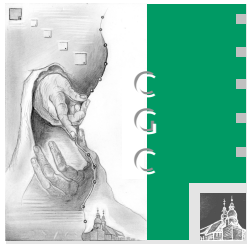


Virtuelle Realität
Interaktion per Computer Vision

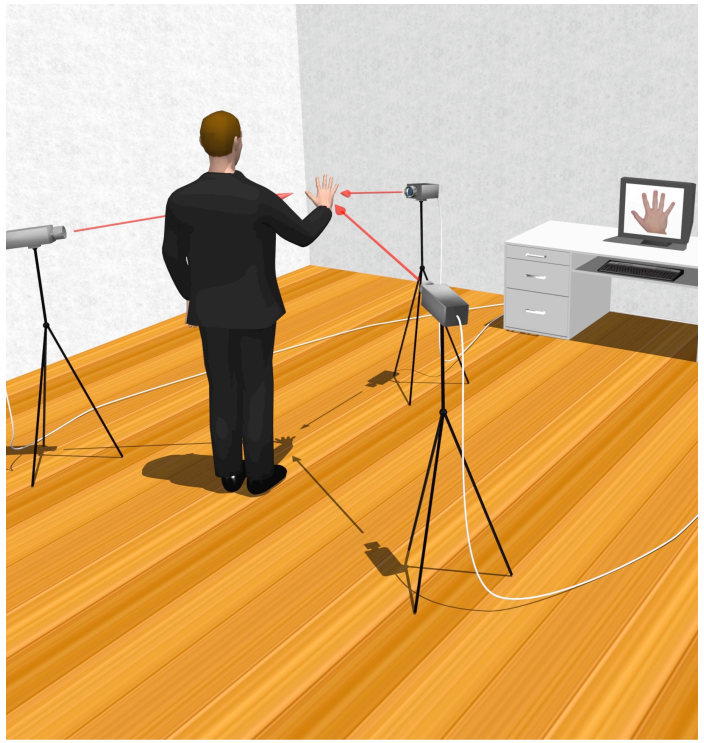
G. Zachmann

Clausthal University, Germany

cg.in.tu-clausthal.de

Was ist Computer-Vision?

- "*The target problem ... is computing properties of the 3-D world from one or more digital images.*"
  [Trucco, Verri: "*Introductory Techniques for 3-D Computer Vision*"]
- "*...computer vision involves computers interpreting images.*" [dito]
- "*Computer vision is the science and technology of machines that see. ... artificial systems that obtain information from images.*"
  [Wikipedia]



Motivation

- Aufgaben:
  - Tracking von Objekten (Körper, Hand, Kopf, Augen)
  - Shape-Erkennung (Geste, Körperhaltung, Gesichtsausdruck)
  - Erkennung eines Bewegungsablaufs (z.B. dynamische Gesten)
- Langfristig der richtige Weg
- Aber: robuste und schnelle Erkennung nicht trivial!
  - Verschiedene Techniken werden eingesetzt, jede hat ihre Vor- und Nachteile

Background Subtraction

- Aufgabe: Bewegliches Interaktionsobjekt (Vordergrund) vom Hintergrund extrahieren
- Idee: Subtrahiere aktuell aufgenommenes Bild von einem Referenzhintergrund

- Naiver Ansatz:
  - Erzeuge Referenzbild $\mathcal{R} : [1, \text{width}] \times [1, \text{height}] \rightarrow [0, 255]^3$
    - nehme Hintergrundbild auf
  - Subtrahiere pixelweise Eingabebild $\mathcal{B}_t$ zum Zeitpunkt t vom Referenzbild

$$P(x, y) = \|\mathcal{R}(x, y) - \mathcal{B}_t(x, y)\|_2$$

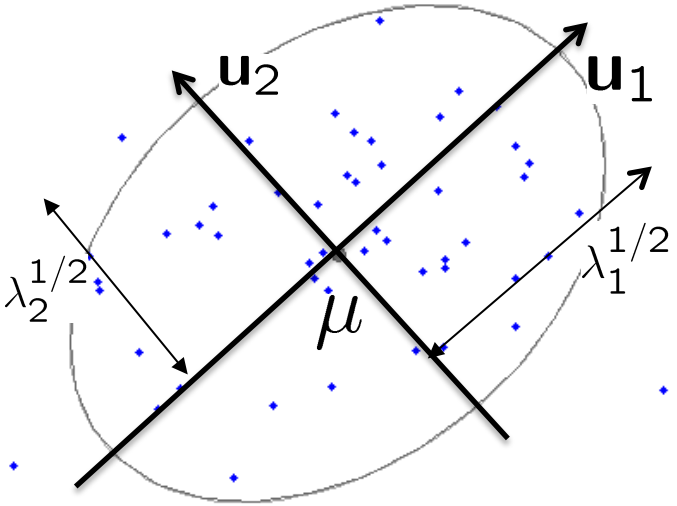
  - Ergebnisbild P ist ein Grauwertbild
  - Pixel mit Differenz größer einem Schwellwert τ gehört zum Vordergrund

