Erkennen von Kopfgesten in Videosequenzen

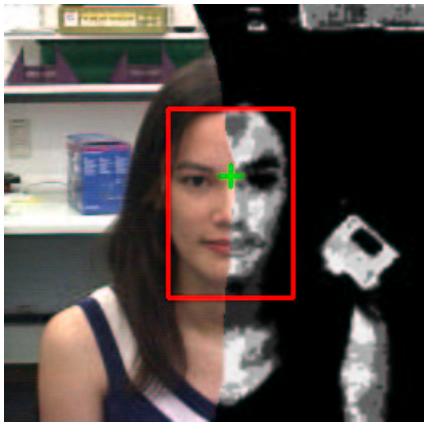
Lindl, Schwartz, Walchshäusl [1]
lindl@in.tum.de, schwartz@in.tum.de, walchsha@in.tum.de

20. Dezember 2002

[1]Frank Althoff, Gregor McGlaun, Prof. Dr. Jayanta Mukherjee

Zusammenfassung

Diese Arbeit ist aus einem Interdisziplinären Projekt am Lehrstuhl Mensch Maschine Kommunikation der Technischen Universität München hervorgegangen. Sie befasst sich mit der Erkennung von Kopfgesten in Videosequenzen. Der eigentlichen Klassifizierung geht eine Kopflokalisierung und Merkmalsextraktion voraus.



Inhaltsverzeichnis

Kapitel 1

Einleitung

Der Mensch verfügt über eine Vielzahl an Möglichkeiten mit seiner Umwelt zu kommunizieren. Das Ziel der Mensch-Maschine-Kommunikation ist es, eine möglichst intuitive und natürliche Kommunikation mit der Technik zu realisieren. Hierfür versucht man alle vorhandenen Kommunikationsmöglichkeiten in optimaler Weise auszuschöpfen.

Die klassischen Eingabemedien für ein Rechensystem sind Tastatur und Maus. Auch wenn Sie eine effiziente Eingabeform darstellen, entsprechen sie nicht dem natürlichen Kommunikationsverhalten des Menschen. Deshalb versucht man Sprache, Schrift, Mimik und Gestik einzubeziehen, um die Interaktion natürlicher zu gestalten. Für Spracheingabe existieren eine Vielfalt von Verfahren, die auch in der Praxis Verwendung finden. Wie zum Beispiel Diktiersysteme, telephonische Fahrplanauskunftssysteme oder Sprachsteuerung im Automobil oder Mobilfunkbereich. Vor allem für den mobilen Einsatz werden Schrifterkennungssysteme verwendet, wie in der Bedienung von Personal Digital Assistants (PDAs). Gestik und Mimik finden zur Zeit kaum praktische Anwendung.

Eine für den Menschen sehr intuitive Form des Informationsaustauschs sind Gesten. Hier sind vor allem zwei Arten von Gesten geeignet zur Interaktion mit einem Rechner. Hand- und Armgesten sowie Kopfgesten, wobei hier die Mimik gesondert betrachtet wird. Gestik als Eingabeform bietet sich vor allem in Bereichen an, wo Spracheingaben wegen zum Beispiel erhöhter Geräuschkulisse nicht zum Einsatz kommen kann, aber auch die Hände nicht zur Eingabe verwendet werden können (zum Beispiel im Automobil).

Im folgenden wird die Extraktion und Erkennung von Kopfgesten in Videosequenzen untersucht.

Kapitel 2

Übersicht über bestehende Verfahren

In diesem Kapitel geben wir einen Überblick über bereits vorhandene Verfahren zur Segmentierung (siehe Kapitel 2.1), zur Objektverfolgung (siehe Kapitel 2.2 auf Seite 15) und zur Klassifizierung (siehe Kapitel 2.3 auf Seite 21). Dieser Überblick beinhaltet sowohl den theoretischen Hintergrund der einzelnen Verfahren als auch deren Bewertung. Zur Evaluierung wurden diese prototypisch implementiert.

2.1 Segmentierung

Unter Segmentierung versteht man im Allgemeinen das Partitionieren eines Bildes in Bereiche, die für das weitere Vorgehen relevant sind, wie zum Beispiel Vordergrund und Hintergrund.

2.1.1 Mustererkennung

Die Hough-Transformation

Die Hough-Transformation dient der Mustererkennung in Graustufenbildern. Als Muster kommen alle mathematisch definierbaren Objekte wie z.B. Geraden, Kreise oder Ellipsen in Frage. Das Ausgangsbild wird durch die Hough-Transformation in den Parameterraum abgebildet. Dieser ist n-dimensional, wobei n der Anzahl der Freiheitsgrade des Suchmusters entspricht. Bei Geraden zum Beispiel genügen zwei Dimensionen. Der Steigungswinkel der Geraden und deren Verschiebung zum Ursprung. Im Parameterraum korrespondiert der Wert jeden Punktes mit der Wahrscheinlichkeit, dass sich das Suchmuster an jener Position befindet.

Vorgehen

Voraussetzung einer Extraktion von Gesichtsmerkmalen, ist die Kenntnis der genauen Kopfposition, da der Bildhintergrund viele zu Gesichtsmerkmalen ähnliche Berei-

Abbildung 2.1: Eingabebild

che enthalten kann. Um in einem Gesicht einzelne Merkmale mit Hilfe der Hough-Transformation zu ermitteln müssen zuerst mathematische Muster der jeweiligen Merkmale erstellt werden. Als Suchmerkmal bieten sich die Augen an, weil diese durch einfache Kreise repräsentiert werden können. Für ein Kreismuster genügen drei Parameter, da Rotationen nicht berücksichtigt werden müssen. Zwei Parameter entfallen für die Mittelpunktsposition und ein Parameter für den Kreisradius. Der zugehörige Parameterraum hat somit drei Dimensionen.

Zuerst wird aus dem Ausgangsbild (siehe Abbildung 2.1) ein Kantenbild erzeugt mit dem Canny-Filter (siehe Abbildung 2.2).

Abbildung 2.2: Bild mit angewendetem Canny-Filter

Das Kantenbild wird durch die Hough-Transformation in den Parameterraum abgebildet (siehe Abbildung 2.3 auf der nächsten Seite).

Im Parameterraum werden die Positionen der lokalen Maxima ermittelt, die Kandidaten für Augen sind. Anschließend müssen durch weitere Schritte (z.B. durch Lagevergleiche) die wahren Augenpositionen aus den Kandidaten ermittelt werden.

Bewertung

Vorteile der Hough-Transformation liegen in der Robustheit gegenüber Hintergrundrauschen und der Tolerierung von Unterbrechungen in den Konturen.

Abbildung 2.3: Auf Ebene projiziertes Bild des Parameterraum

Nachteile ergeben sich bei sehr detailreichen Strukturen, in denen viele Kanten enthalten sind. Diese verursachen meist eine hohe Kongruenz mit dem Suchmuster obwohl keine visuellen Ähnlichkeiten vorhanden sind.

Weiterer Nachteil ist der sehr hohe Rechenaufwand schon bei relativ wenigen Freiheitsgraden des Suchmusters. Die Dimensionen des Parameterraums entsprechen gleichzeitig dem Grad des Polynoms das die Laufzeit-Komplexität angibt. Im Fall der Kreise als Suchmuster ergibt sich somit eine Komplexität von $O(n^3)$. [8] [11] [9]

2.1.2 Lokalisierung der Augen durch den Lidschlag

Der Lidschlag ist ein kurzzeitiges Verschließen des Auges durch die Augenlider, das dadurch erneut von einem Tränenfilm überzogen wird. Daneben unterscheidet man noch den reflektorischen Lidschlag als Schutzreflex des Auges und den willkürlichen und unwillkürlichen Lidschlag. Hierbei schwankt die Frequenz von 2 bis 25 Lidschlägen pro Minute, wobei ein Lidschlag zwischen 200 ms und 600 ms je nach Müdigkeit des Menschen dauern kann. [6]

Vorgehen

Das Öffnen und Schließen des Augenlides erzeugt folglich bei einer Bildfrequenz von 25 Hz eine signifikante Änderung in zwei aufeinanderfolgenden Frames. Diese Differenz kann zur Lokalisierung der Augen verwenden werden. Im optimalen Fall ist während des Lidschlages der restliche Körper vor allem aber der Kopf und der Hintergrund in Ruhe. Somit ergibt sich im Differenzbild zweier Frames nur ein Ausschlag an den Stellen der Augen. Mit einem geeigneten Schwellwert werden nicht signifikante Störungen unterdrückt (siehe Abbildung 2.4 auf der nächsten Seite).

Die Zusammenhangskomponenten im bereinigten Differenzbild bilden die Menge aller möglichen Augenkandidaten. Durch biometrische Verfahren über Eigenschaften wie die Größe und die Form, sowie über die Lage der einzelnen Zusammenhangskomponenten zueinander kann man die Güte der Augenkandidaten bestimmen. Idealerweise erhält man zwei Bereiche, in denen sich mit großer Wahrscheinlichkeit die Augen befinden.

Abbildung 2.4: Differenzbild nach Threshold bei einem optimalen Lidschlag

Abbildung 2.5: Differenzbild nach Threshold mit viel Bewegung

Bewertung

Dieser Ansatz ist nur bedingt für das Segmentieren von Köpfen bzw. Augen geeignet. Unter optimalen Bedingungen können zwar gute Ergebnisse erzielt werden, sobald sich aber der Mensch während des Lidschlages bewegt oder der Hintergrund nicht unveränderlich ist, entstehen starke Differenzen zwischen den Folgebildern, was zu einer großen Anzahl von Augenkandidaten führt. Wenn die Bewegung zu großflächig wird, ist es möglich, dass das komplette Differenzbild zu einer einzigen großen Zusammenhangskomponente zusammenwächst. Beide Effekte machen eine robuste Lokalisierung der Augen aus diesen Differenzbildern unmöglich.

Des weiteren muss zwischen Differenzbildern, die durch normale Bewegung entstanden sind, und zwischen Differenzbildern, die tatsächlich durch einen Lidschlag verursacht worden sind, unterschieden werden (siehe Abbildung 2.5). Es ist schwierig hierfür eine Heuristik zu finden, die robust aus der Menge der möglichen Augenkandidaten die Augen auswählt. Allein die Größe, Form und Lage der einzelnen Zusammenhangskomponenten relativ zueinander ist hier nicht ausreichend. Durch die geringe Frequenz des Lidschlages kann es unter Umständen zu inakzeptabel langen Initialisierungsphasen des Systems kommen, weil nur ein optimaler Lidschlag ausgewertet werden kann.

2.1.3 Eigenfaces

Dieses Verfahren geht auf die Arbeit „Eigenfaces for Recognition" von Turk und Pentland zurück [10]. Grundlage der Eigenface-Methode ist die *Principal Component Analysis* (PCA) . Sie wird zur Dimensionsreduzierung von Vektorräumen eingesetzt. Um PCA anwenden zu können, muss das Bild als ein einziger Vektor der Dimension $B \times H$ angesehen werden, wobei der Grauwert der einzelnen Pixel des Bildes jeweils eine Komponente des Vektors repräsentiert. Der gesamte Vektorraum der Dimension $B \times H$ beinhaltet somit alle möglichen Bilder der Breite B und der Höhe H. Bei den gebräuchlichen Bildauflösungen führt diese Betrachtungsweise unweigerlich zu sehr großen Vektorräumen, die nicht mehr effizient behandelt werden können. Daher versucht man mit der PCA einen Untervektorraum zu bestimmen, der den gesamten Vektorraum gut approximiert. Da sich alle Gesichter in gewisser Weise sehr ähnlich sind, liegen die meisten Punkte der Vektoren relativ nahe zusammen und können somit mit einem kleinen Untervektorraum gut approximiert werden. Die Basisvektoren, die diesen Untervektorraum aufspannen, werden *principal components* genannt. Prinzipiell hängt die Dimension der Basis von der Ähnlichkeit der zu untersuchenden Vektoren und von der geforderten Güte der Approximation ab und kann nicht a priori bestimmt werden.

Vorgehen

Mittels der PCA wird versucht einen Untervektorraum für ein Trainingsset von Gesichtern zu bestimmen. Die Trainingsdaten sollten alle Facetten von möglichen Gesichtern abdecken, damit der Unterraum aussagekräftig genug ist, um unbekannte Gesichter zu lokalisieren. Aus den Eigenvektoren der Kovarianzmatrix der Trainingsdaten lassen sich die *principal components* wie folgt berechnen:
Sei $T = \{\vec{g^1}...\vec{g^m}\}$ die Trainingsdatenmenge wobei $m = |T|$ und $\vec{g^i}$ der Vektor des einzelnen Trainingsbildes ist und $n = B \times H$ der Bildgröße entspricht, dann ist

$$\vec{D_j} = \frac{1}{m} \sum_{i=1}^{m} \vec{g_j^i} \qquad \text{wobei } j = 1..n$$

der Durchschnittsvektor und

$$\vec{\Delta^i} = \vec{g^i} - \vec{D}$$

die Abweichung der einzelnen Trainingsbilder vom Durchschnittsbild. Dann berechnet sich die Kovarianzmatrix wie folgt:

$$C = \mathbb{E}\{ZZ^T\} \qquad \text{wobei } Z = \left(\vec{\Delta^1} \quad \dots \quad \vec{\Delta^m}\right)$$

Die Kovarianzmatrix ist stets symmetrisch und hat die Größe $n \times n$. Somit sind alle Eigenvektoren orthogonal zueinander und bilden eine Basis. Da in der Praxis $m \ll n$ ist, kann man statt den Eigenwerten von C auch die von $\grave{C} = Z^T Z$ berechnen, wobei man zeigen kann, daß alle Eigenwerte von $\grave{C}$ auch Eigenwerte von C sind und umgekehrt.

„⇒"

$\grave{C}\vec{e_i} = \lambda_i \vec{e_i}$ Eigenwertbedingung von $\grave{C}$

$X^T X \vec{e_i} = \lambda_i \vec{e_i}$

$X X^T X \vec{e_i} = \lambda_i X \vec{e_i}$

$C\vec{e_i} = \lambda_i \vec{e_i}$

„⇐"

$C\vec{e_i} = \lambda_i \vec{e_i}$ Eigenwertbedingung von C

$X X^T \vec{e_i} = \lambda_i \vec{e_i}$

$X^T X X^T \vec{e_i} = \lambda_i X^T \vec{e_i}$

$\grave{C}\vec{e_i} = \lambda_i \vec{e_i}$

Um die Dimension des Unterraumes weiter zu reduzieren kann man diejenigen Eigenvektoren vernachlässigen, die einen sehr kleinen Eigenwert haben, da sie sich nicht mehr signifikant auswirken (siehe Abbildung 2.6).

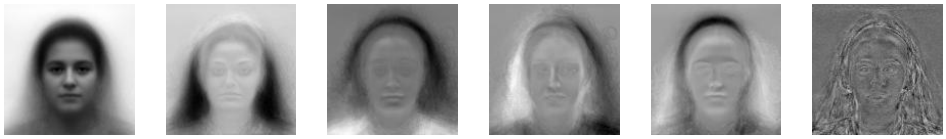


Abbildung 2.6: Eigenfaces von Eigenvektoren mit unterschiedlicher Ordnung (1,2,3,4,5,140) [7]

Aus den Basisvektoren ergeben sich die Eigenfaces nach folgender Formel:

$\vec{b^i} = X\vec{e_i}$ wobei $\vec{e_i}$ der i-te Eigenvektor ist

All diese Eigenfaces spannen nun einen neuen Vektorraum nämlich den *Facespace* F auf.

$F = < \vec{b_0}, ..., \vec{b_m} >$

Dieser *Facespace* wird nun dazu verwendet um Gesichtsregionen in Bildern zu lokalisieren. Dazu wird zuerst das zu untersuchende Bild in den *Facespace* projiziert. Unterscheidet sich die Projektion nur gering von dem Urbild, dann war das Ausgangsbild sehr wahrscheinlich ein Gesicht (siehe Abbildungen 2.7 und 2.8 auf der nächsten Seite). Unterscheidet sich die Projektion nur sehr gering zu einem Trainingsbild aus der Datenbank, dann wurde das Trainingsbild erkannt.

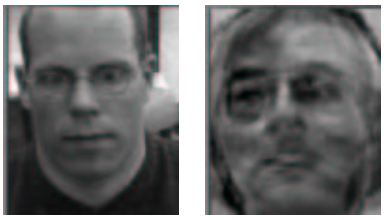


Abbildung 2.7: Urbild und Projektion des Urbildes in den *Facespace*, wobei das Urbild nicht Element der Trainingsmenge war

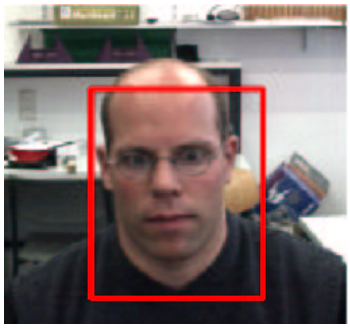


Abbildung 2.8: Ergebnis der Gesichtslokalisation

Trainingsbild (a) Urbild (b) Projektion (c)

Abbildung 2.9: Urbild und Projektion des Urbildes in den *Facespace*, wobei das Urbild Element der Trainingsmenge war

Bewertung

Eigenfaces eignen sich nur dann sehr gut zur Lokalisierung von Gesichtern in Bildern, wenn bestimmte Vorraussetzungen erfüllt sind. Sehr wichtig ist hier eine gute Trainingsmenge, die möglichst viele unterschiedliche Personen beinhalten sollte. Da Eigenfaces weder rotations-, beleuchtungs- noch skalierungsinvariant sind, müsste das Trainigsset zusätzlich noch verschiedene Blickwinkel, Beleuchtungsverhältnisse und Kopfgrößen beinhalten, um diesen Nachteil auszugleichen. Dies läßt sich nur durch sehr großen Aufwand erreichen und würde den Rechenaufwand erhöhen, da die Di-

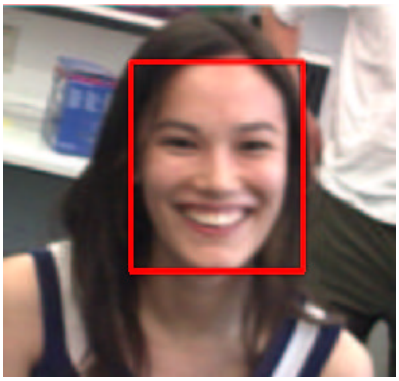


Abbildung 2.10: Ergebnis der Gesichtslokalisation

mension des *Facespace* umso größer wird, je unterschiedlicher die Gesichter sind. Der größte Nachteil des Verfahrens ist aber die sehr schlechte Laufzeit. Um nur ein einziges Bild auf vorhandene Gesichter zu überprüfen, werden schon mehr als 30 Sekunden benötigt. Eine echtzeitfähige Lösung ist somit, auf Basis der OpenCV 0.9.3 Bibliothek, nicht realisierbar.

