

# Mensch - Computer - Interaktion

1/0

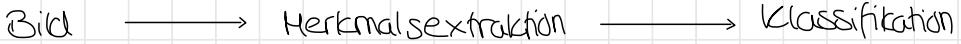
- WIMP Paradigma = Windows, Icons, Mouses, Pointers
- Gesten
- Gesten + Sprache: "Put that there" - Paradigma
- Körperbewegungen

## Anwendungen "sehender" Systeme

- Wahrnehmung des Menschen durch das System ist Voraussetzung für sinnvolle Interaktion und für angemessenes Handeln
- Wer / Was / wo / wann / wie / zu, mit wem?
  - Sprache, Gesik, Körperhaltung, Blickrichtung, Mimik, ..
  - Smart houses, Assisted living
  - Smart cars (Aufmerksamkeit des Fahrers, ...)
  - Smart rooms (Vollst. Protokollierung von Seminaren)
  - Smarte OP-Räume
  - Patienten monitoring
  - Blindenassistenzsysteme

## Computer Vision

Klassisch:



Merkmale:

- Kanten, Farbe, Segmentierung, SIFT, Histogramme, ..

Klassifikation:

- machine learning: SVM, KNN, ...

- ①
- Merkmale stark anwendungsabhängig
  - keine gelernten Features

## modern:

Bild  $\longrightarrow$  Klassifikation

- + · Ende-zu-Ende Lernen
  - $\hookrightarrow$  keine handische Feature extraction
  - hierarchische Daten- Repräsentation
  
- · braucht viele Trainingsdaten
  - rechenintensiv
  - Umgang mit unbekannten / neuen Klassen?

## MUSTERERKENNUNG

### Klassifikation

= gegeben Eingabe  $\vec{x}$  : bestimme Klasse  $w_i$   
 $\hookrightarrow w_i = \text{diskret} \rightarrow$  Klassifikation  
 $\in \mathbb{R}/\text{vector} \rightarrow$  Regression

### Bayesian Classifier

Bayes Regel:

$$P(w_i | \vec{x}) = \frac{p(\vec{x} | w_i) p(w_i)}{p(\vec{x})}$$

↑ likelihood                      ↓ prior probability

↑ normalization factor

wobei  $p(\vec{x}) = \sum_i p(\vec{x} | w_i) P(w_i)$

$\rightarrow$  entscheide Klasse  $w_i$  mit max. posterior prob.

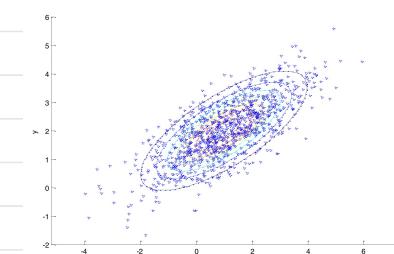
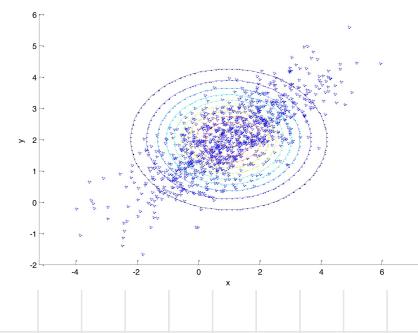
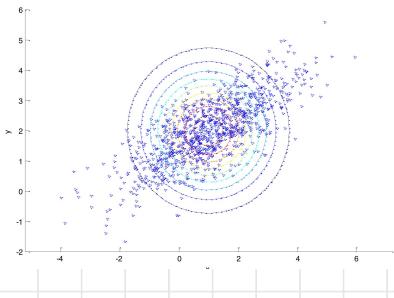
Problem: Verteilung  $p(\vec{x} | w_i)$  oft unbekannt  
 $\Rightarrow$  Schätzen Verteilung

## Gaussian Classification

Annahme  $p(\vec{x} | w_i) \sim N(\mu, \sigma^2)$

=> schätzt nur noch  $\mu, \sigma^2$  aus Daten (mit max. l(h))

- Param reduction durch Restriktion der Kovarianzmatrix:



- Kovarianzmatrix ist Einheitsmatrix

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

- Kovarianzmatrix ist Diagonalmatrix mit Params  $\alpha, \beta$

$$\begin{pmatrix} \alpha & 0 \\ 0 & \beta \end{pmatrix}$$

- Kovarianzmatrix beliebig

$$\begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix}$$



- falls Schätzung der Verteilung (doch) nicht Realität entspricht  
=> sehr schlechter Classifier => GMMs

## Gaussian Mixture Models

- kombiniere mehrere Gauß-Verteilungen mit versch.  $\mu, \sigma^2$

$$p(\vec{x}) = \sum_i w_i \frac{1}{(2\pi)^{d/2} |\sigma|} \exp \left[ -\frac{1}{2} (\vec{x} - \vec{\mu})^\top \cdot \frac{1}{\sigma^2} (\vec{x} - \vec{\mu}) \right]$$

$$\text{mit } \sum_i w_i = 1$$

- + beliebige Verteilungen abschätzbar  
↓
- brauche viele Gauß-Verteilungen, Params schwer zu schätzen

## Expectation Maximization Algo

→ um Params der Verteilungen zu schätzen:

schätze welche Punkte zu welcher Verteilung gehören

1. Init params der GMM zufällig

2. Loop bis Konvergenz:

- Expectation Step:

· berechne  $w_{ik}$  mit  $p_{ij}$ , dass Punkt  $i$  zu Verteilung  $j$

- Maximization Step:

· berechne neue GMM Params basierend auf Zuordnung

## Verfahren

### parametrisch

- Annahme über zugrunde liegende Verteilung  
→ Schätzt Params

- (+) weniger Trainingsdaten nötig
- (-) funktioniert nur falls Modell Daten gut beschreibt

zB Gaussian, GMMs

### generativ

- Methode die  $p(\omega_i)$ ,  
 $p(x_i|\omega_i)$  ↗ erwartet gen. neuer Daten

explizit modelliert

### supervised

- Training mit gelabelten Daten
- (-) viele Daten nötig, teures Labeling

### nicht-parametrisch

- keine solche Annahme
- (+) funktioniert für alle Arten von Verteilungen
- (-) mehr Trainingsdaten nötig

zB Parzen window, k-Near. nei.

### diskriminativ

- $P(\omega_i|x)$  direkt modelliert / direkt Entscheidung w. gegeben  $x$

- (+) einfacher zu trainieren

zB Perceptron, Linear SVM

### unsupervised

- finden von Mustern ohne Labels
- (-) kein Einfluss auf Ergebnis

**Perzepton** findet Trennebene falls Daten linear separierbar

$$y = \vec{w}^T \vec{x} + w_0 \rightsquigarrow y = \vec{w}^T \vec{x} \text{ mit } \vec{w} = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \end{pmatrix}$$

1. Init  $\vec{w} = 0$  (oder zufällig)

2. Berechne Klassifikation für Datum  $\vec{x}$ :

$$y(\vec{x}) = \text{sign}(\vec{w}^T \vec{x})$$

3. Falls korrekt: goto 2

else:

$$\vec{w}' = \vec{w} - \eta \cdot y(\vec{x}) \cdot \vec{x}$$

$$\eta = \text{learning rate}$$
$$0 < \eta < 1$$

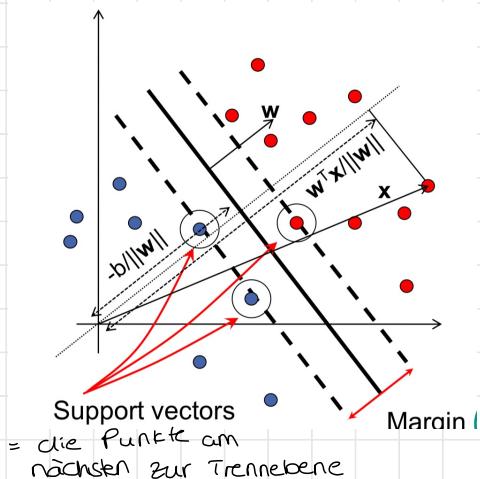
4. Falls alle korrekt: done

else: goto 2

⊖ Problem: viele Trennebenen möglich  $\rightarrow$  welche ist die Beste?

## Support Vector Machines

• Finde Trennebene bei der Daten möglichst weit von Trennebene entfernt bei korrekter Klassifikation



$$\text{Margin} = \frac{2}{\|w\|}$$

$$\Rightarrow \max \cdot \text{margin} \frac{2}{\|w\|}$$

$$\hat{=} \min \cdot \|w\|^2$$

$\Rightarrow$  Optimierungsproblem mit Randbedingung (korrekte klassif.)

= quadratic optimization problem  
 $\rightarrow$  1 Minimum  $\curvearrowleft$

## Soft margin

- Outlier / Noise verschlechtern Ergebnis
  - Ziel: max. margin, ignoriere Outlier
  - $\min \|\omega\|^2 + C \sum_i \xi_i$  ← constraint violations
  - erlaube Violations:
- $$y_i (\omega^\top x_i + b) \geq 1 - \xi_i \quad (\xi_i > 0)$$

## Interpretation $\xi_i$ :

- $\xi_i = 0$ : korrekte Klassifikation außerhalb Margin
- $0 < \xi_i < 1$ : innerhalb
- $\xi_i > 1$ : falsche Klassifikation
- $\sum_i \xi_i$ : obere Grenze für # Trainingsfehler

## nicht-lineare Daten

- transformiere Daten in höher dim. Raum in dem Daten linear separierbar

kernel Trick: → keine Berechnung in höherer Dim. nötig! ☺

- formulierte Algo so, dass Feat. vektoren nur als Punktprodukt nutzen Kernel-Funktion
- ↪ ausprobieren welche am Besten funktioniert

## Linear

- nur 1 Skalarprodukt
- nur  $\vec{w}$
- schnell
- nur 1 Param

## vs.

- Speed
- Memory
- Training

## kernel SVM

- # (sup. vect.) kernel evals
- alle sup. vect.
- langsam
- viele Params

## K-nearest - neighbor

- betrachte K NN von  $x \rightarrow$  weise  $x$  häufigster Klasse zu

(+) · kein Informationsverlust

(-) · viele Daten nötig um Merkmalsraum gut abzudecken  
· viel Speicher nötig  
· gute Distanzmetrik?

## Clustering

### K-means

- init zufällig  $k$  Cluster Mittelpunkte
- Ordne Punkte Clustern zu  $\hookrightarrow$  loop
- berechne Mittelpunkte neu

(+) einfach, effizient

(-) ·  $k$  muss a priori bekannt sein  
· Ergebnisse abh. von init  
· schlecht für überlappende Cluster

### agglomerativ hierarchisch

- jeder Punkt eigenes Cluster  
 $\rightarrow$  iterativ nächste Nachbarn mergen (Ergebnis ist Baum)

Zeit  $O(n^2 \log n)$  Speicher  $O(n^2)$

$\downarrow$   
wo abschneiden  
abh. von Task!

## Curse of Dimensionality

- Intuition von LA nur in niedrigen Dim.
- Classifiers oft besser in niedrig. dim. Räumen

ABER: CV Input oft hoch dim. :/

## Sample Density

für gleiche Dichte in höher dim. Räumen: exponentielle Zunahme

Intervall wkt zB Gauß-Verteilung  $x \sim N(0, \sigma^2 I)$

- 1D: fast alle samples zw.  $-2\sigma, 2\sigma$  ( $P|x_0| < 2\sigma = 95.45\%$ )
- 2D: immer noch
- :
- 100D: fast alle samples außerhalb  $(95.45\%)^{100} = 0.85\%$

## Volumen hyperspheres vs. hypercube

Verhältnis Volumen (sphere) zu Volumen (cube):			
2D:	$\pi$	:	4
3D:	$4\pi/3$	:	8
:	:	:	:
8D:	$\pi^4/24$	:	256
			$\approx 2\%$

## BENCHMARKING

### Klassifikation

		Prediction	
		P	N
Ground Truth	P	TP	FN
	N	FP	TN

$$\text{Precision} = \frac{TP}{TP+FP}$$

$$\text{Recall} = \frac{TP}{TP+FN} = TPR$$

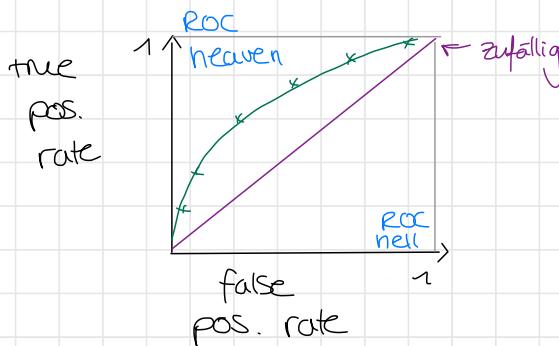
$$1 - \text{Precision} = \frac{FP}{FP+TP}$$

### ROC

$$\cdot \text{ true positive rate} = \frac{TP}{Pos} = \frac{TP}{TP+FN} = \text{Recall}$$

$$\cdot \text{ false positive rate} = \frac{FP}{Neg} = \frac{FP}{FP+TN}$$

=> Plot



### Lokalisierung

Intersection over Union (IoU) = overlap

$$IoU = \frac{|GT \cap PI|}{|GT \cup PI|}$$

GT = ground truth, PI = prediction

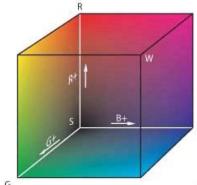
# FACE DETECTION

## Color-Based

Idee: Hautfarbe von Menschen konsistent, versch. von vielen anderen Objekten

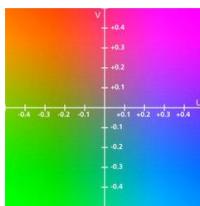
### Farträume

- RGB (Farbwürfel):

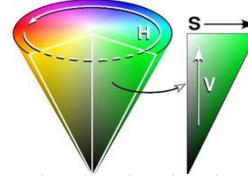


- Class Y Räume:

- Y = Helligkeit
- anderen Z = Farbigkeit

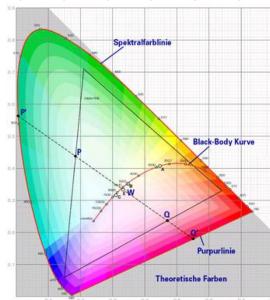


- HSV (Farbkegel):



- perceptually uniform spaces:

- wahrgenommene Farbdifferenz  $\hat{=}$  unterschiedl. der Farbwerte
- euklidische Distanz für Farbvergleiche



### Chromatische Farträume

2 Kanäle codieren Farbinfo

zB HS (von HSV)

uv (von YUV)

rg (von RGB):

$$r = R / (R + G + B)$$

$$g = G / (R + G + B)$$

Motivation: robuster ggüüber versch. Hautfarben, Beleuchtung  
→ nicht der Fall :-)

## Probleme color-based

- ref. Farbe auch abh. vom Lichtspektrum der Lichtquelle

## Histogramm - Basiert (nicht-parametrisches Verfahren)

- + funktioniert in Praxis ganz gut
- hoher memory Bedarf (abh. von Binning)  
brauche viele gelabelte Haut / nicht-Haut Samples  
→ nutze nicht ges. Bild, sondern nur relevante Bildausschnitte

## Histogram Backprojection

- Backprojektion erstellen:
  - mappe jeden Pixel auf Wert des Histogramm-Bins des Farbwerts
- falls Wert  $> \Theta$ : Hautpixel

+ einfach, schnell

- schlecht falls Objekt mehrfarbig

## Histogram Matching

- baue Histogramm auf Bildausschnitt
- vergleiche Histogramm mit Ziel-Histogramm  
↳ Distanzmetrik nötig!

