrekturschritts des Partikelfilters ist. Dabei musste ein sinnvoller Kompromiss zwischen möglichst guten Messergebnissen und des Berechnungsaufwands gefunden werden.

In Zeile 4 von Algorithmus 2 (S. 19) fließt die Messung des Systemzustands durch die Berechnung der Gewichte der Partikel in das Partikelfilter ein. Es wird also für jedes Pixel (u_i, v_j) zum Zeitpunkt t ein Wert benötigt, der angibt, wie wahrscheinlich die Detektion eines Gesichts an seiner Position ist, vorausgesetzt dass sich zum gleichen Zeitpunkt an dieser Stelle tatsächlich ein Gesicht im Bild befindet. Im Folgenden werden zunächst die Rahmenbedingungen des Beobachtungsmodells beschrieben, die zum einen durch die Perspektivprojektionen der beobachteten Szene in die Bildebene der Stereokamera und zum anderen durch die Parameter des Gesichtsdetektors bestimmt werden. Ausgehend von dieser Betrachtung wird anschließend ein Modell entwickelt, das die gesuchte Wahrscheinlichkeitsdichtefunktion möglichst sinnvoll repräsentiert.

4.5.1 Projektionsmodell der Stereokamera

Grundlegend für die anschließenden Berechnungen ist das Lochkameramodell, durch das Linsenkamerasysteme auf einfache Weise approximiert werden können. Voraussetzung für die Gültigkeit dieser Beschreibung ist eine genaue Kalibrierung der Kamera, welche die verwendeten Stereokameras „*Bumblebee*“ und „*Bumblebee 2*“ von Point Grey Research [Reso8] bereits erfüllen: Die Bildebenen beider Kameras liegen komplanar zueinander, sie haben die gleiche Brennweite f und ihre optischen Achsen sind parallel (siehe Abb. 11). Zudem sind die Bildebenen zeilengleich ausgerichtet und es gilt damit $v = v_l = v_r$. Der Ursprung wird in das Projektionszentrum der linken Kamera gelegt $O(0, 0, 0)$. Bei einem Abstand b der Brennpunkte beider Kameras liegt der Brennpunkt der rechten Kamera in $(b, 0, 0)$. Aus dem Strahlensatz folgt für die linke Kamera

$$u_l = \frac{f \cdot X}{Z}, \; v_l = \frac{f \cdot Y}{Z}.$$

Wegen der Verschiebung um b auf der X-Achse ergibt sich für die rechte Kamera

$$u_r = \frac{f \cdot (X - b)}{Z}, \; v_r = \frac{f \cdot Y}{Z}.$$

Der euklidische Abstand zwischen zwei korrespondierenden Bildpunkten im linken und rechten Bild wird als Disparität d bezeichnet. Sie berechnet sich aus

$$d = \sqrt{(u_l - u_r)^2 + (v_l - v_r)^2} = u_l - u_r,$$

da durch die Lage des Koordinatensystems $u_{left} \geq u_{right}$ und durch die zeilengleiche Ausrichtung $v_l - v_r = 0$ gilt. Sind f und b bekannt, so sind alle Informationen zur Rekonstruktion eines Punktes $P = (x, y, z)^T$ aus den Positionen $(u_l, v_l)^T$ und $(u_r, v_r)^T$ des linken respektive rechten Kamerabilds gegeben:

$$Z = \frac{f \cdot X}{u_l} = \frac{f \cdot (X - b)}{u_r} \tag{4.1}$$

$$X = \frac{b \cdot u_l}{u_l - u_r} \tag{4.2}$$

$$Y = \frac{b \cdot v}{u_l - u_r} \tag{4.3}$$

$$Z = \frac{b \cdot f}{u_l - u_r} \tag{4.4}$$

4.5.2 Modellparameter des Gesichtsdetektors

Die Grundlage für den Gesichtsdetektor bildete der nach [LKP03] entwickelte Klassifikator aus der OpenCV-Bibliothek, wie er in Kapitel 2.3 beschrieben wurde. Entgegen der theoretisch vernachlässigbar geringen Wahrscheinlichkeit des Klassifikators fälschlicherweise Gesichtsregionen zu klassifizieren (Gleichung 2.6), treten solche Fälle in der praktischen Anwendung wesentlich häufiger auf. Daher werden in der in OpenCV zur Verfügung gestellten *high-level*-Funktion `cvHaarDetectObjects()` für ein übergebenes Bild die Positionen gefundener Gesichter unter Berücksichtigung einer vorgegebenen minimalen An-

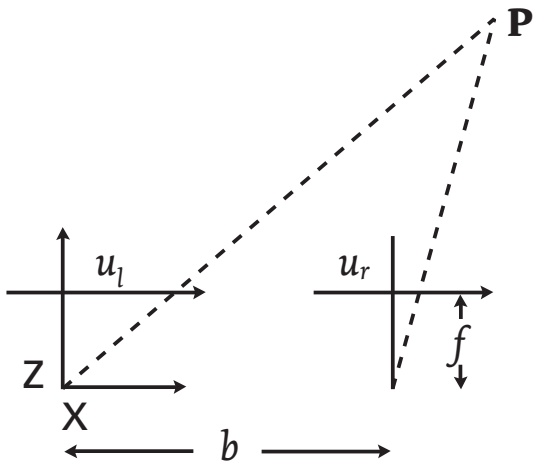


Abbildung 11: Die Projektion der Standardstereogeometrie gilt, wenn beide Kameras die gleiche Brennweite f haben, ihre optischen Achsen parallel sind und ihre Bildebenen zeilengleich ausgerichtet sind. Der Ursprung des Koordinatensystems liegt im Brennpunkt der linken Kamera, wobei die Y-Achse in die Bildebene hinein zeigt.

zahl benachbarter Gesichtsregionen zurückgegeben. Der dafür genutzte Radius wurde von [LKP03] nach empirischen Untersuchungen auf 30% der Breite des Suchfensters gesetzt.

Wie einleitend schon erwähnt, müssen jedoch Aussagen über die bedingten Wahrscheinlichkeiten an möglichen Folgepositionen der Zustandshypothesen verfügbar sein, um ein robustes Beobachtungsmodell erstellen zu können. Insbesondere war von Interesse, einzelne Pixelregionen in einer bestimmten Skalierung zu klassifizieren und damit signifikante Regionen gezielt untersuchen zu können, wodurch auch gleichzeitig die Verarbeitungsgeschwindigkeit des Systems gesteigert werden sollte.

Bei gegebener Auflösung ist die Brennweite der Kamera konstant. Zudem wird angenommen, dass ein durchschnittliches Gesicht eine Höhe von $H = 20$ cm hat und Abweichungen vom Durchschnitt minimal und damit zu vernachlässigen sind. Dabei bezeichnet h die Größe der Gesichter in den Trainingsdaten, die beim verwendeten Klassifikator 20 Pixel betrug. Aus dem Verhältnis der Größe des projizierten Gesichts p zu h ergibt sich die Skalierung:

$$s = \frac{p}{h} \tag{4.5}$$

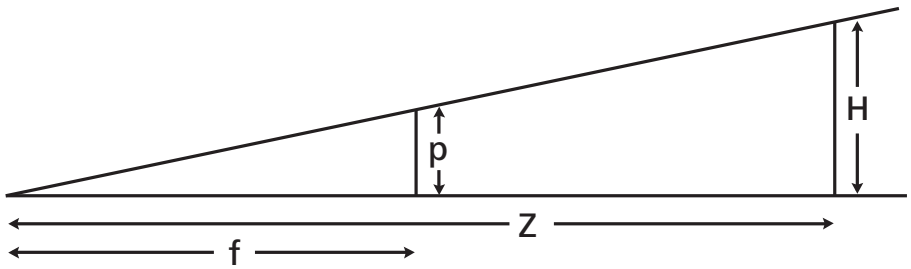


Abbildung 12: Mit Hilfe des Strahlensatzes kann die zu untersuchende Skalierungsstufe aus der Z-Koordinate des Partikels und der konstanten Parameter der natürlichen Gesichtsgröße H, der Brennweite f und der Maskengröße h der Trainingsdaten berechnet werden.