2.1.4 Hintergrunderkennung

Ein simpler Ansatz, um aus einem Kamerabild den Kopf einer Person zu extrahieren, ist den Hintergrund auszumaskieren. Dann würde man davon ausgehen, dass der Teil des Bildes, welcher noch übrig bleibt der Kopf des Nutzers ist. Hierfür ist als Vorbedingung ein statischen Hintergrund, vorzugsweise einfarbig schwarz, von Nöten. Dies ist jedoch nur unter Laborbedingungen zu gewährleisten, und zieht weitere Probleme nach sich, wie Konfusion einer Kamera, die einen automatischen Weißabgleich durchführt.

Um den Ansatz auch bei problematischeren Gegebenheiten weiterzuverfolgen, muss man sein Modell des Hintergrunds fortlaufend anpassen. Dies geschieht, indem man ein Akkumulatorbild hält, in welches langsame Änderungen des Hintergrunds (wie Fluktuation in der Beleuchtung, kleinere Umstellungen etc) aufgenommen werden. Hierzu wird, nachdem ein anfängliches Hintergrundmodell vorhanden ist, ein einfacher adaptiver Filter bei der Einbindung neuer Bilder angewandt. Sei μ_t der Wert des Hintergrundmodells zur Zeit t und y das aktuelle Eingabebild dann ist die Filterfunktion durch

$$\mu_t = \alpha y_t + (1 - \alpha)\mu_{t-1}$$

gegeben, wobei α die Mittelungskonstante ist. Sie wird im allgemeinen auf kleine Werte wie 0.05 gesetzt, so dass sich Bewegungen im Vordergrund des Bildes nicht auf das Hintergrundmodell auswirken.

Bewertung

Dieser Ansatz erweist sich jedoch für Gestenerkennung als nicht praktikabel, da wegen der Ruhephasen und der relativ statischen Position des Kopfes, die Information über den Hintergrund im Kopfbereich mit der Zeit verloren geht. Beziehungsweise der Kopf wird durch das Hintergrundmodell assimiliert.

2.1.5 Farbbasierte Segmentierung

Als Grundlagen dieses Verfahrens dient ein Basishistogramm des zu segmentierenden Objektes, das vor schwarzem Hintergrund erstellt wird. Um das Objekt zu lokalisieren wird das normierte Basishistogramm mit dem Histogramm des Eingabebildes multipliziert. Nach der Rückprojektion des neuen Histogramms sind alle Bereiche des Bildes, die nicht im Farbspektrum des Objektes lagen, ausgeblendet.

Vorgehen

Zur Detektion von Gesichtern in Farbbildern bietet sich die menschliche Hautfarbe an. Im Folgenden wird versucht ein möglichst dichtes und exaktes Basishistogramm der Hautfarbenverteilung zu generieren. Da die Farbe menschlicher Haut auch bei verschieden dunklen Hauttypen, hauptsächlich in der Luminanz variiert, liegen alle Hautfarben nur in einem kleinen Bereich der CbCr-Ebene des YCbCr Farbraums . Der Luminanzkanal kann deshalb im weiteren ignoriert werden. Die Abbildung des RGB-Farbraumes in den YCbCr Farbraum berechnet sich wie folgt:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Um die Verteilung innerhalb CbCr zu bestimmen, sind wir davon ausgegangen, dass die Hautfarben normalverteilt sind. Zur Vereinfachung der Verteilung bietet sich daher eine Approximation durch folgende Gaußfunktion an (siehe Abbildung 2.11 auf der nächsten Seite). Um deren Parameter zu bestimmen, wurden aus 40 Stichproben von unterschiedlichen Hauttypen der Mittelwert und die Kovarianzmatrix berechnet.

Mittelwert:

$$\vec{m} = \mathbb{E}\{\vec{x}\} \text{ wobei } \vec{x} = \begin{pmatrix} cr \\ cb \end{pmatrix} \tag{2.1}$$

Kovarianzmatrix:

$$C = \mathbb{E}\{(\vec{x} - \vec{m})(\vec{x} - \vec{m})^T) \tag{2.2}$$

Die Gaußverteilung berechnet sich nun mittels

$$Pr(cr, cb) = \exp[-0.5(\vec{x} - \vec{m})^T C^{-1} (\vec{x} - \vec{m})] \tag{2.3}$$

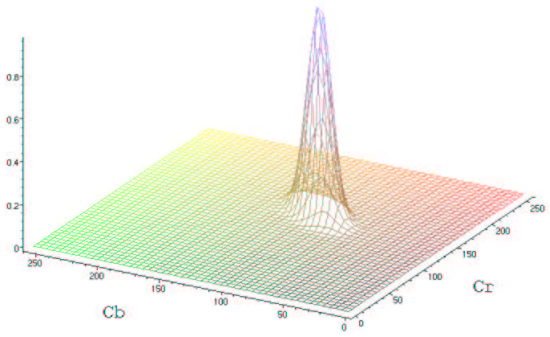


Abbildung 2.11: Approximierte Farbverteilung der Hauttypen in der CbCr-Ebene

Das Histogramm des CbCr-Anteils des Eingangsbildes (siehe Abbildung 2.12 (a)) wird mit der Gaußfunktion multipliziert, um nichthautfarbene Bereiche herauszufiltern. Mit dem daraus entstandenen Histogramm, wird das CbCr-Bild in ein Graustufenbild projiziert. Als Ergebnis erhalten wir ein 8 bit Graustufenbild, (siehe Abbildung 2.12 (b)) in dem jeder Hautfarbton durch einen Grauwert repräsentiert wird, je nach Funktionswert der Verteilung an der Stelle $Pr(cr, cb)$.

Ausgangsbild (a)

Projektion mit neuem Histogramm (b)

Abbildung 2.12: Ergebnis der Hautfarbensegmentierung

Bewertung

Für farbbasierte Verfahren existieren schnelle Implementierungen. Sie sind robust auch bei bewegten und fein strukturierten Hintergrundflächen. Die Orientierung und Skalierung der zu erkennenden Objekte ist irrelevant. Probleme hingegen treten bei Objekten wie zum Beispiel Holz, blonde Haare, Textilien auf, deren Farbe ähnlich dem ermittelten Hautfarbspektrum ist. Diese werden fälschlich als Haut klassifiziert .

2.1.6 Zusammenfassung

Verfahren	Bewertung
Hough-Transformation (2.1.1)	⊖ schlechte Erkennungsraten bei fein strukturiertem Hintergrund ⊖ hohe Leistungsanforderungen an die Hardware ⊕ beleuchtungs- und rotationsinvariant
Lokalisierung der Augen durch den Lidschlag (2.1.2)	⊖ schlechte Ergebnisse bei starken Bewegungen bzw. Beleuchtungsänderungen ⊖ lange Initialisierungszeit ⊕ sehr schnell zu berechnen
Eigenfaces (2.1.3)	⊖ passender, genügend großer Trainingskorpus ⊖ komplexer, ineffizienter Algorithmus ⊕ auch zur Gesichterkennung geeignet
Hintergrunderkennung (2.1.4)	⊖ Hintergrund muss weitgehend statisch sein ⊖ ausreichend Dynamik im Vordergrund notwendig ⊕ einfaches, effizientes Verfahren
Farbbasierte Segmentierung (2.1.5)	⊖ Fehlklassifizierung von Objekten, die ein ähnliches Spektrum wie Hautfarbe aufweisen ⊖ Probleme bei verdeckten oder nicht optimal beleuchteten Gesichtern ⊕ rotations- und skalierungsinvariant ⊕ weitgehend unabhängig vom Hautfarbentyp ⊕ schneller Algorithmus

Fazit

Da unsere Kopfgestenerkennung Echtzeitansprüchen genügen soll, fallen die Verfahren Hough-Transformation und Eigenfaces wegen zu langsamer Algorithmen weg.

Die Lokalisierung der Augen durch den Lidschlag hat sich als zu ungenau bei starken Bewegungen im Bild erwiesen. Man könnte zwar zur Initialisierung des Systems vom Benutzer ein bewusstes Blinzeln ohne Kopfbewegung abverlangen, was aber nicht sehr ergonomisch wäre.

Um durch die Hintergrunderkennung den Bildhintergrund erfolgreich vom Bildvor-

dergrund zu separieren muss der Hintergrund statisch und der Vordergrund dynamisch sein. Beides ist im allgemeinen nicht gegeben.

Bei der farbbasierten Segmentierung bleiben zwar immer noch einige Probleme bestehen, aber im Vergleich bieten sich doch die entscheidenden Vorteile. Sie ist rotations- und skalierungsinvariant und die Berechnung ist sehr schnell. Ausserdem benötigt sie keine Initialisierungsphase und ist auch gegen Bewegungen im Bildhintergrund immun.

Somit haben wir uns letztlich für dieses Verfahren zur Kopfsegementierung entschieden. (siehe auch Kapitel 3.1 auf Seite 32)

2.2 Tracking

Unter Tracking versteht man die räumliche und zeitliche Erfassung von sich bewegenden Objekten in Bildsequenzen. Alle Positionen des Objektes bilden die sogenannte Trajektorie .

2.2.1 Optischer Fluß

Optischer Fluss wird zur Verfolgung von Objekten eingesetzt. Dabei werden Luminanzänderungen in Bildsequenzen, als Bewegung interpretiert.

Einschränkungsgleichung

Sei $I(x, y, t)$ die Luminanz des Bildes an Position (x, y) zur Zeit t. Falls sich nun ein Objekt um (dx, dy) bewegt gilt folgende Gleichung, vorausgesetzt die Eigenhelligkeit des Objektes bleibt konstant:

$$I(x + dx, y + dy, t + dt) = I(x, y, t) \tag{2.4}$$

Die Taylorentwicklung der linken Seite ergibt

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt \tag{2.5}$$

nach Einsetzen in *Formel 2.4*, Subtraktion von $I(x, y, t)$ und Division durch dt erhält man:

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{2.6}$$

Mit $\vec{v} = \begin{pmatrix} \frac{\partial x}{\partial t} \\ \frac{\partial y}{\partial t} \end{pmatrix}$ ergibt sich die Einschränkungsgleichung

$$\nabla I \cdot \vec{v} + I_t = 0 \tag{2.7}$$

Die Einschränkungsgleichung ist ein unterbestimmtes lineares Gleichungssystem für die unbekannte Geschwindigkeit $\vec{v}$. Um dennoch Lösungen zu erhalten, existieren verschiedene Ansätze, wie zum Beispiel der Horn-Schunck-Algorithmus oder der Lucas-Kanade-Algorithmus .

Lucas-Kanade-Algorithmus

Der Lucas-Kanade-Algorithmus ist im Gegensatz zum Horn-Schunk-Alogrithmus eine lokale Methode, die nur in einem kleinen Fenster Ω operiert. Dabei werden die einzelnen Bildpunkte $\vec{m}$ unterschiedlich gewichtet. Punkte, die im Zentrum liegen erhalten größeres Gewicht. Wir definieren die Gewichtsfunktion $W(\vec{m})$ mit $\vec{m} \in \Omega$. Der Optische Fluss des Fenstermittelpunkts lässt sich nun wie folgt berechnen:

$$\min_{\vec{v}} E = \sum_{\vec{m}\in\Omega} W^2(\vec{m})(\nabla I \cdot \vec{v} + \frac{\partial I}{\partial t})^2 \tag{2.8}$$

Aufgeteilt und vereinfacht ergibt sich aus *Formel 2.8*:

$$\frac{\partial E}{\partial v_x} = \sum_{\vec{m}\in\Omega} W^2(\vec{m})(\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t})\frac{\partial I}{\partial x} = 0 \tag{2.9}$$

$$\frac{\partial E}{\partial v_y} = \sum_{\vec{m}\in\Omega} W^2(\vec{m})(\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t})\frac{\partial I}{\partial y} = 0 \tag{2.10}$$

Um das System zu lösen definieren wir:

$$A = \begin{pmatrix} \frac{\partial I_1}{\partial x_1} & \frac{\partial I_1}{\partial y_1} \\ \vdots & \vdots \\ \frac{\partial I_n}{\partial x_n} & \frac{\partial I_n}{\partial y_n} \end{pmatrix}_{n\times 2}$$

$$W = diag(W(\vec{m}_1), \ldots, W(\vec{m}_2))_{n\times n}$$

$$\vec{b} = -\begin{pmatrix} \frac{\partial I_1}{\partial t} \\ \vdots \\ \frac{\partial I_n}{\partial t} \end{pmatrix}$$

Nach Einsetzen erhält man:

$$A^T W^2 A\vec{v} = A^T W^2 \vec{b} \tag{2.11}$$

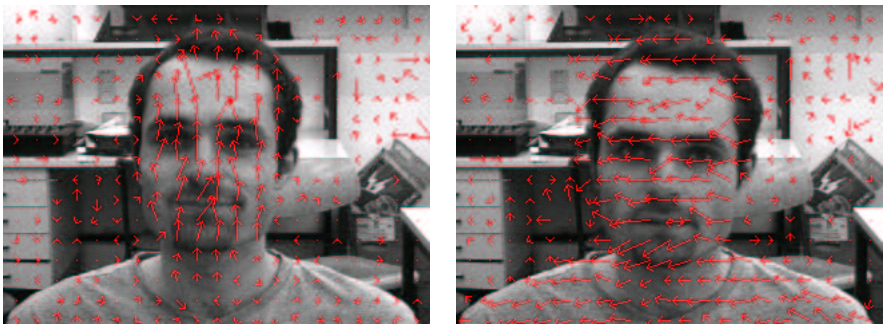
Der Optische Fluss für den Bildpunkt $\vec{m}$ ist somit:

$$\vec{v} = (A^T W^2 A)^{-1} A^T W^2 \vec{b} \tag{2.12}$$

Bewertung

Allein durch den gemittelten optischen Flusses lässt sich die Bewegungsrichtung in einer Bildsequenz feststellen. Damit lassen sich Gesichtsgesten wie Kopfnicken und Kopfschütteln eindeutig unterscheiden. (siehe Abbildung 2.13 auf der nächsten Seite) Leider genügt der Durchschnittsvektor allein nicht, um zwischen Kopfrotationen und Kopftranslationen zu differenzieren.

Die benötigte Rechenzeit hält sich in Grenzen, wenn man den Bildausschnitt auf einen kleinen Bereich beschränkt.



Kopfnicken (a) Kopfschütteln (b)

Abbildung 2.13: Optischer Fluss in Gesichtsgesten

2.2.2 Template Matching

Sind Muster in einem Bild schwer oder nicht robust durch ein normales Segmentationsverfahren zu extrahieren, oder ist das Muster gar zum Implementationszeitpunkt unbekannt bzw. ändert sich häufig, wird die Methode des Template Matching angewandt.

Allgemein gibt es für Template-Matching-Verfahren unterschiedliche Anforderungen. Zum einen könnte das Muster beliebig im Bild rotiert sein, daher besteht eine starre Abbildung zum Template . Das Muster könnte seine Größe ändern, wodurch man eine linearen Ähnlichkeitsabbildung erhält. In besonderen Fällen lässt man auch affine Abbildungen zu, wenn zum Beispiel eine Drehung des Objekts im Dreidimensionalen möglich ist.

Beim Templatematching wird das Objekt, welches zu finden ist, durch seine Grauwertmatrix beschrieben (dh. diese bildet das Muster bzw. Template). Um nun das Muster im Bild zu matchen, wird das Template über alle möglichen Positionen des Bildes geschoben und ein, weiter unten noch zu definierendes, Ähnlichkeitsmaß berechnet. Ist der Wert des berechneten Ähnlichkeitsmasses größer als ein gewisser, vom Benutzer festgelegter, Schwellenwert, so wird das Muster als erkannt angenommen.

Summe der absoluten Grauwertdifferenzen

Als Ähnlichkeitsmaß dient ein geeignetes Maß auf dem Bildraum, daher einer Abbildung von Template und Eingabebild auf $\mathbb{R}^+$. Sei $t(x, y)$ ein Template das in $\mathbb{R}$ definiert ist und $b(x, y)$ ein Bild dann erzeugen der Absolutbetrag der Grauwertdifferenzen ($n = |R|$)

$$a(x,y) = \frac{1}{n} \sum_{(u,v)\in R} |t(u,v) - b(x+u, y+v)|$$

ein übliches Ähnlichkeitsmaß. Intuitiv ist klar, dass um so kleiner das Ähnlichkeitsmaß, die Übereinstimmung zwischen dem gegebenen Bild und dem Template umso

besser ist.

Daher gilt folgendes Vorgehen um ein Muster in einem Bild aufzufinden. Das Ähnlichkeitsmaß wird über das gesamte Bild berechnet und der im letzten Abschnitt erwähnte Threshold auf das Ähnlichkeitsbild angewendet. Dann werden lokale in den übrigen Regionen die lokalen Minima bestimmt.

Das einfache Ähnlichkeitsmaß der Grauwertdifferenzen ist offensichtlich stark anfällig für Änderungen der Beleuchtung. Dies versucht man zu umgehen, indem man zum Beispiel Template und Bild auf ihre jeweiligen durchschnittlichen Grauwerte normalisiert, was jedoch bei multiplikativen Beleuchtungsänderungen keinen Erfolg verspricht.