+ bessere Ergebnisse

- rechenintensiver

## Gaussian Models (parametrisches Verfahren)

- Annahme: Hautfarbe normalverteilt mit  $G(x; \mu, C)$  Covarianzmatrix

$\rightarrow \mu, C$  geschätzt basierend auf Trainingsset

Pixel ist Haut falls:  $p(x|skin) > \theta$  oder  $p(x|skin) > p(x|non-skin)$

## GNMs

- beschreibe Hautfarbe durch  $> 1$  Gauß-Verteilung
- Schätze Params mit EM-Algo

## Bayes Classifier

- Pixel ist Haut falls  $\frac{p(x|skin)}{p(x|non-skin)} \geq \frac{p(non-skin)}{p(skin)}$

$\rightarrow p(x|w)$  schätzen mit Histogrammen:

$$p(x|w_i) = \frac{h_i(x)}{\sum_x h_i(x)} \leftarrow \begin{array}{l} \text{Pixel der Klasse } w_i \\ \text{mit Wert } x \end{array}$$

## Hautfarbe $\rightarrow$ Gesichter

### Post-processing

- Erosion (min. Wert im Filter-kernel-Bereich)
  - $\hookrightarrow$  entfernt Randpixel
- Dilatation (max. Wert ... )
  - $\hookrightarrow$  verbreitert Ränder
- Opening: Erosion  $\rightarrow$  Dilatation
- Closing: Dilatation  $\rightarrow$  Erosion

## Heuristiken zur BBox - Findung

- größtes Haut - Cluster
- Ellipse - fitting
- zeitliche (tracking) Info
- facial feature detection

## Neuronale Netze

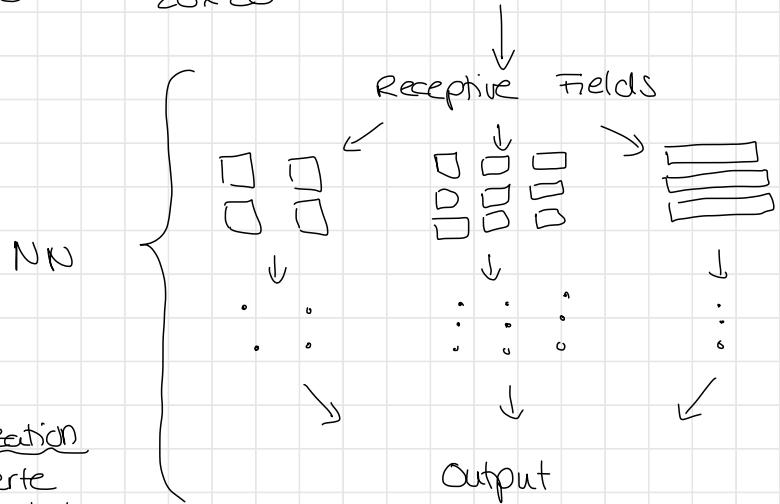
End 2 End Ansatz : Bild  $\leadsto$  BBox

### erstes einfaches NN

(Rowley et al. 1998)

Fenster

Input  $\rightarrow$  versch. Größe  $\rightarrow$  Bildausschnitte  $\rightarrow$  Preprocessing  
Bild  $\rightarrow$  20x20



### Preprocessing:

Histogram Equalization

- dehnen Pixelwerte auf ges. Intensitätsspektrum

1. Berechne  $p(x_i) = \frac{n_i}{n}$   
= Wkt dass Pixel wert i hat

2. Berechne kumulative Verteilungsfkt.  $c(i) = \sum_{j=0}^i p(x_j)$

3. Transformiere Pixel  $y_i^t = c(i) \cdot (\text{max} - \text{min}) + \text{max}$

# DEEP LEARNING BASICS

end-to-end learning

## DL Aufgaben

Klassifikation



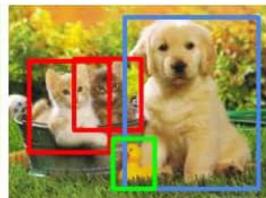
CAT

Klassifikation  
+ Lokalisierung



CAT

Object  
Detection



CAT, DOG, DUCK

Instance  
Segmentation



CAT, DOG, DUCK

## ImageNet Dataset

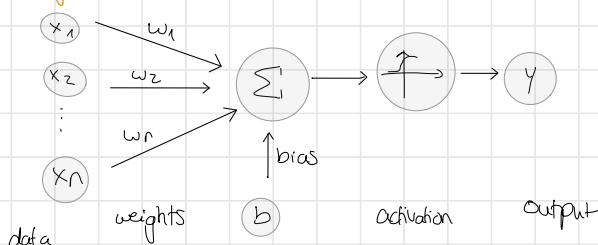
- > 1 Mio gelabelte Daten
- 1000 Klassen

## Vorteile

- End2End learning  $\Rightarrow$  einfacher
- oft besser als klassische Ansätze
- hierarchische Features
  - ↪ frühe Layer : low-level Features (Kanten, ...)
  - ↪ späte : high-level Features (Augen, ...)

## MLP

### Single Layer Perception:



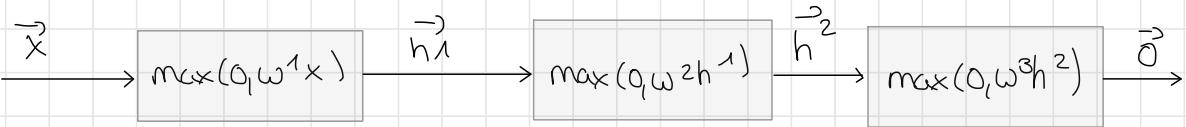
$$y = \text{activation} (w_1 x_1, w_2 x_2, \dots, w_n x_n + b)$$

activation muss nicht-linear sein

→ Kombination linearer Funktionen ist linear

=> nur lineare Probleme lösbar

## Multi-Layer Perceptron



$$\vec{\theta} = \text{Modell} = \omega^i, b^i \text{ für alle Layer}$$

$$\vec{h}_1 = \max(0, \omega^1 x + b^1)$$

$x \in \mathbb{R}^D$  \*Pixel ausgerollt

$$\omega^1 \in \mathbb{R}^{N_1 \times D}$$

$$b^1 \in \mathbb{R}^{N_1}$$

$$h^1 \in \mathbb{R}^{N_1}$$

⋮

$$\vec{o} = \max(0, \omega^3 h^2 + b^2)$$

- hidden units ≈ Klassifikator / Feature Computer

## Parameter

- Gewichte (werden gelernt)
  - Activation function
  - Loss function
  - Learning rate
  - # layers
  - # units pro layer
- } ausprobieren

# Netz-Bausteine

## Softmax

- Wkt von Klasse k geg. Input  $\vec{x}$

$$p(c_k = 1 | \vec{x}) = \frac{e^{o_k}}{\sum_j e^{o_j}}$$

$$\vec{o} = \begin{pmatrix} o_1 \\ o_2 \\ \vdots \\ o_k \\ \vdots \\ o_c \end{pmatrix} = \text{Ausgabe}$$

(! keine echte Wkt-Verteilung)

## Lernverfahren

### Gradient Descent

$$\vec{\theta} = \vec{\theta} - \eta \frac{\partial L}{\partial \theta} \quad 0 < \eta < 1 \text{ Learning rate}$$

## Momentum

- betrachte Gradient vom Schritt davor

$$\Delta_t = \mu \Delta_{t-1} - \eta \frac{\partial L}{\partial \theta} \quad \mu \text{ ablicherweise } 0,9$$

$$\vec{\theta} = \vec{\theta} + \Delta_t$$

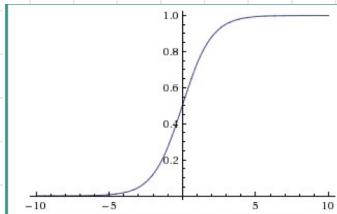
=> Schnellere Änderung falls Gradient in gleiche Richtung  
 $\rightarrow$  schnellere Konvergenz  
 langsamere andere  
 $\rightarrow$  weniger Fluktuation

## Tricks

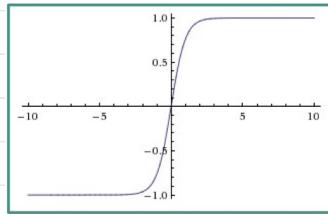
- Dropout (gegen Overfitting durch "Hot-Paths" gut)
- Pre-Training (lernt generische Features, z.B. Kanten,..  $\rightarrow$  als init.)
- Data Augmentation
- Hard negative mining (Netz trainieren, dann nochmal auf fehlerhaften nachtrainieren)

## Activations

Sigmoid  $\sigma = \frac{1}{1+e^{-x}}$



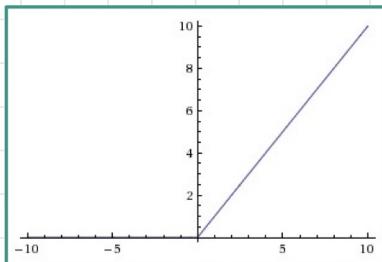
tanh(x)



⊖ vanishing gradient

→ Ableitung  $\approx 0$  für große / kleine x

ReLU = max(0, x)



⊕ kein van. grad

⊖ dying ReLU

→ Lsg: leaky ReLU  $\max(0, \alpha x, x)$

## Loss - Funktion

negative - log

Klassifikation

hinge - loss

$$L(\vec{x}, y; \theta) = -\sum_j y_j \log p(c_j | \vec{x})$$

$$L(x, y; \theta) = \sum_j \max(0, 1 - x_j y_j)$$

L1 (MAE)

Regression

L2 (MSE)

$$L(\vec{x}, y; \theta) = \sum_j |y_j - x_j|$$

$$L(\vec{x}, y; \theta) = \sum_j (y_j - x_j)^2$$

## Backpropagation

Modell  $\vec{\Theta} = \vec{w}^n, \vec{b}^n \quad \forall n \in \text{Layer}$

→ Ableitung mit Kettenregel

$$1. \frac{\partial L}{\partial o} = p(c|x) - y$$

$$2. \frac{\partial L}{\partial w^3} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial w^3} = (p(c|x) - y) \cdot h^2$$

$$3. \frac{\partial L}{\partial h^2} = \frac{\partial L}{\partial o} \cdot \frac{\partial o}{\partial h^2} = (p(c|x) - y) \cdot w^3$$

$$4. \frac{\partial L}{\partial w^2} = \frac{\partial L}{\partial h^2} \cdot \frac{\partial h^2}{\partial w^2}$$

$$5. \frac{\partial L}{\partial h^1} = \frac{\partial L}{\partial h^2} \cdot \frac{\partial h^2}{\partial h^1}$$

# CNNs

## Probleme FC:

- $200 \times 200 \text{ px} \rightarrow 40K \text{ hidden units} \Rightarrow \approx 1.6 \text{ B params}$   
→ Ressourcenschwund
- keine lokalen Korrelationen
- nicht genügend Trainingsdaten

## → Conv Layer:

- $200 \times 200 \text{ px}$
- 86 kernels  $\approx 11 \times 11 \Rightarrow \approx 1 \text{ M params}$

- Verbindung hidden unit  $\rightarrow$  Bildausschnitt
- geteilte Gewichte
- viele Filter möglich
- "Rezeptive Felder"

## Convolution Layer

$$h_j^n = \max \left( 0, \sum_{k=1}^K h_k^{n-1} * w_{kj}^n \right)$$

↑  
output  
feature map

↑ activation

↑ input feat.  
map

↑ Filter-kernel

## Dimensionen

$$\text{Input } H^{l-1} \times W^{l-1} \times n_c^{l-1} \xrightarrow{\text{filter in layer } l-1} \text{Output } H^l \times W^l \times n_c^l$$

$$H^l = \left\lceil \frac{H^{l-1} + 2p^l - f^l}{s^l} + 1 \right\rceil, W^l \text{ analog}$$

## Parameter Layer l

$$n_c^l \times \underbrace{f \times f}_{\text{\# Filter}} \times n_c^{l-1}$$

↑  
kernel-size  
↑  
# Kanäle Input

## Pooling Layer

= kompakte Zusammenfassung von Infos

- Summierung pooling
- Max pooling

- (+) · Invarianz gegenüber Bildtransformationen  
 · bessere Robustheit gegen Rauschen

üblich: Overlapping pooling  
 → reduziert spatial layer size

typisch  $S=2$ ,  $f=3$

## Dimensionen

$$\text{Input} \quad w^{l-1} \times h^{l-1} \times n_c^{l-1}$$

$$\text{Output} \quad w^l \times h^l \times n_c^{l-1}$$

$\uparrow$  gleich?

$$w^l = \left\lfloor \frac{w^{l-1} - f^l}{s^l} + 1 \right\rfloor, h^l \text{ analog}$$

## Batch Normalization

Problem: Internal Covariance Shift

- Layer  $l$  bekommt als Eingabe die Ausgabe von Layer  $l-1$
  - durch Grad Dec. ändern sich diese Werte ständig
  - je höher Learning rate desto stärker die Änderung
  - Layer  $l$  lernt langsam da sich Input ständig ändert
- Lösung: Normalisiere Eingaben auf  $\mu=0, \sigma^2=1$

Alg.: Normalisiere pro Featutemap über alle Elemente eines Batches

1) Berechne empirisches Mittel  $\mu_k$ , Varianz  $\sigma_k^2$  des Batches für jede Dimension (Featutemap) einzeln

2) Normalisiere:

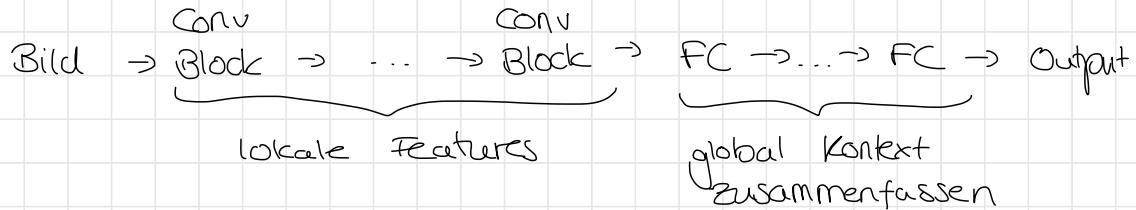
$$\hat{x}_k = \frac{x_k - \mu_k}{\sqrt{\sigma_k^2 + \epsilon}} \quad \leftarrow \text{kleine konst.} \rightarrow \text{numerische Stab.}$$

3)  $y_k = \gamma_k \hat{x}_k + \beta_k \rightarrow$  alle Activations auf  $\mu=0, \sigma^2=1$   
 $\uparrow \quad \nearrow$  limitiert Netz  
lernbare Params  $\rightarrow$  erlaubt Reskalierung durch Netz  
 $\rightarrow$  lernet Netz selbst durch  $\gamma_k, \beta_k$   
( $\beta_k$  macht  $b'$  überflüssig)

## Conv Block

Conv → Nicht-lin. → Normalisierung → Pooling

## CNN



## Dropout

→ z.B. Wkt 50%

- beim Training zufällig Neuron-Output auf 0 setzen  
→ verringere Abh. zw. Neuronen  
⇒ jedes Neuron gezwungen zu lernen

## Implementierung

- CNNs  $\hat{=}$  Matrixmult. / - add.  $\Rightarrow$  GPUs !

## Multi GPU :

- Datenparallelismus : Worker trainieren gleiches Modell auf versch. Daten  
→ Gradienten teilen
- Modellparallelismus : Worker trainieren versch. Teile des Modells auf gleichen Samples  
→ Neuronenaktivierungen teilen

# FACE RECOGNITION

## Face Recognition > Biometrische Verfahren

- keine physische Interaktion nötig
- keine extra HW nötig (nur Kamera)
- passive Identifikation von Masse

## Anwendungen

- Entertainment: Spiele, VR, Trainingsprogramme, ...
- Sicherheit: Device login, Surveillance, ...

LFW Dataset = labeled faces in the wild

- pre-deep learning: Menschen deutlich besser

## Schwierigkeiten

- allg. Object recogn.: viele Klassen, große Unterschiede  
face recogn.: eine Klasse, kleine Unterschiede  
→ Beleuchtungsänderungen, Orientierung  
= größer Unterschiede als versch. Gesichter ::
- geringe Bildauflösung
- Make-up

## Paradigma

### Closed-Set

- Gesichter aus DB wiedererkennen "Wer ist das?"  
→ Metrik = # korrekte

## Open-Set

- kenne ich diese Person und wenn ja wer ist das?
  - False accept: Pers. als bekannt erkannt, aber unbekannt
  - False reject: unbekannt bekannt
  - False classify: falsche Person erkannt

# Klassische Ansätze

## Feature-based

- fiducial points  
(Mundwinkel, ...)
- Distanzen, Winkel, Bereiche, ...
- geometrisch

vs

## Appearance-based

- heuristisch
- fiducial regions
- statistisch

## Beobachtungen

- 3D-Bild  $\rightarrow$  2D-Bild
- hochauflösendes 2D-Bild  $\rightarrow$  3D-Bild
- 4-5 gute 2D-Bilder  $\rightarrow$  3D-Bild / 2D + 3D-Bilder

## Feature-based

- zB
- Augenbrauendicke, vertikale Pos. der Brauen
  - Augenmittelpunkte
  - Biegsungsgrad der Braue
  - Nasenpos. + -breite
  - Mund pos. + -breite + -höhe
  - Kinnform
  - Gesichtsbreite auf Nasenhöhe
  - ...