$$P(x, y) = \begin{cases} \text{Vordergrund} & : \quad p(x, y) > \tau \\ \text{Hintergrund} & : \quad p(x, y) \leq \tau \end{cases}$$

- Problem: Schon leichte Variation des Hintergrundes (z.B. Beleuchtungsvariation) stört das Ergebnis

Etwas bessere Idee

- Farbverteilung für Hintergrund pro Pixel aufbauen
  - Hintergrund über längeren Zeitraum aufnehmen
  - Für jedes Pixel eigene Farbverteilung
    - Die einfachste Möglichkeit ist eine unimodale Funktion, d.h. eine Funktion mit nur einem Extremum
    - Häufigsten verwendete unimodale Funktion in der CV-Community ist die Gaussfunktion



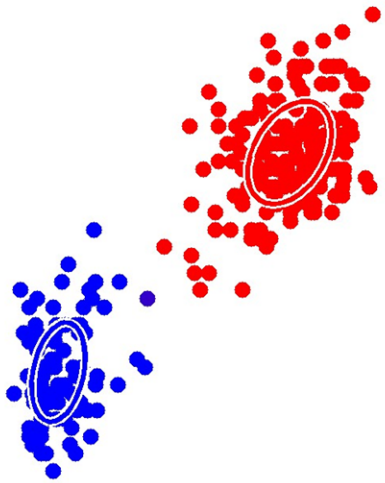
$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x} \mid \mu, \mathbf{\Sigma}) = \frac{1}{(2\pi)^{3/2}|\mathbf{\Sigma}|^{1/2}} \exp(-\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{\Sigma}^{-1}(\mathbf{x}-\mu))$$

- μ ist Mittelwert; $\mathbf{u}_1, \mathbf{u}_2$ die Eigenvektoren von Σ und λ_1, λ_2 die Eigenwerte mit $\lambda_1 > \lambda_2 > 0$
- Funktioniert ganz gut z.B. bei Beleuchtungsvariation

- Unimodale Fkt versagt schon bei leicht beweglichem Hintergrund (z.B. wedelnde Bäume)
- Verbesserung: Mixture of Gaussians (Parameter z.B. durch Clustering-Verfahren bestimmen)

$$p(\mathbf{x}) = \sum_{k=1}^{K} \omega_k \mathcal{N}(\mathbf{x}|\mu, \mathbf{\Sigma})$$

  - Im Falle des Baumes, K=2, eine Gaussfkt für den grünen Baum, die anderen für den weiss-blauen Himmel

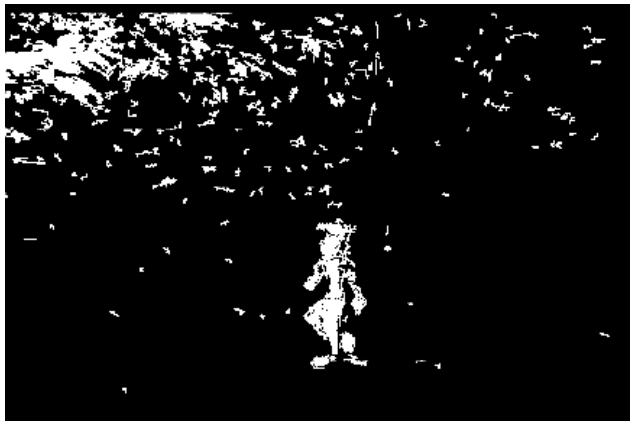


- Non-parametric Model bei stärker variierendem Hintergrund. z.B. Kernel Density Estimator