Grauwertkorrelation

Ein besseres Ähnlichkeitsmaß ist die Grauwertkorrelation

$$c(x,y) = \sum_{(u,v)\in R} t(u,v)b(x+u,y+v)$$

Die Grauwertkorrelation nimmt, im Gegensatz zum Ähnlichkeitsmaß der Grauwertdifferenzen , einen hohen Wert bei hoher Übereinstimmung an. Das Problem der Grauwertkorrelation ist die Abhängigkeit vom Kontrast des Bildes. So kann es geschehen, dass ein Objekt, welches nicht passt, im Bild eine größere Korrelation hat, als ein gut passendes, da dessen Kontrast kleiner ist. Auch ist die Korrelation nicht beleuchtungsinvariant.

Der Vorteil der Grauwertkorrelation ist, dass sie durch Multiplikation entsteht und sowohl multiplikative als auch additive Beleuchtungsveränderungen ausgeglichen werden können, wenn die Grauwerte auf den mittleren Grauwert bezogen werden und mit ihrer Standardabweichung normiert werden. So ergibt sich

$$c(x,y) = \frac{1}{n} \sum_{(u,v)\in R} \frac{t(u,v) - m_t}{\sqrt{s_t^2}} \cdot \frac{b(x+u,y+v) - m_t(x,y)}{\sqrt{s_b^2(x,y)}}$$

wobei m_t der mittlere Grauwert des Templates ist, $m_b(x,y)$ der mittlere Grauwert des Bildes in der Maske um (x,y) und $\sqrt{s_t^2}$ beziehungsweise $\sqrt{s_b^2(x,y)}$ die entsprechenden Standardabweichungen sind. Die so genannte „normierte Korrelation" hat somit einen Wertebereich zwischen -1 und 1 einschließlich. Aus $c(x,y) = 1$ folgt $b(x+u,y+v) = at(u,v) + b$ mit $(a > 0)$ ebenso für $c(x,y) = -1$ folgt $b(x+u,y+v) = at(u,v) + b$ mit $(a < 0)$

Zur Effizienzsteigerung läßt sich die Berechnung der Grauwertkorrelation noch umformen:

$$\begin{aligned}
&\tfrac{1}{n} \sum_{(u,v)\in R} \tfrac{t(u,v)-m_t}{\sqrt{s_t^2}} \cdot \tfrac{b(x+u,y+v)-m_t(x,y)}{\sqrt{s_b^2(x,y)}} \\
&= \tfrac{1}{n} \frac{\sum_{(u,v)\in R} t(u,v)b(x+u,y+v) - m_t m_b(x,y)}{\sqrt{s_t^2 s_b^2(x,y)}} \\
&= \frac{n \sum_{(u,v)\in R} t(u,v)f(x+u,y+v) - \sum_{(u,v)\in R} t(u,v) \sum_{(u,v)\in R} b(x+u,y+v)}{\sqrt{\left(n \sum_{(u,v)\in R} t(u,v)^2 - \left(\sum_{(u,v)\in R} t(u,v)\right)^2\right)\left(n \sum_{(u,v)\in R} b(x+u,y+v)^2 - \left(\sum_{(u,v)\in R} b(x+u,y+v)\right)^2\right)}}
\end{aligned}$$

Nach dieser Umformung spart man sich $O(n)$ Subtraktionen.

Im direkten Vergleich zwischen dem Ähnlichkeitsmaß der Summe der Betragsdifferenzen und der Grauwertkorrelation stehen für die Summe der Betragsdifferenzen die schnelle Berechnung sowie, dass ein einfaches Abbruchskriterium gefunden werden kann, wenn der geforderte Schwellwert nicht mehr unterschritten werden kann (siehe unten). Dagegen ist der Berechnungsaufwand für die normierte Grauwertkorrelation hoch, aber sie ist im Gegensatz zur Summe der Betragsdifferenzen auch gegenüber multiplikativen Beleuchtungsänderungen invariant. Es lassen sich zwar auch Kriterien für den Abbruch der Berechnung finden, wann der geforderte Schwellwert nicht mehr überschritten werden kann, jedoch müssen entweder die Standardabweichung oder die Korrelation (die nicht normierte Korrelation) über das ganze Bild berechnet werden.

Optimierungen

Sei der gewählte Schwellwert für die Summe der Betragsdifferenzen w so gilt

$$a(x,y) = \frac{1}{n}\sum_{i=1}^{n} |t(u_i,v_i) - b(x+u_i, y+v_i)| \leq w$$

$$a(x,y) = \sum_{i=1}^{n} |t(u_i,v_i) - b(x+u_i, y+v_i)| \leq nw$$

und sei die Summe der ersten j Differenzen mit $\hat{a}_j(x,y)$ bezeichnet, so ist

$$a(x,y) = \hat{a}_j(x,y) + \underbrace{\sum_{i=j+1}^{n} |t(u_i,v_i) - b(x+u_i, y+v_i)|}_{\geq 0} \leq nw$$

Daher kann die Berechnung sicher abgebrochen werden, falls $\hat{a}_j > nw$.

Nun betrachten wir die normierte Grauwertkorrelation:

$$\hat{c}(x,y) = \sum_{i=1}^{n} \frac{t(u_i,v_i) - m_t}{\sqrt{s_t^2}} \cdot \frac{b(x+u_i, y+v_i) - m_b(x,y)}{\sqrt{s_b^2(x,y)}} \geq nw$$

Die Berechnung des Mittelwerts und der Standardabweichung für das Template kann natürlich unabhängig vom Bild geschehen und geht somit nicht in die Laufzeit mit ein. Normalerweise fordert man, damit ein Bild als erkannt gilt $c(x,y) \gg 0$. Um ein Abbruchkriterium bei bekanntem Schwellwert w zu finden gibt es zwei Ansätze.

Zum einen kann man die (unnormierte) Korrelation $c_u(x,y)$ vollständig über das Bild berechnen und ebenso den Mittelwert $m_b(x,y)$. Dann muss die Standardabweichung nur berechnet werden, falls $c_u(x,y) > 0$. Ein Abbruchkriterium für die Berechnung der Standardabweichung folgt aus der Umformung zu:

$$\sqrt{s_b^2(x,y)} \leq \frac{c_u(x,y) - m_t m_f(x,y)}{\sqrt{s_t^2} nw}$$

Zum anderen kann man den Mittelwert $m_b(x, y)$ und die Standardabweichung $s_b(x, y)$ über das ganze Bild berechnen, wobei sich dies bei rechteckigen Mustern, auf die man sich meist beschränkt effizient über rekursive Filter geschehen kann. Sei nun $\hat{c}_j(x, y)$ die Summe der ersten j Korrelationen:

$$\hat{c}(x,y) = \hat{c}_j(x,y) + \sum_{i=j+1}^{n} \underbrace{\frac{t(u_i,v_i) - m_t}{\sqrt{s_t^2}} \cdot \frac{b(x+u_i, y+v_i) - m_b(x,y)}{\sqrt{s_b^2(x,y)}}}_{\geq 0} \geq nw$$

Daher ist ein Abbruch auf jeden Fall möglich falls

$$\hat{c}_j(x,y) + (n-j) < nw \leftrightarrow \hat{c}_j(x,y) < (w-1)n + j$$

gilt.

Optimierung durch Skalierungen

Da die Berechnung des Ähnlichkeitsmasses über das ganze Bild schon für kleine Dimensionen sehr teuer ist, sucht man das Templatematching noch durch Skalierung in Bildpyramiden zu optimieren. Hierzu wird vom Bild und von dem Template eine Bildpyramide berechnet. Man zieht eine Bildpyramide mit Mittelwertfilter einer Gaußpyramide vor, da sich durch die Gaußfilterung das Bild von Ebene zu Ebene um 0.25 Pixel verschiebt. Die Filterungen sind notwendig, um Aliasingeffekte auszugleichen, die das Auffinden des Templates im Bild verhindern wuerden. Nun wird das Templatematching nur auf der kleinsten Ebene durchgeführt. Dies ergibt schon bei vier Ebenen eine Ersparnis vom Faktor 4092 da die Bilder um den Faktor $2^{4^2} = 64$ kleiner sind. Nun werden die die auf der obersten Ebene (mit einem vereinfachten Schwellwert) gefundenen Matches durch die Ebenen „nach oben" verfolgt, bis ein Match in der obersten Ebene gefunden wird.

2.2.3 Zusammenfassung

Verfahren	Bewertung
Optischer Fluss (2.2.1)	⊖ Verfälschung des Ergebnisses möglich, verursacht durch Helligkeitsänderungen im Bild ⊖ Mehrdeutigkeiten, da die Lösungen aus einem unterbestimmten Gleichungssystem hervorgehen ⊕ relativ robustes Ergebnis bei Mittelung über einen grösseren Bildausschnitt
Template Matching (2.2.2)	⊖ je nach Ähnlichkeitsmaß mehrere gute Übereinstimmungen an verschiedenen Postitionen im Bild möglich ⊖ Mehraufwand bei rotierten oder skalierten Mustern ⊕ beleuchtungsinvariant bei Graukorrelations-Ähnlichkeitsmaß ⊕ bei richtiger Initialisierung gute Objektverfolgung gewährleistet

Fazit

Durch das Template Matching lässt sich ein Ähnlichkeitswert zwischen einem Suchmuster und einem Bildausschnitt ermitteln. Das Template Matching benutzen wir zur Verifikation von Kopfkandidaten und zur Bestimmung der Nasenwurzelposition im Gesicht. (siehe auch Kapitel 3.2.1 auf Seite 36)

Mit dem Optischen Fluss lässt sich zu einzelnen Bildblöcken der zugehörige Bewegungsvektor bestimmen. Die Algorithmen zur Bestimmung des Optischen Flusses versuchen dabei Lösungen für die Einschränkungsgleichung zu finden. Das Ergebnis ist umso robuster, je grösser der jeweilige Block ist, auf dem der Algorithmus operiert. In unserer Arbeit verwenden wir den Optischen Fluss zur Detektion von Kopfrotationen um die Nasenwurzel. (siehe auch Kapitel 3.2.2 auf Seite 39)

2.3 Klassifizierung

2.3.1 Neuronale Netze

Das menschliche Gehirn liegt bei der Ausführung numerischer Algorithmen hinter der Leistungsfähigkeiten technischer Maschinen. Trotzdem ist es im Bereich der Mustererkennung heutigen Rechnern weit überlegen. Sei es im Wiedererkennen von Gesichtern oder Verstehen von Sprache. Selbst bei unvollständigen oder verfälschten Eingaben gelingt es dem menschlichen Gehirn das Wesentliche zu extrahieren. Was liegt somit näher, als die Arbeitsweise des menschlichen Gehirns auf Maschinen zu übertragen.

Definitionen

Ein künstliches Neuron ist ein Tupel $(\vec{x}, \vec{w}, f_a, f_o, \vec{y})$ bestehend aus einem Eingangsvektor $\vec{x}$, einem Ausgangsvektor $\vec{y}$, einem Gewichtungsvektor $\vec{w}$, einer Aktivierungsfunktion f_a mit $f_a : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ und einer Ausgabefunktion f_o, mit $f_o : \mathbb{R} \rightarrow \mathbb{R}$. Der Ausgangswert des Neurons ergibt sich nun durch $\vec{o} = f_o(f_a(\vec{x}, \vec{w}))$.

Ein künstliches Neuronales Netz ist ein gerichteter Graph (N, V), mit den Neuronen N als Knoten und den Verbindungen zwischen den Neuronen V als Kanten. Dabei können die Verbindungen noch unterschiedlich stark gewichtet werden. Die Schnittstellen zur Aussenwelt erfolgt durch die Eingabe- und Ausgabeschicht bzw. Eingangneuronen $\{i_1 ... i_n\}$ und Ausgangsneuronen $\{o_1 ... o_n\}$. Alle Neuronen die nicht von der Aussenwelt zugänglich sind heissen verborgene Neuronen oder *hidden neurons* (siehe Abbildung 2.14).

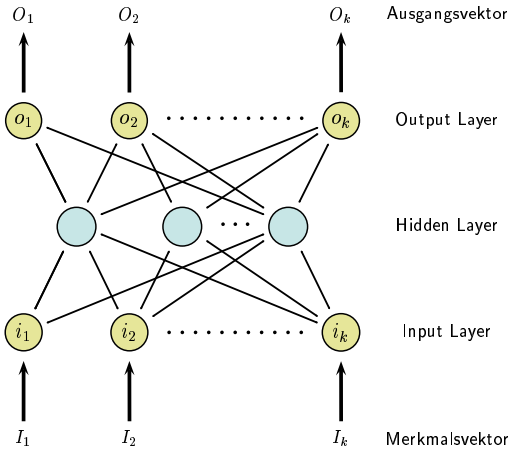


Abbildung 2.14: Neuronales Netz mit Ein- und Ausgabeschicht

Prägung

Bevor ein Neuronales Netz einsatzfähig ist muss es erst trainiert werden. Es lernt indem es sich selbst modifiziert. Zum Beispiel durch Veränderungen der Gewichte, Ausprägung neuer Neuronen oder Verbindungen, löschen existierender Verbindungen oder Anpassen der Aktivierungsschwellenwerte. Man unterscheidet dabei verschiedene Lerntechniken :

- Überwachtes Lernen (=*supervised learning*)

  Beim Überwachten Lernen wird die korrekte Ausgabe mit der Ausgabe des Netzes verglichen. Die Differenz zwischen den beiden Ausgaben wird minimiert durch Anpassen der Gewichte des Neuronalen Netzes. Diese Technik setzt allerdings voraus, dass viele Trainingsdaten vorliegen.

- Bestärkendes Lernen (=*reinforcement learning*)

Beim Bestärktem Lernen wird dem Netz nur mitgeteilt, ob die Ausgabe korrekt war, nicht die Abweichung zur korrekten Lösung.

- Unüberwachtes Lernen (=*unsupervised learning*)

  Beim Unüberwachtem Lernen versucht das Netz ohne äussere Beeinflussung die Ausgaben in Ergebnisklassen zu partitionieren. Diesen Klassen werden dann seperat bewertet.

Bewertung

Charakteristisch für ein Neuronales Netz ist die verteilte Wissensrepräsentation. Es ist somit sehr robust gegenüber Ausfällen von einzelnen Neuronen oder Verbindungen zwischen Neuronen. Darüber hinaus werden Informationen massiv parallel verarbeitet. Dadurch lassen sich Neuronale Netze effizient auf Multiprozessor-Architekturen implementieren. Neuronale Netze sind adaptiv. Sie könnnen sich wechselnden Bedingungen durch Änderung der Verbindungsstrukturen anpassen.

Schwierig ist es für eine bestimmte Anwendung die passende Topologie des Netzes bzw. den optimalen Trainingsdatensatz zu ermitteln. [2] [5] [1]

2.3.2 Hidden Markov Modelle (HMM)

Hidden Markov Modelle (HMMs) kommen ursprünglich aus dem Bereich der Spracherkennung. Sie sind aber nicht nur auf diese Domäne beschränkt sondern können auch zur Klassifizierung in anderen Gebieten verwendet werden. Grundlage der Erkennung sind stochastische Modelle . In unserem Fall sind das stochastische Modelle der Featurefolgen, während der einzelnen Gesten. Somit erhält jede Geste ein eigenes Modell, das diese repräsentiert.

Definition

Ein HMM ist ein doppelt stochastischer Prozeß. Die Markovkette bildet den ersten Prozeß der HMM (siehe Abbildung 2.15 auf der nächsten Seite). Sie besteht aus den Zuständen $Q = \{q_1 \dots q_n\}$ und den Transitionen. Unter einer Transition versteht man den Übergang von einem Zustand q_i in einen neuen Zustand q_j. Für jeden dieser Zustandsübergänge wird eine Wahrscheinlichkeit $a_{i,j}$ angegeben, mit der genau dieser Übergang auftritt. Diese Übergangswahrscheinlichkeiten werden in der Matrix A gespeichert. Ausserdem wird für jeden Zustand q_i eine Wahrscheinlichkeit π_i angegeben, mit der dieser Zustand der Anfangs- bzw. Einsprungszustand ist. Somit lässt sich die Markovkette formal wie folgt definieren: $M = (A, \vec{\Pi})$

Der zweite Zufallsmechanismus gibt zu jedem Zeitpunkt t mit einer bestimmten Beobachtungswahrscheinlichkeit ein Symbol $O_t \in$ Ausgabealphabet aus. Dessen Wahrscheinlichkeit $b_j(O_t) = P(O_t|q_j)$ hängt davon ab, in welchem Zustand q_j sich der Prozeß gerade befindet. Diese Symbole können beobachtet werden, wobei von außen jedoch nicht festzustellen ist, in welchem Zustand q sich der Prozeß jeweils befindet. Daher auch der Name „verdeckt" oder „hidden" Markov-Modelle. Alle Emissionswahscheinlichkeiten werden in einer Matrix B gespeichert.

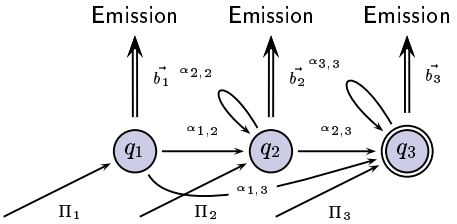


Abbildung 2.15: Einfache Markovkette mit 3 Zuständen

Definition eines HMMs: $\lambda_{HMM} = (A, B, \Pi)$

Berechnung

Wird eine bestimmte Folge von Symbolen $O = (O_1, O_2, \ldots, O_T)$ der Länge T beobachtet, so ist $P(O|\lambda_{HMM})$ die Wahrscheinlichkeit, dass O vom Modell M erzeugt wurde. Da nicht bekannt ist durch welche Zustandsfolge die Beobachtung O entstanden ist, müssen die Wahrscheinlichkeiten aller n möglichen Zustandsfolgen $Q = (q_1, \ldots, q_T)$ aufaddiert werden.

$$P(O|\lambda_{HMM}) = \sum_{i=1}^{n} P(O|Q_i)P(Q_i)$$

Dieser einfache Ansatz führt aber zu einer sehr ungünstigen Komplexität von $O(Tm^T)$, wobei m =Anzahl der Zustände und T die Länge der Ausgabe ist. Es existieren aber effektivere Verfahren um $P(O|\lambda_{HMM})$ zu berechnen, auf die hier aber nicht weiter eingegangen wird:

der Algorithmus von Baum-Welch

der Viterbi Algorithmus

der Vorwärts-Rückwärts Algorithmus

Anwendung als Klassifizierer

Um HMMs zur Klassifizierung verwenden zu können, muss für jede Klasse der Klassifizierung ein eigenes HMM erstellt und trainiert werden. Um eine Beobachtungsfolge zu klassifizieren wird für jedes Klassen-HMM die Wahrscheinlichkeit berechnet, dass die Beobachtungsfolge von diesem Modell erzeugt worden ist. Die Beobachtung gehört dann zu der Klasse mit der höchsten Wahrscheinlichkeit.