## → Klassifikation

NN mit Distanzmetrik Mahalanobis Distanz

$$\Delta_j(x) = (x - m_j)^T \Sigma_j^{-1} (x - m_j)$$

Eingabe Gesicht       $\uparrow$       aug. Vektor der j. Person beschreibt       $\uparrow$       Kovarianzmatrix

- Personen durch aug. feature Vektor beschrieben
- Verteilung auf Trainingsdaten geschätzt

## Appearance - Based

holistic

- ges. Gesicht verarbeiten

local / fiducial

- Regionen (Mund / Augen...) getrennt verarbeiten

## Preprocessing

- Gesichter an Landmarken ausrichten z.B. Pupillen
- entferne Translation, Rotation, Skalierung, Hintergrund

## Holistische Verfahren

### Eigenfaces

- Gesicht = Punkt im hochdim. Bildraum
- versch. Gesichter teilen Eigenschaften
- > beschreibe Gesichter bzgl. der Unterschiede
  - ~ projiziere auf niedr. dim. Raum
  - ~ Klassifikation durch Distanzmaße

### PCA

Ziel: finde Vektoren die Verteilung der Gesichter im Bildraum am Besten beschreibt

→ principal components = Eigenvektoren der Kovarianzmatrix der Gesichter

### Training:

- Trainingsset  $Y = [y_1, \dots, y_k]$  aus  $k$  Gesichtern

- mean face  $m = \frac{1}{k} \cdot \sum y_i$

- Kov. matrix  $C = (Y - m)(Y - m)^T$

$$\Rightarrow D = U^T C U \quad D = \text{Eigenwerte}, \quad U = \text{Eigenvektoren}$$

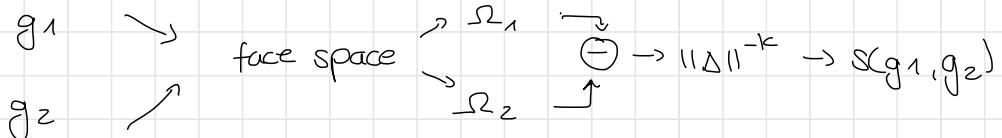
$\Rightarrow$  wähle  $\Omega = U^T (Y - m)$  als repr. Koeffizienten

## Testing:

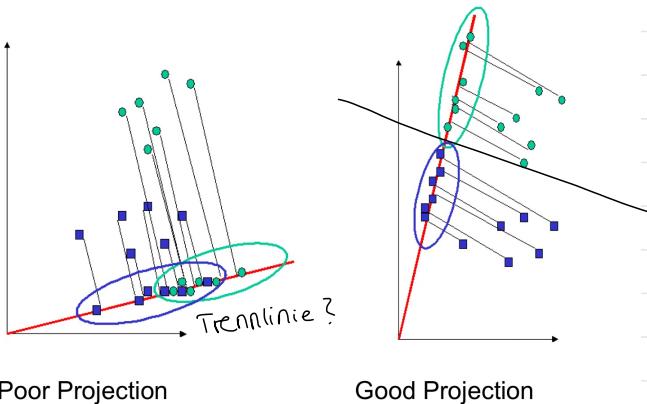
projizierte Gesicht in face space:

$$\underline{\Omega} = \mathbf{U}^T (\mathbf{y} - \mathbf{m})$$

→ finde Kandidat  $k$  mit  $\min \epsilon_k = \| \underline{\Omega} - \underline{\Omega}_k \|$



- kein Wissen über kritische Aussehenänderung einbezogen (durch zB Gesichtsausdruck)
- keine Unterscheidung zw. Form, Aussehen  
→ Beleuchtungsänderungen problematisch
- PCA nutzt keine Klasseninfo  
→ nicht optimal für Diskrimination



## Erweiterung: View-based Eigenspaces

- geg.  $N$  Personen mit  $M$  Orientierungen  
↳ bau  $M$  versch. Eigenspaces

## Bayesian Face Recognition

### Bayesian Similarity Measure

- Unterschiede zw. 2 Bildern nur durch typische Änderungen innerhalb einer Person (Gesichtsausdruck,...) miteinbeziehen
- Vergleiche inter-Personen-Varianz und intra-Personen-Varianz

=> **dual PCA**

$$\Omega_I = \{ \Delta = x_i - x_j : L(x_i) = L(x_j) \}$$

$$\Omega_E = \{ \Delta = x_i - x_j : L(x_i) \neq L(x_j) \}$$

$$\xrightarrow{\text{Bayes rule}} P(\Delta | \Omega_I) P(\Omega_I)$$

$$\text{Similarity } S = P(\Omega_I | \Delta) = \frac{P(\Delta | \Omega_I) P(\Omega_I)}{P(\Delta | \Omega_I) P(\Omega_I) + P(\Delta | \Omega_E) P(\Omega_E)}$$

$P(\Delta | \Omega)$  aus Trainingsdaten

## FisherFace

= Linear Discriminant Analysis

Ziel: max. Interklassenvarianz  
min. Intraklassenvarianz



$$\text{Interkl. v. } S_B = \sum_{i=1}^C \|x_i\| (\mu_i - \mu) (\mu_i - \mu)^T \quad C = \# \text{ Klassen}$$

$$\text{Intrakl. v. } S_W = \sum_{i=1}^C \sum_{x_k \in X_i} (x_k - \mu_i) (x_k - \mu_i)^T$$

$$\Rightarrow w_{\text{fd}} = \arg \max_w \frac{|w^T S_B w|}{|w^T S_W w|}$$

- (+) · abstrahiert Licht-/Ausdrucksunterschiede  
· behält Klassenseparation

## Local Approaches



- lokale Änderungen im Gesicht: nur lokale Region betroffen (bei  $\downarrow$  holl. ganzes Feature)
- versch. statistische Verteilungen regional besser repräsentiert
- Gewichtung der lokalen Features an ges. Klasse möglich

## Modular Eigenspaces

- Klassifikation auf fiducial regions statt ges. Gesicht

## Local / Modular PCA

- Bild aufteilen in  $N$  Bilder  
 $\rightarrow$  PCA auf allen  $N$

## Gabor Filter

2D sin-wellen

$\hookrightarrow$  Filter versch. Skalierungen, Orientierungen  
( $\approx$  rezeptive Felder im visuellen Kortex)

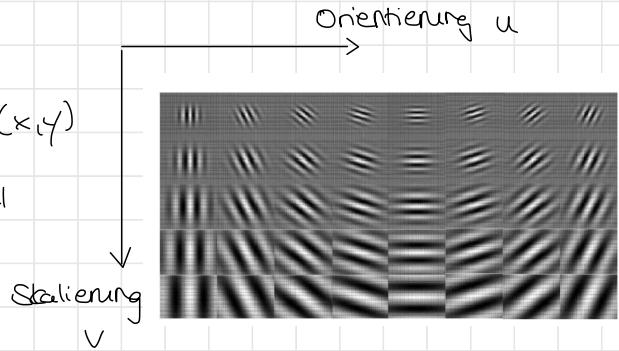
## Gabor Wavelet Transformation

$$O_{u,v}(x,y) = I(x,y) * \Psi_{u,v}(x,y)$$

Bild      Faltung      kernel

$\leadsto$  Filter der Skalierung,  
Orientierung verstärkt

$\rightarrow O_{u,v}(x,y)$  hoch dim.  $\Rightarrow$  PCA anwenden!



## Elastic Bunch Graphs

- Jet = Set von 40 Gabor Koeff. für einen Bildpunkt
- Graph = N facial landmark points, annotiert mit Jets  
Kanten annot. mit Distanzvektoren
- bunch graph = versch. Jets für versch. Posen, Aussehen  
(geschl. (geöffn. Augen, ...))

## Local Binary Pattern Histogram

- Bild in Zellen unterteilen
- vergleiche jede Zelle mit Nachbarn  
→ falls  $wert_{Nachbar} \geq \text{thresh.}$  : 1  
else : 0
- im Uhrz. S. ablesen → Binärzahl  
→ Dezimalzahl
- Histogramm über alle Zellen  
→ damit dann Klassifikation (zB SVM)

zB :

5	9	1	thresh=4	1	1	0
4	4	6	→	1	•	1
7	2	3		1	0	0

→  $11010011_2 = 21_{10}$

## Dense Features

- Features auf versch. Skalierungen des Bildes + auf überlappenden Bildbereichen (sliding window) berechnen

Problem: Stacking aller Features in 1 Vektor = sehr hohe Dim  
 zB  $160 \times 125 \text{ Img, } s=1, f=24, 5 \text{ Skalierungen, } 128 \text{ Features}$   
 $\Rightarrow 260000 \cdot 128 \approx 3,3 \text{ Mio Dim}$

⇒ Encoding in kompaktere Form

## Bag of Visual Word Model

- = Objekt als Histogram über Visual words
- visual words = features

## Fisher Encoding

- GMM mit versch. Gauß-Verteilungen für versch. Regionen
- Speichere Vektor der Unterschiede erster + zweiter Ordnung zw. dense features und GMM centers

$$\phi_k^{(1)} = \frac{1}{N\sqrt{\omega_k}} \sum_{p=1}^N \alpha_p(k) \left( \frac{x_p - \mu_k}{\sigma_k} \right)$$

$$\phi_k^{(2)} = \frac{1}{N\sqrt{2\omega_k}} \sum_{p=1}^N \alpha_p(k) \left( \frac{(x_p - \mu_k)^2}{\sigma_k^2} - 1 \right)$$

$$\rightarrow \Phi = [\phi_1^{(1)}, \phi_1^{(2)}, \dots, \phi_K^{(1)}, \phi_K^{(2)}]$$

Dim =  $2 \times K \times d \leftarrow$  dim. der dense feat. Vektoren

zB SIFT:  $d=128$   
 $K=512$  GMMs      }  $\Phi$  130 000 dim.

⇒ weitere Reduktion durch weniger  $K$  (zB 64) oder  
subspace learning methods

## Across Pose Matching

Problem: 2 Bilder von 2 Gesichtern frontal ähnlicher als 2 Orientierungen einer Person

## Pose Normalization

- berechne facial feature mesh  
↳ normalisierte damit Gesichtspose

→ 2D Active Appearance Models

- unabh. shape, app. models:

$$\text{shape } s = (\underbrace{x_1, y_1, x_2, y_2, \dots, x_v, y_v}_\text{zB Pos. der Nase})^T = s_0 + \sum_{i=1}^n p_i s_i$$

↑  
mean shape  
↑  
shape Eigenvec.

$$\text{appearance } A(x) = A_0(x) + \sum_{i=1}^m \lambda_i A_i(x) \quad \forall x \in S_0$$

→ Ziel: finde Params so dass

$$\arg \min_{p, \lambda} \sum_{x \in S_0} \left[ A_0(x) - \sum_{i=1}^m \lambda_i A_i(x) - l(w(x; p)) \right]^2$$

~  $p_i$  verändern → Änderung Gesichtsform (z.B. runder)  
 $x_i$  → Aussehen (z.B. mehr Bart)

=> nutze model um Gesicht zu frontal pose zu transformieren

### 3D morphologisches Modell

schätzen 3D Gesichtsform + -textur von Einzelbildern durch Fitter eines statistischen, morphologischen Modells

$$\text{Shape } S = \sum_{i=1}^m \alpha_i s_i; \quad \text{texture } T = \sum_{i=1}^m \beta_i t_i;$$

Referenzgesicht  $I_0$ :

- Referenz Shape:  $S_0 = (x_1, y_1, z_1, \dots, x_n, y_n, z_n)^T$
- Referenz texture:  $T_0 = (R_1, G_1, B_1, \dots, R_n, G_n, B_n)^T$

→ PCA auf Set von shape Vektoren, texture Vektoren

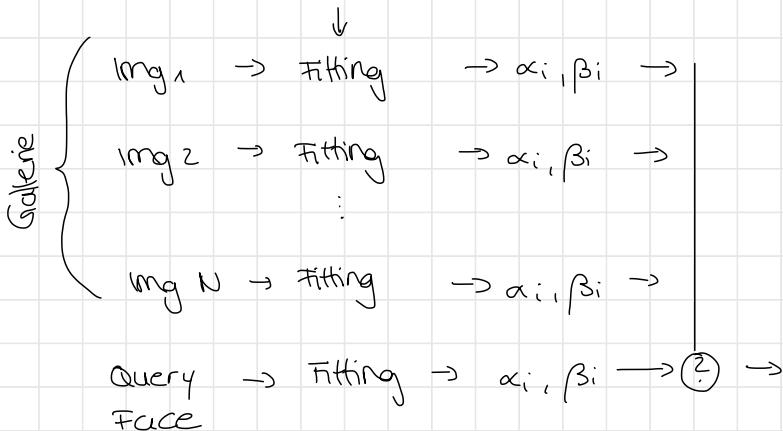
Inferenz:

$$\text{Ziel: minimiere } E_i = \sum_{x,y} \| I_{\text{input}}(x,y) - I_{\text{model}}(x,y) \|^2$$

→  $\alpha_i, \beta_i$ , pose angles, translation, Beleuchtung, ... anpassen

⊖ iterative Optimierung ⇒ sehr langsam

3D Scans → Morphologisches Modell  
DB



# Deep Learning

## DeepFace

- 7 Schichten, 20 Mio Params
- 4 Mio RGB Trainingsbilder

Facebook

4M - 500M

Training samples

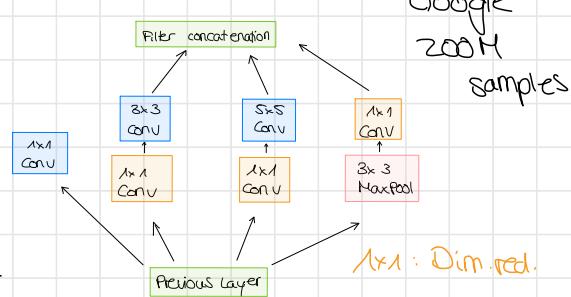
## Alignment

1. 2D Warp mit 6 fiducial points
2. 3D Model mit 67 points
3. frontale Ansicht ins NN

## FaceNet

Basis: GoogleNet

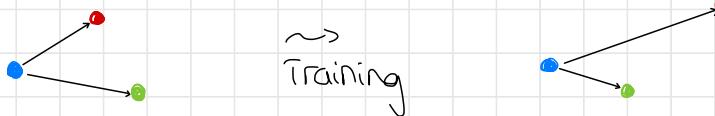
- Inception Modules
- Auxiliary Loss Layers



## Triplet Loss

Bild als 128D Vektor codieren

Anchor a       $\exists$  gleiche Pers.  
Positive p  
Negative n  $\rightarrow$  andere Pers.



$$L = \sum_i^N \left[ \underbrace{\|f(x_i^a) - f(x_i^p)\|_2^2}_{\text{pos.}} - \underbrace{\|f(x_i^a) - f(x_i^n)\|_2^2}_{\text{neg.}} + \alpha \right]$$

$\rightarrow$  Ziel:  $= 0$       Ziel:  $= \alpha$

$\rightarrow$  wähle Negative  $x_i^n$  sodass (semi-hard negatives)

$$\|f(x_i^a) - f(x_i^p)\|_2^2 < \|f(x_i^a) - f(x_i^n)\|_2^2$$

# Deep Face Recognition

Ziel: gleiche Ergebnisse wie Google, fb  
aber weniger Daten

Oxford Vgg  
2,6 M Samples

## VGG Face Dataset

1. Finde Namen von Stars (z.B. IMDb)

→ 5000 Namen

2 Verifiziere Stars:

- sammle repräsentative Bilder je Name

→ entferne Stars mit vielen falschen Bildern

3 Bilder filtern

- 2000 Bilder pro Name

- Ranking der Bilder nach Suchmaschinenergebnisse

4. Entferne (fast-) Duplikate

5. Manuelle Checks

## Netzarchitektur

· very deep architecture

·  $3 \times 3$  CONVs

$S = 1$

· ReLU

· 3 FCs

· Triplet Loss

· Feature Embedding 1024 dim.

· Finetuning der letzten Layer auf anderen Datensets

# FACIAL FEATURE DETECTION

## Facial Features

- markante Regionen mit hoher Aussagekraft  
Augen, Augenbrauen, Nase, Mund, ...