Mit Hilfe des Strahlensatzes lässt sich bei konstanter tatsächlicher Gesichtshöhe H, gegebener Maskengröße h und Brennweite f für jede Entfernung $Z > 0$ des Gesichts von der Bildebene die Skalierung berechnen, in der die Messung durchgeführt werden soll (siehe Abbildung 12):

$$\frac{H}{Z} = \frac{p}{f} \tag{4.6}$$

$$s = \frac{H * f}{Z * h} \, . \tag{4.7}$$

4.5.3 Erstellung des Beobachtungsmodells

Unter Verwendung der vorangegangenen Betrachtungen wird im Folgenden ein Modell entwickelt, das die gesuchte Wahrscheinlichkeitsdichtefunktion $p(z_t \mid x_t)$ approximiert. Hierbei werden zu erwartende Fehler auf ihre Ursachen untersucht und versucht, diese zu modellieren.

Prinzipiell kann man Fehlerquellen des vorliegenden Sensorsystems in zwei Komponenten unterscheiden: Zum einen sind durch die Projektion der Linsenoptik leichte Verzerrungen in den Kamerabildern zu erwarten. Da aufgrund der Einfachheit und der schnellen Berechenbarkeit ein Lochkameramodell verwendet wurde, entstehen Fehler in der Position projizierter Partikel, zumal die zweidimensionalen Koordinaten auf ganze Pixelpositionen gerundet werden. Außerdem birgt der Gesichtsklassifikator Fehlerquellen, die auf der einen

Seite durch den Prozess des Anlernens bedingt sind, auf der anderen Seite von Umwelteinflüssen abhängen.

Zwar werden die Eingabedaten vom Gesichtsdetektor mittels einer Beleuchtungskorrektur vorverarbeitet (siehe 2.3.3), bevor die eigentliche Klassifikation durchgeführt wird. Dennoch können starke Schwankungen in den Lichtverhältnissen dazu führen, dass die zu verarbeitenden Bilder so stark unter- bzw. überbelichtet sind, dass eine erfolgreiche Klassifikation insbesondere in relativ großen Entfernungen von der Kamera nicht möglich ist.

Außerdem kann eine Person, die ihr Gesicht von der Kamera abwendet, so dass im Extremfall nur noch ihr Hinterkopf zu sehen ist, ab einem bestimmten (z.B. nach 5.1.3 zu untersuchenden) Winkel nicht erkannt werden, da der verwendete Klassifikator nur auf frontalen Gesichtsbeispielen trainiert wurde. Bei erfolgreicher Klassifikation eines um die Y-Achse gedrehten Gesichts können allerdings Fehler in der Tiefenberechnung auftreten, wenn auf diese Weise rotierte Gesichter kleineren Skalierungsstufen zugeordnet werden, da sie geringere Ausmaße im Kamerabild einnehmen, obwohl der Abstand vom Mittelpunkt des Kopfes zur Bildebene konstant geblieben ist.

Um die beschriebenen Messfehler und die Verzerrungen der Linsenoptik im Vergleich zum Lochkameramodell zu berücksichtigen, sollte ursprünglich in einer dreidimensionalen Umgebung um die Position des projizierten Partikels gesucht und ein Konfidenzwert über diese Umgebung gemittelt werden. Ein solches Qualitätsmerkmal ließe sich etwa unter Annahme der stochastischen Unabhängigkeit der Messungen multiplikativ erzeugen, oder auch als eine gemittelte Summe berechnen. Dabei würde die Unsicherheit der Projektion so modelliert, dass die Detektorantwort an der projizierten Position stärker gewichtet wird, als die Werte für die Positionen in der Umgebung. Jedoch ist der Rechenaufwand für die Klassifizierung einer Pixelregion relativ hoch, und daher die Bildung eines solchen Mittelwerts in einem Echtzeitsystem nicht praktikabel, da die Anzahl der durchzuführenden Klassifikationen bei einer Umgebung der Größe N um den Faktor N^3 steigt.

Stattdessen wurde das Messmodell sehr einfach gehalten und der Klassifikator nur an Pixelpositionen angewandt, auf die ein Partikel projiziert wurde. Ausgehend von der Annahme, dass der Abstand b der Brennpunkte der Kameras auch in geringen Entfernungen keinen Einfluss auf das winkelabhängige Verhalten des Gesichtsdetektors hat, werden Treffer nur dann gezählt, wenn der Klassifikator in beiden Kamerabildern korrespondierende Pixelregionen als Gesicht klassifiziert. Daher wird für einen Partikel nur dann im rech-

ten Kamerabild eine Klassifikation durchgeführt, wenn im linken Bild bereits ein Treffer berechnet wurde.

Zusammengefasst kann das Beobachtungsmodell beschrieben werden als die Bildung des Minimums der Messungen in den Stereobildern an den korrespodierenden projizierten Partikelpositionen:

$$p(z \mid x) := \begin{cases} c & \text{wenn korrespond. Pixelregionen als Gesicht klassifiziert} \\ 1 - c & \text{sonst} \end{cases} \tag{4.8}$$

Die Implementierung des Klassifikators von [LKP03] in OpenCV liefert bei einem Treffer lediglich 1 zurück und anderenfalls eine negative Zahl, die angibt, bis zu welcher Stufe die Kaskade durchlaufen wurde. Auf Grundlage der Messwerte des Gesichtsdetektors wird jedoch der *resampling*-Schritt des Partikelfilters durchgeführt. Partikel, die aufgrund der Messungen kein Gesicht repräsentieren, würden ein Gewicht von $w = 0$ erhalten und somit keine Information mehr zur Zustandsvorhersage liefern können. Um die genannten Unsicherheiten der Klassifikation zu berücksichtigen, wird c entsprechend der Gleichungen 2.5 und 2.7 bewertet.

5 Experimente und Evaluation

Um einen Eindruck vom Verhalten des Prototyps zu gewinnen, wird davon zunächst eine Beschreibung gegeben. Anschließend werden die zu untersuchenden Aspekte des Systems vorgestellt, zu denen in den folgenden Abschnitten Entwürfe von Experimenten entwickelt werden.

Zum Programmstart wird der von der Stereokamera aufgespannte Sichtkörper mit einer Wolke von Partikeln gefüllt. In der 3D-Ansicht des Programms ist zu diesem Zeitpunkt die Partikelmenge in Form eines Pyramidenstumpfs zu erkennen 13(a). Zum Zeitpunkt der Aufnahme dieses Screenshots wurde das Gesicht noch nicht gefunden, da alle Partikel nur mit sehr geringer Wahrscheinlichkeit die Position des Gesichts repräsentieren. Dies kann dadurch bedingt sein, dass noch kein Partikel in der Umgebung des Gesichts existiert. Andererseits kann dies auch der ungünstigen Beleuchtung geschuldet sein, durch die es dem Klassifikator nicht gelingt, das Gesicht in dieser geringen Auflösung als solches zu erkennen. Der Entwurf eines Experiments, welches die Toleranz des Klassifikators gegenüber unterschiedlichen Beleuchtungssituationen untersuchen soll, wird in Kapitel 5.1.3 vorgestellt.