Bewertung

HMMs eignen sich besonders gut für das Modellieren von Pausen oder zeitlich gestreckter Abläufe, da diese in der zugrundeliegenden Markov Kette durch Self-Loops

nachgebildet werden. Desweiteren liefert das Verfahren automatisch die Güte der Klassifizierung. Um gute Ergebnisse zu erzielen, sind vor Allem gute Trainingsdaten und Klassen, die möglichst gut separiert sind, erforderlich. Ausserdem sollten die ausgewählten Featurevektoren die jeweilige Gesten bzw. Klassen gut approximieren. Diese Eigenschaft läßt sich oft nur durch Kombination von mehreren Einzelfeatures wie z.B. Kopfmittelpunkt und Optischen Fluß der linken- und rechten Gesichtshälfte erreichen. (siehe auch Kapitel 3.3 auf Seite 40) Dennoch stellen die HMMs ein mächtiges, flexibles, einfaches und vergleichsweise schnelles Verfahren dar.

2.3.3 Support Vektor Machines (SVM)

Auf Grund der Menge an Theorie kann, im Rahmen dieser Arbeit, auf die genauen Algorithmen für Support Vektor Maschinen nicht eingegangen werden. Im Folgenden soll ein Überblick über das Konzept, seine Möglichkeiten und eine prinzipielle Darstellung der Umsetzung gegeben werden. Für die Details sei auf die weiterführende Literatur [12] oder [13] verwiesen. Eine gute Einführung findet sich in [3].

Das Problem der Mustererkennung lässt sich vereinfacht folgendermaßen beschreiben. Als Eingabe erhält man einen Featurevektor $\vec{x}$ des weiteren k Klassen die hier mit $1, 2, \ldots, k$ bezeichnet werden sollen. Weiter sind richtige Mappings $(\vec{x}_i, y_i)$ gegeben, das Trainingsset. Zu dem gegebenen $\vec{x}$ soll das Label $y \in \{1, \ldots, k\}$ bestimmt werden, so dass die Wahrscheinlichkeit des Fehlers minimiert wird. Für den Fall das nur zwei Klassen unterschieden werden sollen, vereinfachen wir die Label zu $y \in \{-1, 1\}$. Auf diesen Fall wollen wir uns im Folgenden, der Einfachheit halber, beschränken.

Eine Maschine, die die gegebenen Mappings lernen soll, ist nun einfach durch alle möglichen Abbildungen $\vec{x} \mapsto f(\vec{x}, \alpha)$ gegeben. Der Parameter α kann nun angepasst werden. Zu einem festen α gehört daher das, was wir eine trainierte Maschine nennen. Der Erwartungswert des Fehlers, für eine trainierte Maschine, beträgt somit:

$$R(\alpha) = \int \frac{1}{2}|y - f(\vec{x}, \alpha)| dP(\vec{x}, y)$$

Die „expected risk" für ein gegebenes Trainingsset ergibt sich zu:

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^{i} |y_i - f(\vec{x}_i, \alpha)|$$

$\frac{1}{2}|y_i - f(\vec{x}_i, \alpha)|$ wird „loss" genannt. Gefordert ist nun ein α zu finden, so dass $R(\alpha)$ minimal wird.

Linear-separierbare Daten

Für den linear separierbaren Fall von Punkten, im n-dimensionalen Raum, wählt man eine Hyperebene im Raum, die die beiden Punkteklassen voneinander trennt (siehe Abbildung 2.16 auf der nächsten Seite).

Dies ist auf viele verschiedene Weisen möglich. Gesucht ist natürlich die optimale. Die Idee besteht darin den „Rand" zu den Daten möglichst groß zu halten. Abbildung 2.17 auf der nächsten Seite zeigt eine „gute" Hyperebene, (im zweidimensionalen

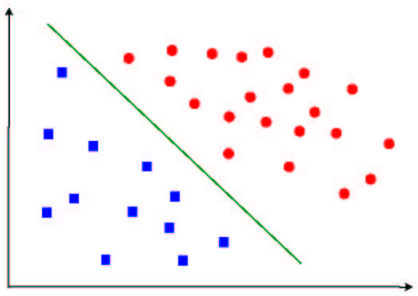


Abbildung 2.16: Linearseparierbarer Fall: Punkteklassen durch Hyperebene getrennt

Fall ist dies nur eine Gerade) wohingegen Abbildung 2.18 eine schlechte Separierung der selben Punktemenge zeigt.

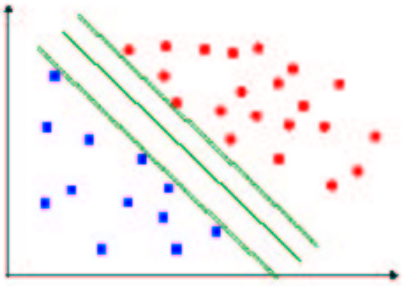


Abbildung 2.17: Linearseparierbarer Fall, „breiter Rand"

VC Dimension

Die VC (*Vapnik Chervonenkis*) Dimension einer lernenden Maschine (daher einer Menge von Funktionen $f(\alpha)$, wobei α wieder als Parametermenge verwendet wird) ist als die Kardinalität, der maximal zu trennenden Punktemenge, definiert. (f trennt die Punktemenge, falls die Menge in jede beliebige 2 Untermengen zerlegt werden kann) Für Hyperebenen gilt offensichtlich, dass die VC-Dimension gleich der Dimension des Featureraums + 1 ist, wie man sich leicht am 2-dimensionalen Fall klar macht. (3 Punkte entsprechend angeordnet lassen sich in jeder beliebigen Weise zerlegen, für 4 Punkte ist dies hingegen nicht mehr möglich)

Man könnte annehmen, dass die Performance einer Menge von Funktionen (dh. einer Maschine) mit ihrer VC-Dimension zunimmt, da sie ja mehr Punkte zu trennen vermag. Dies ist jedoch irreführend. In [3] wird eine Beispielmenge angegeben, deren VC-Dimension unendlich ist, für die jedoch einfach eine Menge von vier Punkten

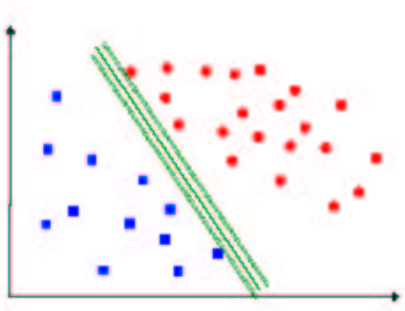


Abbildung 2.18: Linearseparierbarer Fall, schlecht separiert

angegeben werden kann, die nicht getrennt werden können.

Es lässt sich folgende Schranke für das Risiko zeigen. Die VC-Dimension sei mit h bezeichnet und wir wählen ein $0 \leq \eta \leq 1$ dann gilt für loss's die diese Werte mit Wahrscheinlichkeit $1 - \eta$ annehmen.

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\left(\frac{h(\log(2l/h) + 1) - \log(\eta/4)}{l}\right)}$$

Nun wieder zu unserem Fall der linearseparierbareren Daten, die wir durch Hyperebenen trennen wollen. Sei die Hyperebene in Normalform gegeben, daher durch einen Vektor $\vec{w}$ und eine Konstante b, so dass für alle Punkte $\vec{x}$ auf der Ebene gilt: $\vec{x} \cdot \vec{w} + b = 0$. Nun nehmen wir an, dass die Punkte des Trainingssets auf dem „Rand" um die Hyperebene genau die Entfernung 1 zur Hyperebene haben. Dies können wir durch Skalierung der Daten garantieren. So müssen die folgenden Ungleichungen gelten:

$\vec{x}_i \cdot \vec{w} + b \geq 1$ für $y_i = 1$

$\vec{x}_i \cdot \vec{w} + b \leq -1$ für $y_i = -1$

was sich zu folgender Ungleichung zusammenfassen lässt.

$$y_i(\vec{x}_i \cdot \vec{w} + b - 1) \geq 0 \quad \forall i \tag{2.13}$$

Durch die Forderung dass die Punkte auf dem Rand (d.h. unsere Supportvektoren) Entfernung 1 zur Hyperebene haben, berechnet sich der Abstand der Supportvektoren $\vec{w} \cdot \vec{x_i} + b = \pm 1$ und somit der Rand zu $\frac{2}{\|\vec{w}\|}$. Somit können wir den Rand Maximieren, indem wir unter Einhaltung der Ungleichung (2.13) $\|\vec{w}\|$ minimieren. Dies ist ein Konvexes Problem und lässt sich mit der Lagrange Methode lösen. Das duale Problem enthält wesentlich weniger Ungleichungen, wodurch die Lösung vereinfacht

wird. Nach dem Dualitätsprinzip stimmen die optimalen Lösungen des primalen mit denen des dualen überein (wobei natürlich bei dem einen Minima und bei dem anderen Maxima gesucht sind). Im separierbaren Fall lässt sich daher relativ einfach die optimale Lösung bestimmen.

Nichtseparierbarer Fall

Im ersten Ansatz werden so genannte Slack Variablen eingeführt, die im Rand oder gar auf der „anderen Seite" liegen können. Dadurch wird der Algorithmus etwas komplexer, da die Fehler „bestraft" werden müssen, aber im Prinzip wird der selbe Algorithmus wie oben verwendet. Als Beispiel für enthaltene Fehler siehe, Abbildung 2.19.

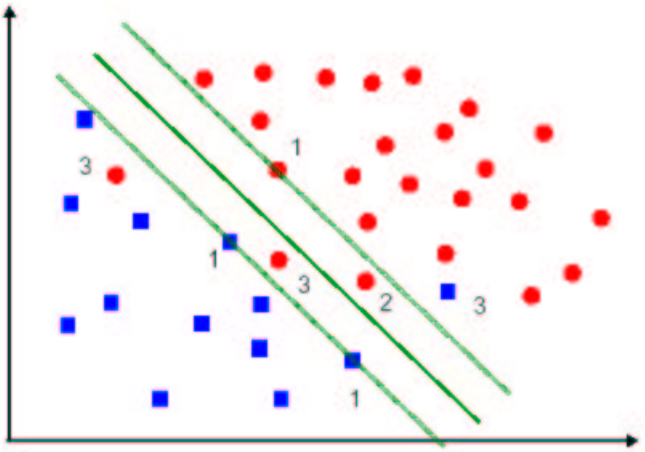


Abbildung 2.19: Nicht separierbar mit tolerierten Fehlern

Kernelmaschinen

In höheren Dimensionen ist die Separierbarkeit wahrscheinlicher, daher verfolgt man den Ansatz die Eingabe in einen Hochdimensionalen Featureraum zu transformieren. Hierfür definieren wir uns eine Funktion Φ die in unseren Hochdimensionalen (möglicherweise unendlichdimensionalen) Featureraum abbildet. Der Trainingsalgorithmus muss jetzt nur auf Skalarprodukten im Featureraum arbeiten, daher auf Ergebnissen von $\Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$. Wenn wir nun eine Kernelfunktion hätten mit folgender Eigenschaft: $K(\vec{x_i}, \vec{x_j}) = \Phi(\vec{x_i}) \cdot \Phi(\vec{x_j})$ müssten wir nur K in unserem Trainingsalgorithmus verwenden, ohne jemals ein Skalarprodukt im Featureraum wirklich auszurechnen, bzw. überhaupt die Funktion Φ explizit zu kennen.

Es lassen sich Bedingungen für Kernelfunktionen aufstellen (Mercer's Condition) so dass eine zugehörige Abbildung Φ und passender Featureraum existieren. Siehe

hierzu auch [4]. Hier wollen wir nur noch einige Standardkernel angeben. Z.B. Polynomiell: $K(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y} + 1)^p$ Gauss: $e^{-\|\vec{x}\cdot\vec{y}\|^2/2\sigma^2}$, oder $\tanh(\kappa\vec{x} \cdot \vec{y} - \delta)$. Wobei der letzte Kernel nach [3] eine spezielle Art eines zweischichtigen sigmuidalen Neuronalen Netzes ergibt. In Abbildung 2.20 ist ein Polynom dritten Grades als Kernel für ein linear separierbares Trainingsset verwendet worden, und die Lösung ist immer noch annähernd linear. Wohingegen in Abbildung 2.21 der linear nicht separierbare Fall durch die Verwendung eines Polynoms 3. Grades separierbar geworden ist.

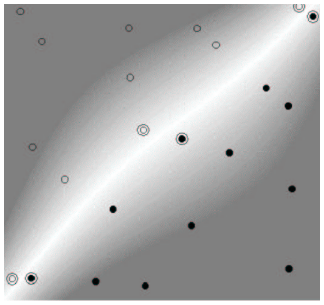


Abbildung 2.20: Polynomieller Kernel 3. Grades

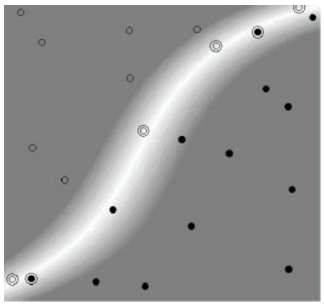


Abbildung 2.21: Polynomieller Kernel 3. Grades

Bewertung

Support Vektor Maschinen sind für Erkennungsaufgaben sehr gut geeignet, jedoch nur für den eher statischen Fall, dass die Eingangsdaten nicht wie in unserem Fall der Kopfgesten zeitliche Stauchungen erfahren können. An die Anforderungen, die diese Mustererkennungsaufgabe stellt können Support Vektor Maschinen nicht einfach angepasst werden.

2.3.4 Zusammenfassung

Verfahren	Bewertung
Neuronale Netze (2.3.1)	⊖ ungeeignet für logische Operationen oder rekursive Berechnungen ⊖ schlecht geeignet bei zeitlichen Stauchungen der Eingabe ⊕ effiziente Implementierungsmöglichkeiten auf parallelen Architekturen ⊕ robust gegenüber Ausfall einzelner Komponenten ⊕ kleiner Trainingskorpus
Hidden Markov Modelle (HMM) (**??**)	⊖ passender, genügend großer Trainingskorpus ⊕ unabhängig von zeitlicher Dauer der Eingabe ⊕ einfache Behandlung von Pausen in der Eingabe
Support Vektor Maschinen (SVM) (2.3.3)	⊖ schlecht geeignet bei zeitlichen Stauchungen der Eingabe ⊕ gute Erkennungsleistungen im statischen Fall

Fazit

Support Vektor Maschinen und Neuronale Netze sind eher ungeeignet für den Einsatz der Kopfgestenerkennung. Die beiden Verfahren kommen nur schlecht mit zeitlichen Stauchungen, wie sie bei Kopfgesten vorkommen, zurecht.

HMM's hingegen verfahren gut mit zeitlich gestreckten oder gestauchten Eingaben und sind darüberhinaus robust gegenüber Pausen. Auch wenn das Training von HMM's mehr Aufwand bedeutet als das von NN's, haben wir uns entschlossen erstere zur Klassifizierung der Kopfgesten zu verwenden. (siehe auch Kapitel 3.3 auf Seite 40)

Kapitel 3

Systemübersicht

Unsere Implementierung lässt sich grob in vier Abschnitte gliedern. Zuerst erfolgt das Lesen des Eingabebildes. Dieses kann in einem MPEG Stream kodiert sein, unkodiert in einer BMP Bildreihe vorliegen, oder über eine Videograbberkarte , mit Hilfe von Video 4 Linux (V4L) , eingelesen werden. Anschließend wird die Kopfposition und -größe lokalisiert (siehe Kapitel 3.1 auf der nächsten Seite). Die Featurebestimmung ermittelt die Postion der Nasenwurzel und die Bewegungsrichtungen der angrenzenden Bereiche. (siehe Kapitel 3.2 auf Seite 35) Die daraus resultierenden Merkmalsvektoren dienen der abschließenden Gestenklassifizierung (siehe Kapitel 3.3 auf Seite 40) als Berechnungsgrundlage.