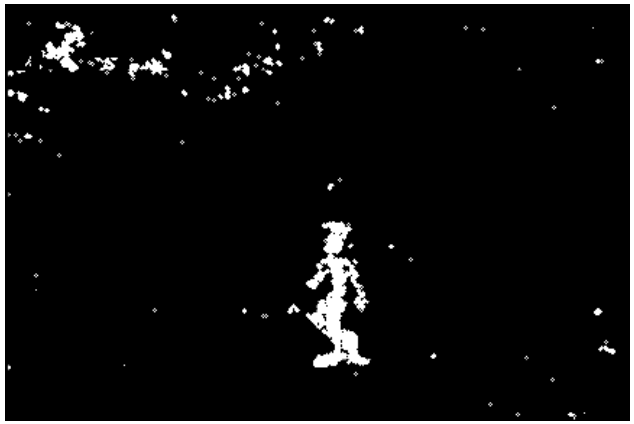
$$p(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} k\left(\frac{\mathbf{x} - x_n}{h}\right)$$

Beispiel

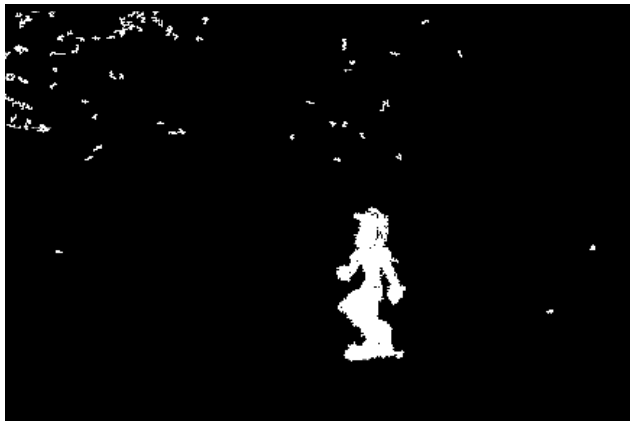
Source



Einfaches Thresholding



Non-parametric Model



Mixture of Gaussians

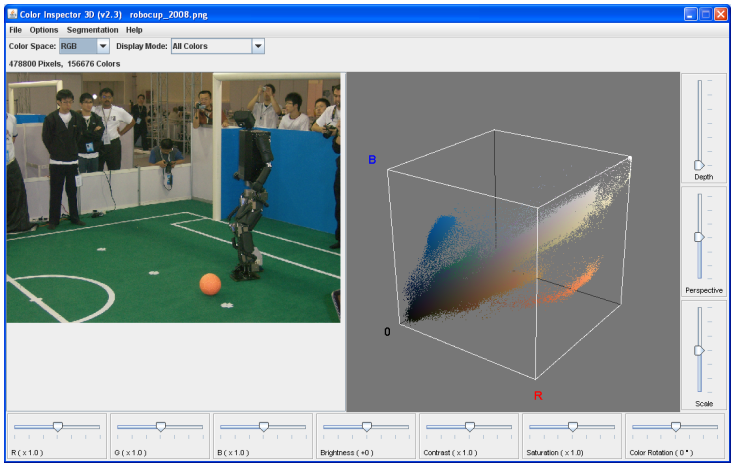
Grenzen des Verfahrens

- Variation des Hintergrundes nur in begrenztem Umfang
  - Objekte/Personen (die ich nicht tracken will) im Hintergrund dürfen sich nicht bewegen
  - Kameraposition statisch
- Tracking über längeren Zeitraum kaum möglich
  - Problem des sich stetig verändernden Hintergrunds, z.B. Beleuchtungsvariation

Farbsegmentierung

- Aufgabe: Bestimme alle Pixel im Bild die zu dem zu detektierenden Objekt gehören
- Voraussetzung: homogene Farbe des Zielobjektes
- Idee:
  - Farbverteilung des Objektes erstellen
  - Segmentierung der Vordergrundpixel über Farbraum
- Vorteil: Funktioniert auch bei beweglicher Kamera und variierendem Hintergrund

Robocup 2008



Farbhistogram
[Kai Uwe Barthel, FHTW Berlin]

Notationen

- Gegeben die Zufallsvariablen
  - X ein 3-dim Zufallsvektor, der Farbwerte repräsentiert
  - Y eine Zufallsvariable, welche die Segmentierung repräsentiert, d.h. sie kann die Werte *fg* (foreground) und *bg* (background) annehmen
  - *rgb* repräsentiert einen konkreten Farbwert z.B. (255,0,0)
  - P(X=*rgb*) gibt die Wahrscheinlichkeit für das Vorkommen des Farbwertes *rgb* in einem Bild an
  - P(Y=*fg*) ist die Wahrscheinlichkeit, dass ein Pixel im Bild zum Vordergrund gehört, P(Y=*bg*) analog
  - P(X=*rgb* | Y=*fg*) ist die bedingte Wahrscheinlichkeit, dass der Farbwert rgb zum Vordergrund gehört, analog für *bg*
  - Im Folgenden kürzen wir P(X=*rgb* | Y=*fg*) mit P(*rgb* | *fg*) ab, analog für *bg*