## Anwendungen

- ID
  - Normalisierung bei appearance based recognition
  - head pose estimation
- Blick - Tracking
- Expression recognition
- age modeling

## Schwierigkeiten

- Unterschiede zw. versch. Menschen
- bei anderem Gesichtsausdruck
- Variation von Pose
- Skalierung
- Beleuchtung
- Verdeckung (von Haaren, Brille, ...)

## alte Ansätze

- Viola & Jones
- Modular Eigenspaces
- Morphable 3D Model

## Statistical Appearance Ansätze

Idee: nutze Vorwissen, zB Modelle

- Deformierbare Modelle → können auf Veränderungen reagieren

## Kriterien gutes Modell

- Bildet Charakteristika des Objekts gut ab
- Kompakte Repräsentation
- Robust gegen Rauschen

## Appearance Model

- shape + texture model

### Training

#### 1. Shape Model mit PCA

- alle Shapes alignen  $\rightarrow$  entfernt Transl., Rot., Skal.

$$\Rightarrow \text{Landmarken } x = [x_1, y_1, \dots, x_n, y_n]^T$$

$$\begin{aligned} - \text{PCA} \quad x &\approx \bar{x} + P_S b_S & \leftarrow \text{Eigenvekt. der Kov.-matrix} \\ &\quad \uparrow & b_S = P_S^T (x - \bar{x}) \\ &\quad \text{mean} & = \text{shape Params im} \\ && \text{proj. Raum} \end{aligned}$$

$$\Rightarrow \text{Dim}(x) > \text{Dim}(b_S)$$

#### 2. Texture Model (repr. Grauwerte an jedem Punkt)

- warpe Bild, sodass Punkte auf mean shape passen (= shape free patch)

$$- \text{normalisierte Grauwerte} \Rightarrow h = [g_1, g_2, \dots, g_k]^T$$

$$\begin{aligned} - \text{PCA} \quad h &\approx \bar{h} + P_g \cdot b_g & \leftarrow b_g = P_g^T (h - \bar{h}) \\ &\quad \uparrow & \text{mean} \end{aligned}$$

#### 3. Modelliere Korrelation zw. shape, texture Models

$$\begin{aligned} - \text{konkat. Vektor} \quad b &= \begin{pmatrix} w_S b_S \\ b_g \end{pmatrix} & \begin{array}{l} = \text{shape} \\ = \text{texture} \end{array} \end{aligned}$$

$$\begin{aligned} - \text{PCA} \quad b &= P_C \cdot c = \begin{pmatrix} P_{Cs} \\ P_{Cg} \end{pmatrix} \cdot c & \leftarrow \text{kontrolliert shape} \\ && \& \text{texture} \end{aligned}$$

$\Rightarrow$  shape model

texture model

$$x = \bar{x} + P_S w_S^{-1} P_{Cs} c$$

$$g = \bar{g} + P_g P_{Cg} c$$

- (-) · Paramfindung schwierig  
- Kostenfunktion?  
↳ schweres Optimierungsproblem

## Active Shape Model

Init: grobe Startpos.

→ init Instanz  $x$  des Models mit Shapeparams  $b$ ,  
Translation  $T = (x_t, y_t)$ , Skal.  $s$ , Rot.  $\theta$

- Loop:
1. Finde besten Nachbar-Match für Punkt  $x_i$  in Umgebung von  $x_i$
  2. Update  $(b, T, s, \theta)$  Params um neues Modell  $x$  zu beschreiben
  3. Iteration bis Konvergenz

- (-) sehr schwierig zu impl.  
Paramsuche nur mit shape constraints → Texturinfo ignoriert

## Profilsuche

- Suche nächsten Punkt entlang einer Profil-Normalen

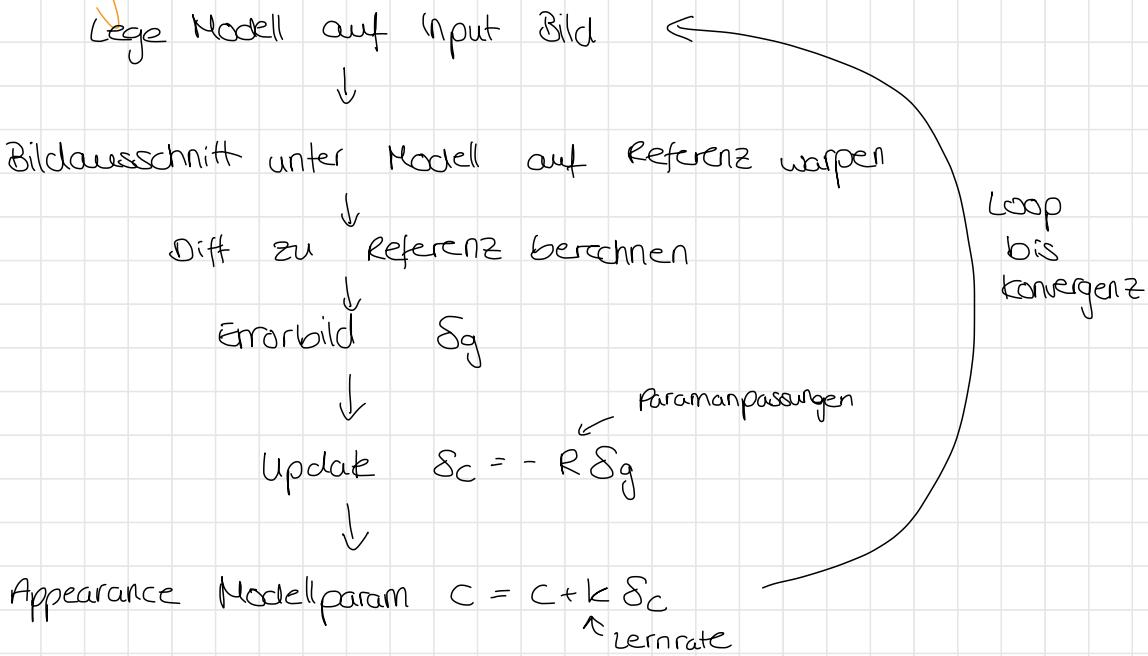
## Multi-resolution Suche

- Suche optimale Params hierarchisch
  1. Sehr grobe Bildauflösung
  2. Verfeinere Params mit besserer Auflösung
  3. -"-
  - ...

## Active Appearance Models

- nutze Shape + Texture zur Suche
  - optimiere Params, sodass Differenz zw. synteth. Bild + Zielbild minimiert
- Gradient Descent

### Training



$R$  Matrix kann gelernt werden mittels multi-variate lin. regress.

# Regression Forests

## Regression Tree

- Knoten: Entscheidungen durch vgl. von Zahlen
- Blätter: Zahlen / Multidim. Vekt. von Zahlen

## Regression Forest

- = Set von Trees
- Trees auf versch. Trainings-Subsets trainiert
- Prediction durch mitteln aller Einzelpredictions

## Facial Features

- versch. Bäume für versch. Kopfposen
- Blätter akkumulieren votes für versch. facial fiducial points

## Training

- jeder Baum auf zufälligem Set von Bildern
- Patches in jedem Bild extrahieren

Ziel: Akkum. Wkt für Feature  $C_n$  gegeben Patch  $P_i$  am Blattknoten

$$p(C_n | P_i) \propto \exp\left(-\frac{|d_i^n|}{\lambda}\right)$$

- Patch  $P_i$  repräsentiert durch appearance feature  $I_i$ , displacement Vektor  $D_i$  (= offsets)

$$P_i = \{I_i, D_i\} \quad D_i = (d_i^1, d_i^2, \dots, d_i^n)$$

- Node splitting Kriterium

$$f_\theta(P) = \frac{1}{|R_1|} \sum_{q \in R_1} I^\alpha(q) - \frac{1}{|R_2|} \sum_{q \in R_2} I^\alpha(q)$$

## Conditional Regr. Forests

### Training:

- berechne wkt für Kopfpose
- für jede Pose: teile Trainingsset in Subsets je Pose
- Regr. Forests je Pose trainieren

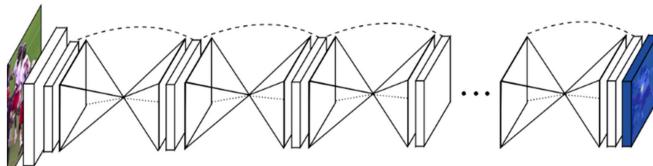
### Testing:

- schätzt wkt für jede Pose
- wähle entsprechende Bäume aus versch. Forests
- schätzt Dichtefunktion für alle facial feat. points
- Output: Clustering über alle feature candidate votes für einen geg. fac. feat. pt.

## CNN basiert

### Stacked Hourglass Network

- fully conv net
- wiederholtes Down- + upsampling + skip connections
- Input: RGB Bild
- Output: 1 Heatmap je Landmark  
↳ Heatmap durch DSNT zu koord. transf.



DSNT = Differentiable Spatial to Numerical Transform

· keine Params

· backprop möglich

$$\begin{array}{c} \hat{z} \\ \cdots \\ \begin{matrix} 0.0 & 0.1 & 0.0 \\ 0.1 & 0.6 & 0.1 \\ 0.0 & 0.1 & 0.0 \end{matrix} \\ \cdots \end{array} \quad \begin{array}{c} x \\ \cdots \\ \begin{matrix} 0.0 & 0.4 & 0.8 \\ 0.0 & 0.4 & 0.8 \\ 0.0 & 0.4 & 0.8 \end{matrix} \\ \cdots \end{array} \quad \begin{array}{c} y \\ \cdots \\ \begin{matrix} -0.4 & -0.4 & -0.4 \\ 0.0 & 0.0 & 0.0 \\ 0.4 & 0.4 & 0.4 \end{matrix} \\ \cdots \end{array}$$

$$\rightsquigarrow x = \langle \hat{z}, x \rangle_F = \begin{pmatrix} 0.1 \times 0.4 & + \\ 0.1 \times 0.0 & + \\ 0.1 \times 0.4 & + \end{pmatrix} = 0.4$$

$$y = \langle \hat{z}, y \rangle_F = \begin{pmatrix} 0.1 \times -0.4 & + \\ 0.1 \times 0.0 & + \\ 0.1 \times 0.4 & + \end{pmatrix} = 0.0$$

# FACIAL EXPRESSION ANALYSIS

## Facial expression

- = Änderungen im Gesicht als Reaktion zum internen emotionalen Zustand, den Intentionen einer Person oder sozialer Kommunikation
- wichtiger Bestandteil menschl. Kommunikation:
  - schnellste, natürlichste Art Emotion zu kommunizieren

## Anwendungen

- Schmerzanalyse
- Psycho - Analyse (Depression / Autismus)
- Workload - / Stress - Analyse
- Werbung
- Smart Cars (Fahrer aufmerksam / müde / gestresst / ...)

## Schwierigkeiten

- Emotionen aus Gesichtern ablesen auch für Menschen schwer  
≈ 50% accuracy  
→ Menschen sehr gut wenn zusätzl. Kontext, zB Körperhaltung
- Gesichtsausdruck kann versch. Emotionen als Ursache haben  
zB Grinsen aus Freude oder Frustration?
- Variabilität der Expression abh. von Geschlecht, Alter, Herkunft
- Beleuchtungsunterschiede, Verdeckungen (Bart, Brille, ...)
- gefakete Emotionen
- Gesichtsausdrücke gehen fließend ineinander über
- ⇒ Beste Ergebnisse wenn kombiniert mit Gesten, Körperhaltung, Sprache

## Datasets

### Cohn-Kanade

- 100 Schauspieler vor neutralem Hintergrund, frontal
- 23 versch. Gesichtsausdrücke
  - ↳ einzelne Aus
  - + Kombis von Aus
- Veränderte Lichtverhältnisse

### Ru-FACS

- 100 Personen
- spontane Gesichtsausdrücke
- Aus annotiert

### AFEW

- in the wild expressions
- aus Filmen
- annot. mit 6 Basisemtionen
- 330 Personen (1 - 70 Jahre)

# Beschreibung von Gesichtsausdrücken

## 6 Basis-Emotionen (+ Neutral)

nach Ekman

- Annahme: sind angeboren  
→ überall für alle Menschen gleich

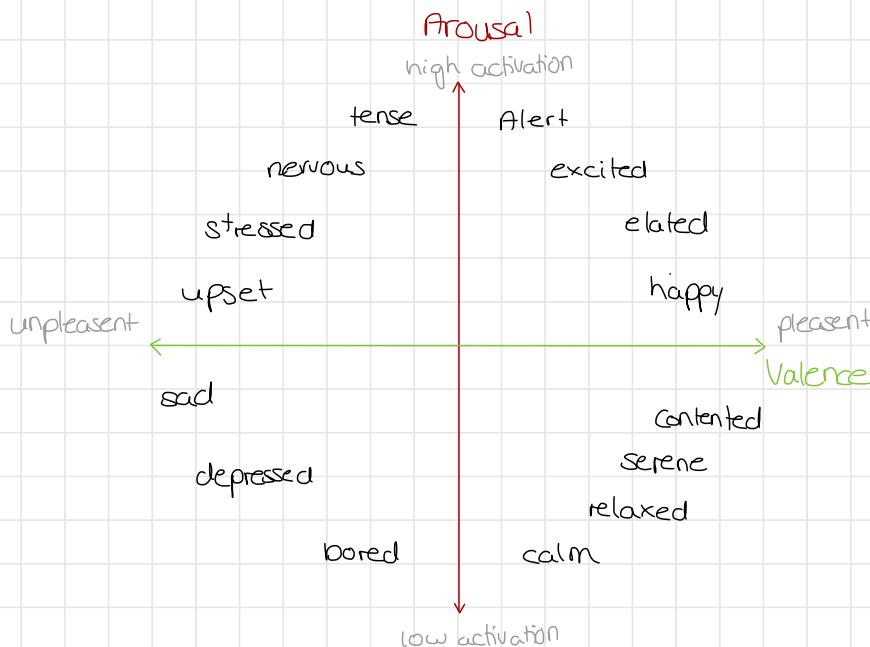
⊖ umstritten ob das so richtig, trotzdem oft verwendet

## 6 diskrete Klassen

- |                  |               |        |
|------------------|---------------|--------|
| - Freude / Glück | - Traurigkeit | - Ekel |
| - Überraschung   | - Angst       | - Wut  |

## Dimensional Emotion Models

- kontinuierliche Emotion entlang 2-3 Achsen  
↳ häufig: 2 Achsen nach Russel



# FACIAL ACTION CODING SYSTEM

nach Ekman, Friesen

## Action Units

44 AUs

- 30 relativieren Kontraktionen von bestimmten Muskeln im Gesicht
  - 12 oberes Gesicht (zB Blinzeln, Zwinkern, Brauen heben)
  - 18 unteres (zB Kinn fallen lassen, Lippen spitzen)
- 14 "verschiedenes" (zB Zunge zeigen, Lippen beißen)
- abh. von Intensität: 5 Stufen abh. vom Grad der Kontraktion

→ können kombiniert werden

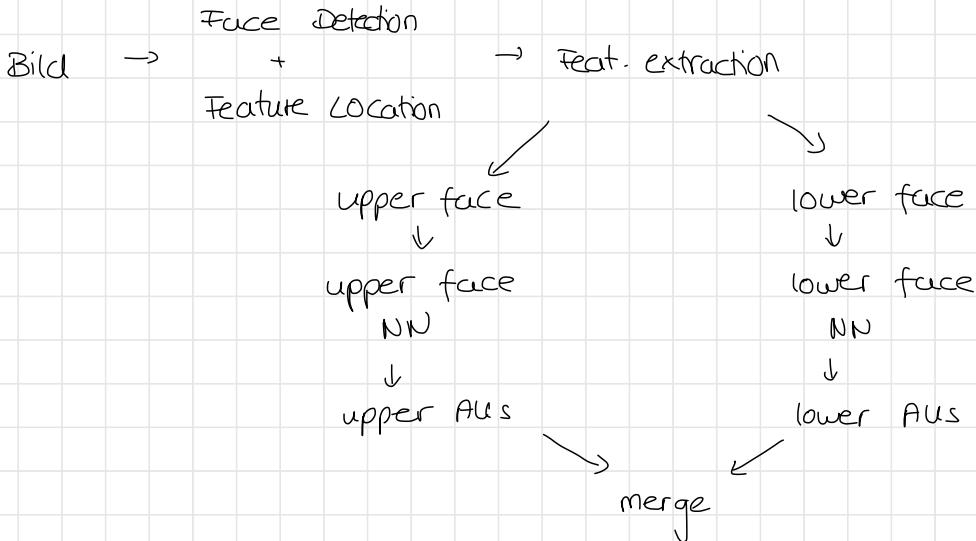
- additiv: Erscheinung einzelner AUs unverändert
- nicht-additiv: verändert

## Interpretation

- Gesichtsausdruck korreliert mit Emotionen
- Annahme: Lügen / Wahrheit ablesbar

## Systeme

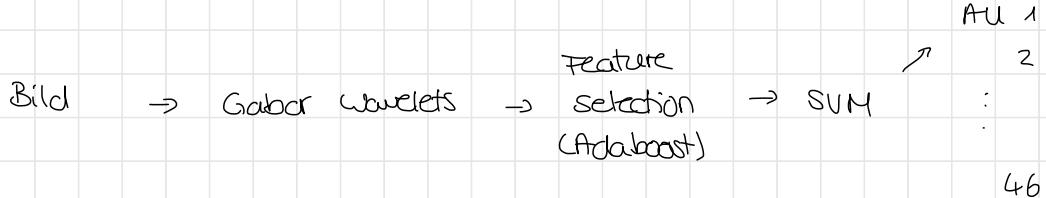
### AFFA (2001)



## Feature Extraction

- Modellbasierte facial features
  - permanente (z.B Mund, Nase, Augen,...)
  - transiente (z.B Falten, Grinsen,...)
- ⊖ nur auf KC- Database
  - in the wild sehr schlecht
  - nur wenige Aus

## Spontaneous Expressions (2006)



- + keine Modellanpassungen nötig
- immer noch manuelle Filter

## AdaBoost

Init: Set von Features

↗ Features: finde das, was Klassifikation am meisten verbessert  
→ behalte

immer weiter Features dazu nehmen bis ausreichend genau

⇒ Dimensionsreduktion

# PEOPLE DETECTION

## Anwendungen

- Sicherheit, Überwachung
- Smart Cars (Fußgänger, ...)
- Interaktion (Xbox Kinect)
- Medizin (Pat. überw., ...)
- kommerziell (Kunden zählen, ...)