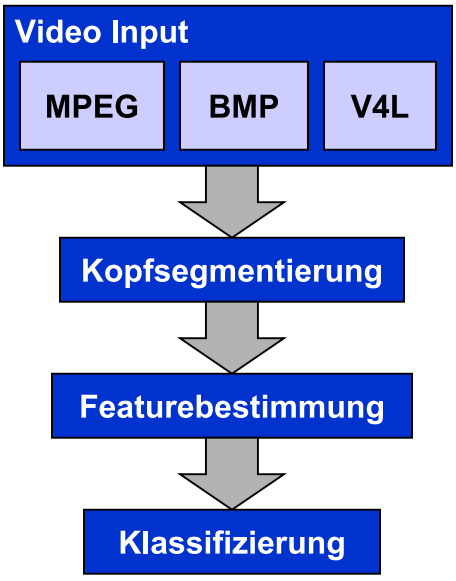


Abbildung 3.1: Flussgramm des gesamten Systems

3.1 Segmentierung

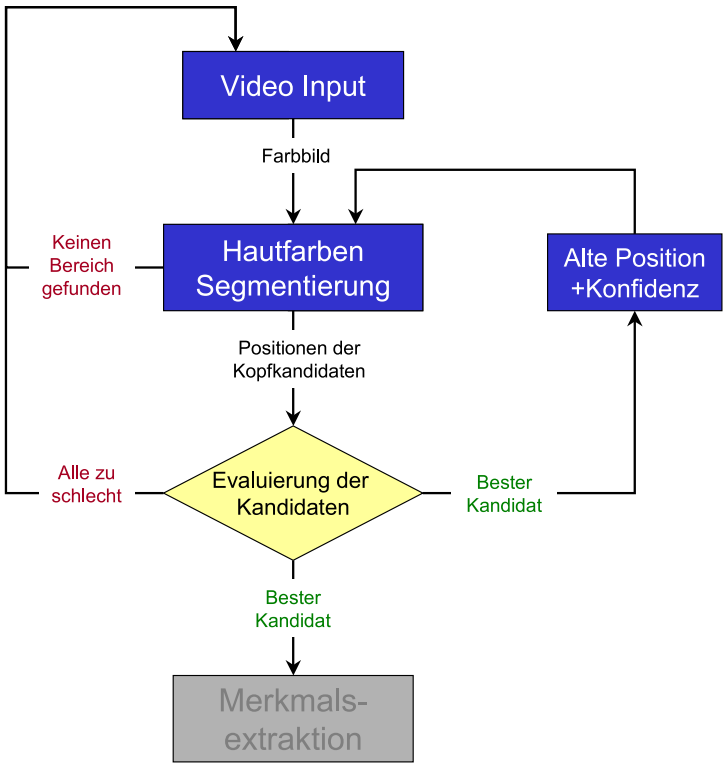


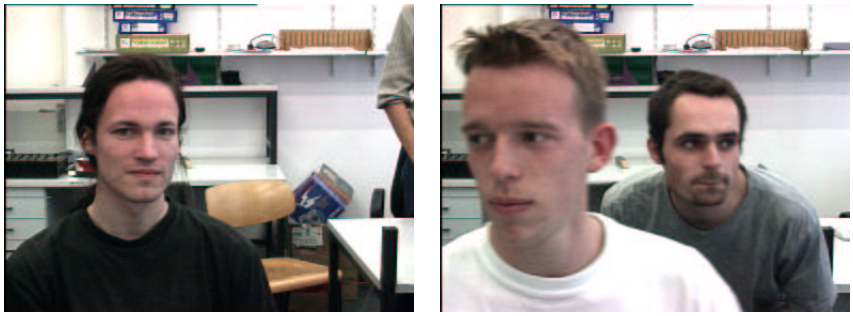
Abbildung 3.2: Flussdiagramm zur Kopfsegmentierung

Das Ergebnis unserer Segmentierung ist ein Rechteck, das die Kopfposition im Eingabebild zurückliefert. Wir verwenden ein farbbasiertes Verfahren , das die Farbinformation der menschlichen Haut als Basis benutzt. (siehe auch Kapitel 2.1.5 auf Seite 12)

3.1.1 Vorgehen

Das Eingabebild liegt im RGB-Format vor. Jeder Kanal hat dabei eine Farbtiefe von 8 bit. Die vorausgehende Konvertierung des Bildes vom RGB-Farbraum in den YCbCr Farbraum gewährleistet eine helligkeitsinvariante Detektion der Hautfarben. Da auch unterschiedliche Hautfarbentypen sich im Wesentlichen in der Luminanz unterscheiden und nicht in der Farbinformation, kann man den Y-Kanal ignorieren, und lediglich die CbCr-Ebene zur Segmentierung betrachten. Die Hautfarben bedecken in dieser Ebene nur einen relativ kleinen Bereich und liegen annähernd normalverteilt vor. Somit läßt sich diese Verteilung durch eine entsprechende Gaußfunktion approximieren. Die

Parameter dieser Funktion haben wir durch 40 Hautfarbenstichproben ermittelt. Für jeden der $256 \cdot 256$ verschiedenen Farbvektoren (cb, cr) lässt sich dessen Hautfarben-Wahrscheinlichkeit bestimmen.



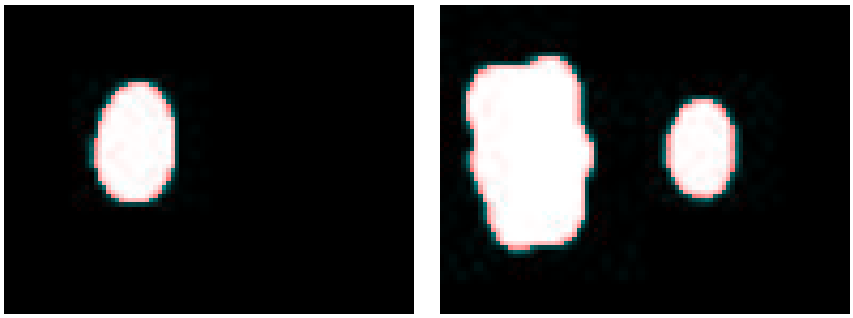
Eingangsbild

1. Hautfarbeninformation in Graustufenbild kodiert

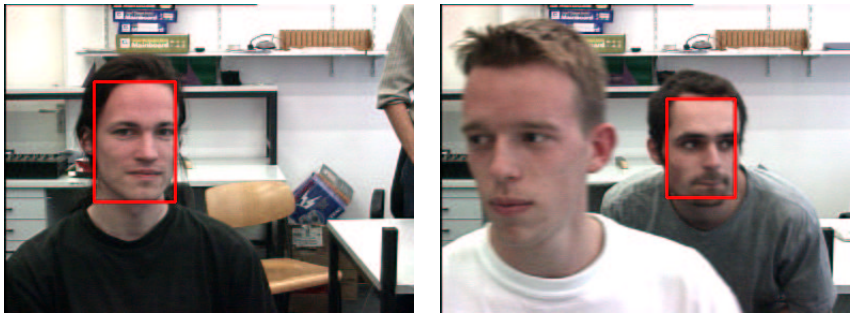
2. Binarisiertes Graustufenbild mit Schwellenwert

1. Daraus berechnet sich ein 8 bit Graustufenbild. Über jeden Bildpunkt kann man nun eine Aussage treffen, mit welcher Wahrscheinlichkeit er hautfarben ist.

3. Morphologische Operationen



4. Abschliessendes Closing



Kopfposition nach der Evaluierung aller Kopfkandidaten

2. Um im Folgenden verschiedene *morphologische Operationen* effizient auf das Graustufenbild anwenden zu können wird es binarisiert. Dabei legt ein Schwellenwert fest, ab welchem Graustufenwert der Bildpunkt als Hautfarbe bzw. als Nicht-Hautfarbe kodiert wird. Anschließend werden verschiedene morphologische Operationen auf das binarisierte Bild angewendet.

3. Durch ein *Closing* mit einer kleinen Ellipse werden kleine Verunreinigungen

aus dem Bild gefiltert. Ein *Opening* mit einem mittelgroßen Rechteck versucht dunkle Bereiche, wie sie in der Augengegend entstehen, abzudecken. Anschließend werden bei allen verbleibenden Zusammenhangskomponenten etwaige Löcher gefüllt. Diese Löcher können unter anderem in der Augengegend (die ja nicht hautfarben ist) auftreten und verhindern, dass das Gesicht erfolgreich segmentiert werden kann.

4. Das abschließende *Closing* mit einer länglichen, größeren Ellipse entfernt alle Bereiche, die nicht die passende Größe haben. Übrig bleiben eine oder mehrere Zusammenhangskomponenten. Die Positionen der kleinsten, umschließenden Rechtecke (*bounding box*) dieser Zusammenhangskomponenten sind die resultierenden Kopfkandidaten.

3.1.2 Evaluierung der Kopfkandidaten

Für jeden Kopfkandidaten wird berechnet, wie gut ein Templatebild der Nasenwurzel mit dem jeweiligem Bildbereich korreliert (siehe auch Kapitel 3.2.1 auf der nächsten Seite). Liegt dieser Korrelationswert unter einer gewissen Schranke, wird dieser Kopfkandidat nicht als neue Kopfposition akzeptiert. Überschreiten mehrere Kopfkandidaten diesen Schwellenwert, wird derjenige genommen, der die beste Übereinstimmung erreicht. Dieses Vorgehen ist zeitaufwendig und wird deshalb nur in der Initialisierungsphase angewendet.

Wurde ein Gesicht gefunden, wird die Initialsierungsphase verlassen und in die Laufphase gewechselt. Dort geht jede alte Kopfposition in die Folgeberechnung mit ein. Alle Rechenschritte werden dann nur mehr innerhalb der alten Kopfposition vorgenommen, bis die Größe des Bereiches zu klein wird oder keine Zusammenhangskomponenten mehr gefunden werden. Erst dann wird in die Initialisierungsphase gewechselt und die Suche wieder auf das gesamte Bild ausgedehnt. Dieses Prinzip garantiert eine robuste Kopflokalisation sowie ein schnelles Kopf-Tracking.

3.1.3 Zusammenfassung

Aus einem Farbbild werden die Positionen aller hautfarbenen Bereiche ermittelt. Möglich wird dies durch ein passendes Hautfarbenmodell, das durch Stichproben erstellt wurde. Anschliessend werden Bereiche, die keiner zulässigen Kopfform entsprechen, durch morphologische Operationen, eliminiert. Übrige Kopfkandidaten werden durch einen Templatematching-Vorgang evaluiert. Resultat ist eine neue Kopfposition, falls die Evaluierung erfolgreich war.

3.2 Merkmalsextraktion

In diesem Zusammenhang versteht man unter Merkmalsextraktion das Auffinden und Verfolgen markanter Punkte oder Eigenschaften in einem Gesicht. Der zeitliche Verlauf dieser Merkmale bildet die Grundlage der Klassifizierung (siehe auch Kapitel 3.3 auf Seite 40). Prinzipiell stellt das Ergebnis der Segmentation (siehe auch Kapitel 3.1 auf

Seite 32) auch ein Merkmal dar, das zur Klassifikation verwendet werden kann. Es hat sich jedoch herausgestellt, dass eine Boundingbox des Kopfes alleine nicht ausreichend ist, um gute Ergebnisse in der Klassifizierung zu erzielen. Das liegt zum einen daran, dass sich bei Rotationen des Kopfes um eine Achse, die in der Bildebene liegt, die resultierende Boundingbox nur geringfügig verändert. Besonders gut läßt sich dieses Verhalten bei Personen beobachten, deren segmentierter Bereich sich über den Kopf hinaus bis zur Brust erstreckt (siehe auch Abbildung 3.3). Hier würde eine Nickbewegung des Kopfes komplett im bereits segmentierten Bereich zu liegen kommen. Aus

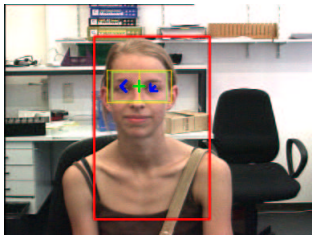


Abbildung 3.3: Segmentierter Bereich ist größer als der Kopf und kann somit nicht zum Tracken von Nickbewegungen des Kopfes verwendet werden

diesem Grund wird das Ergebnis der Segmentierung nur dazu verwendet, um den Suchbereich für die Merkmalsextraktion einzuschränken.

Im folgenden werden drei Merkmale betrachtet:

- Position der Nasenwurzel
- Optischer Fluss des rechten oberen Gesichtsquadranten
- Optischer Fluss der linken oberen Gesichtsquadranten

Der gesamt Prozeß ist in der Abbildung 3.4 auf der nächsten Seite dargestellt.

3.2.1 Extraktion der Nasenwurzel

Um die Bewegung des Kopfes zu verfolgen, ist ein markanter, unveränderlicher Punkt oder Region im Gesicht erforderlich. Diese Region sollte möglichst unabhängig vom Gesicht des Benutzers sein, um ein robustes Tracking zu gewährleisten. Naheliegend wäre es die Augen als Region zu wählen, diese sind jedoch nicht unveränderlich. So verschmelzen die Augen beim Lidschlag (siehe auch Kapitel 2.1.2 auf Seite 6) mit dem Rest des Gesichtes. Auch der Mund scheidet als Kandidat aus, da er sich beim Reden nicht nur bewegt, sondern auch seine Form stark verändert. Als letzter markanter Punkt im Gesicht bleibt die Nase übrig. Sie erfüllt die Anforderung unveränderlich zu sein. Die Nase bzw. Nasenwurzel alleine ist jedoch nicht markant genug, um sie in jedem Gesicht robust zu finden. Dieser Nachteil wird durch Hinzunahme eines Teiles der Augenbrauen und der Augen ausgeglichen (siehe Abbildung 3.5 auf der nächsten Seite), da diese Region eine hohe Symmetrie aufweist, welche einmalig im Gesicht ist. Um markante Punkte im Gesicht lokalisieren zu können, wird ein Template Image (siehe

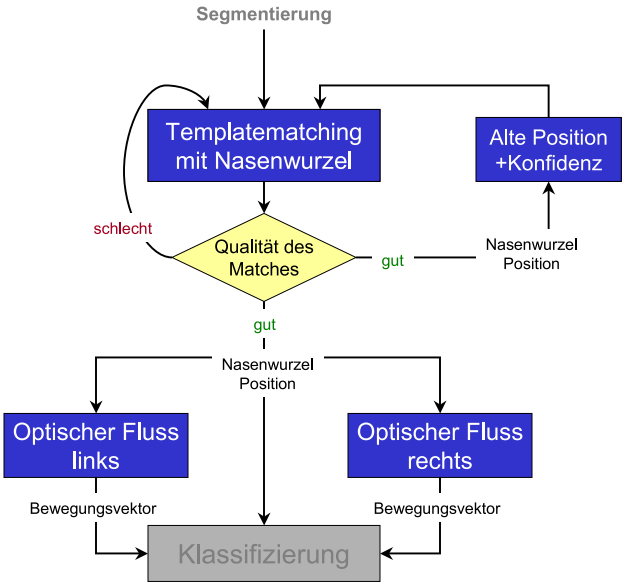


Abbildung 3.4: Flussdiagramm zur Merkmalsextraktion

auch Kapitel 2.2.2 auf Seite 17) über das Bild geschoben und für jeden Pixel ein Wert der Übereinstimmung ermittelt. Die Qualität des Ergebnisses ist hierbei von der Güte des verwendeten Template Bildes und der Art des Templatematchings abhängig. Die Grauwertkorrelation hat sich als das beste Verfahren erwiesen. Mit ihr ist es möglich Störeinflüsse wie z.B. Brillen zu ignorieren, da diese zwar den Bildbereich verändern, die Symmetrie bleibt allerdings unverändert, da Brillen zur selben Achse wie das Template Bild symmetrisch sind.

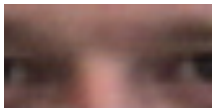


Abbildung 3.5: Die Nasenwurzel mit Augenbrauen und einem Teil der Augen

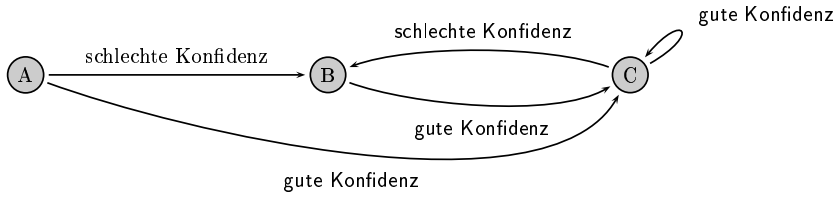
Vorgehen

Ausgangspunkt für die Lokalisierung ist das Rechteckt, das vom Segmentierungsprozess geliefert wird. Es ist anzunehmen, dass in diesem Bereich nur ein Punkt sehr gut mit dem Template übereinstimmt. Desweiteren kann davon ausgegangen werden, dass sich die Nase in der oberen Hälfte des gefundenen Kopfrechtecks befindet, was jedoch bei zu großzügiger Segmentierung zu Fehlern führen kann. Diese Annahme dient zum einem zur Steigerung der Geschwindigkeit zum Anderen kann durch Hinzunahme

von Zusatzinformationen der Suchraum verkleinert werden und somit Fehler minimiert werden. Wenn zusätzlich die Information, dass sich die Nasenwurzel von einem Bild zum anderen nur in einem bestimmten Rahmen bewegen kann, berücksichtigt wird, so kann der Suchraum sogar auf einen kleinen Bereich um die zuletzt gefundene Position der Nasenwurzel beschränkt werden. Diese extreme Einschränkung kann jedoch nur unter zwei Voraussetzungen zu guten Ergebnissen führen:

- die Position der Nasenwurzel im letzten Bild ist bekannt
- die Konfidenz des letzten Matches ist ausreichend hoch

Die erste Bedingung führt dazu, dass ein Initialsierungszustand eingeführt werden muss, dessen Suchbereich sich über die komplette oberen Hälfte des Kopfes erstreckt. Um der zweiten Voraussetzung gerecht zu werden, muss bei einem zu schlechten Match in den Initialisierungszustand zurückgesprungen werden (siehe Abbildung 3.6). Als Ergebnis des Prozesses wird eine Position und ein Konfidenzmaß zurückgegeben. Diese beiden Werte dienen im nächsten Schritt als Ausgangsgrößen für den optischen Fluss.



A Initialisierungsphase

B Reinitialisierungphase

C Arbeitszustand

Abbildung 3.6: Zustandsautomat für den Suchbereich der Nasenwurzellokalisierung

Bewertung

Die Nasenwurzel ist ein sehr robustes Merkmal zum Tracken. Folgende Einschränkungen sind zu beachten:

- Die Nasenwurzel darf nicht wesentlich durch Haare verdeckt sein.
- Bei zu großen Rotationen des Kopfes nach links oder rechts kann die Nasewurzel nicht robust lokalisiert werden, da große Teile des Templates nicht mehr sichtbar sind.
- Rotationen des Kopfes um 180° führen zu schlechten Ergebnissen. Dies kann aber vernachlässigt werden, da solche Fälle in der Praxis nicht auftreten.

- Bei Rotationen des Kopfes um die Z-Achse, wobei die Z-Achse durch die Nase geht, bewegt sich die Nasenwurzel nur unwesentlich. Diese Verhalten läßt sich beim seitlichen Neigen des Kopfes zur Schulter beobachten.
- Das Gesicht sollte einen minimalen und maximalen Abstand zur Kamera nicht überschreiten, da das Templatematching nicht größeninvariant ist.

3.2.2 Optischer Fluss im Gesicht

Es existieren Kopfgesten, die nicht durch die Position der Nasenwurzel alleine repräsentiert werden können. So verändert sich zum Beispiel bei Rotationen um die Z-Achse die Position der Nasenwurzel nur unwesentlich. Auch eine Betrachtung des Durchschnittes des Optischen Flusses der einzelnen Pixel des gesamten Kopfes, führt in diesem Fall zu keiner Verbesserung, da sich der Optische Fluss der linken- und rechten Gesichtshälfte gegenseitig auslöschen würden. Dies kann man umgehen, indem man nur den Durchschnitt des Optischen Fluss eines Gesichtsquadranten bzw. Gesichtshälfte betrachtet.