Segmentierung

- Pixel wird als Vordergrund klassifiziert, wenn

$$\frac{P(fg|rgb)}{P(bg|rgb)} > \tau$$

- Berechnung der Wahrscheinlichkeiten über *Regel von Bayes*

$$\frac{P(fg|rgb)}{P(bg|rgb)} = \frac{P(rgb|fg)P(fg)\cancel{P(rgb)}}{P(rgb|bg)P(bg)\cancel{P(rgb)}}$$

- Satz: *Regel von Bayes*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Ground-Truth-Daten erstellen

- Wahrscheinlichkeiten auf der rechten Seite der Gleichung

$$\frac{P(fg|rgb)}{P(bg|rgb)} = \frac{P(rgb|fg)P(fg)}{P(rgb|bg)P(bg)}$$

  sind einfach zu ermitteln z.B. durch manuelles Labeln und Histogram Counting

- Training: Zielobjekt aus verschiedenen Viewpoints und unter mehreren Beleuchtungsbedingungen aufnehmen

$$P(rgb|fg) = \frac{fg(rgb)}{\#\text{Pixel}_{\text{fg}}} \qquad P(rgb|bg) = \frac{bg(rgb)}{\#\text{Pixel}_{\text{bg}}}$$

$$P(fg) = \frac{\#\text{Pixel}_{\text{fg}}}{\#\text{Pixel}_{\text{fg}} + \#\text{Pixel}_{\text{bg}}} \qquad P(bg) = \frac{\#\text{Pixel}_{\text{bg}}}{\#\text{Pixel}_{\text{fg}} + \#\text{Pixel}_{\text{bg}}}$$

Grösse der Trainingsdaten

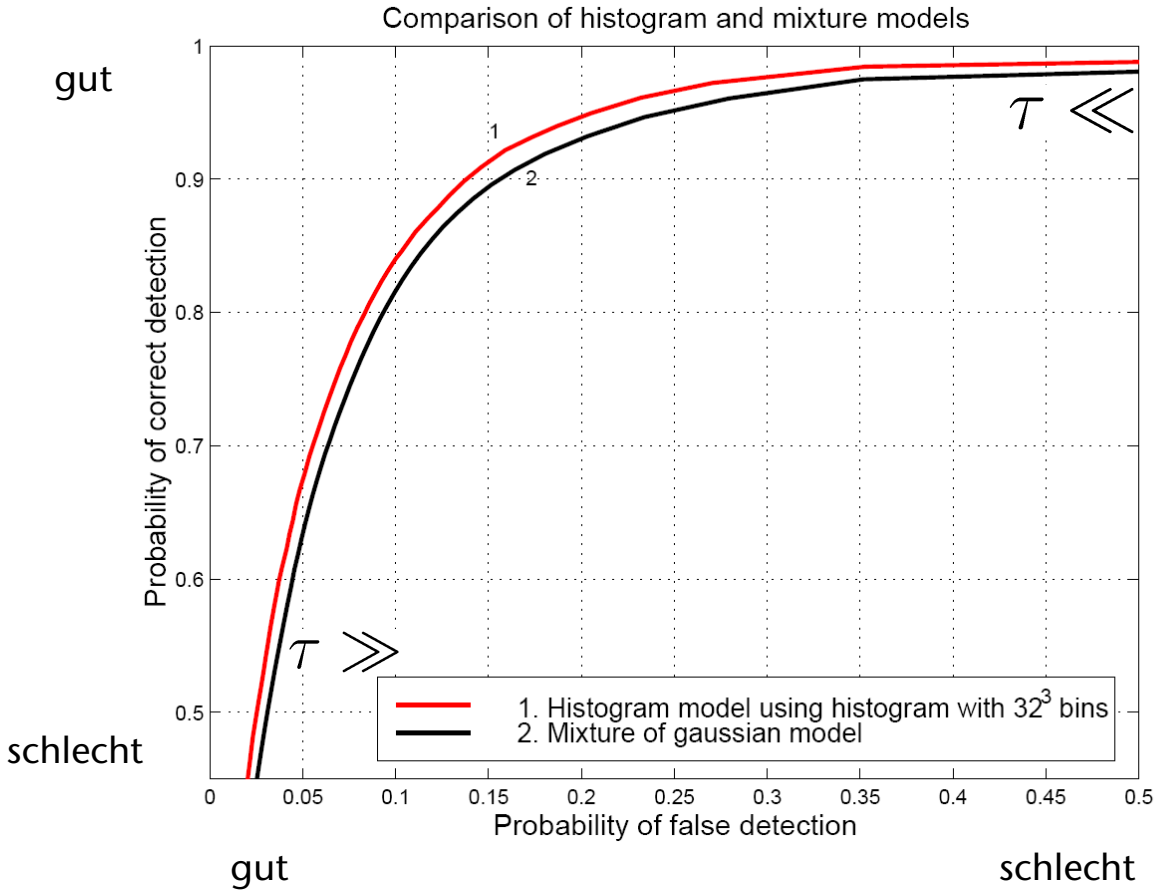
- Ziel: robuste Verteilung mit wenigen Bildern; z.B. durch
  - Mixture of Gaussians
  - Kernel Density Funktion
- Histogramm zwar genauer, **aber** viel größerer Trainingsdatensatz erforderlich
- Beispiel: Hautsegmentierung