→ gut falls keine Gesichter erkennbar / zu schlechte Auflösung

## Schwierigkeiten

- Unterschiede in Kleidung, Accessoires (Rucksack, Schirm, ...)
- viele versch. Körperposen
- Überlappung vieler Personen (Menschenmengen)

## Ansätze

### Bild

vs

### Video

- Grauwerte / RGB / Radar...
- ⊕ anwendbar in versch. Kontexten
- ⊖ komplizierter  
funktioniert schlechter

- zeitl. Infos
- optischer Fluss
- kombi mit Bild-verfahren mögl.
- ⊖ komplizierter in unbekanntem  
kontext (Hintergrund? bewegte  
Kamera?)

### global

vs

### Ausschnitte

- holistisches Modell
- ↪ 1 Feature für ges. Person
- ⊕ einfaches Modell
- gut bei geringer Auflösung
- ⊖ Probleme mit Okklusion,  
Artikulation

- Körperteile einzeln Modellieren
- ⊕ bewegte Körper besser modelliert
- können mit Okkl., Überlappung  
umgehen
- teilen von Trainingsdaten
- ⊖ komplexe Modelle
- Probleme bei geringer Aufl.

## diskriminativ

- geg. Daten: entscheide ob Person

⊕ · gut falls kein Modell gute Ergebnisse

⊖ · nicht interpretierbar

· keine Unsicherheit abbildung

vs.

## generativ

- modelliert wie Daten generiert werden

⊕ · interpretierbar (warum ja/nein)

· Samples erzeugbar

⊖ · Modellvielfalt für Klassifk. unnötig

· gutes Modell schwer

Klassifikator → Detektor

Sliding window → Feature Extraction → Object Classifier → ja/nein

## HOG Deskriptor

global, diskriminativ

x-Filter:

### Gradienten:

$$\text{1-seitig : } f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h}$$

$$\frac{f(x+h) - f(x)}{h}$$

$$\boxed{-1 | 1}$$

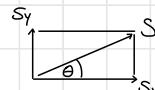
2-seitig :

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x-h)}{2h}$$

$$\boxed{-1 | 0 | 1}$$

### Magnitude:

$$S = \sqrt{s_x^2 + s_y^2}$$



### Orientierung:

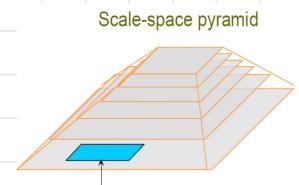
$$\theta = \arctan\left(\frac{s_y}{s_x}\right)$$

## Lernphase

1. normalisierte Trainingsdaten erzeugen:  
= Set mit gescrepten Bildern von Fußgängern in the wild
2. Bilder in feat. spaces kodieren  
= globaler Deskriptor pro Bild / Person ( $\approx 4000$  dim. Vektor)
3. Binären Klassifikator trainieren  
= SVM

## Inferenz

1. Sliding window über versch. Skalierungen
  2. Klassifikator auf Ausschnitten (SVM)  
↳ ja/nein - Objekt Entscheidungen
  3. mehrere Detektionen fusionieren (Clustering)  
→ 3D Pos. + Skalierung
- ⇒ Objekt Detektion mit BBoxes



## Deskriptor

### 1. Gradienten berechnen

Input: Bildregion  $64 \times 128$  px

- Falten mit  $[-1, 0, 1]$  in  $x, y$  Richtung
- $S, \theta$  berechnen
- je Farbkanal separat ⇒ nehme  $S, \theta$  vom Kanal mit größtem  $S$

### 2. Zellen Histogramme

1 Zelle  $\hat{=} 8 \times 8$  Pixeln

⊕ kompaktere Darstellung

robuster ggüber Rauschen (einzelne Grad. weniger Einfluss)

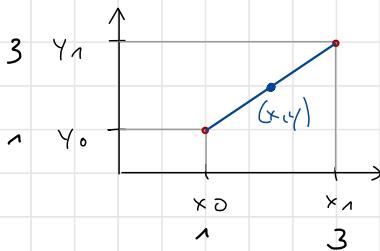
- 9 Bins - Histogramm erstellen für  $\Theta \in [0, 180]$

| 0 | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 |

→ in Bins wird S eingetragen basierend auf  $\Theta$

### Interpolation

- Linear:  $y = y_0 + (x - x_0) \frac{y_1 - y_0}{x_1 - x_0}$



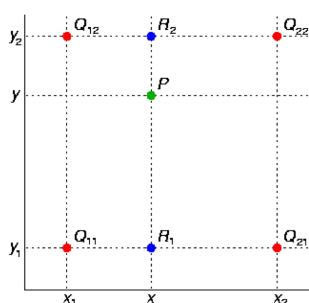
- Bilinear:  $R_1 = (x, y_1)$

$$R_2 = (x, y_2)$$

$$f(R_1) \approx \frac{x_2 - x}{x_2 - x_1} \cdot f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} \cdot f(Q_{21})$$

$$f(R_2) \approx \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

$$\Rightarrow f(P) \approx \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$



$Q_{ij}$  = Zellen-Centers

## Rechenbeispiel

$$\Theta = 85^\circ \quad S = 4.8$$

1. Lineare Interpolation :  $\Theta$  zu Bin-Centers (10, 30, 50, ...)

bin 70 :  $15^\circ$   
 90 :  $5^\circ$  Breite des Bins  
 $\Rightarrow$  Gewichte  $5/20 = 1/4$   
 $15/20 = 3/4$

2. Bilin. Interpolo.. Distanz zu Cell-Centers

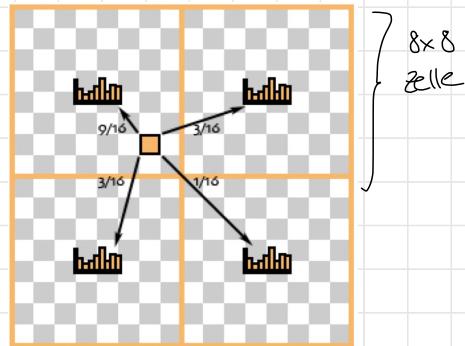
- links : 2 rechts : 6
  - oben : 2 unten : 6
  - linko/rechts  $6/8, 2/8$
  - oben/unten  $6/8, 2/8$
- $\Rightarrow$  Gewichte top left :  $6/8 \cdot 6/8 = 9/16$   
 r :  $6/8 \cdot 2/8 = 3/16$   
 b l :  $2/8 \cdot 6/8 = 3/16$   
 b r :  $6/8 \cdot 6/8 = 9/16$

$$\Rightarrow \text{top left } 70^\circ \text{ Bin} : 4.8 \cdot 1/4 \cdot 9/16 = 27/4$$

$$90^\circ \text{ Bin} : 4.8 \cdot 3/4 \cdot 9/16 = 81/4$$

$$\text{top right } 70^\circ \text{ Bin} : 4.8 \cdot 1/4 \cdot 3/16 = 9/4$$

⋮  
⋮



## 3. Histogramm - Normalisierung

- Histogramme normalisieren über überlappende  $2 \times 2$  Zellen  
 $\hookrightarrow$  konkatenieren  $\rightarrow$  normalisieren
- zB L2 Norm / L2 Hysterese

- Magnituden gewichtet gemäß Gaussian spatial window  
→ weit entfernte Gradienten tragen weniger bei

#### 4. konkatenieren

- alle Blockdeskriptoren konkatenieren  
 $7 \times 15 \times 4 \times 9$   
→ 3780 dim Vektor

#### Silhouette Matching

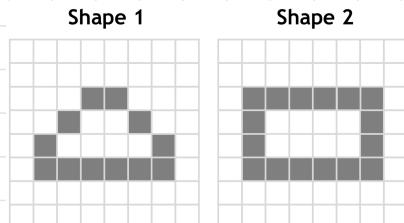
Ziel: aligne bekannte Objektformen mit Bild global, diskriminativ

Komplexität:

$$\mathcal{O}(\# \text{Positionen} \cdot \# \text{templates} \cdot \# \text{konturpixel} \cdot \text{sizeof(searchregion)})$$

#### Distance Transform

- Vergleiche + Align 2 Shapes



1. ∀ Pixel: berechne Distanz zum nächsten Konturpixel

8	6	5	4	4	5	6	8
6	5	3	2	2	3	5	6
5	3	2	0	2	3	5	
3	2	0	2	2	0	2	3
2	0	2	2	2	2	0	2
2	0	0	0	0	0	0	2
3	2	2	2	2	2	2	3
5	4	4	4	4	4	4	5

= Distance Transform

2. Legt shape2 auf Distance Transform, summiere Distanzen entlang shape2

→ so lange verschieben bis Minimum gefunden

8	6	5	4	4	5	6	8
6	5	3	2	2	3	5	6
5	3	2	0	2	3	5	
3	2	0	2	2	0	2	3
2	0	2	2	2	2	0	2
2	0	0	0	0	0	0	2
3	2	2	2	2	2	2	3
5	4	4	4	4	4	4	5

$$d = 14$$

# Chamfer Matching

1. Distance Transform DT berechnen
  2. A m<sup>g</sup>gl. Objektpos.:
    - lege Shape auf DT
    - Berechne Distanz d
    - behalte Instanz mit  $d < \theta$

$$d = \frac{1}{N} \sum_{i \in F} DT(i)$$

## Implementierung

z Scans über Bild

- Forward Scan

Von links oben nach rechts unten ↓

→ Naske a b a zB 3 2 3  
b c 2 0

- ## - Backward Scan

von rechts unten nach links oben

→ Maske      C B      O 2  
              a b a      zB      3z?

→ Maskenwerte auf lokale Distanz addiert

→ c updaten auf minimum der S Summen

$\Rightarrow$  lineare Kompl

- + · schnell
  - + · gute Ergebnisse für Bilder mit wenig Hintergrundstruktur (uncluttered)
  - · schlechte Erg. für cluttered Images
  - · braucht enorm viele Silhouetten

## Verbesserungen

### Hierarchische Templates

- Shapes nach Ähnlichkeit clustern

### Grob $\rightarrow$ Fein - Suche

- zunächst auf grober Auflösung suchen
- $\rightarrow$  nur Bereiche mit geringen Distanzen behalten
- $\rightarrow$  Suche auf feinerer Auflösung
- $\rightarrow \dots$

### Kantenorientierung

pro Pixel mit einbeziehen

## Teil - basierte Modelle

Schwierigkeit: wie von Einzeldefektionen  $\rightarrow$  Gesamtentscheidung?  
wurde es Person oder nur einige Fehlentscheidungen?

### erstes Modell 1973

- Körperteile (= 2D Bildausschnitte)
- + Struktur (= Konfiguration der Körperteile)
- $\rightsquigarrow$  räuml. Anordnung

### Fixed spatial layout:

- Teile meistens gleiche Pos., Orient. respektive Zentrum
- Bewegung relativ zueinander gering
- $\rightarrow$  z.B. für Gesichter

### Flexible Spatial Layout:

- Teile variabel in Pos., Orient., Skal.
- relative Positionen probabilistisch modellieren
- $\rightarrow$  z.B. für Personen (Arme, Beine stark beweglich)

## Verbindungsstrukturen

- Fully connected
  - Stern Topologie
  - K-Fan



- Baum
  - Bag of Features = keine räuml. Anordn.  
⋮ ⋮
  - Hierarchie
  - Sparse flexible model

# Mohan Detector

4 Körperteile : Gesicht + Schultern

## Beine

rechter Arm

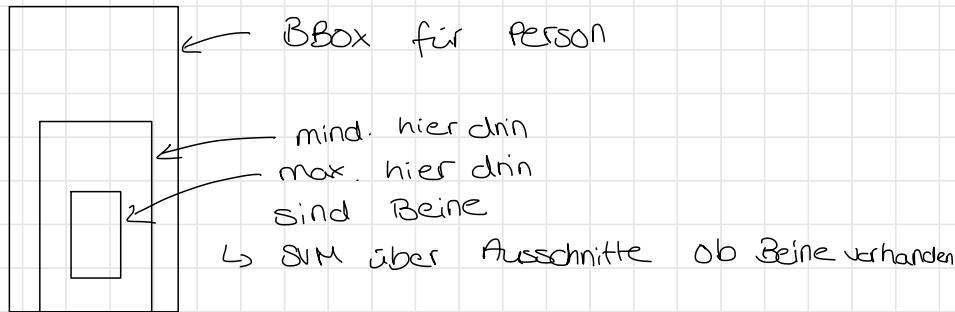
linker Arm

→ Kombination : SVM Classifier

} fixe Anordnung

## Detection

- HOG Deskriptor
  - fixe räuml. Anordnung, erlaube lokale Abweichungen



~ für alle 4 Teile machen, wenn genügend Teile erkannt:

Ja / Nein ausgeben

## Implicit Shape Model

Idee: Lerne große Teile die Objekt ausmachen  
= Visual Vocabulary / Bag of words  
→ Lerne relative Anordnung respektive Zentrum  
(= Stern-Topo)

### 1. Part Detection / Localization

- manuell definieren

⊖ menschl. Verständnis u.U. nicht optimal für Computer

- Lernen:

Anforderungen:

- wiederholbar: unabh. von Rotation, Perspektive, Beleuchtung, ...
- distinkt
- kompakt (nicht Kopf bis Fuß oder komische Strukturen)
- effizient
- Abdeckung des Objekts ausreichend

Local Features = Keypoints ( $x, y, s = \text{Scale}$ )

1. keypoint Detection
2. ROI um Keypoint
3. Normalisiere ROI
4. Deskriptor berechnen
5. Deskriptoren vergleichen

### Detection

- finde Punkte mit 2-dim. Signalfärbung (z. Ecken)

→ Second Partial Derivative Test

$$\text{Hesse-Matrix } (I) = \begin{bmatrix} I_{xx} & I_{xy} \\ I_{xy} & I_{yy} \end{bmatrix}$$

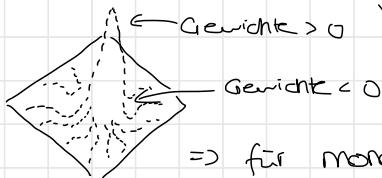
$$\det = I_{xx} I_{yy} - I_{xy}^2 > 0 \text{ dann lok. Max/Min}$$

→ Antwort hängt. an Ecken, stark texturierten Bereichen

## Skalierung:

Mexican-Hat-fct.