Das grüne Ellipsoid in der 3D-Ansicht stellt die Kovarianz der Partikelwolke dar und die blaue Kugel verdeutlicht die Position des Schwerpunkts der Wolke. Betritt eine Person die Szene, und befindet sich ein Partikel in unmittelbarer Umgebung ihres Gesichts, so versammeln sich die Partikel innerhalb weniger folgender Zeitschritte um diese Position. In der 3D-Ansicht ist deutlich zu erkennen, dass die Kovarianz stark abgenommen hat (siehe Abbildung 13(b)). Ebenso ist gut zu sehen, dass die Streuung in z-Richtung, also in den Sichtkörper hinein, wesentlich größer ist, als in x- und y-Richtung, was durch die prinzipbedingte Ungenauigkeit in der Berechnung der Projektion entsteht. Verliert das Partikelfilter die Position des Gesichts, dann vergrößert sich aufgrund des zufälligen Prozessmodells die Streuung der Partikel. Wird das Gesicht nicht wieder gefunden, dann füllen die Partikel nach einer gewissen Zeitspanne den Sichtkörper wieder vollständig aus.

Im Wesentlichen hängt der Erfolg des Gesichts-Trackers von seinen zwei Hauptbestandteilen ab. Der Gesichtsdetektor liefert die Grundlage für das Messmodell des Partikelfilters und somit ist seine Fähigkeit, Gesichter auch unter schwierigen Bedingungen zu lokalisieren von entscheidender Bedeutung für den Einsatz in realen Umgebungen. Außerdem wirken sich die Parameter des Partikelfilters stark auf Geschwindigkeit und Erfolg der Objektverfolgung aus. Anhand von ausgewählten Experimenten sollen die Auswirkungen auf

das Gesamtverhalten des Trackingsystems analysiert werden. Es werden verschiedene Szenarien beschrieben, von denen aber aus zeitlichen Gründen nur einige ausgewählte durchgeführt werden.

5.1 Gesichtsdetektor

Als Kernbestandteil des Messmodells des Partikelfilters ist es entscheidend, dass der Gesichtsdetektor auch in schwierigen Situationen zuverlässige Ergebnisse liefert, damit der Partikelfilter erfolgreich Gesichter verfolgen kann. Schwierigkeiten bereiten in natürlichen Umgebungen beispielsweise die variable Beleuchtung, und die Tatsache, dass Gesichter in den häufigsten Fällen nicht gleichzeitig frontal und aufrecht zur Bildebene hin ausgerichtet sind.

5.1.1 Variation der Anzahl der Skalierungsstufen

Da das Berechnen der Repräsentation des Klassifikators für jede beliebige Skalierungsstufe zur Laufzeit zu rechenaufwändig ist, werden diese Kaskaden nur für eine vorgegebene Anzahl von Skalierungsstufen berechnet. Deswegen wird ein Partikel auf die seiner Z-Koordinate am nächsten gelegene skalierte Repräsentation projiziert und an dieser Pixelposition geprüft, ob der Gesichtsdetektor ein Gesicht vorfindet. Durch die Diskretisierung des Skalenraums ist zu erwarten, dass beim Gewichten der Partikel Fehler entstehen, da abhängig von der Anzahl der Stufen der Abstand zwischen der eigentlich zu untersuchenden und der tatsächlich untersuchten Skalierungsstufe unterschiedlich groß ausfallen kann. Besonders bei wenigen überprüften Skalierungen kann dies dazu führen, dass kein Gesicht mehr gefunden wird, da die Maxima der Wahrscheinlichkeitsdichtefunktion zwischen den einzelnen Stufen liegen.

Anhand von einem Experiment soll nun analysiert werden, wie sich die Anzahl der Skalierungsstufen auf die Verlässlichkeit der Messung auswirkt. Dafür wird ein einzelnes Kamerabild in mehreren Durchläufen in jeweils einer anderen Anzahl von Skalierungsstufen untersucht. Dabei ist insbesondere interessant, ob positive Detektorantworten tatsächlich nur in einer begrenzten Pixel- und Skalenumgebung erzielt werden und wie groß diese Umgebungen sind. Bei den aufgenommenen Bildern werden zunächst per Hand die Gesichtsmitten in den Bildern bestimmt, sowie die Größe des Gesichts, aus der sich nach Gleichung 4.5 die tatsächliche Skalierung ermitteln lässt.

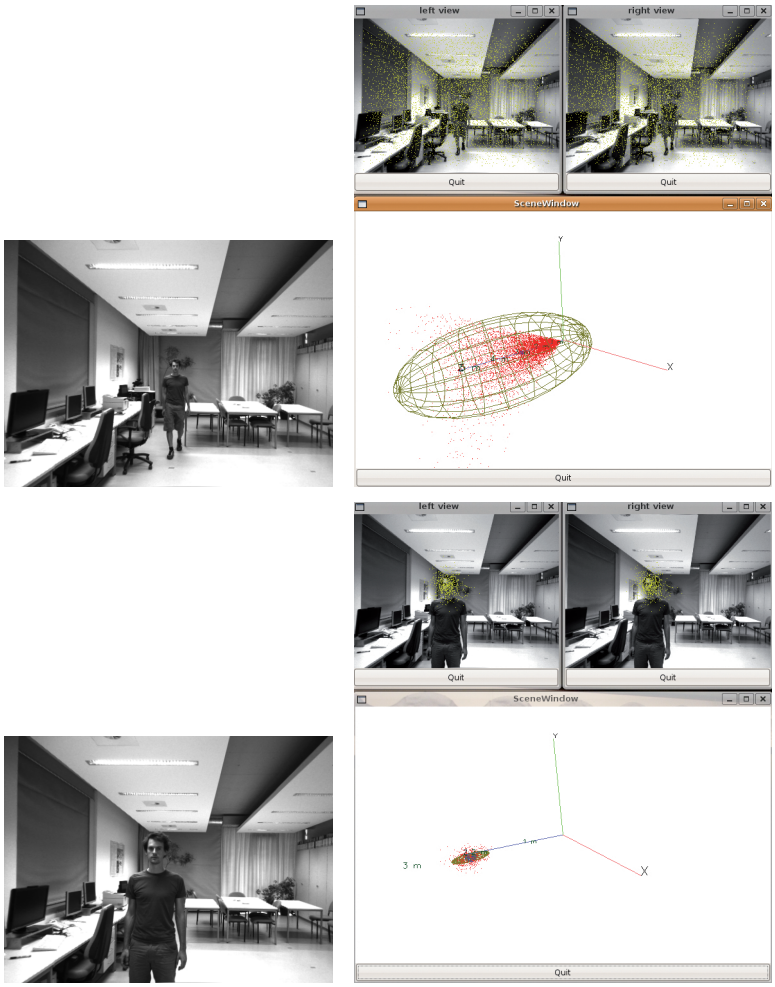


Abbildung 13: Screenshots des Prototypen. Auf der linken Seite ist das linke Kamerabild zum Bild 150 bzw. 500 einer Testsequenz abgebildet. Die drei Ansichten auf der rechten Seite zeigen die entsprechenden linken und rechten Kamerabilder, die mit den projizierten Partikeln überlagert sind. Darunter sind die jeweilige Lage der Partikelwolke relativ zum Urpsprung der Szene dargestellt, welcher im Brennpunkt der linken Kamera liegt. Im linken Screenshot ist eine Momentaufnahme nach 150 Bildern zu sehen. Wie in der 3D-Ansicht zu sehen ist, sind die Partikel sind im gesamten Sichtkörper der Kamera verteilt. Das grüne Ellipsoid stellt die Kovarianz der Partikelwolke dar und erstreckt sich über einen großen Teil des Raums. Der rechte Screenshot zeigt den Zustand des Systems bei Bild 500 der Sequenz. Alle Partikel haben sich um das Gesicht versammelt. In der 3D-Ansicht ist zu sehen, dass die Kovarianz deutlich kleiner ist und sich der Schwerpunkt (blaue Kugel) der Partikelwolke bei etwa zwei Meter befindet, was in etwa der tatsächlichen Entfernung des Gesichts von der Kamera entspricht.