Vorgehen

Als erstes muss das Gesicht in Quadranten eingeteilt werden. Dazu gibt es prinzipiell zwei Möglichkeiten. Zum einen könnte man die *bounding box* des Kopfes verwenden, um die Qaudranten zu bestimmen. Hier kann es jedoch passieren, dass zu viel vom Hintergrund in die Berechnung einfließt und somit das Ergebnis verfälscht, wenn der Hintergrund selbst in Bewegung ist. Dieser Effekt tritt vor allem dann auf, wenn das Ergebnis der Segmentierung zu große Bereiche liefert. Die andere Möglichkeit zur Einteilung liegt darin, dass man die bereits lokalisierte Nasenwurzel als Mittelpunkt des Gesichtes betrachtet und relativ zu dieser Position die Lage der einzelnen Quadranten berechnet. Dies hat den Vorteil, dass der optische Fluss immer innerhalb des Gesichtes berechnet wird. Außerdem läßt sich die Güte des optischen Flusses durch die Güte der Nasenwurzelposition bestimmen. Desweiteren ist die Nasenwurzel ziemlich genau das Zentrum der kritischen Rotationen um die Z-Achse und ist daher als Koordinaten Mittelpunkt sehr gut geeignet.

In den so definierten Quadranten wird nun unabhängig voneinander der durchschnittliche Optische Fluss berechnet (siehe auch Kapitel 2.2.1 auf Seite 15). Die Größe der Quadranten ist durch das System vorgegeben. Die so entstandenen Bewegungsvektoren werden als Features an den Klassifizierer weitergegeben.

Bewertung

Für die meisten Gesten liefet der optische Fluss des linken- und rechten Gesichtsquadranten ein nahezu identisches Ergebnis, welches isoliert betrachtet einen guten Identifikator für Kopfgesten darstellen würde. In Verbindung mit der Nasenwurzel als Koordinatenursprung für die Quadranten und der relativen Bewegung der Nasenwurzel, läßt sich sogar horizontale Bewegung des gesamten Menschen von Kopfschütteln unterscheiden. Diese Tatsache ergibt sich dadurch, dass der Optische Fluss bei horizontalen

Bewegungen Null ist, weil der Offset, der durch die Bewegung entstanden ist, durch den Offset des Koordinatensystems ausgeglichen wird. Somit stellt der Optische Fluss ein gutes Feature für die Klassifizierung dar. Probleme bereiten allerdings die Augen, die beim Lidschlag oder beim Bewegen der Pupillen das Ergebnis verfälschen können.

3.2.3 Zusammenfassung

Die Merkmalsextraktion ist die Grundlage für die Klassifizierung. Es werden drei verschieden Merkmale extrahiert. Zum einen die Position der Nasenwurzel als markante und exponierte Stelle in der Mitte des Gesichtes und zum anderen der optische Fluss der rechten bzw. linken Gesichtshälfte. Die Güte der Nasewurzel ist direkt von dem Ergebnis der Segmentierungsphase abhängig und der optische Fluss ist direkt von der Nasenwurzel abhängig. Diese drei Features zusammen bilden eine gute Basis, um alle Kopfgesten klassifizieren zu können.

3.3 Klassifizierung

Die Klassifizierung von Kopfgesten stellt an die verwendeten Verfahren besondere Ansprüche. Vor allem muss die Erkennung bzw. die Berechnung der Features tolerant gegenüber Streckungen bzw. Stauchungen in der Zeit sein. Auch soll natürlich die Erkennung einer Geste unabhängig vom Ort, wo diese durchgeführt wird, sein. Letzteres ist jedoch relativ einfach durch relative Koordinaten bzw. Bewegungsvektoren zur Featureberechnung zu erreichen.

In der durchgeführten Implementierung werden zur Erkennung bzw. Klassifizierung der Gesten diskrete Hidden Markov Modelle (HMMs) verwendet. Den Ausschlag für die Wahl der HMM als Klassifikatoren gab deren hohe Tolleranz gegenüber zeitlicher Variationen der Gesten. Die anderen in Kapitel 2.3 beschriebenen Verfahren lassen sich nicht leicht in der Weise modifizieren, dass sie dies gleichermassen unterstützen. Die Evaluierung der anderen Methoden wurde nicht selbst durchgeführt, stattdessen stützen wir uns auf bekannte Ergebnisse aus der Sprachverarbeitung, wo an das Erkennen vergleichbare Anforderungen gestellt sind, wie bei der Gestenerkennung. Vor allem die guten Ergebnisse bei der Verwendung von HMM in der Spracherkennung haben die Entscheidung beeinflusst, da hier ebenfalls das Problem der zeitlichen Dehnung bzw. Stauchung sehr wichtig ist. Ebenso sind bei der Spracherkennung Pausen im Sprachfluss ein häufiges Phänomen, dass von HMMs gemeistert werden kann. Die Verwendung von HMM zur Gestenerkennung wird unter anderem in [**?**] beschrieben.

Diskrete HMM benötigen, wie z.B. [**?**] erwähnt, generell mehr Parameter, jedoch ist die Berechnung beim Erkennen einfacher. Die Performance wiederum hängt wesentlich von der Aufgabe ab. Da eine freie Implementierung von diskreten HMM vorhanden war, aber keine für das Zielsystem dieser Arbeit geeignete Implementierung von kontinuierlichen HMM, konnte eine durch eigene experimentelle Daten abgestützte Entscheidung nicht getroffen werden.

Die zum Einsatz kommenden HMM sind einer Beispielimplementierung von Richard Myers und James Whitson entnommen, welche auch schon in einer Arbeit am

Lehrstuhl MMK der TUM verwendet wurden, so dass hier auf diese Erfahrungen zurückgegriffen werden konnte.

Eine gute Klassifizierungsperformance ist mit HMM mit 5 Zuständen erreicht worden. (Auf das genauere Design konnte aufgrund der verwendeten Implementierung nicht genauer Einfluss genommen werden). Die Anzahl der Zustände wurde experimentell bestimmt, indem die Anzahl erhöht wurde, bis die Erkennung für die wichtigen Grundgesten Kopfschütteln und Nicken auf unseren Testdaten bei fast 100% lag.

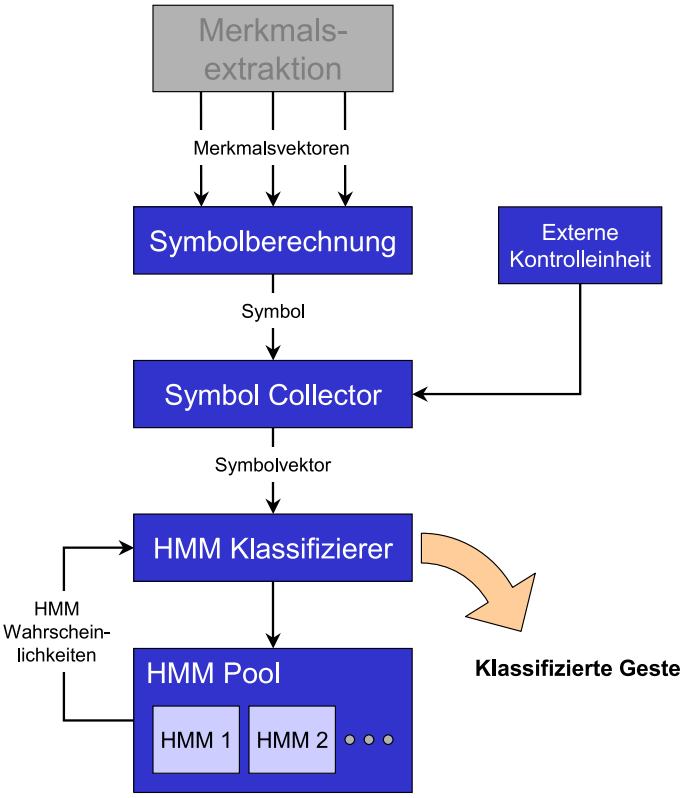


Abbildung 3.7: Flussdiagramm zur Klassifikation

Die Generierung der (diskreten) Symbole für die Erkennung durch das HMM gliedert sich in die folgenden Schritte. Zuerst werden der Optische Fluss über die der Nasenwurzel benachbarten Regionen berechnet und jeweils der Mittelwert berechnet (siehe Kapitel 3.2.2 auf Seite 39). Dann werden diese beiden Vektoren sowie der Geschwindigkeitsvektor der Nasenwurzel, deren Position durch Templatematching bestimmt wurde, wie in Kapitel 3.2.1 beschrieben, jeweils zu Integer Werten zwischen 0 und 5 diskretisiert. Hierbei stellt das Symbol 0 keine Bewegung dar. Die Symbole 1-4 stehen für Bewegungen in die 4 Himmelsrichtungen. Eine feinere Unterscheidung ist bei diskreten Pixelbewegungen mit Werten kleiner 2-3 Pixeln nicht sinnvoll. Diese drei Symbole werden nun kanonisch in einen Wert kodiert, der das endgültige Symbol bil-

det. Dies macht die verwendete HMM Implementierung nötig. (Die Überführung der 3 Werte in ein Symbol erfolgt nach der üblichen Formel $a + b * 5 + c * 5^2$)

Die Berechnung des Optischen Flusses, bzw. die Verwendung zur Symbolgenerierung für die Klassifikation ist nötig, um die Erkennung von Gesten zu ermöglichen, bei denen der Kopf zum Beispiel um sein Zentrum gedreht wird, so dass der Geschwindigkeitsvektor der Nasenwurzel alleine keine Identifizierungsmöglichkeit bietet. Hier deuten die Vektoren des optischen Flusses in entgegengesetzte Richtungen auf- bzw. abwärts, wodurch eine Klassifizierung solcher Gesten ermöglicht werden kann. Für einfachere Fälle kann auch ein simpler Symbolgenerator verwendet werden, der nur für einen Vektor ein Symbol berechnet. In der Frühphase des IDP wurden auch hiermit für Ja/Nein-Gesten zufriedenstellende Ergebnisse erzielt.

Die Klassifizierung wird durch den Nutzer initialisiert. Zuerst startet dieser die Berechnung der Symbole, die von einem Symbolcollector gesammelt werden. Auf eine weitere Aktion des Benutzers hin, wird nun die eigentliche Klassifizierung eingeleitet. [1] Hierzu testet der HMM-Klassifizierer die Symbolsequenz auf jedem HMM und erhält so einen Wahrscheinlichkeitsvektor für jedes HMM. Nun kann entweder dieser Vektor zurückgegeben werden, was eine höhere Flexibilität für die Anwendung bedeutet, oder nur das zur höchsten Wahrscheinlichkeit gehörende HMM. Die zugehörige Geste gilt dann als erkannt. (siehe auch Abbildung 3.7 auf der vorherigen Seite)

[1] Eine automatische Trennung von Gesten im Videostrom (sogenanntes Spotting) stand Ausserhalb des Bereichs dieser Arbeit, und wurde deshalb nicht untersucht

Kapitel 4

Evaluierung

In diesem Kapitel zeigen wir die Konditionen auf, die das System von seiner Umgebung fordert. Ausserdem ermitteln wir die Systemlaufzeit.

4.1 Vorbedingungen

Hier werden alle Randbedingungen und Voraussetzungen erläutert, die erforderlich sind, damit das System optimal funktioniert.

Bereich	Bedingungen
Ausgangsvideo	• ein in MPEG1 kodierter Videostream • Einzelbilder alle Einzelbilder eines Videostreams werden im BMP Format in forlaufender Nummerierung in einem Verzeichnis gespeichert • eine Video4Linux konforme Grabber-Karte
Ausgangsbild	• die Höhe und die Breite sind beliebig, müssen aber ein vielfaches von Vier sein • 288 Pixel für die Höhe und 384 Pixel für die Breite sind ausreichend um gute Ergebnisse zu erreichen • RGB Farbbild

Umgebung	• grundlätzlich ist die Umgebung irrelevant. Es sollten sich jedoch nicht zu viele Objekte, mit ähnlicher Fabrverteilung wie die der menschlichen Haut, im Hintergrund befinden. • mehr als ein Mensch bzw. Gesicht beeinflusst das Ergebnis insofern nicht, dass sich der Segmentierer in der Initialisierungsphase für ein Gesicht entscheidet, das er dann verfolgt. Bei Reinitialisierungen können allederings Probleme auftreten. • das Segmentierungsergebnis ist Abhängig von der Umgebungsbeleuchtung, wobei hierfür zu einem großen Teil die verwendete Kamera verantwortlich ist.

4.2 Benchmarking

Unser Testsystem, auf dem wir die Geschwindigkeitsmessungen vornahmen, war mit einem auf 2.4 GHz getakteten Intel Pentium IV Prozessor (512 kb Cachespeicher) bestückt. Das System verfügte über 1 GB Hauptspeicher. Als Betriebssystem kam Linux (Kernel 2.4.20) zum Einsatz. Zur Laufzeitmessung verwendeten wir tau in der Version 2.12 (*http://www.acl.lanl.gov/tau*).

Wir betrachteten als Testeingabe mehrere Videosequenzen. Diese unterschieden sich in der Anzahl der Kopfkandidaten, in der Grösse des erkannten Kopfbereiches und in der Erkennungsrate. An drei exemplarisch herausgenommen Beispielen vergleichen wir in den folgenden Ausführungen die jeweiligen Laufzeiten.

Die Balkendiagramme zeigen die Teillaufzeiten der einzelnen Programmmodule und die Gesamtlaufzeit. Die Werte sind dabei über die gesamte Lauflänge der jeweiligen Videosequenzen gemittelt. Die einzelnen Teilmodule sind

Bildoperationen (BO)	Zeit, die für Bildskalierungen, Farbtiefenkonvertierungen und Bildkopien anfällt
Kopflokalisation (KL)	Zeit zur Bestimmung aller Kopfkandidaten
Optischer Fluss (OF)	Zeit, zur Berechnung des Optischen Flusses
Template Matching (TM)	Zeit, die ein Templatematching Vorgang benötigt
Bild laden (BL)	Zeit, um ein Bild von dem Datenträger zu laden

Die Gesamtzeit (G) ist die durchschnittliche Zeit, die ein Bild der jeweiligen Videosequenz, zur Berechnung pro Zyklus benötigt. Dabei ergibt die Zeit der einzelnen Teilmodule nicht zwingend die Gesamtzeit, da einzelne Module wie z.B. das Template Mat-

ching mehrmals pro Durchlauf ausgeführt werden können (siehe auch Kapitel 3.1.2 auf Seite 35). Eine Verfeinerung der KL-Laufzeit stellen die Kreisdiagramme (siehe Abbildung 4.5 auf Seite 47) dar.

- Morphologische Operation (MO): Closing mit Kopfellipse
- MO: Löschen von kleinen Partikeln
- MO: Füllen der Augenregionen
- Bestimmung der Hautfarbenanteile
- Zusammenhangskomponenten (ZK) bestimmen
- Löcher in ZK schliessen

Testdaten

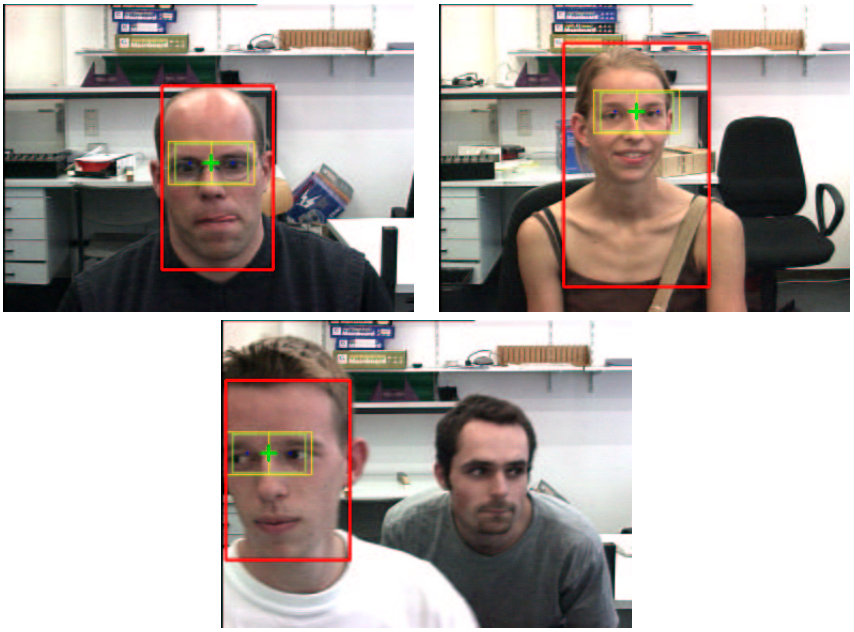


Abbildung 4.1: Ausschnitte aus den Videosequenzen *alt_mov*, *sintje1_mov*, *rudjus-leo_mov*

1. Video *alt_mov* (siehe Abbildung 4.1) Die Kopf- bzw. Nasenwurzelerkennungsrate des Videos liegt bei nahezu 100%. Somit sind kaum zeitaufwendige Neuinitialisierungen notwendig. Der lokalisierte Kopfbereich ist relativ klein, was sich in einer guten TM-Laufzeit niederschlägt. Die Gesamtlaufzeit liegt bei 17 ms. (siehe Abbildung 4.2 auf der nächsten Seite)

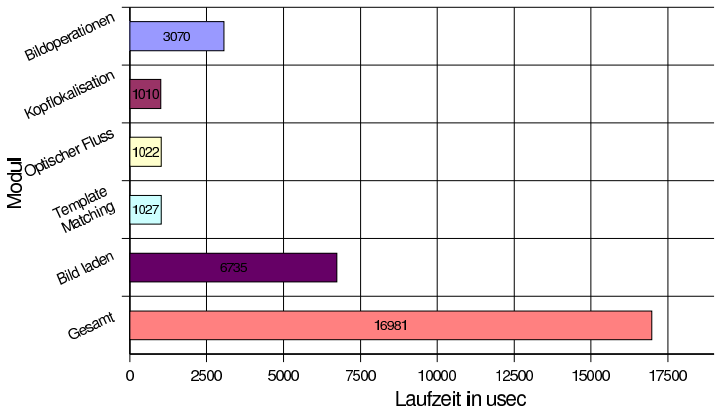


Abbildung 4.2: Laufzeiten der einzelnen Module (*alt_mov*)