- "Blob" Detektor: Laplacian-of-Gaussian  
= 2. Ableitung der Gauß-Kurve



- => für monotone Bildregionen heben die sich auf  
=>  $\Delta w = 0$
- => für Regionen mit Kanten abh. von Radius  $\Delta w \neq 0$
- nutze als Skalendektor  
(nutze Skala mit max.  $\Delta w$ )

## 2. Part Description

### Local Descriptors

- beschreibe Region um keypoint

### Rot. invarianz:

- berechne Orientierungshistogramm (ähnlich HOG)
- wähle dominante Orientierung (= Bin mit meisten Werten)
- normalisiere auf diese Orientierung

### SIFT

Region auf  $16 \times 16$  px skalieren

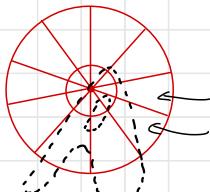
→ in  $4 \times 4$  Regionen unterteilen

→ Histogramm (16 Stück, je 8 Bins, gewichtete Gradienten)

→ 128 dim. Vektor

### Shape Context Descriptor

· = keypoint



zähl # Punkte je Bin  
Count - 1  
Count = 4

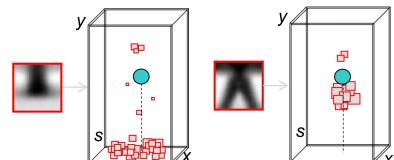
### 3. Learning Part Appearances

#### Visual Vocabulary

- keypts + Desk. & Personen - Trainingsamples
  - gruppier ähnliche lokale Deskriptoren (Clustering)
    - = Körperteile
  - speichert jeweils die durchschn. Segmentierungsmaske
- ~ typisch: Vocabsize 10000+

### 4. Learning Spatial Layout of Parts

- Vokabulareinträge auf Trainingsbilder matchen
- $(x, y, \text{Scale})$  respektive Zentrum
- Verteilung je Körperteil schätzen
- GMM



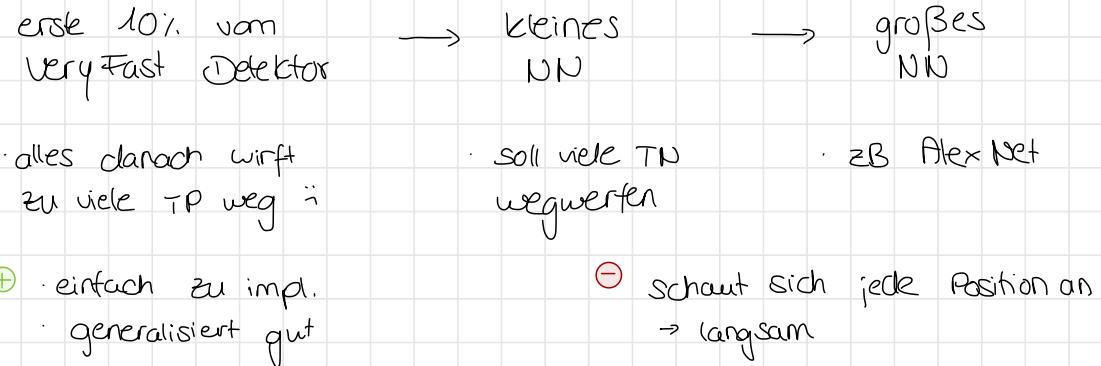
### 5. Combination of Part Detection

- Bild
- ↓
- Keypoint Detection
- ↓
- Probabilistisches Voting:
  - ∀ Keypoints: was ist nächster Vokab. eintrag?
  - zu weit entfernt? Keypoint ignorieren
- ↓
- 3D Voting Space:
  - jedes Körperteil wählt wo relativ zu ihm Körperzentrum ist
- ↓
- Back-Projection:
  - vom Zentrum aus: wenn hier Person, dann muss etwa dort zB Kopf sein
  - ~ Keypoint an Stelle + prüfen ob Kopf detektiert
- ↓
- Segmentierung:
  - weise Bildregionen (zB Kopf) durchschn. Segmaske zu

# Deep Learning

- (+) · hohe Genauigkeit, gute Generalisierbarkeit
- (-) · kein manuelles Feature-Engineering / Tuning nötig
- (-) · vergl. weise langsam

## Network Cascades



## R-CNN

1. extern: generiere Region Proposals
2. skaliere Regions auf gleiche Größe
3. Regions → CNN für Featuregenerierung
4. Für jede Ausgabe: Klassifikation mit SVM  
BBox Regression

## Region Proposals

Grouping methods:

- hierarchische Gruppierung von Bildregionen
- (+) bessere lokal.

## Window scoring methods:

- generiere viele Ausschnitte
- berechne "objectness" score
- ⊕ schneller

## Selective Search:

1. Segmentierung über Intensity  
→ BBoxen drumrum
2. Nachbarn die ähnlich sind mergen bis # Boxen ok  
⇒ ROIs

## Training

1. AlexNet auf ImageNet vortrainieren
2. letzte Layer neu auf neue # Klassen
3. Trainiere 1 SVM je Klasse
4. BBox Regression ( $d_x, d_y, dw, dh$ )



- sehr langsam
- CNN, SVMs separat trainiert
- komplexes Netz

## Fast R-CNN

1. Schicke Bild durch CNN → nur 1mal Feat. map berechnen
2. extern: generiere Region Proposals
3. ROI Pooling
4. FC Layers
  - Linear + softmax → Klassifikation
  - Linear → BBox Regression

## ROI Pooling

- FC erwartet fixe Eingabegröße

Input: Feat. maps  $C \times W \times H$  + Region Proposals

→ splitte ROI in  $w \times h$  Regionen

→ maxpool

→  $C \times w \times h$  features

→ FC layers

## Training

- EZE: Multi-Task-Loss

- neg. log. lik. für Klassifikation
  - L1 für Regression
- } gewichtet  
addieren

- (-) Region Proposals extrem  $\Rightarrow$  noch nicht mit gelernt  
 $\hookrightarrow$  langsam!

## Faster R-CNN

= Fast R-CNN



## Region Proposal Network

Input:  $C \times W \times H$  Feat. Maps

Output: p Region Proposals + "objectness" score

$p \times 6$  (4 für BBox, 2 für Objekt ja/nein)

RPN = mini CNN

→ gehe mit  $s=1$  über Feat. map

→ an jedem Punkt: evaluiere  $k$  Anchor auf objectness

$\Rightarrow W \times H \times k$  proposals

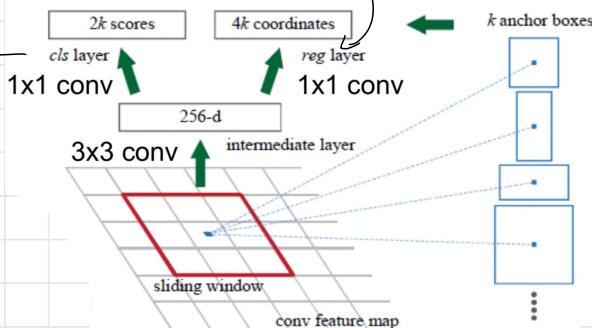
## Anchor:

- Referenzboxen versch. Skalierungen, aspect ratios
- 3

Regr. der  
Anchor

=> 9 Anchor

Klassifik.



## Loss

- Positiv Bsp.: IoU > 0.7
- Negativ : IoU < 0.3
- restl. kein Einfluss auf Training

→ Multi-Task Loss

$$L(\hat{p}_i; \hat{g}, \hat{t}_i; \hat{g}) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \cdot \frac{1}{N_{\text{reg}}} \sum_i p_i^* L_{\text{reg}}(t_i, t_i^*)$$

so wählen, dass etwa gleich gewichtet

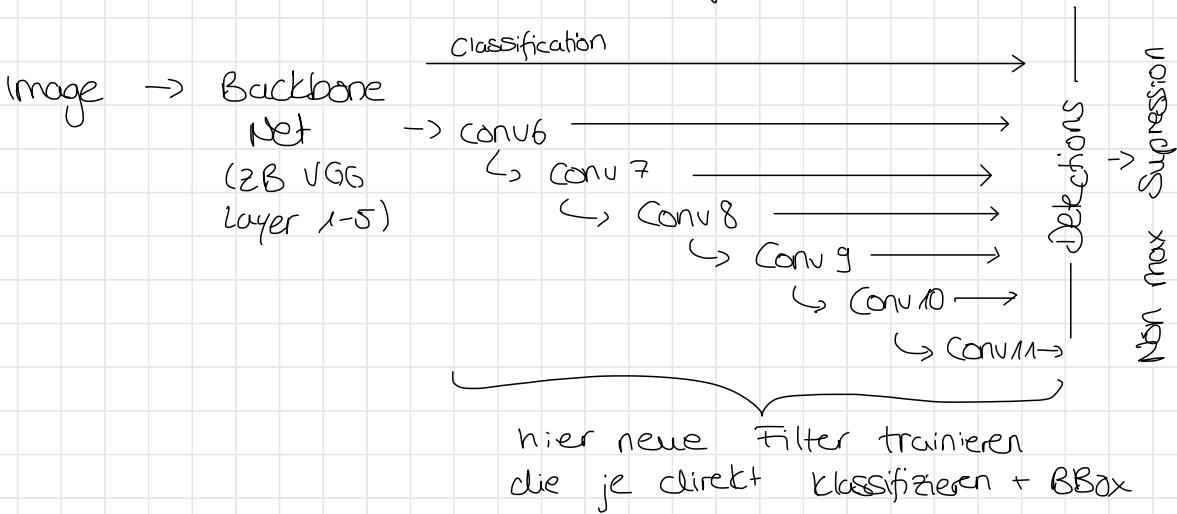
## Training

- alles zusammen EZE
- 4 losses gleichzeitig
  - Objectness Classification ↗ RPN
  - Anchor regression
  - Object class classif.
  - Detection Regression

## SSD Detector

= Single Shot Multibox Detector

- wende versch. Anchor Boxes an versch. Stellen im CNN an
  - ↳ auf Layern versch. Größe
  - ≤ versch. Auflösung



- keine FC Layer mehr
- Conv Layer unabh. von Eingabegröße (mit FC schon)

## Mask R-CNN

- zusätzlicher Segmentierungstask am Ende

## Design Entscheidungen

- Input (Patches, Pyramiden, ...)
- Backbone (VGG16, ResNet, ...)
- Neck (FPN, ...)
- Head (YOLO, SSD, RPN, ...)
- Anchor Boxes / Anchorless
- Loss Functions
- Building Blocks (Attention, ...)
- Trainingsmethoden

## TRACKING

- bestimme Zustand eines Objekts / Person über Sequenz von Beobachtungen (Bilder) hinweg
    - Location
    - Position
    - Rotation
    - Pose
    - ...
- Lokalisierung, ... in jedem Frame

Idee:

- nutze Infos aus >1 Bildern
- nutze Vorwissen zB Systemdynamik

## Sensor Setups

- Einzelkamera
- Mehrere Kameras
- Aktive Kameras
- Multimodal: Kamera + Mikros

## Beobachtungsmodalitäten

- Vorlagen
- Farben
- Segmentierung
- Kanten
- optischer Fluss
- Detektoren
- ...

## Zustandsschätzung

Ziel: sage Zustand vorher (Position, Pose, ...)

→ kann nicht direkt gemessen werden

→ kann nur Beobachtungen machen

→ Beobachtungen sind verzerrt! => Fehler

$\Rightarrow$  finde Zustand  $x_t$  so dass

$$\operatorname{argmax} p(x_t | Z_t)$$

$x_t$  = Zustand zum Zeitpunkt  $t$

$Z_t$  = Beob./Messung zum Zeitpunkt  $t$

$Z_t = z_0 \dots z_t$  alle Beob. bis  $t$

## Bayes Filter

Annahme: Zustand  $x$  ist Markov-Prozess

$\hookrightarrow$  Markov-Anannahme:  $x_t$  nur abh. von  $x_{t-1}$

$$p(x_t | x_{t-1}, \dots, x_0) = p(x_t | x_{t-1})$$

Zustand  $x_t$  generiert Beobachtungen  $z_t$

$$p(z_t | x_t, x_{t-1}, \dots, x_0) = p(z_t | x_t)$$

$\rightarrow$  schätze wahrsch. Zustand  $x_t$  mit  $\operatorname{argmax} p(x_t | Z_t)$   
rekursiv



### Predict:

$$p(x_t | Z_{t-1}) = \int_{x_{t-1}} p(x_t | x_{t-1}) p(x_{t-1} | Z_{t-1}) dx_{t-1}$$

$\nwarrow$  wkt dass von Zustd.  $x_{t-1}$  nach  $x_t$   
 $\nwarrow$  wkt aus letztem Schritt

## Update:

$$p(x_t | Z_t) = \frac{p(z_t | x_t) \cdot p(x_t | Z_{t-1})}{p(z_t | Z_{t-1})}$$

$\uparrow$  konstant bzgl.  $x$

$$\approx \propto \cdot p(z_t | x_t) \cdot p(x_t | Z_{t-1})$$

## Voraussetzungen:

- Prozessmodell  $p(x_t | x_{t-1})$  (= Übergangswkten)
  - Messmodell  $p(z_t | x_t)$  (= Beobachtungswkten)
- müssen bekannt sein (oder geschätzt werden)

## Kalman Filter

### Annahmen:

- Bewegungs-, Messmodell linear
- Bewegungs-, Messrauschen Gauß-verteil

$\leadsto$  analytisch berechenbare Lösung

$$x_t = Ax_{t-1} + w_{t-1} \quad p(w) \sim N(0, Q)$$

$$z_t = Hx_t + v_t \quad p(v) \sim N(0, R)$$

↓  
schätzen  
falls unbekannt

$A$  = Übergangswahrscheinlichkeiten (Matrix)

$H$  = Messmatrix

$p(w)$  = Bewegungsrauschen

$p(v)$  = Messrauschen = wie verfälscht sind Messungen

## Algorithmus:

Init: Schätze  $\hat{x}_{t-1}^-, P_{t-1}$  für  $t=0$

### Time update: (Predict)

(1) Schätze State:  $\hat{x}_t^- = A \hat{x}_{t-1}^-$

(2) Schätze Fehlermatrix:  $P_t^- = A P_{t-1}^- A^\top + Q$

### Mess-Update: (Correct)

(1) Kalman Gain:  $K_t = \frac{P_t^- H^\top}{H P_t^- H^\top + R}$

(2) Schätzung Updaten mit Messwert  $z_t$ :

$$\hat{x}_t = \hat{x}_t^- + K_t (z_t - H \hat{x}_t^-)$$

(3) Fehlerkovaianzmatr. updaten:

$$P_t = (I - K_t \cdot H) P_t^-$$

↑ Einf. matr.



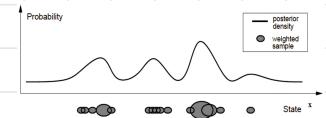
- => Je höher Messrauschen  $R$  desto niedriger Kalman Gain
- => Prediction weniger von Messung beeinflusst
- => Je höher Prozessrauschen  $Q$  desto höher Kalman Gain
- => Prediction mehr von Messung beeinflusst

(-)

- nicht anwendbar falls nicht-linearer Prozess oder Beziehung Prozess -> Messung nicht-linear
- $\rightsquigarrow$  Verbesserung: Extended kalman Filter
- mehrere Objekte: welche Messung soll welchen Zustand updaten?

## Particle Filter

- beliebige Wktverteilungen möglich  
 $\rightarrow$  repr. durch gewichtete Samples = Partikel
- numerischer Ansatz



## Voraussetzungen

- Messmodell  $p(z_t | x_t)$  = Wkt dass  $z_t$  beobachtet geg. dass  $x_t$  aktueller Zustand
- Bewegungsmodell  $p(x_t | x_{t-1})$  = Wkt dass Zustand  $x_t$  geg. davor  $x_{t-1}$

$\rightsquigarrow$  Set von N Partikeln zum Zeitpunkt t

$$\{ (s_t^{(i)}, \pi_t^{(i)}) \mid i=1, \dots, N \}$$

$\nwarrow$  Gewicht von Partikel i zum Zeitpunkt t

$\rightarrow$  Bewegungsmodell update  $s_t^{(i)} \leftarrow s_{t-1}^{(i)}$

Messmodell

$\pi_t^{(i)} \leftarrow s_t^{(i)}$   
 $\rightsquigarrow$  Gewichtsupdate abh. davon wie gut Partikel die Messung wiedergeben

$\downarrow$   
 beliebig festlegbar

## Condensation Algo

- 1) Select: neues Partikelset  $M_t = \{S_t^{(1)}, \dots, S_t^{(N)}\}$  gemäß  $\pi_{t-1}^{(i)}$   
- ziehe N mal mit zurücklegen aus gewichtetem Set  $M_{t-1} = \{S_{t-1}^{(1)}, \dots, S_{t-1}^{(N)}\}$
- 2) Predict: Propagiere Partikel gemäß Bewegungsmodell
- 3) Measure: Berechne Gewichte für  $M_t$  geg. Messmodell

$$\pi_t^{(i)} = p(z_t | x_t = S_t^{(i)})$$

- ≈ Zielposition durch  
- Clustern der Partikel  
- stärksten Partikel auswählen  
- ...