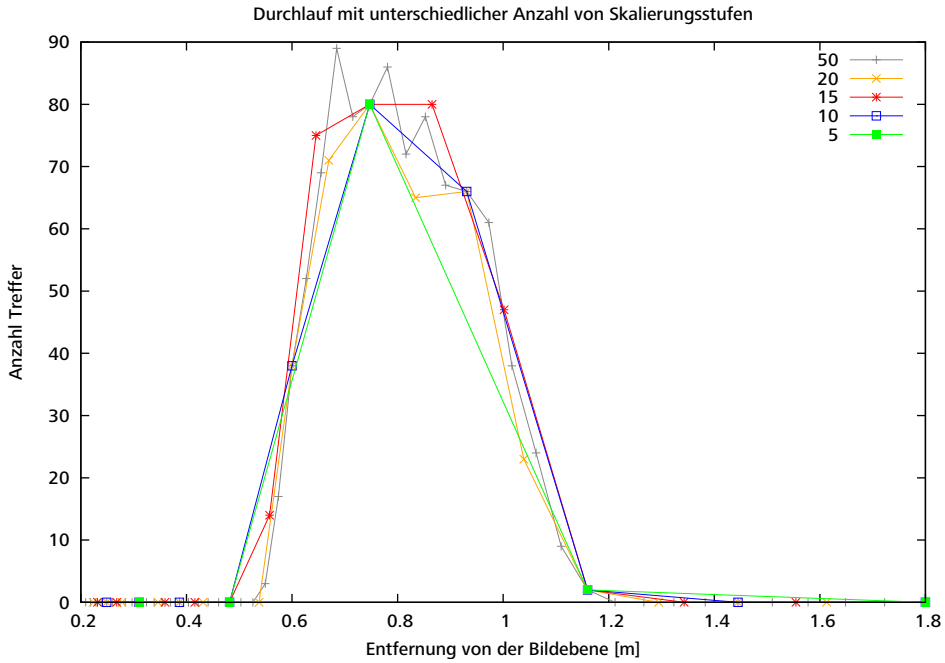


Abbildung 14: Dieses Experiment (siehe 5.1.1) zeigt deutlich, dass der Klassifikator nur in einer begrenzten Umgebung der tatsächlichen Entfernung des Gesichts von der Kamera positive Werte liefert.

Mit diesen Daten werden die Werte des Gesichtsdetektors verglichen, indem in einer Pixelumgebung von maximal 30% der Gesichtsgröße um die per Hand ermittelte Position gesucht wird. Außerdem werden nur Treffer gezählt, deren Gesichtsgröße innerhalb ± 50% der tatsächlichen Größe liegen (siehe [LKP03]). Schließlich werden alle Treffer pro Durchlauf aufsummiert .

In diesem Experiment befand sich das Gesicht in einem Abstand von einem Meter von der Kamera. In Abb. 14 ist gut zu sehen, dass der verwendete Klassifikator nur in einer deutlich begrenzten Umgebung des tatsächlichen Gesichts Pixelregionen als Gesicht klassifiziert. Es lässt sich jedoch nicht genau ausmachen, wie viele Skalierungsstufen für eine verlässliche Detektion notwendig sind. Zu vermuten ist jedoch, dass ab etwa 15 Skalierungsstufen kein Zugewinn an Genauigkeit zu erwarten ist.

Daher wurde in einem zweiten Experiment untersucht, ob die Trefferrate in der oben beschriebenen Umgebung des Gesichts von der Anzahl der Skalierungsstufen abhängt. Dafür wurde die gleiche Vorgehensweise wie im vorhergehenden Experiment genutzt. Für jeden

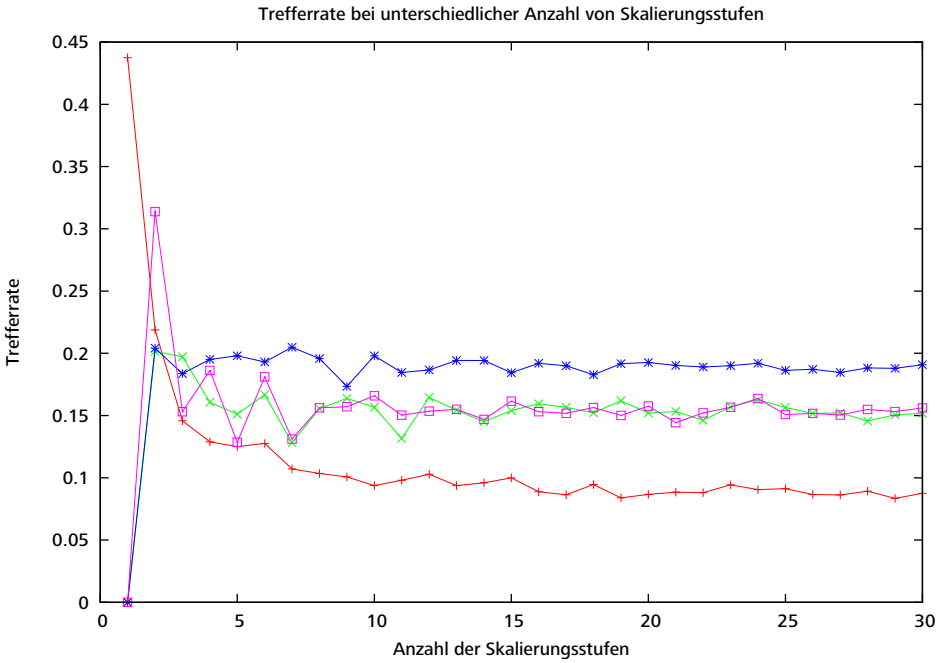


Abbildung 15: Ergebnisse von vier verschiedenen Einzelbildern von Gesichtern, die jeweils mit einer unterschiedlichen Anzahl von Skalierungsstufen in einer Umgebung von ±0,5 m untersucht wurden. Die Anzahl der Treffer wurde jeweils mit der Anzahl der untersuchten Pixelpositionen der jeweiligen Skalierungsstufe ins Verhältnis gesetzt. Es zeigt sich, dass ab etwa 15 Skalierungsstufen die Trefferrate in etwa konstant bleibt.

Durchlauf wurde die absolute Anzahl der Treffer in Relation zur Anzahl aller untersuchten Positionen gesetzt. Dabei zeigt sich, dass ab etwa 15 Skalierungsstufen die Trefferrate nahezu konstant bleibt (Abb. 15), was die aus dem vorhergehenden Experiment getroffene Vermutung zu bestätigen scheint. Mit zunehmender Anzahl der untersuchten Positionen steigt dann lediglich der Verarbeitungsaufwand, mit einer höheren Genauigkeit ist jedoch nicht zu rechnen.

5.1.2 Rotation in der Bildebene (X-Y-Ebene)

Ebenfalls auffällig war bei den ersten Tests des Tracking-Systems die Sensibilität des Gesichtsdetektors gegenüber in der Bildebene (um die z-Achse) gedrehter Gesichter. In einem Experiment soll verdeutlicht werden, wie stark ein Gesicht gegenüber der vertikalen Achse rotiert sein darf, damit es vom Detektor noch als solches klassifiziert werden kann. Für