Originalbild

Hautsegmentierung

Threshold

- Wahl des Schwellwertes ist Kompromiss zwischen "false positives" und "false negatives"

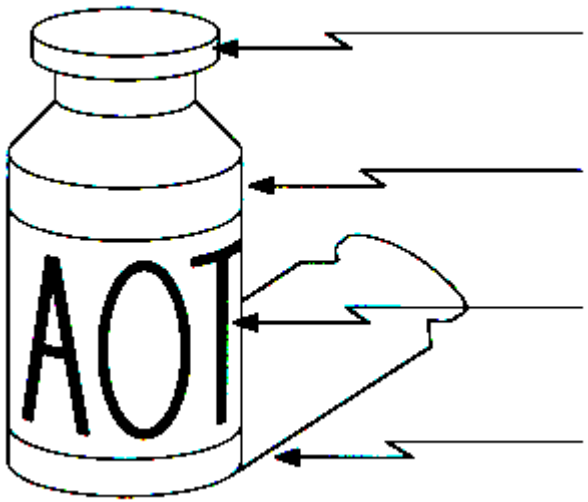


Grenzen des Verfahrens

- Hintergrundabhängig
  - Objektfarbe darf nicht zu häufig im Hintergrund vorkommen
- Beleuchtungsabhängig
  - Helligkeit der Lichtquelle(n)
  - Farbe der Lichtquelle(n)
- Speziell bei Hautsegmentierung: Personenabhängig
  - verschiedene Menschen haben verschiedene Hautfarbe

Bildkanten

- Bisher Objektfarbe als Feature
- Kanten beschreiben Form des Objektes
- Kanten entstehen durch mehrere Faktoren



Diskontinuität der Oberflächen Normale

Unterschied im Tiefenwert

Farbunterschied

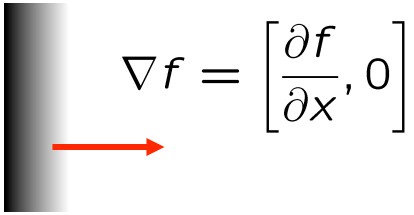
Beleuchtungsvariation

Extraktion

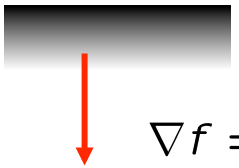
- Kanten sind Farbunterschiede pro Bildraumabstand, wird also durch Bildgradient repräsentiert

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

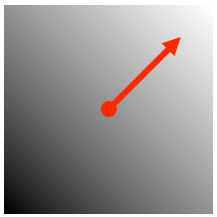
- Gradient zeigt in Richtung der größten Änderung



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- Richtung und Intensität des Gradienten

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right) \qquad \|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- Analytische Funktion eines Bildes nicht bekannt
- Deshalb diskrete Ableitung

$$\frac{\partial f}{\partial x}(x,y) \approx f(x+1,y) - f(x,y)$$

- Bekannteste Kantenoperatoren:

$$\begin{bmatrix} +1 & 0 \\ 0 & -1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix}$$

$$\begin{bmatrix} 0 & +1 \\ -1 & 0 \end{bmatrix} \quad \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} \quad \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}$$