2. Video *sintje1_mov* (siehe Abbildung 4.1 auf der vorherigen Seite)

   Durch den grossen Hautanteil, wird der Kopfbereich über den gesamten Oberkörper ausgedehnt. Dadurch ergibt sich eine längere TM-Laufzeit. Die Gesamtlaufzeit liegt bei 19,5 ms. (siehe Abbildung 4.3)

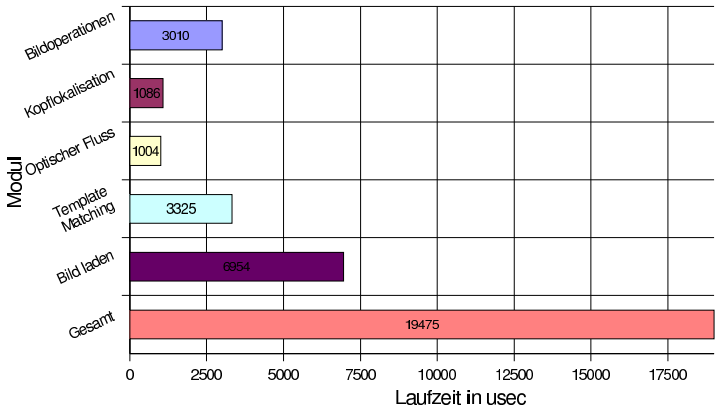


Abbildung 4.3: Laufzeiten der einzelnen Module (*sintje1_mov*)

3. Video *rudjusleo_mov* (siehe Abbildung 4.1 auf der vorherigen Seite)

   Die Grösse des Kopfbereiches ist vergleichsweisse klein. Die hohe TM-Laufzeit ist auf die zahlreichen Initialisierungen zurückzuführen. Dabei treten oft mehrere Kopfkandidaten auf. Die Gesamtlaufzeit beträgt 17,6 ms. (siehe Abbildung 4.4 auf der nächsten Seite)

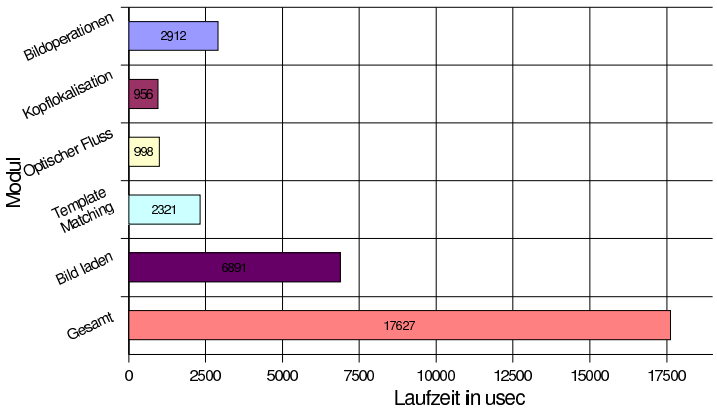


Abbildung 4.4: Laufzeiten der einzelnen Module (*rudjusleo_mov*)

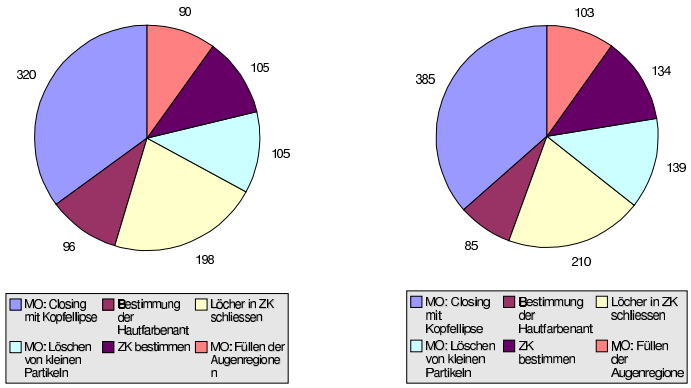


Abbildung 4.5: Laufzeitenanteile am Modul FindHead (*sintje1_mov*, *rudjusleo_mov*)

Zusammenfassung

Durchschnittlich erreichen wir eine Gesamtlaufzeit von ca. 18 ms. Das entspricht über 55 Bilder pro Sekunde. Zeiten für die Darstellung des Videos, basierend auf den Funktionen der *highgui*-Bibliothek, lagen bei zusätzlichen 10 ms. Die Zeiten zur Klassifizierung liegen vernachlässigbar bei unter 1 ms und wurden deshalb nicht extra aufgeführt.

4.3 Bewertung unserer Implementierung

Folgende Ausschnitte aus Videosequenzen zeigen Sonderfälle bzw. Grenzen auf, bezüglicher der Segmentierung und der Merkmalsextraktion. Durch Feineinstellung der einzelnen Variablen und Konstanten kann man sicherlich den einen oder anderen Problemfall noch optimieren. Das folgende Testset wurde jedoch nur mit den Defaultinitialisierungen erstellt (siehe auch Kapitel 5.2 auf Seite 53).

- Abbildung 4.6 zeigt die erfolgreiche Kopfextraktion, trotz einer anderen grossen Zusammenhangskomponente. Die richtige Auswahl wurde allein durch die Evaluierung der Kopfkandidaten getroffen (siehe auch Kapitel 3.1.2 auf Seite 35).

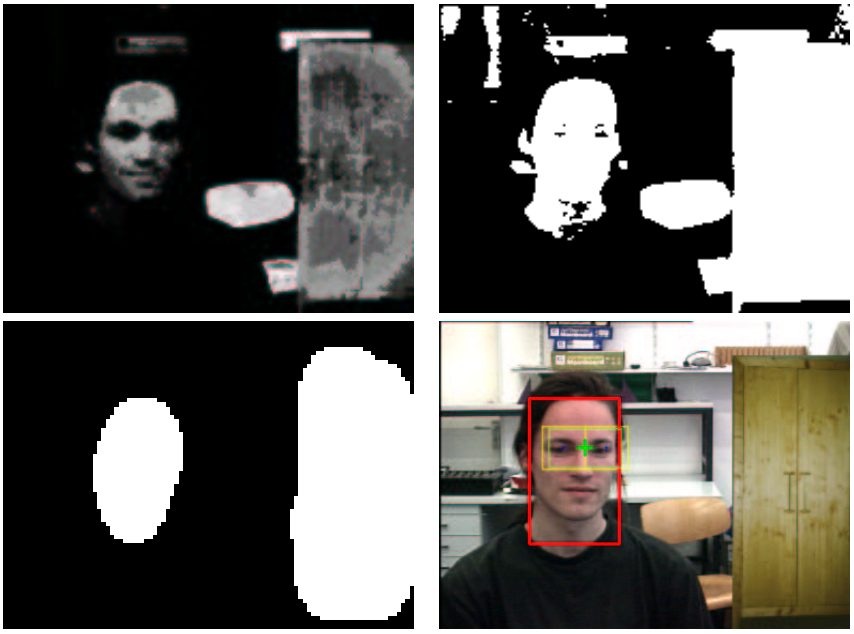


Abbildung 4.6: Erfolgreiche Kopflokalisation trotz mehrerer ZK

- In Abbildung 4.7 überschreiten mehrere Kopfkandidaten den nötigen Schwellenwert. Als Ergebnis wird derjenige zurückgeliefert, bei dem das Nasentemplate am besten matched.
- Abbildung 4.8 auf der nächsten Seite zeigt einen erfolglosen Segmentierungsversuch. Durch die dunklen Farben und dem stark bedecktem Gesicht ist es nicht mehr möglich einen Kopf zu segmentieren.
- Der Kopfbereich in Abbildung 4.9 auf der nächsten Seite wird trotz grosser Hautanteile im Bild erfolgreich segmentiert, da die restlichen ZK keinen Platz für die Kopfellipse bieten

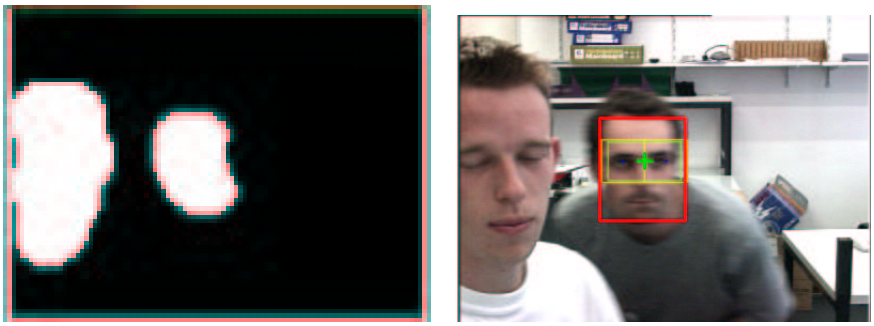


Abbildung 4.7: Besserer Match beim Rudi

Abbildung 4.8: Schlechte Farbverhältnisse und verdecktes Gesicht durch Bartwuchs

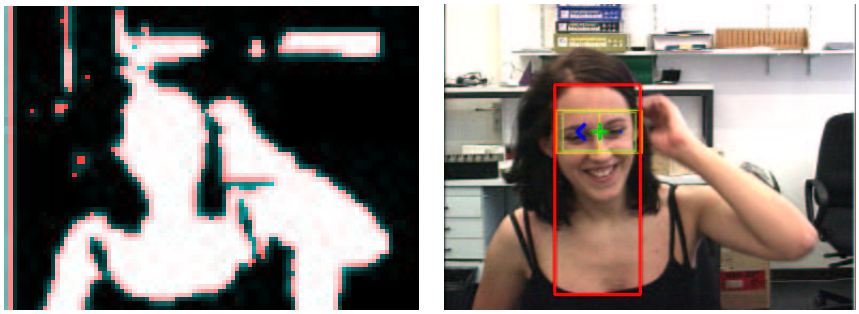


Abbildung 4.9: Erfolgreiche Erkennung trotz sehr grosser Zusammenhangskomponenten

- Durch die sehr dunklen Augenregionen in Abbildung 4.9 entstehen zwei grosse Löcher in der Kopfzusammenhangskomponente. Das rechte kann geschlossen werden, da es nach den morphologischen Operationen komplett umschlossen ist, das linke bleibt offen. Resultat beim abschliessenden Closing ist ein verkleinerter Kopfbereich oder eine erfolglose Kopfsuche.

Abbildung 4.10: Ungefülltes Auge, da nicht umschlossen von ZK

- Ebenfalls ein verkleinertes Kopfrechteck ensteht in Abbildung 4.11. Grund hierfür ist die starke Reflexion auf der Stirn. Dieser Bereich fällt komplett aus dem Hautspektrum und wird somit zum Hintergrund.

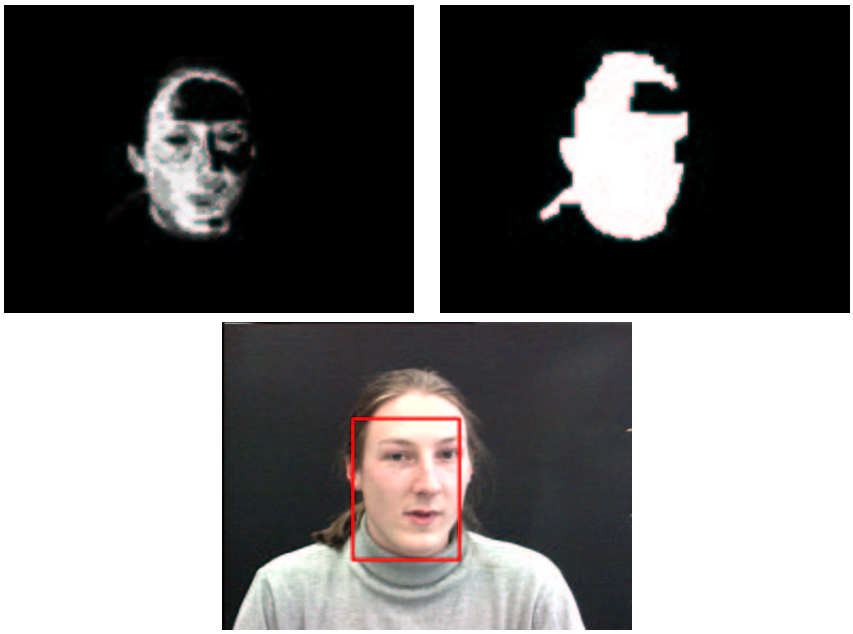


Abbildung 4.11: Verkleinertes Kopfrechteck wegen ungünstiger Beleuchtung

- Abbildung 4.12 auf der nächsten Seite zeigt einen erfolgreiche Kopfextraktion. Gut zu erkennen ist, wie die Nichthautfarbenen Bereiche nach den morphologischen Operationen geschlossen werden. Auch kann das Hemd durch das abschliessende Closing eliminiert werden, da es nicht die passende Form hat.
- Der grosse Kopfbereich in Abbildung 4.13 auf der nächsten Seite entsteht durch

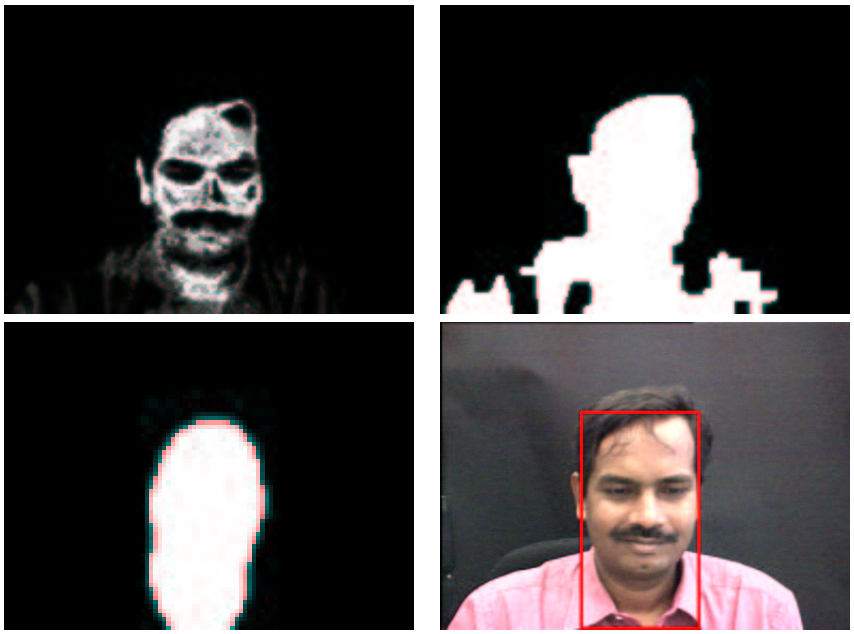


Abbildung 4.12: Erfolgreiche Segmentierung

den grossen Hautbereich am Oberkörper. Trotzdem wird die Nasenwurzel noch erfolgreich gefunden. Die Laufzeit dafür ist aber höher.

Abbildung 4.13: Sehr grosser Kopfbereich

- Die falschen Matches in Abbildung 4.14 auf der nächsten Seite werden einerseits natürlich durch die Kopfposition verursacht. Dadurch kann das Template nicht mehr in voller Grösse matchen. Andererseits werden die schlechten Übereinstimmungen durch den grossen Kopfbereich gefördert. Dieser überdeckt zum Teil den Bildhintergrund. Somit ergeben sich in der oberen Hälfte des Kopfbereiches Kandidaten für die Nasenwurzel, die den Schwellenwert überschreiten.

Abbildung 4.14: Falscher Match der Nasenwurzel

4.4 Ausblick

Unsere Implementierung kann aus Farbbildern erfolgreich Köpfe segmentieren, die Nasenwurzelposition lokalisieren und anhand dieser Merkmale Gesten klassifizieren. Jedoch sind einige Verbesserungen denkbar.

- Die Evaluierung der Kopfkandidaten könnte noch robuster gemacht werden, wenn mehrere Gesichtsmerkmale extrahiert würden. Zum Beispiel, zusätzlich zur Nasenwurzel, noch die Position der Augen und des Mundes. Gleichzeitig wäre auch die Lokalisierung der Nasenwurzel robuster. Man würde die Position nur akzeptieren, wenn auch die anderen Merkmale im richtigen Umkreis gefunden werden.
- Das Ergebnis der Segmentierung könnte noch stärker als Feature herangezogen werden. So könnte die Orientierung der Zusammenhangskomponente oder auch ihr Schwerpunkt betrachtet werden.
- Um auch Köpfe vor hautfarbenen Hintergrund erfolgreich segmentieren zu können, müsste zusätzlich noch ein formbasiertes Verfahren implementiert werden
- Das verwendete Hautfarbenmodell könnte man dynamisch der wechselnden Umgebung anpassen. Z.B. bei anderer Beleuchtung.
- Falls man nur die Kopfposition zur Verfügung hat und keine Nasenwurzel lokalisieren kann, könnte man trotzdem, allein durch den Optischen Fluss, noch Gesten klassifizieren.
- Das sogenannte Spotting, also die richtige Partitionierung der Gesten, ist nicht implementiert. Momentan muss der Benutzer Anfang und Ende einer zu klassifzierenden Geste manuell steuern.

Kapitel 5

Details der Implementierung

5.1 Klassenübersicht

Der Graph 5.1 auf Seite 56 zeigt die wichtigsten Klassen und Abhängigkeiten unserer Implementierung. Für eine detaillierte Auflistung und Beschreibung aller Klassen, Methoden und Variablen sei auf die, dem Source Code beiliegende, *doxygen*-Dokumentation verwiesen.

5.2 Initialisierungswerte

Tabelle 5.2 auf Seite 55 zeigt die Initialisierungswerte der wichtigsten Variablen bzw. Konstanten auf. Viele dieser Werte sind Erfahrungswerte und entstanden durch „try and error".