## Anzahl Partikel N

- abh. von Dimension des Zustandsraums

typisch: 1 Objekt in 2D Bildraum: 50 - 500

→ Curse of Dim. :	1D	→ $S^1$ Partikel
	2D	→ $S^2$
	3D	→ $S^3$



## Multi-Object Tracking

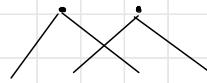
- je Objekt 1 Tracker
- + schnell, parallelisierbar
- ⊖ bestes Assignment schwer zu finden
- 1 Tracker im gesharten Raum
- + optimale Log. einfacher zu finden
- ⊖ # Objekte muss bekannt sein, Curse of Dim.

# Multi-Camera-Systems

## Topologien

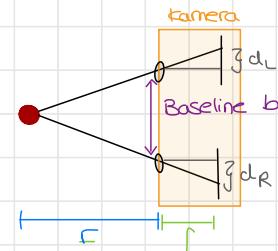
### Stereo-Kamera

- gleiche Orientierung, geringer Abstand
- Objekt-Aussehen  $\approx$  gleich in beiden Kameras
- Disparitätsberechnung möglich



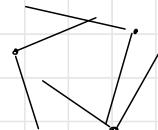
Disparität: Distanz Objekt  $\leftrightarrow$  Kamera  
 $\hookrightarrow$  geht nur wenn korrespond. Punkte  
 in beiden Bildern gefunden  
 $\Rightarrow$  müssen ausreichend Struktur haben

$$\Gamma = \frac{b \cdot f}{d_L - d_R}$$



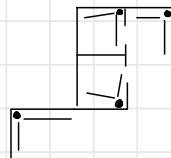
### Multi-Kamera

- beliebige Abstände, Orientierungen
- Überlappungen der Blickwinkel
- Objekt-Aussehen versch je Kamera
- 3D-Lokalisierung möglich

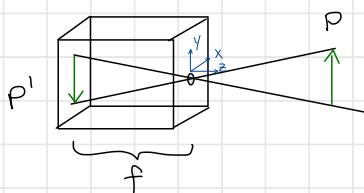


### Netzwerke

- keine Überlappung
- Objekt-Aussehen stark verschieden



### Lochkameramodell



$$P = \begin{pmatrix} x \\ y \\ z \end{pmatrix} ; P' = \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

$$z' = -f$$

$$\frac{y'}{-f} = \frac{y}{z} \Rightarrow y' = -\frac{fy}{z}$$

$$\frac{x'}{-f} = \frac{x}{z} \Rightarrow x' = -\frac{fx}{z}$$

=> Pixelkoordinaten  $(u, v)$

optisches Zentrum

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} k_u & 0 \\ 0 & -k_v \end{pmatrix} \begin{pmatrix} x' \\ y' \end{pmatrix} + \begin{pmatrix} c_x \\ c_y \end{pmatrix}$$

$k_u, k_v$  = Skalierungsfaktoren zw. Welt-, Pixelkoord.

=> **intrinsische Kameraparameter**

modellieren Linsenverzerrung

$f, k_u, k_v, (c_x, c_y); k_1, \dots, k_n$   
müssen bekannt sein

-> schätzen durch Kalibrierung

**extrinsische Parameter**

= Lokalisierung der Kamera resp. glob. Koord. Sys.

$T$  = Translationsvektor

$R$  =  $B \times 3$  Rotationsmatrix

~ Weltpunkt  $p^* = (x, y, z)$  nach Kamera

$$p = R(x \ y \ z)^T + T$$

**Triangulierung**

Annahme: Objektpos. von mehreren Blickwinkeln bekannt

→ Schnittpunkte von mehreren Blickachsen

~ ggf. Least-Squares Approximation

$\approx^2$

**Korrespondenz-Problem**

- mehrere Personen → mehrere mögl. Schnittpunkte

→ teuer alle Mögl.keiten zu berechnen

**ohne Triangulierung**

- für jede 3D-Hypothese: schaue ob Beobachtungen passen  $\Rightarrow$  Partikel-Filter

zB Personenlecker feiert

## Beispiel: Smart Room

- Ziel: mehrere Personen  $\rightarrow$  trackte Sprecher
- Sensoren: mehrere fixierte Kameras, Mikros

$\rightarrow$  Multimodales-Tracking: nutze mehrere Features

## Background Models

- Szene ohne Personen aufnehmen  
= Hintergrundmodell (HM)
- jeder Pixel der sich signifikant von HM unterscheidet  
 $\Rightarrow$  Person

- (-) · nur für statische Hintergründe  
 $\hookrightarrow$  Beleuchtungsänderungen?  
kamera bewegt sich?  
Objekte im Bild verschoben?  
Schatten von Personen?
- $\Rightarrow$  Modell anpassen

## Tracking: Partikel-Filter

Zustand:  $s_i = (x_i, y_i, z_i)$  = Kopf des Sprechers

Bew.-modell: Partikel mittels Gaussian Diffusion propagiert

Beob.-modell:  $\pi_i$ ; bestimmt aus visueller + akustischer Info

$\hookrightarrow$  z.B.: Körper-BBox um  $s_i$

$\rightarrow$  gewichtete gemäß # Vordergrundpixel in BBox

$\hookrightarrow$  Gesichts-Detektor im Bereich um  $s_i$

$\rightarrow \pi_i$  größer falls Gesicht

$\hookrightarrow \dots$

$\Rightarrow$  gewichtet alle c Merkmale dynamisch  $\rightarrow$  Democratic Integration

$$p(z_t | s_t) = \sum_c r_c p_c(z_t | s_t)$$

## Visuelle Hülle

= approx. der Silhouette

## Volume Culling

- fixe Voxelgröße zB 5x5cm

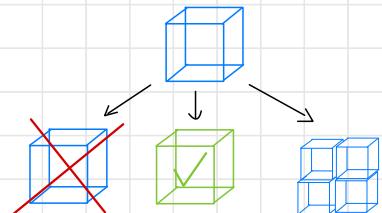
→ prüfe H Voxel: liegt Voxel in der Silhouette

→ ja: Voxel gehört zur visuellen Hülle

→ aufwendig ?

⇒ divide & conquer

- 1 Voxel am Anfang
- Actions: del, accept, split



## BODY POSE

### Kinect Sensor

- billiger, schneller RGB-D Sensor
- RGB Kamera
- IR-Emitter + -Detektor für depth
- Mikros mit noise cancelling
- Tilt + accelerometers

### Structured Light

- simuliert Stereo-Kamera
- Lichtmuster wird emittiert
- durch Oberfläche von Objekten deformiert
- Deformation umrechnen auf Entfernung  
(Block Matching) ✓

# Pose Recognition

Shotton et al. (2011)

## 1. Pixel Klassifikation

- nutze disparity image (Pixeldifferenzen)
- alle Pixel unabhängig
- Random Decision Forests
- gleichzeitige Feature Extraction

→ braucht viele Trainingsdaten → synthetische Daten

## 2. Joint Estimation

- mean shift clustering auf Pixeln  
→ Gaussian Kernel um Zentren zu finden
- Clustering in 3D aber gewichtet für Tiefeinvianz

→ Summe der gewichteten Pixel = Konfidenz

⊖ Bias in Richtung frontale Posen

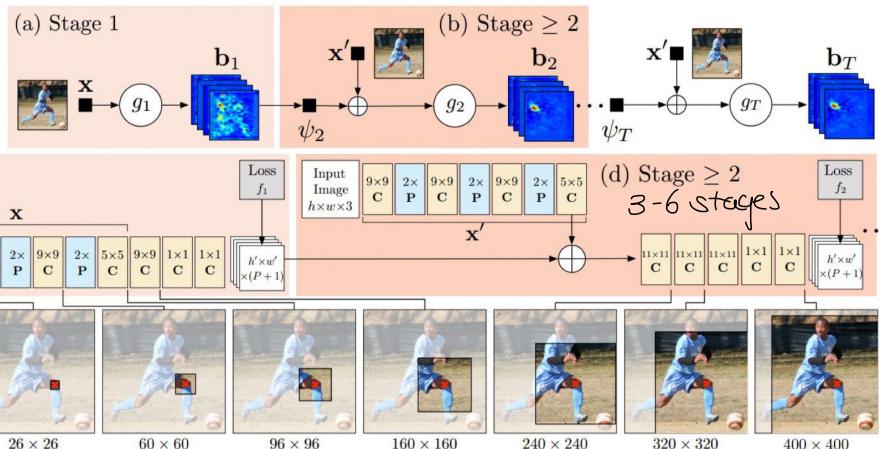
# CNNs

(ohne Kinned)

Wei et al. (2016)

Convolutional Pose Machines  
( $T$ -stage)

P Pooling  
C Convolution



· confidence maps je Gelenk  
→ in mehreren stages verbessern

Idee: lokale Bildinfo (= Stage I) schwach ABER kann guter Hinweis sein  
→ alle confidence maps der vorherigen Stufe mit berücksichtigen

- keine FC Layers ( $\Rightarrow$  beliebige Bildgrößen)

## Training

- 28 000 gelabelte Daten (mit annotiertem Skelett)  
 $\rightarrow$  ground truth als kleine Gaussian-fcts. an Gelenk pas.
- Euklidische Loss Fkt

$$E = \sum_{p=1}^{P+1} \sum_{z \in Z} \| b^p(z) - b_{gt}^p(z) \|_2^2$$

P = Körperteile

Z = Bereiche der confidence map

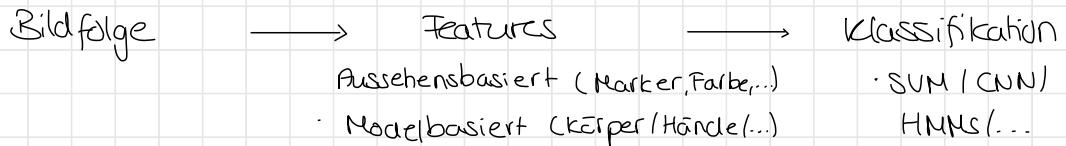
# GESTURE RECOGNITION

## Gesture

= Nutzen von Bewegung von Körper/Gliedmaßen um etwas auszudrücken

## Gesture Recognition System

- generiert semantische Beschreibung für bestimmte Bewegungen
- nutze non-verbale Kommunikationskanäle für HCI



## Anwendungen

- Multimodale Interaktion
  - Gesten + Sprache
- Mensch - Roboter - Interaktion
- Smart Environments

} Menschen (noch) besser verstehen

## Arten von Gesten

- Hände + Arme
  - Zeigende Gesten
  - Zeichensprache
  - Hallo, Daumen hoch, call-me, ...
- Kopf
  - Nicken, zeigen, ...
- Körper
  - Schulterzucken, ...

! stark kultur-spezifisch  
Viele Gesten koordiniert mit Sprache

# Taxonomie von Gesten

## Hand / Arm Bewegungen

### Gesten

unbeabsichtigt

#### manipulativ

- "put that there"
- Navigation in virtuellen Räumen

kommunikativ = Spontan

### Aktionen

#### mimetic

- zB Zigarette nachmachen

#### deictic

- Zeigegesten

### Symbole

#### referential

#### modalizing

· statisch vs. dynamisch

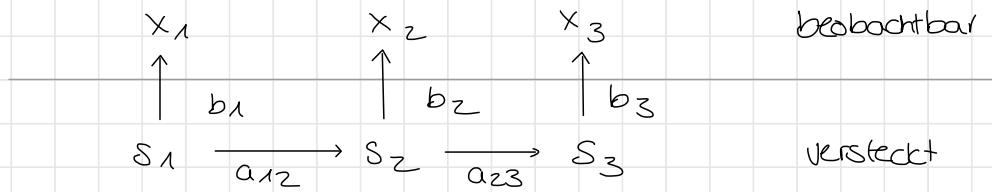
## Semaphorische Gesten

· akkurat definiert → spezifische Symbole eines Alphabets

## Konversationelle Gesten

intuitiv während Kommunikation genutzt

# Hidden Markov Models



## Markov Annahme

nächster Zustand nur abh. von aktuellem Zustand

$$\text{HMM} = (S, \pi, A, B, V)$$

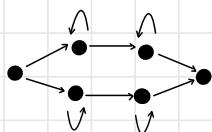
- $S = \{s_1, s_2, \dots, s_n\}$  = Zustände
- $\pi$  = initiale Wkt Verteilung  $\sum \pi_i = 1$   
 $\rightarrow \pi(s_i) = \text{Wkt, dass erster Zustand } s_i \text{ ist}$
- $A$  = Zustandsübergangsmatrix  $\sum a_{ij} = 1$   
 $\rightarrow a_{ij} = \text{Wkt, dass } s_j \text{ auf } s_i \text{ folgt}$
- $B = \{b_1, b_2, \dots, b_n\}$  = Ausgabewahrscheinlichkeiten  
 $\rightarrow b_i(x) = \text{Wkt, dass } x \text{ beobachtet wenn Zustand } s_i$
- $V = \{x_1, x_2, \dots, x_n\}$  = Beobachtungsraum (diskret)  
 $= \mathbb{R}^d$  (kontinuierlich)

## Topologien

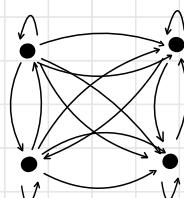
links - nach - rechts



alternative Pfade



ergodisch



## Beobachtungsmodell

$P(x_t | s_j)$  meistens Gauß-Mischverteilung

## Aufgaben mit HMMs

### Evaluationsproblem

Berechne Wkt der Beobachtung  $x_1, x_2, \dots, x_T$  geg. HMM  $\lambda$

$$P(x_1, x_2, \dots, x_T | \lambda) \rightarrow \text{Forward-Algo}$$

### Dekodierproblem

Berechne wahrscheinlichste Zustandsfolge  $s_{q1}, s_{q2}, \dots, s_{qT}$

$$\sim \operatorname{argmax}_{q_1, \dots, q_T} P(q_1, \dots, q_T | x_1, x_2, \dots, x_T, \lambda)$$

$\sim$  Viterbi-Algo

### Lern-/Optimierungsproblem

Finde HMM  $\lambda'$  sodass Beobachtung wahrscheinlicher  
 $\leftarrow$  Trainingssequenz

$$P(x_1, x_2, \dots, x_T | \lambda') > P(x_1, x_2, \dots, x_T | \lambda)$$

$\sim$  Viterbi-Algo / Baum-Welch-Algo

## Algorithmen

### Forward Algo

$\alpha_t(i) = \underbrace{P(x_1, x_2, \dots, x_t)}_{=X} ; q_t = s_i | \lambda$  = machen + Endzustand  $s_i$

Init:  $\alpha_1(i) = \pi_i \cdot b_i(x_1) \quad 1 \leq i \leq |S|$

Loop:  $\alpha_t(i) = \left( \sum_{j=1}^{|S|} \alpha_{t-1}(j) a_{ji} \right) \cdot b_i(x_t) \quad 1 \leq t \leq T$

Terminierung:  $P(x_1, x_2, \dots, x_T | \lambda) = \sum_{j=1}^{|S|} \alpha_T(j)$

## Viterbi - Algo

≈ Fwd - Algo, aber argmax in Loop statt Summe

## Baum - Welch - Algo

= EM Algo

1. Forward Step:

- Berechne  $\alpha_t(i)$  mittels Fwd - Algo

2. Backward Step:

- Berechne wkt der Endsequenz  $y_{t+1}, \dots, y_T$  gegeben Startzustand:

$$\beta_t(i) = p(y_{t+1}, \dots, y_T | q_t = s_i; \lambda)$$

3. Update step:

- Berechne Params  $\lambda$  neu mit Bayes und

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{\sum_{j=1}^{|S|} \alpha_t(j) \beta_t(j)}$$

# Systeme

## Sign Language Recognition

ASL = 6000 Gesten

→ nehme 40 Wörter ins Vokabular

### Feature extraction

- Kamera als 1st - & 2nd - Person - View
- Köpfe durch Farbsegmentierung (= blobs)

→ Features  $x, y, \underbrace{\Delta x, \Delta y}_{\text{relative Merkmale}}, \text{size}, \text{blob "angle"}, \text{eccentricity of ellipse}$   
 $\rightarrow$  robuster gg. Verschiebung

### Training

- jedes Wort : 4 - Zustands HMM  
 → einzeln trainiert
- automatische Segmentierung von Sätzen in 5 Teile
- Initiale Schätzung : Viterbi - Algo  
 → Verfeinerung : Baum - Welch - Algo
- kein Kontakt

### Metric

· Genauigkeit  $ACC =$

$$\frac{N - D - S - I}{N}$$

$N = \# \text{ Wörter}$

$D = \# \text{ Deletions}$

$S = \# \text{ Substitutions}$

$I = \# \text{ Insertions}$

## Haushalts - Robo

- pointing- Gesten als Armbewegung in Richtung target

- Multimodal :

- Personen - Tracking + Gesten - Erkennung
- Sprach - Erkennung
- Sprach - Synthese
- Mobile Platform

} ARMAR

## Tracking

Kopf + Hände

Stereo Kamera  $\rightarrow$  linkes rechtes Bild  $\rightarrow$  Features: Farbe + Disparität  $\rightarrow$  3D Model

- ~ Partikel - Filter :

Beobachtungsmodell = Punktwolke der Skin - Pixel

$$s_t^* = \arg \max_{s_t} P(o_t | s_t) \cdot P(s_t | s_{t-1}) \cdot P(s_t)$$

$\nearrow$   
Bewegungsmodell       $\nwarrow$  Posen - Wkt.