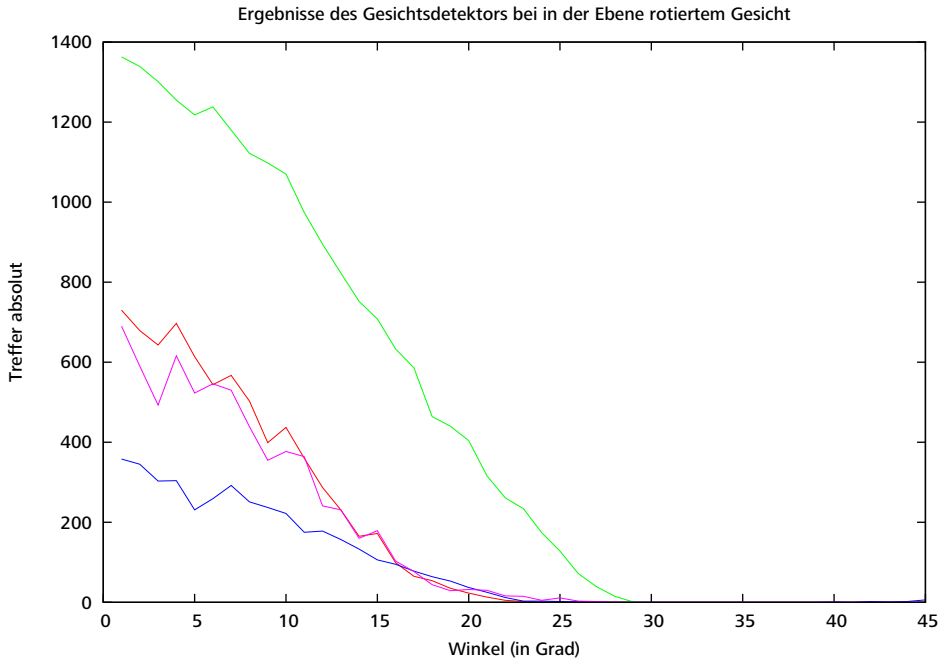


Abbildung 16: Abhängigkeit des Gesichtsklasifikators von der Rotation des Gesichts in der Bildebene. Bereits bei einer Rotation um 10 bis 15º nimmt die Anzahl der Treffer um etwa die Hälfte ab.

dieses Experiment bietet es sich an, von einem Kamerabild um jeweils sukzessive 1° rotierte Duplikate zu erstellen. Die Vorgehensweise bei der Auswertung gleicht der der beiden vorhergegangenen Experimente, es wird die Erfolgsquote des Detektors gemessen an den per Hand ermittelten Ergebnissen. Anhand der Ergebnisse ist zu sehen, dass eine relativ geringe Neigung des Kopfes zur Seite bereits eine starke Beeinträchtigung der Klassifikationsergebnisse bedeutet. Im Einsatz in multimedialen Umgebungen ist jedoch anzunehmen, dass dieser Effekt durch die Verfolgung des Gesichts mittels des Partikelfilters in den meisten Fällen annähernd aufgefangen werden kann. Eine Möglichkeit, auch in der Bildebene rotierte Gesichter zuverlässig zu klassifizieren, ist das Anlernen eines Klassifikators mit entsprechenden Trainingsdaten. Der wesentliche Verbesserungsvorschlag zum Gesichtsklassifikator von Viola et. al in [LKP03] – die Erweiterung der Menge der Merkmalstypen durch um 45º gedrehte Merkmale – könnte dies unterstützen, da der verwendete Klassifikator mit lediglich drei unterschiedlichen Typen angelernt wurde, die zudem nicht rotiert wurden.

5.1.3 Weitere Möglichkeiten für Experimente am Gesichtsklassifikator

Im Folgenden werden zwei Ideen für Experimente am Gesichtsklassifikator beschrieben, die weitere wichtige Aspekte beleuchten könnten, aufgrund des höheren Aufwands aber nicht durchgeführt wurden.

Variation der Beleuchtungssituation Obwohl die OpenCV-Implementierung des Algorithmus von VIOLA ET. AL intern zuerst eine Normalisierung der Bilddaten vornimmt, haben sich in ersten Tests des Trackers Probleme mit der Toleranz des Gesichtsdetektors gegenüber unterschiedlichen Ausleuchtungen von Gesichtern gezeigt, indem ein Gesicht nicht verfolgt wurde, obwohl es für das menschliche Auge unschwer auf dem Kamerabild zu erkennen war. Eine zusätzliche auf das Gesicht gerichtete Lichtquelle führte zu einer erfolgreichen Erkennung.
Um das Verhalten des Gesichtsdetektors in unterschiedlichen Beleuchtungssituationen zu testen, soll eine Reihe von Bildern aufgenommen werden, die in Position und Orientierung konstant sind, sich jedoch in der Beleuchtung unterscheiden. Interessant sind etwa Gegenlichtsituationen, in denen sich ein Gesicht vor sehr hellem Hintergrund befindet und nur teilweise bzw. schlecht ausgeleuchtete Gesichter.
Für die Auswertung würde es sich anbieten, wiederum die Anzahl der vom Gesichtsdetektor gefundenen Gesichter innerhalb der Gesichtsregion aufzusummieren. Als Maß für die Qualität der Beleuchtung kann beispielsweise der WEBER-Kontrast [SAG+05] der jeweiligen Gesichtsregion dienen:

$$K = \frac{L_{max} - L_{min}}{L_{min}} ,$$

wobei L_{max} und L_{min} die maximale respektive minimale Leuchtdichte bezeichnen.

Die Durchführung dieses Experiments erfordert gut kontrollierbare und reproduzierbare Beleuchtungssituationen. Diese herzustellen hätte aber einen hohen zeitlichen Aufwand bedeutet. Unter normalen Bedingungen funktioniert die Gesichtsdetektion ausreichend zuverlässig. Im prototypischen Einsatz zur Evaluation der Raumklangsteuerung ist zudem die Möglichkeit der ausreichenden Beleuchtung der Gesichter gegeben, weswegen die Dringlichkeit dieses Experiments als eher gering eingestuft wurde.

Rotation aus der Bildebene heraus (X-Z-Ebene) In realen Umgebungen wird es häufig vorkommen, dass ein Gesicht nicht direkt in die Kamera blickt, so dass im Bild nur

sein Halbprofil bzw. im Extremfall nur das Profil zu sehen ist. Obwohl der verwendete Gesichtsklassifikator ausdrücklich zur Klassifikation von frontalen Gesichtern trainiert wurde (siehe [LKP03]), scheint er relativ robust gegen Drehungen um die Y-Achse zu sein. Um diese Vermutung zu bestätigen, könnte man ein dem vorhergehenden Experiment entsprechendes durchführen, das sich im Wesentlichen durch die verwendete Testsequenz unterscheidet: Eine Person wird von der Kamera in unterschiedlichen Winkeln zur Bildebene aufgenommen (z.B. in 10°-Schritten von 0° bis 90°). Die Auswertung würde wieder analog zu den vorher beschriebenen Experimenten erfolgen. Für aussagekräftige Ergebnisse wäre eine sorgfältig erstellte Sequenz von Testdaten erforderlich gewesen. Aufgrund der ausreichenden Robustheit des Klassifikators wurde daher auf die Durchführung dieses Experiments verzichtet.

5.2 Partikelfilter

Um die Echtzeitfähigkeit des Tracking-Systems zu ermöglichen, wurden insbesondere für das Messmodell Entscheidungen getroffen, die sich negativ auf die Zuverlässigkeit der Positionsbestimmung auswirken können. Das Prozessmodell wurde sehr einfach gehalten, um eine niedrige Dimensionalität des Zustands zu erreichen. Durch das Messmodell wird die Wahrscheinlichkeitsdichtefunktion über dem Zustand nur durch zwei diskrete Werte approximiert und Abweichungen zwischen Projektionsmodell der Stereokamera und dem verwendeten Lochkameramodell wurden ignoriert. Diese Entscheidungen wurden ausführlich in Kapitel 4 beschrieben und begründet. An dieser Stelle sollen Experimente zum Überprüfen ihrer Auswirkungen auf den Erfolg der Gesichtsverfolgung hin skizziert werden. Außerdem sollen Annahmen über Parametereinstellungen überprüft werden.