Roberts Prewitt Sobel

  - Implementierung als Faltung

Canny-Edge-Detector

- Für viele CV-Algorithmen wird ein binäres Kantenbild benötigt
- Wir können bisher nur Kantenintensitätsbild berechnen

- Guter Algorithmus: Canny Edge Detector

- Canny's Zielsetzung:
  - Gute Kantendetektion: Der Algo soll soviele echte Kanten finden wie möglich
  - Gute Lokalisierung: Von Algo gefundene Kanten sollten so nah wie möglich an tatsächlichen Kanten liegen
  - Gute Kantenantwort: Eine Kante sollte nicht mehrfach gefunden werden
- Lösung kann durch Ableitung der Gaussfunktion approximiert werden

Algorithmus

- Gaussian Smoothing Filter auf Bild anwenden, z.B.

$$\frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

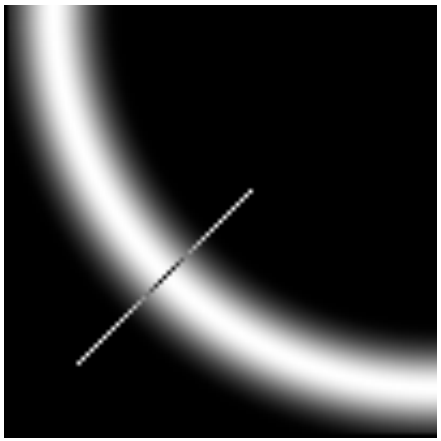
- Ableitung des resultierenden Bildes berechnen, z.B. Sobel Op.
- Non-Maximum-Suppression Algo anwenden (nächste Folie)
- Hysteresis Thresholding:
  - Sei K das Kantenbild und T_1, T_2 zwei Schwellwerte mit $T_2 > T_1$
  - markiere alle Pixel $K(x,y) \geq T_2$ als Kante
  - markiere alle Pixel $K(x,y) < T_1$ als nicht Kante
  - Markiere alle Pixel $T_1 \leq K(x,y) < T_2$ als Kante, wenn es einen Pfad von einem Kantenpixel nach *(x,y)* gibt, der nur Pixel mit Wert $\geq T_1$ hat

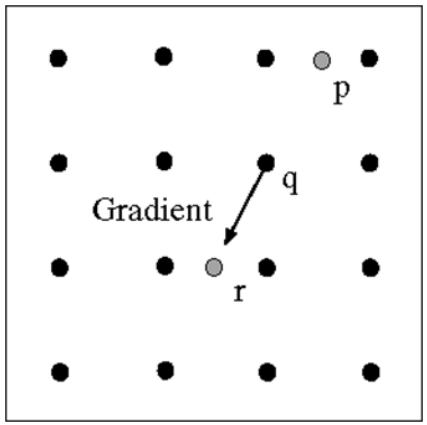
Non-Maximum-Suppression

- Idee: Wähle lokales Maximum als Pixel und lösche alle anderen Kantenpixel
- Algorithmus:
  - Betrachte jedes Pixel **q** im Kantenbild *K*
  - *Teste für benachbarte Pixel* **p** *und* **r***:*

$$K(\mathbf{p}) < K(\mathbf{q}) \ \wedge \ K(\mathbf{r}) < K(\mathbf{q})$$

  - *Falls ja* $\rightarrow$ **q** *ist lokales Maximum ;* setze **q** als *Kantenpixel*
    Sonst; setze **q** als *Hintergrundpixel*





Beispiel

Kantenoperator

Non-max-
suppression

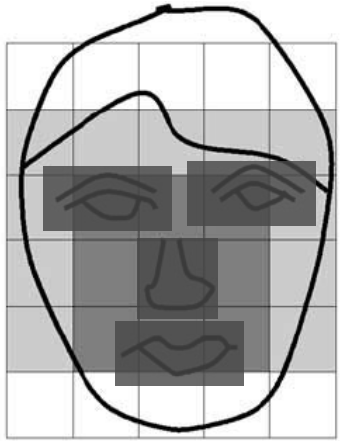
Hysteresis
Thresholding

Gesichtserkennung

- Unterscheide zwischen vier verschiedenen Problemstellungen
  1. Lokalisation: finde Position aller Gesichter in einem Bild
  2. Erkennung: bei gegebener Position des Gesichtes, ordne Gesicht einer bestimmten Person zu
  3. Emotion/Ausdruck: erkennen ob eine Person lacht, traurig ist etc.
  4. Tracking: Gegeben initiale Position, verfolge Gesicht über einen längeren Zeitraum
- Wir behandeln primär Lokalisation
- Tracking kann als Lokalisation pro Frame formuliert werden;
  - Ab Frame 2 ist grobe Position des Gesichtes bekannt
  - Daher ist die abzusuchende Bildregion deutlich kleiner (unter der Voraussetzung kontinuierlicher Bewegung)