Tabelle 5.1: Initialisierungswerte verschiedener Variablen

Beschreibung	**Variablenname**	**Initialisierungswert**
Globale Variablen/Konstanten		
Breite des Eingabebildes	`Idp::Image::width`	384
Höhe des Eingabebildes	`Idp::Image::height`	288
Grösse der Optischen Fluss Fenster um die Nasenwurzel	`optflow_area_size.width`	40
	`optflow_area_size.height`	40
Variablen/Konstanten der Klasse `Idp::FindHeadSkin`		
Schwellenwert für Klassifizierung als Hautfarbe	`skin_threshold`	2

| **Beschreibung** | **Variablenname** | **Initialisierungswert** |

Tabelle 5.1: Initialisierungswerte verschiedener Variablen

Beschreibung	**Variablenname**	**Initialisierungswert**
Mittelwerte des Maximums der Hautfarbenfarbverteilung	`mid_cr` `mid_cb`	$0.5969 * 255$ $0.4234 * 255$
Einträge der Kovarianzmatrix der Hautfarbenverteilung	`covar_[pos]_value`	$\begin{pmatrix} 94.9412 & 66.7807 \\ 66.7807 & 139.2085 \end{pmatrix}$
Grösse des strukturierenden Elements zur Kopfsegmentierung	`head_ellipse_size.width` `head_ellipse_size.height`	16 22
Grösse des strukturierenden Elements zur Rauschfilterung	`dirt_ellipse_size.width` `dirt_ellipse_size.height`	2 2
Grösse des strukturierenden Elements zum Füllen von Löchern	`eye_rect_size.width` `eye_rect_size.height`	5 5

Variablen/Konstanten der Klasse `Idp::FindHead`

Schwellenwert zur Akzeptanz eines Kopfkandidaten nach dem Templatematching	`init_template_match_threshold`	0.7
Zusätzliche Suchbreite/höhe zum aktuellen Kopfbereich	`delta_search_area.width` `delta_search_area.height`	3 3
Minimale Kopfgrösse	`min_head_size_width` `min_head_size_height`	10 10
Maximal erlaubte Grössenänderung des aktuellen Kopfbereiches bevor eine Reinitialisierung erfolgt	`min_jumping_width` `min_jumping_height`	20 20
Maximal erlaubte Positionsänderung des aktuellen Kopfbereiches bevor eine Reinitialisierung erfolgt	`min_jumping_posx`	20

Beschreibung	**Variablenname**	**Initialisierungswert**

Tabelle 5.1: Initialisierungswerte verschiedener Variablen

Beschreibung	Variablenname	Initialisierungswert
	`min_jumping_posy`	20
Wird der Suchbereich eingeschränkt oder nicht	`search_whole_image`	false

Variablen/Konstanten der Klasse `Idp::TemplateMatcher`

Schwellenwert ab dem das Template akzeptiert wird	`match_min_value`	0.7
Grösse des Suchbereichs	`neighborhood_size.width` `neighborhood_size.height`	7 4

Beschreibung	Variablenname	Initialisierungswert

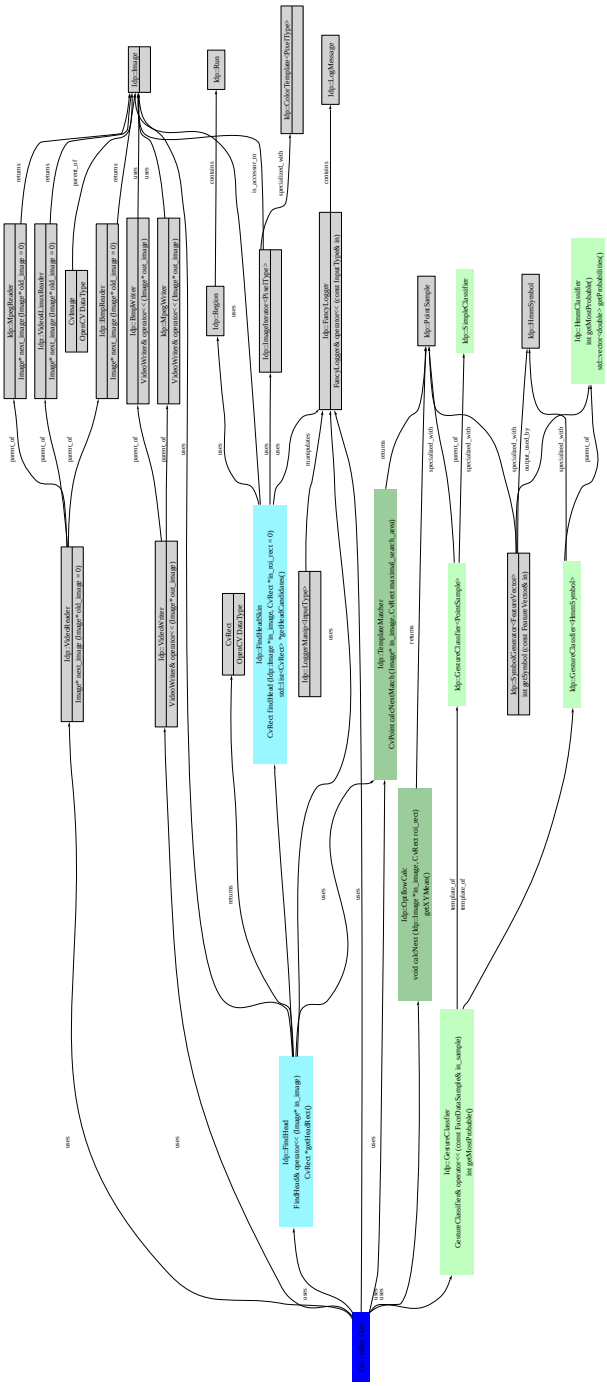


Abbildung 5.1: Detaillierte Klassenabhängigkeiten

Kapitel 6

Graphical User Interface (GUI)

Diese Kapitel beschreibt die IDP-GUI. Die GUI fasst alle Bereiche und Features des kompletten Systems in einem einzelnen Programm zusammen. Desweiteren lassen sich mit Hilfe GUI alle relevanten Parameter einfach und in realtime ändern. Die Oberfläche ist in vier Bereiche unterteilt (siehe Abbildung 6.1):

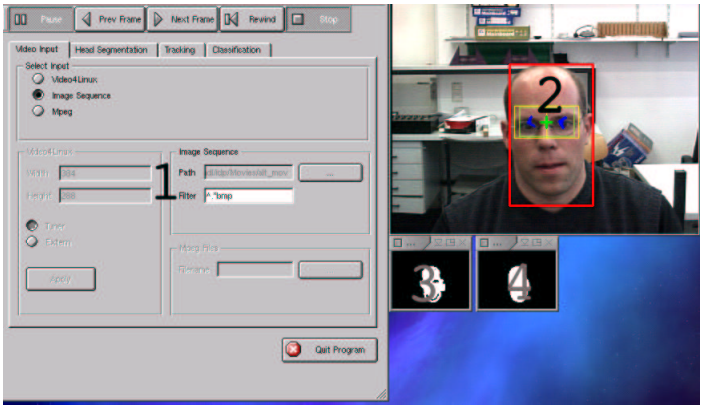


Abbildung 6.1: Die 4 Bereiche der GUI

- Hauptfenster (1):
  Hier können alle Einstellungen vorgenommen werden. Dieses Fenster untergliedert sich in fünf Bereiche, die in den nachfolgenden Sektionen erklärt werden.
- Hauptausgabefenster (2):
  Hier wird das Viedeobild und alle aus dem Videobild gewonnnen Daten ausgegeben. Ein segmentierter Kopf wird durch ein rotes Rechteck, die Nasenwurzel durch ein grünes Kreuz, der Suchbereich für die Nasenwurzel durch ein grünes Rechteck, der Optische Fluss links und rechts neben der Nasenwurzel durch blaue Pfleile und der Suchbereich für den optischen Fluss wird durch ein gelbes Rechteck gekennzeichnet.

- Ausgabefenster der Hautfarbenseqmentierung (3):
  Hier wird das Ergebnis der Farbbasierten Segmentierung angezeigt. Es wurden noch keine Morphologischen Operationen angewandt um das Ergebnis zu verbessern.
- Ausgabefenster für das Ergebnis der Morphologischen Operationen: (4)
  Die engültige Segmentierung des Kopfes nach Anwendung der Morpgologischen Operationen auf das Ausgabebild (3).

6.1 Videokontrolle

Mit der Videokontrolle ist es möglich das Videomaterial in Einzelschritten zu durchlaufen. Es stehen vier verschieden Buttons zur Verfügung (siehe Abbildung 6.2):

Abbildung 6.2: Die Video-Kontroll-Leiste

- Pause:
  Das Video wird angehalten, nicht aber die Verarbeitung des aktuellen Bildes. Somit kann man die Auswirkungen der einzelnen Parameter auf ein Standbild untersuchen. Der Optische Fluss liefert hier zwangsweise keine Ergebnisse mehr.
- Prev Frame:
  Es wird ein Bild im Video zurückgesprungen. Dies ist nur bei BMP-Videos möglich.
- Next Frame:
  Es wird ein Bild im Video vorgesprungen. Bei Video4Linux Videos ist dies meistens nicht der chronologisch Nachfolger des aktuellen Frames.
- Pause:
  Das Video und die Verarbeitung des Videos wird angehalten. Somit kann man die Auswirkungen der einzelnen Parameter vor allem aber den optischen Fluss auf Einzelbilder untersuchen, sofern man diese mit “Prev Frame” oder “Next Frame” einzeln abarbeitet.

6.2 Eingabemedien

In diesem Bereich lassen sich die verschiedenen Eingabemedien der Videosequenzen konfigurieren. Es stehen drei verschieden Arten von Videoquelle zur Verfügung (siehe Abbildung 6.3 auf der nächsten Seite):

- BMP-Seqenzen:
  In einem Verzeichnis liegen alle Bilder eines Videos als Einzelbilder im BMP-Format. Das Eingabefeld “Path” gibt hierbei den absoluten Pfad des Verzeichnisses an. Über den regulären Ausdruck im Eingabefeld “Filter” läßt sich das

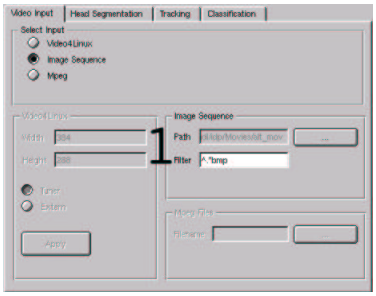


Abbildung 6.3: Die verschiedenen Video Eingabeformate

Einlesen des Verzeichnisses beeinflussen, indem nur Dateien eingelesen werden, die dem Regulären Ausdruck entsprechen.

- MPEG Dateien:
  Das Video wird aus einer MPEG 1 Datei gelesen.

- Video4Linux:
  Das Video wird direkt von einem Video4Linux-Device gelesen. Getestet wurde das Interface an einer Hauppauge PCI TV-Karte. Bei TV-Karten wird zwischen Tuner und einer externen Videoquelle unterschieden. Prinzipiell sollten aber auch andere Video4Linux-Devices wie z.B. Webcams funktionieren auch wenn diese keien Unterscheidung zwischen Tuner und Extern besizten. Um das Device benutzen zu können, muss zuerst die gewünschte Capture-Auflösung und Tuner oder Extern eingestellt werden. Mittels dem Button “Apply” wird das Capturing gestartet.

6.3 Segmentierung

In diesem Bereich lassen sich alle Parameter der Kopfsegmentierung einstellen (siehe Abbildung 6.4). Alle Parameter wirken sich sofort auf das Ergebnis der Seqmentierung aus.

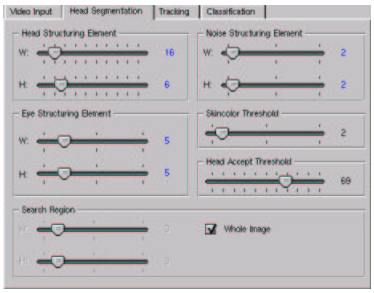


Abbildung 6.4: Die Parameter der Kopf-Seqmentierung

- Whole Image:
  Die Segmentierung wird für jedes Bild komplett auf dem ganzen Bild neu berechnet. Ansonsten findet die Segmentierung nur in einem kleinen Bereich um die letzte Kopfposition statt.

- Search Region:
  δx,δy angegeben in Pixel und bezogen auf $\frac{1}{8}$ des Originalbildes. Diese Werte geben an um wieviel sich der Suchbereich im Vergleich zur Größe des zuletzt gefundenen Kopfes vergrößert. Diese Option ist nur wirksam, wenn nicht über das gesamte Bild segmentiert wird (siehe oben).

- Skincolor Threshold:
  Threshold für die Grundsegmentierung auf Basis von Hautfarben. Je geringer dieser Threshold ist, desto mehr Pixel werden als Hautfarbe klassifiziert, wobei sich jedoch der Klassifizierungsfehler erhöht.

- Noise Structuring Element:
  Die Höhe und breite eines Rechteckes angegeben in Pixel und bezogen auf $\frac{1}{8}$ des Originalbildes. Alle Zusammenhangskomponenten die kleiner sind als dieses Rechteck werden gelöscht.

- Eye Structuring Element:
  Die Höhe und breite eines Rechteckes angegeben in Pixel und bezogen auf $\frac{1}{8}$ des Originalbildes. Alle Löcher und Buchten, innerhalb bzw. am Rand von Zusammenhangskomponenten, die kleiner sind als dieses Rechteck werden aufgefüllt. Oft entstehen solche Löcher durch Augen, weil diese keiner Hautfarbe entsprechen.

- Head Structuring Element:
  Die Höhe und breite einer Ellipse angegeben in Pixel und bezogen auf $\frac{1}{8}$ des Originalbildes. Nur ein Kopf der mindestens so groß ist wie diese Ellipse wird weiterverwendet.

- Head Accept Threshold:
  Dieser Wert gibt an, ab welcher Güte ein Kopfkandidat als Kopf klassifiziert werden soll.

6.4 Tracking

In diesem Bereich lassen sich alle Parameter des Trackings einstellen (siehe Abbildung 6.5 auf der nächsten Seite). Alle Parameter wirken sich sofort auf das Ergebnis des Trackings aus.

- Nosebridge Accept Threshold:
  Dieser Wert, angegeben in Prozent, gibt an, ab welcher Güte eine Nasenwurzel als gefunden klassifiziert werden soll.

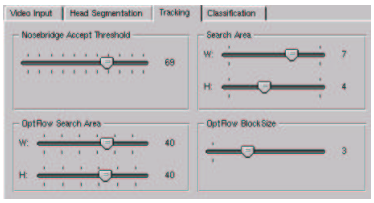


Abbildung 6.5: Die verschiedenen Parameter des Tracking

- Search Area:
  Δx,Δy angegeben in Pixel und bezogen auf $\frac{1}{4}$ des Originalbildes. Diese Werte geben die Gröβe des Suchbereiches für die Nasenwurzel an, wobei Δx und Δy auf die Position der zuletzt gefundenen Nasenwurzel bezogen ist.
- Opflow Search Area:
  Δx,Δy angegeben in Pixel. Diese Werte geben die Gröβe des Suchbereiches für den Optischen Fluss an, wobei Δx und Δy auf die Position der zuletzt gefundenen Nasenwurzel bezogen ist. Der Optische Fluss wird seperat links und rechts neben der Nasenwurzel berechnet.
- Opflow Block Size:
  Die Größe der quadratischen Blöcke, die für die Brechnung des Optischen Flusses verwendet werden.

6.5 Klassifizierung

Die Klassifizierung unterteilt sich in zwei Bereiche. Zum einen in die Generieung von Featurevektoren und zum Anderen in die Klassifizierung des Featurevektor mit Hilfe von HMMs. Mittels des Buttons “Start” wird die Generierung des Featurvektors gestartet und beendet.

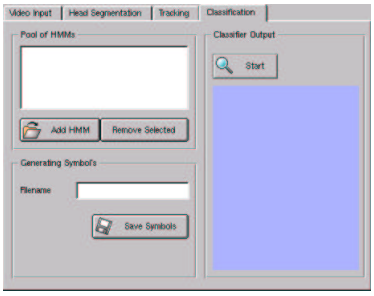


Abbildung 6.6: Die Möglichkeiten der Klassifizierung

- Pool of HMMs:
  Hier kann eine Liste von HMMs, die zur Klassifizierung herangezogen werden

sollen, verwaltet werden. Ergebnis der Klassifizierung ist dasjenige HMM, was die größte Wahrscheinlichkeit für den Featurevektor hatte.

- Classifier Output:
  Liste mit den Ergebnisen der Klassifizierung.

- Generating Symbols:
  Hiermit kann der erzeugte Featurevektor für das externe Trainieren von HMMs gepseichert werden.

Literaturverzeichnis

[1] Tu berlin. URL: http://user.cs.tu-berlin.de/~spohlexx/marvin/4_5_1_1_erkennung.html.

[2] Neuro-Fuzzy AG. Einführung in neuronale netze. URL: http://wwwmath.uni-muenster.de/SoftComputing/lehre/material/wwwnnscript/einl.html.

[3] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[4] R. Courant and D. Hilbert. *Methods of Mathematical Physics*, volume 1. Interscience Publishers, Inc, New York, 1953.

[5] Fraunhofer Gesellschaft. Mustererkennung - neuronale netze. URL: http://www.umsicht.fhg.de/WWW/UMSICHT/Produkte/SP/oe_320/Neuro/Topologie.html.

[6] V. Hargutt H. Tietze. Zweidimensionale analyse zur beurteilung des verlaufs von ermüdung. URL: http://www.psychologie.uni-wuerzburg.de/methoden/forschung/vortraege/tietze&hargutt_teap01.pdf.

[7] ITC. Eigenfaces and codification. URL: http://www.itc.ucm.es/PROYECTOS/EIGENCAR/eiweb2_e.html.

[8] Kay Uwe Kasemin. Houghtransformation. URL: http://www.physik.uni-osnabrueck.de/nonlinop/Hough.

[9] Andreas Leich. Anwendung der hough-transformation für aufgaben der digitalen bildverarbeitung im verkehrswesen. URL: http://vini25.vkw.tu-dresden.de/vinije/leich/diplom/Seite1.html.

[10] A. Pentland M. Turk. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 1991.

[11] Simon Perkins Robert Fisher. Hipr. URL: http://www.dai.ed.ac.uk/HIPR2/hough.htm.

[12] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, New York, 1995.

[13] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998. forthcoming.

Abbildungsverzeichnis

Index