5.2.1 Variation der Partikelanzahl

Ausschlaggebend für die erfolgreiche Gesichtsverfolgung ist, dass das Partikelfilter weder über zu viele noch zu wenige Partikel verfügt, da im Extremfall kein Gesicht verfolgt werden kann bzw. der Rechenaufwand so sehr steigt, dass die Echtzeitfähigkeit des Systems eingebüßt wird. Im folgenden einfachen Experiment wird eine Bildsequenz herangezogen, in der sich ein Gesicht sich im Raum bewegt und stets aufrecht in die Kamera blickt. In mehreren Durchläufen wird nun die Partikelanzahl von anfangs 10000 sukzessive um 500 reduziert und der Erfolg der Objektverfolgung beurteilt. Erfolgreich ist der Test, wenn sich am Ende der Sequenz der Schwerpunkt der Partikelwolke in einem Abstand von maximal $\frac{1}{3}$

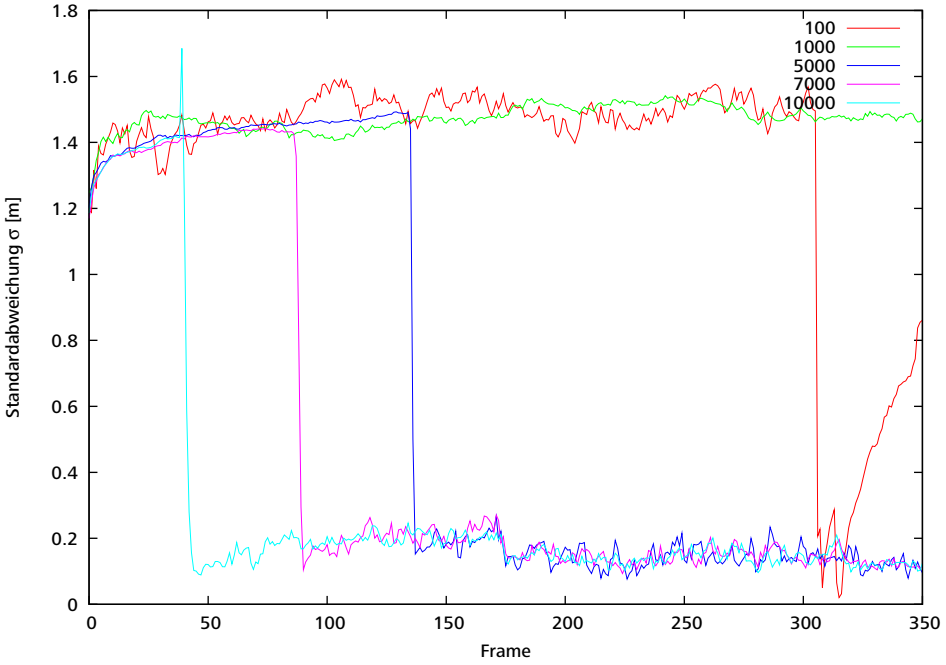


Abbildung 17: In dieser Testreihe wurde untersucht, wie sich die Anzahl der Partikel auf die Verlässlichkeit des Systems auswirkt (5.2.1). Bei einer niedrigen Anzahl von Partikeln ist keine verlässliche Objektverfolgung möglich.

der Gesichtsgröße [LKP03] vom tatsächlichen Gesicht entfernt befindet und die Standardabweichung der Partikel innerhalb 150% der Gesichtsgröße liegt. Für die Testsequenz bietet es sich an, das Gesicht in einer Diagonalen durch den von der Stereokamera aufgespannten Sichtkörper wandern zu lassen, da dadurch im Fall, dass der Partikelfilter das Gesicht nicht verfolgen kann, die Wahrscheinlichkeit des zufälligen Wiederfindens gering gehalten wird.

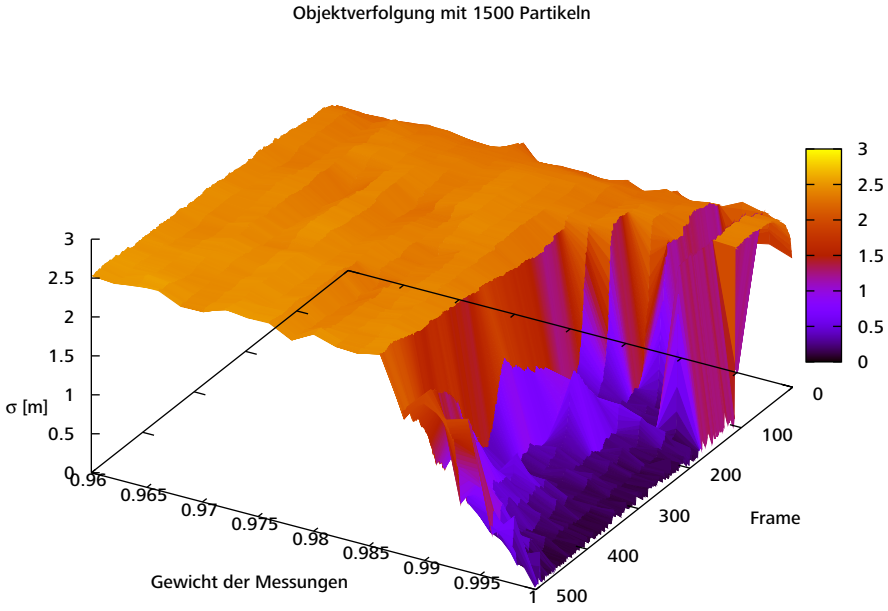
In Abbildung 17 ist das Verhalten des Partikelfilters bei unterschiedlicher Anzahl von Partikeln zu sehen. Bei relativ wenigen Partikeln wird unter Umständen über die gesamte Sequenz kein Gesicht gefunden und verfolgt. Ist die Anzahl zu gering, und ein Partikel befindet sich zufälligerweise in einer Gesichtsregion, so sammelt sich die Partikelwolke um diese Position. Nach kurzer Zeit verliert das Partikelfilter aber das zu verfolgende Gesicht wieder. Je höher die Anzahl der Partikel ist, desto früher wird in der Bildsequenz das Gesicht gefunden. Zudem scheint die Varianz der Standardabweichung bei höherer Partikelzahl geringer zu sein. Für eine echtzeitfähige Verarbeitung sind etwa 7000 Partikel als praktikabel erschienen, wobei etwa fünf Bilder pro Sekunde erreicht wurden.

5.2.2 Abhängigkeit zwischen Partikelanzahl und positiver Detektorantwort

Aufgrund der Entscheidung, zugunsten der Echtzeitfähigkeit ein sehr einfaches Messmodell zu verwenden (wie in Kapitel 4.5 beschrieben), wird für jedes Partikel nur an maximal zwei Pixelpositionen in den korrespondierenden Kamerabildern in jeweils einer Skalierung der Gesichtsdetektor angewendet. Die Bildung des Minimums der Rückgabewerte des Detektors hat zur Folge, dass die Wahrscheinlichkeitsdichtefunktion nur zwei verschiedene Werte annehmen kann. Erhöht man nun die Anzahl der Partikel, so wird das Gesicht zwar weiterhin erfolgreich verfolgt, die Streuung der Partikel wird jedoch größer. Nach ersten Versuchen mit einem Heraufsetzen der positiven Detektorantwort und gleichzeitigem Verringern des entsprechenden negativen Wertes (Gleichung 4.8) drängte sich die Vermutung auf, dass das Messmodell zu instabil und abhängig von der Mächtigkeit der Partikelmenge sei. Um diese Behauptung zu untersuchen, wird wieder die Testsequenz aus 5.2.1 verwendet. In mehreren Experimenten soll dabei sowohl die Partikelanzahl verändert werden als auch der Wert des Gesichtsdetektors, der im Fall für die Klassifikation eines Pixels als Gesichts zurückgeliefert wird. Als Qualitätskriterium dient hierbei die Streuung der Partikelwolke, die durch die Standardabweichung angegeben wird und möglichst gering ausfallen sollte. In den Abbildungen 18(a) und 18(b) ist deutlich zu sehen, dass im Messmodell ein sehr hohes Gewicht für Treffer gesetzt werden muss, damit eine erfolgreiche Verfolgung möglich ist. Es bestätigt sich auch die Vermutung, dass bei hohen Partikelanzahlen und einem Gewicht zwischen etwa 0.98 und 0.99 die Standardabweichung höher sein kann, als bei einer kleineren Partikelmenge. Diese Tatsache lässt sich dadurch erklären, dass das geringe Gewicht von Partikeln, die kein Gesicht repräsentieren in der Summe relativ hoch ist. Im das *Resampling*-Schritt ist die Wahrscheinlichkeit, gering gewichtete Partikel zu ziehen daher relativ hoch im Vergleich zur „Anziehungskraft“ von Partikeln, die Gesichter repräsentieren. In der Praxis hat es sich bewährt, bei einer Anzahl von 7000 Partikeln die Gewichte für Treffer auf 0.9995 zu setzen.