Video

Template-Matching-Ansatz

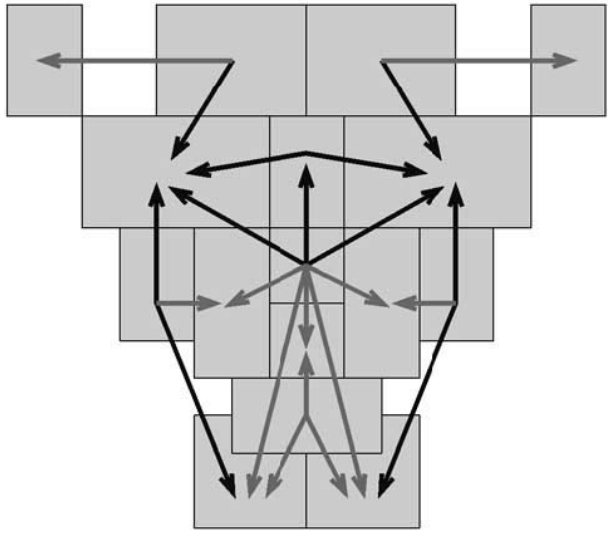


- Idee:
  - Erstelle ein Muster des Gesichtes.
  - Suche im Bild nach diesem Muster.
  - Melde Position des Gesichtes bei hinreichend guter Übereinstimmung
- Vorgehensweise
  - Extraktion invarianter Features, d.h. Features die möglichst robust gegenüber Störungen (*Beleuchtung, Hintergrund,* unterschiedliche *Gesichtsform)* sind
- Berechnung der Wahrscheinlichkeit für das Vorhandensein eines Gesichtes durch Kombination der Features

- Extraktion invarianter Features:
  1. Extraktion von low-level Features (z.B. Kanten, Hautfarbe)
  2. Basierend auf den low-level Features, berechne durch Template Matching potentiellen Positionen von Gesichtsfeatures wie
     1. *Augen(-brauen)*
     2. *Nase(-nlöcher)*
     3. *Mund/Lippen*
     4. *Wangen*
     5. *Gesichtsrand*

Konkretes Verfahren

- Idee: Helligkeitswerte im Gesicht variieren zwar von Pixel zu Pixel, aber Relationen im Großen bleiben erhalten



- Erstellen des Gesichts-Templates:
  - Definiere Template als Menge von rechteckigen Teilregionen des Gesichtes
  - Jede Region entspricht einem signifikanten Gesichtsfeature wie Augen, Mund, Nase, Wangen, Stirn
  - Feature sind durchschnittliche Helligkeitswert für jede dieser Regionen
    - kann entweder über Ground-Truth-Datensatz berechnet werden oder
    - durch Expertenwissen (d.h. manuell festlegen)
  - Speichere zu jedem paar von Regionen (mit Pfeilen markiert) Quotient der Helligkeitswerte

- Lokalisierung des Gesichtes im Bild:
  - Berechne im Eingabebild für alle möglichen Positionen die Regionen
  - Ein Regionenpaar matcht mit dem Temlate, wenn der Helligkeitsquotient einen bestimmten Schwellwert überschreitet
  - Bildposition wird als Gesicht klassifiziert, wenn mehr als eine bestimmte Anzahl an Regionenpaaren passen

Gestenerkennung

- Aufgabe: Gegeben *n* Handgesten, entscheide ob bzw. welche Geste im Bild zu sehen ist
- Anwendungen:
  - Gesten als Shortcuts in Programmen
  - Simple Navigations in VR
  - Automatische Übersetzung von Gebärdensprachen in Wort/Schrift
- Zwei Grundsätzlich verschiedene Herangehensweisen
  - Modellbasierte Methoden
  - Bildbasierte Methoden