5.2.3 Weitere Möglichkeiten für Experimente am Partikelfilter

Aus zeitlichen Gründen konnten die folgenden Experimente nicht durchgeführt werden. Sie werden an dieser Stelle dennoch skizziert werden, da sie weitere interessante Aspekte des Partikelfilters beleuchten sollten.



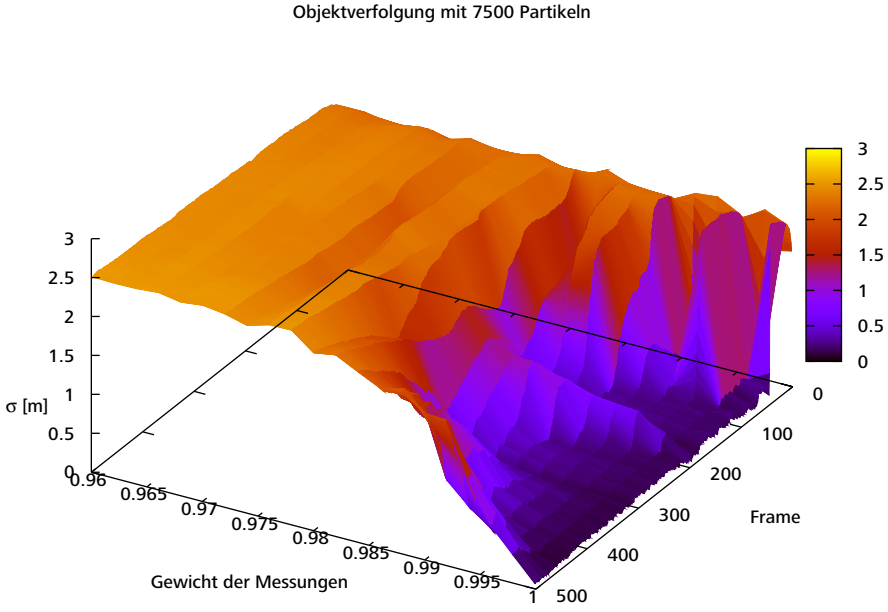


Abbildung 18: In beiden Fällen ist zu sehen, dass das Gewicht von Partikeln, die eine Gesichtsposition repräsentieren, relativ hoch sein muss, damit das Partikelfilter ein Gesicht erfolgreich verfolgen kann. Eine höhere Anzahl von Partikeln approximiert die a-posteriori-Wahrscheinlichkeitsverteilung $bel(x_t)$ besser und liefert stabilere Ergebnisse.

Variation des Bewegungsmodells Ein physikalisches Bewegungsmodell, das zumindest die Geschwindigkeit der Partikel enthält, wurde aufgrund des erhöhten Rechenaufwands und der These, dass das Partikelfilter in höheren Dimensionen zu unstabileren Ergebnissen führt [DdFG01] aufgegeben. Stattdessen wird die neue Position eines Partikels aus einer Gauß-Verteilung um seine alte Position gezogen. Die Standardabweichung für die drei Koordinatenachsen dieser Verteilung wird dem Programm als Parameter übergeben. Interessant bei diesem einfachen Bewegungsmodell ist wie groß die Varianz sein muss, damit natürliche menschliche Bewegungen erfolgreich modelliert werden können. In einem ersten Test sollen die x-, y- und z-Komponente als stochastisch unabhängig betrachtet werden. Die Verteilung, aus der zufällig gezogen wird ist somit rotationssymmetrisch. Anschließend soll ausgehend von den aus diesem Experiment gewonnenen, möglichst optimalen Werten für die x- und y-Komponenten von Sigma ein Optimum für die z-Komponente gefunden werden. Wiederum bietet sich die Sequenz aus 5.2.1 an. Als Maß für das Optimum kann eine möglichst kleine Norm der Standardabweichung dienen, die das Verfolgen während der gesamten Testsequenz ermöglicht.

Abweichung der Trajektorien zwischen Stereobild-Tracking und einem monokularen System Den Großteil der Rechenzeit benötigt das Tracking-System, um für jeden Zeitschritt die Integralbilder zu berechnen und den Klassifikator auf alle Partikel anzuwenden. Die Anzahl der durchzuführenden Klassifikationen wurde bereits durch die Erstellung des Messmodells und des Zwischenspeicherns der Werte bereits klassifizierter Positionen reduziert. Würde man jedoch statt einer Stereokamera eine monokulare Kamera einsetzen, so ist zu vermuten, dass sich die Rechenzeit nahezu halbiert. Allerdings leidet die Genauigkeit der Ortung von Gesichtern darunter, da die notwendigen Informationen für eine Triangulierung fehlen und sich dadurch die Berechnung der z-Komponenten der Positionen alleine auf die Skalierungsinformationen des Gesichtsdetektors stützen muss (siehe Gleichung 4.7). Für den Vergleich der Genauigkeit der Lokalisierung von stereobasiertem Gesichtstracker und der monokularen Lösung werden beide Programme auf die gleiche Bildsequenz angewandt und jeweils der Schwerpunkt sowie die Kovarianz der Partikelwolken zu jedem Zeitschritt ausgelesen. Ein Plot der Trajektorien des jeweiligen Systems in der X-Z-Ebene erlaubt die Beurteilung der Abweichung in der Positionsbestimmung des einfacheren Ansatzes. Zusätzlich wäre ein Vergleich der Verläufe der Streuung der Partikelwolken über die Zeit hinweg hilfreich, um eine qualitative Aussage zu diesem Experiment

treffen zu können. Aus zeitlichen Gründen wurde von der Durchführung dieses Experiments abgesehen.

5.3 Zusammenfassung der Experimente

Die beschriebenen Experimente haben gezeigt, dass der entwickelte Prototyp mit den in der Modellierung getroffenen Einschränkungen im Prozess- und Messmodell unter gewissen Bedingungen eine erfolgreiche Objektverfolgung ermöglicht.

Die Vermutung, ohne große Genauigkeitsverluste nur eine bestimmte Anzahl diskreter Skalierungsstufen zu verwenden, hat sich in einem ersten Experiment bestätigt. Es wurde zum einen deutlich, dass der Klassifikator nur in einer stark begrenzten Umgebung um ein Gesicht in benachbarten Skalierungsstufen angeschlagen hat. Außerdem wurde gezeigt, dass die Genauigkeit bei einer höheren Anzahl als etwa 15 bis 20 Skalierungsstufen nicht steigt, da die Treffererate dann nahezu konstant bleibt.

In einem zweiten Experiment wurde untersucht, wie stark sich die Rotation eines Gesichts in der Bildebene negativ auf die Ergebnisse der Klassifikation auswirken. Dabei wurde deutlich, dass bereits ein Rotation von etwa 20º dazu führen kann, dass ein Gesicht nicht zuverlässig als solches erkannt wird. Im protoypischen Einsatz stellt diese Erkenntnis jedoch kein Problem dar. Um einen gegenüber einer derartigen Rotation robusten Klassifikator anzulernen, könnte man einen Trainigsdatensatz erstellen, in dem alle Gesichtsbeispiele auch in rotierten Varianten vorliegen. Beim Anlernen könnte man zusätzlich in [LKP03] vorgeschlagene, um 45º rotierte Merkmale hinzuziehen.