Video

Modellbasierte Methoden

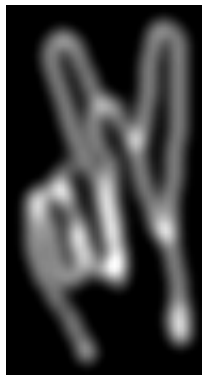
- Basis ist ein Modell der Hand bzw. der Handgesten
  - Als Modell kann z.B. eine künstliche/gerenderte Hand dienen
- Ziel:
  - Vergleiche die Gesten, generiert durch das Handmodell mit dem Eingabebild
  - Bestimme das Modell, das am besten zum Bild passt
    - Kantenabstandsmass (*Chamfer*, *Hausdorff*, direkter Vergleich)
    - Überlappung der Handregionen von hautsegmentiertem Eingabebild und Template
  - Passt die Modellgeste hinreichend gut, wird die Geste vom Verfahren erkannt, sonst wird Bild als Hintergrund erkannt

Ein modellbasierter Ansatz

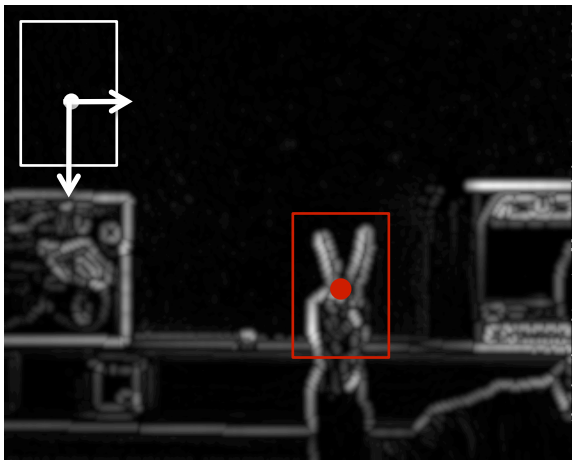
- Gegeben: Eine Handgeste *G* und Eingabebild *I*

- Ziel: Finde die Position im Eingabebild *I*, die am besten zur Handgeste *G* passt
- Verwende Kanten als Feature zum Vergleich von Eingabebild und Handgeste

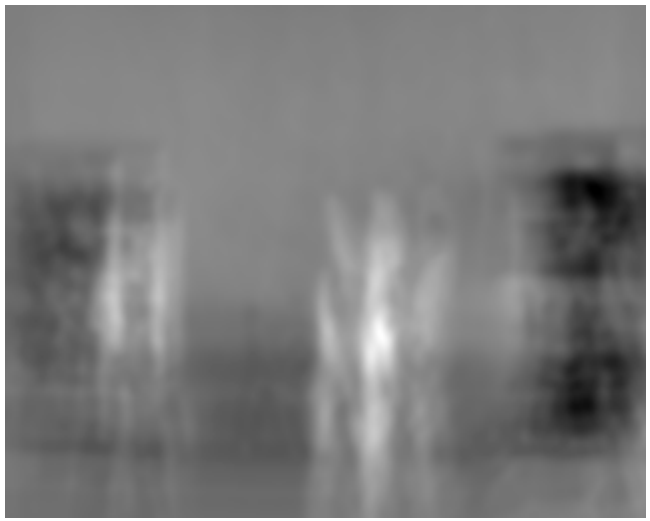
Confidence Map



- Template T_G zur Handgeste *G* generieren
  - Rendere Handgeste *G* und extrahiere daraus Kanten
  - Wende Smoothing auf Kantenbild an



- Matching Algorithmus:
  - Für alle Pixel im Eingabebild
    - Extrahiere Kanten aus Eingabebild *I*, Ergebnis ist Kantenbild I_K
    - Berechne "Ähnlichkeit" zwischen Template und lokale Bildumgebung zum Pixel durch pixelweise Mutliplikation von Template und Kantenbild
- Dies beschreibt eine Faltung des Kantenbildes I_K mit dem Template T_G (genauer mit Spiegelung des Templates)



- Ergebnis nennt man *Confidence Map*
  - Gleiche Auflösung wie Eingabebild
  - Lokale Maxima geben beste Kandidaten für Position der Handgeste an
- Aufwand der Faltung ist O(*n***m***w***h*)
  - *nxm* ist Auflösung von I_K und *wxh* Auflösung von T_G
- Analogie zwischen Faltung im Ortsraum und Mutliplikation im Fourierraum (nächste Folie)
- Aufwand für *Fast-Fourier-Transformation(FFT):* O(*n***m**log*n**log*m*)
- Aufwand für Mutliplikation im Fourierraum O(*n***m*)
- Faltung über Fourierraum lohnt wenn log*n**log*m* < *c***w***h*

Video


Original image
Template index: our approach; Position: hand labeled
Chamfer based approach
Our approach