Die Anzahl der Partikel ist wesentlich für eine zuverlässige Objektverfolgung. Ein weiteres Experiment wurde durchgeführt, um zu ermitteln, wie groß diese sein muss. Es zeigte sich, dass ab etwa 5000 Partikeln die Verlässlichkeit ausreichend war, auch. Dennoch wurde das Gesicht in der Testsequenz erst etwa 100 Bilder später gefunden, als in einem Durchlauf mit 10000 Partikeln. Aufgrund der erhöhten Verarbeitungszeit bei steigender Partikelanzahl wurde als guter Kompromiss zwischen Genauigkeit der Objektverfolgung und Echtzeitfähigkeit eine Anzahl von 7000 Partikeln ermittelt.

Schließlich wurde ein Experiment durchgeführt, das den Zusammenhang der Anzahl der Partikel und der gewählten Gewichtung der als Gesicht klassifizierten Partikel anhand der sich dadurch verändernden Standardabweichung verdeutlichen sollte. Festzuhalten ist, dass die das Partikelfilter betreffenden Experimente exemplarischer Natur sind, da Parti-

kelfilter nicht deterministisch sind. Dennoch können anhand von ihnen Tendenzen erkannt werden.

Um mit dem entwickelten Prototyp Gesichter erfolgreich zu verfolgen, hat es sich bewährt, das System mit 7000 Partikeln bei einem Gewicht c des Messmodells (siehe Gleichung 4.8) von 0,9995 zu konfigurieren. Die Komponenten der Standardabweichung σ des Prozessmodells wurden jeweils auf 0,1 Meter gesetzt. Bei einem maximalen Abstand von der Kamera von sechs Metern wurden 14 Skalierungsstufen verwendet.

6 Zusammenfassung

In dieser Arbeit wurde ein System vorgestellt, das es ermöglicht, eine Person in einem begrenzten Raum in Echtzeit über ein Stereokamerasystem im dreidimensionalen Raum zu verfolgen. Das Hauptaugenmerk wurde auf Robustheit und Echtzeitfähigkeit gelegt, um dem Einsatzziel in einem interaktiven multimedialen Kontext zu entsprechen. Aufgrund dieser Anforderungen fiel die Wahl auf ein Partikelfilter, dessen Beobachtungsmodell auf einer mittels AdaBoost angelernten Klassifikator-Kaskade basiert.

Nach einer allgemeinen Übersicht über die Kategorisierung verschiedener Techniken zur Erkennung von Gesichtsregionen in Bildern wurden zunächst die Grundlagen der verwendeten Verfahren beschrieben.

Aufgrund der starken Einschränkungen in der Modellierung aus Gründen der Erreichung der Echtzeitfähigkeit des Systems wurde eine Reihe von Experimenten entworfen, anhand derer Informationen über die Robustheit ermittelt werden können. Aus zeitlichen Gründen wurden nur einige besonders interessante von ihnen durchgeführt. Anhand der Ergebnisse konnten jedoch manche Designentscheidungen gerechtfertigt und Eindrücke aus ersten Tests belegt werden.

Probleme bereitet das insbesondere das Beobachtungsmodell des Partikelfilters, da der Gesichtsklassifikator als dessen Kernbestandteil nur im Fall von aufrechten Gesichtern zuverlässige Ergebnisse liefert. Dies ließe sich lösen, wenn man einen Klassifikator nach Li et al. in [LZZ+02] verwenden würde, der auf dem in dieser Arbeit vorgestellten Algorithmus nach Viola et al. basiert. Rotationen in der Bildebene von werden durch das Anwenden einer Detektor-Pyramide auf das Originalbild sowie um $\pm 30º$ rotierte Kopien abgedeckt. Außerdem modellieren sie Drehungen von Gesichtern aus der Bildebene heraus, indem sie einen Klassifikator mittels *FloatBoost* – einer Abwandlung des AdaBoost Algorithmus – anlernen. Dabei enthält der Trainingsdatensatz Gesichter, die in Schritten von zehn Grad rotiert wurden. Einen anderen Ansatz zur robusteren Modellierung von Drehungen von Gesichtern aus der Bildebene heraus wurde in [KSS+06] vorgestellt. Hier werden mehrere Klassifikatoren für jeweils unterschiedliche Ansichten mittels AdaBoost angelernt und zur Laufzeit ausgewählt.

Für den Anwendungsfall in multimedialen Räumen wäre es von Interesse, mehrere Personen gleichzeitig verfolgen zu können. Zum einen könnte eine solche Lösung auf einem Partikelfilter mit *stratified sampling* [TBF05] basieren, das die Partikelwolke in *Cluster* unterteilt, und aus diesen etwa nach *low variance sampling* die a-posteriori-Verteilung generiert.

Einen anderen Ansatz stellten Vermaak, Doucet und Peres in [VDP03] vor, indem sie für jedes zu verfolgende Objekt einen eigenen Partikelfilter erzeugten, und deren Verteilungen aufgrund der Partikelgewichtung zu einem *mixture particle filter* kombinierten.

Abbildungsverzeichnis

Literatur

[Alp08] Ethem Alpaydin. *Maschinelles Lernen*. Oldenbourg Wissenschaftsverlag, 2008.

[Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, August 2006.

[DdFG01] Arnaud Doucet, Nando de Freitas, and Neil Gordon. An introduction to sequential monte carlo methods. In Arnaud Doucet, Nando De Freitas, and Neil Gordon, editors, *Sequential Monte Carlo methods in practice*. Springer-Verlag, 2001.

[FHL+03] Dieter Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2(3):24–33, July-September 2003.

[FS95] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of online learning and an application to boosting. In *EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory*, pages 23–37, London, UK, 1995. Springer-Verlag.

[HL01] E. Hjelmås and B.K. Low. Face detection: A survey. *Computer Vision and Image Understanding*, 83(3):236–274, September 2001.

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

[JL05] Anil K. Jain and Stan Z. Li. *Handbook of Face Recognition*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[KF00] Oliver D. King and David A. Forsyth. How does condensation behave with a finite number of samples? In *ECCV '00: Proceedings of the 6th European Conference on Computer Vision-Part I*, pages 695–709, London, UK, 2000. Springer-Verlag.

[KSS+06] Yoshinori Kobayashi, Daisuke Sugimura, Yoichi Sato, Kousuke Hirasawa, Naohiko Suzuki, Hiroshi Kage, and Akihiro Sugimoto. 3d head tracking using the particle filter with cascaded classifiers. In *In: BMVC (2006)*, 2006.

[LKP03] Rainer Lienhart, Alexander Kuranov, and Vadim Pisarevsky. Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In *Pattern Recognition*, volume 2781/2003 of *Lecture Notes in Computer Science*, 2003.

[LZZ+02] Stan Z. Li, Long Zhu, ZhenQiu Zhang, Andrew Blake, HongJiang Zhang, and Harry Shum. Statistical learning of multi-view face detection. In *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part IV*, pages 67–81, London, UK, 2002. Springer-Verlag.

[PCD08] Nicolas Pinto, David D Cox, and James J DiCarlo. Why is real-world visual object recognition hard? *PLoS Comput Biol*, 4(1):e27, Jan 2008.

[Res08] Point Grey Research. `http://www.ptgrey.com/products/bumblebee2/`, 2008.

[SAG+05] Peter Shirley, Michael Ashikhmin, Michael Gleicher, Stephen R Marschner, Erik Reinhard, Kelvin Sung, William B Thompson, and Peter Willemsen. *Fundamentals of Computer Graphics*. AK Peters, July 2005.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[VDP03] J. Vermaak, A. Doucet, and P. Perez. Maintaining multimodality through mixture tracking. *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1110–1116 vol.2, Oct. 2003.

[VJ01] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1, pages 511–518, 2001.

[Web02] Andrew R. Webb. *Statistical Pattern Recognition*. John Wiley and Sons Ltd., 2 edition, 2002.

[YKA02] Ming-Hsuan Yang, David J. Kriegman, and Narendra Ahuja. Detecting faces in images: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(1):34–58, 2002.