## Gestenmodell

3 Phasen  $\Rightarrow$  3 + 1 Modelle

- Beginn
- Halten
- Ende
- Garbage = für alles was keine Geste ist

## Zugrichtung

- nutzt Blickwinkel + Unterarmorientierung + Hand - Kopf - Linie

## Video Wand

- Visuelle 3D - Hülle (Voxel - Culling)

- Detection + Tracking des Arms

- Interaktion: Annäherung  $\rightarrow$  Halten  $\rightarrow$  Zurückziehen

Bewegung

# ACTION & ACTIVITY RECOGNITION

## Anwendungen:

- Video Indizierung / Analyse
- smart - rooms
- Patientenüberwachung
- Surveillance
- Roboter
- ...

Event = "etwas das stattfindet oder passiert"

zB Gester

Aktionen (Gessen, trinken, aufstehen,...)

Aktivitäten (Essen zubereiten, spielen,...)

Naturereignis (Feuer, Sturm,...)

## Human Actions

### 1) Physical body motion

- einfache Bewegungsmuster
  - idR einzelne Person
  - kurze Dauer
- zB. laufen, klatschen,...

### 2) Interaktion mit Umwelt

- um spezifisches Ziel zu erreichen
- zB. Tür öffnen, Auto starten,...

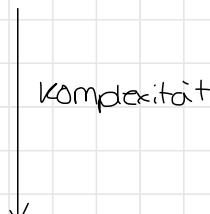
## Activities

- = komplexe Sequenz von Actions
  - . ggf. mehrere Personen
  - . idR. längere Dauer
- zB Essen zubereiten, Meeting,...

## Hierarchie

Actions = Bewegungsprimitive

- Hand heben, Objekt greifen, ...



## Activities

- Tee trinken, telefonieren, ...

## Events

- Fußballspiel, Party, ...

## Schwierigkeiten

- viele mögliche Actions / Activities
- variierender Kontext
- variable Umgebung
- Einzelpers. vs. Gruppen

## Spatio Temporal Features Descriptor

### Features

- Temporal Templates
  - zB Motion history Image
  - ⊕ einfach, schnell ( $\rightarrow$  absolute Frame-Differenz)
  - ⊖ sensitiv ggüber Segmentierungsfehler, Kamerabewegung
- Active Shape Models
  - ⊕ shape Regularisierung
  - ⊖ sensitiv ggüber init und Trackingfehlern
- Tracking with Motion priors
  - ⊕ gleichzeitiges Tracking + action recogn.
  - ⊖ sensitiv ggüber init und Trackingfehlern
- Motion-based recognition
  - ⊕ generische Deskriptoren  $\rightarrow$  weniger appearance-abhängig
  - ⊖ sensitiv ggüber Lokalisierungs-, Trackingfehlern

## Space-time-Features

Idee: kombiniere räuml. + zeitl. Bewegungsdeskriptoren

→ HOG um Aussehen zu beschreiben

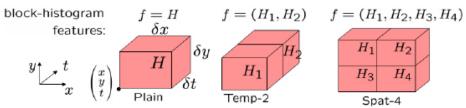
HOF um Bewegung im Video zu beschreiben

↪ histogram of oriented flow

3 Arten von Features:

### Plain:

- Beschreibe Videosequenz als Würfel ( $\delta x, \delta y, dt$ )  $\rightarrow f = H$
- beschreibe Würfel mit HOG, HOF



### Temp-2:

- Konkateniere 2 Plain-Features in zeitlicher Abfolge  
 $\rightarrow f = (H_1, H_2)$

### Spat-4:

- Konkateniere 4 Plain-Features räumlich  $\rightarrow f = (H_1, H_2, H_3, H_4)$

=> Würfel ist space-time-Würfel

- idR mit Kopf im Zentrum

=> Block encodieren mit HOG, HOF

$$\Theta = (\underbrace{(x, y, t)}_{\text{Ort}}, \underbrace{\delta x, \delta y, dt}_{\text{Space-time-Breite}}, \beta, \emptyset)$$

Art des Histogramms (HOF, HOG)

↑  
Art des Blocks  
(Plain, Temp-2, Spat-4)

=> aus Video sehr viele Features  $f_\Theta$  generieren

**HOG:** hier vereinfacht: 4 Bins

**HOF:** = histogram of optical flow : 4 Bins (+1 für keine Bewegung)

≈ hier: normalisierter Action-Würfel hat Größe  $14 \times 14 \times 8$ ,  
je Unit  $5 \times 5 \times 5$  Pixel

=> > 1 Mio mögl. Features für

### Action learning

- Boosting (zB AdaBoost) um Features eines Action-Würfels zu klassifizieren

### Space-time interest points + BoW



Erweiterung des Harris-Operators

- ↪ Suche entlang Zeitachse
- ↪ auf versch. Skalierungen (= dense scale Sampling)

### Bag of words

1. Vokabular lernen:

Clustering lokaler Features  
→ zentren = Wörter

2. Lokale Features ähnlichstem vis. Wort zuweisen

→ BoW feature = Histogramm vis. Wort-Häufigkeiten in Region

## Dense Trajectories

### Feature Trajectories

- = Tracking von Bildpunkten über mehrere Frames  
zB kLT tracker / SIFT Deskriptoren

### Dense Trajectories

- Trajektorien durch optical flow tracking auf densely sampled points
- Feature point alle  $\leq 5$  px auf versch. Skalierungen
- untrackbare Punkte entfernen

### Tracking

- Punkte versch. Frames konkateniert = Trajektorie
- Trajektorie max  $\ell$  Frames lang
- Form der Trajektorie als Sequenz

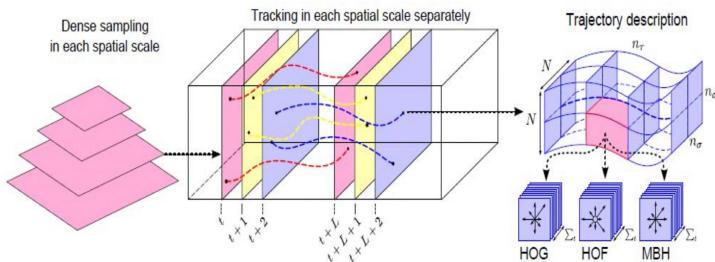
$$S = (\Delta P_t, \dots, \Delta P_{t+\ell-1})$$

$$\Delta P_t = (P_{t+1} - P_t) = (x_{t+1} - x_t, y_{t+1} - y_t)$$

$$\Rightarrow \text{Normalisieren} \quad S' = \frac{S}{\sum_{j=t}^{t+\ell-1} \|\Delta P_j\|}$$

### Features

- entlang jeder Trajektorie: HOG, HOF, MBH berechnen  
= motion boundary histogram



① Kamerabewegung erzeugt irrelevante Trajektorien

=> Improved Dense Trajectories

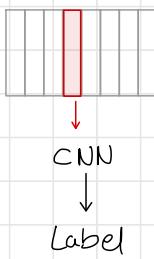
Kompensierte Kamerabewegung mittels SURF + RANSAC

## CNNs

### Zeitliche Verknüpfung

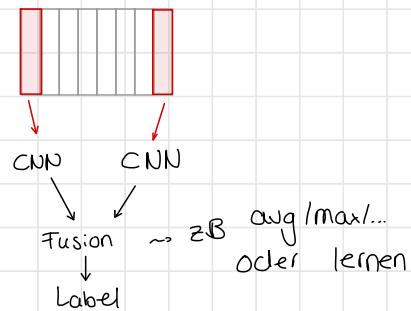
#### Single Frame

- je 1 Frame
- keine zeitl. Info



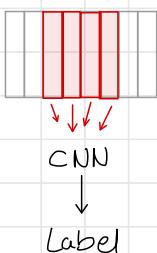
#### Late Fusion

- 2 Frames mit N Frames Abstand
- beide einzeln durch CNN
- letzte Layers haben zeitl. Infos



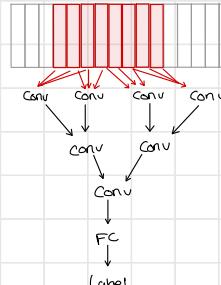
#### Early Fusion

- frühe Conv Layer haben zeitl. Info
- Filter mit  $1 \times 11 \times 3 \times T$   
 $T = \# \text{ Frames}$



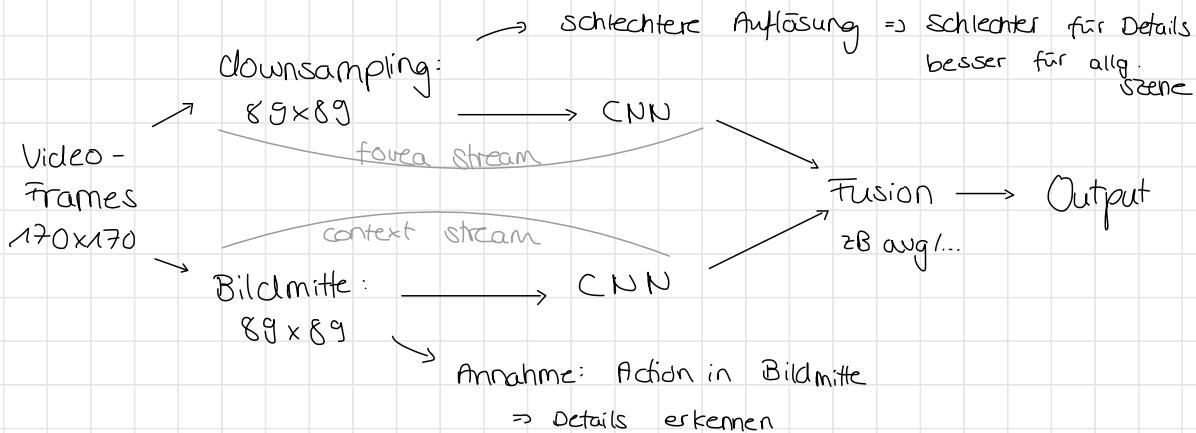
#### Slow Fusion

- Spätere Layer = größerer zeitl. Kontext
- Bewegungsmuster auf versch. Hierarchien

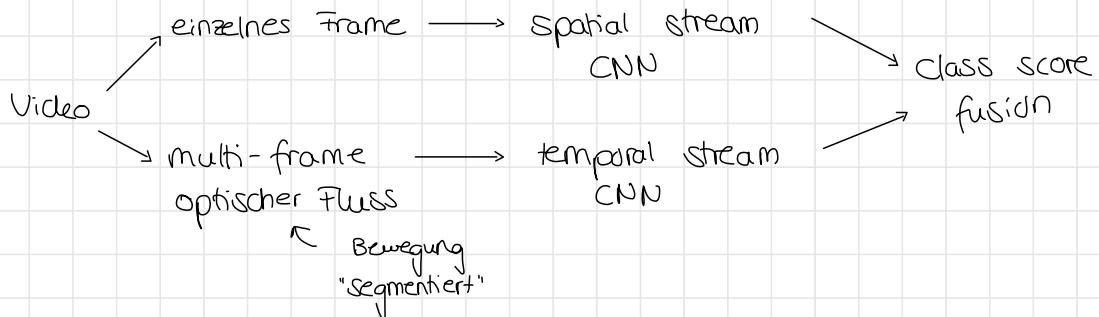


## Multi-resolution CNN

- 3D-Fitter  $\rightarrow$  viele Params  $\rightarrow$
- => reduzierte input size



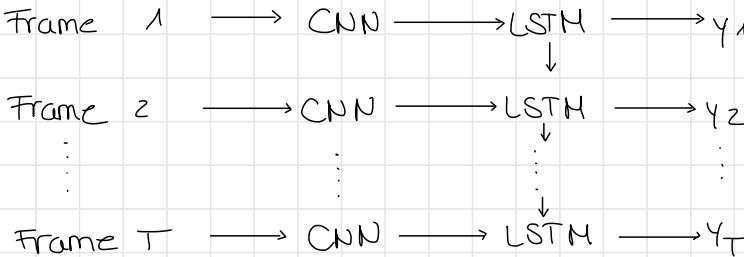
## Two-Stream CNNs



## C3D

- mehrere Frames auf einmal  $\rightarrow$  3D Convolutions
  - $3 \times 3 \times 3$  Convolutions mit  $S=1$
- (-)
- Pretraining nicht direkt möglich  
 $\rightarrow$  zu wenige Daten für E2E

## CNN - LSTM



Input                  Visuelle Features                  temporaler kontext                  Output

## RNN

- für Inputsequenzen
- interner State
- Gewichte über alle Zeitschritte geteilt



$$\begin{aligned} h_t &= f_w(h_{t-1}, x_t) \\ &= \tanh(w_{hh} \cdot h_{t-1} + w_{xh} \cdot x_t) \end{aligned}$$

$$y_t = w_{hy} \cdot h_t$$

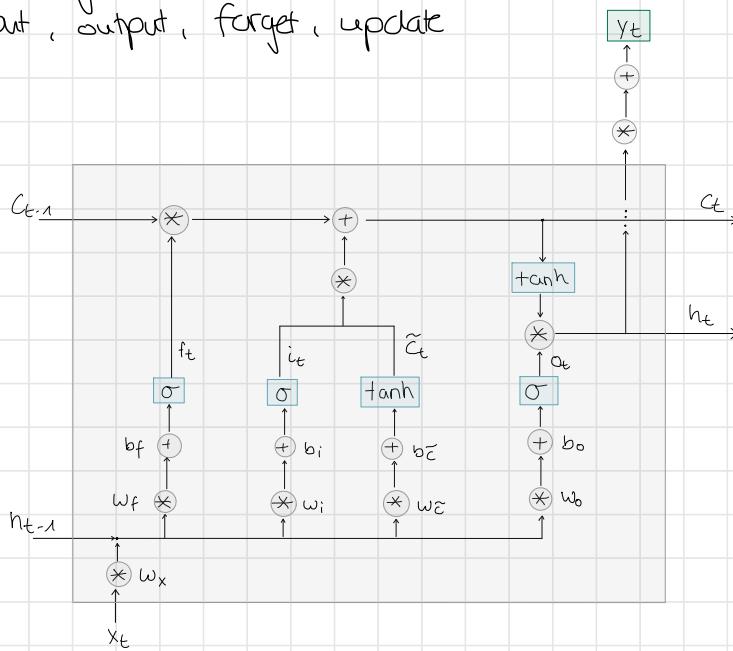
Training: Backprop through time

- Netz ausrollen
- Fwd Pass berechnen
- prediction scores berechnen  $\rightarrow$  neg log lik.
- Soft-Max (Klassifikation)
- Backprop Gradienten von allen Ausgaben

(-) Vanishing / exploding gradients

## LSTM

Idee: Änderungen am Zustand durch Gates kontrollieren  
 → input, output, forget, update

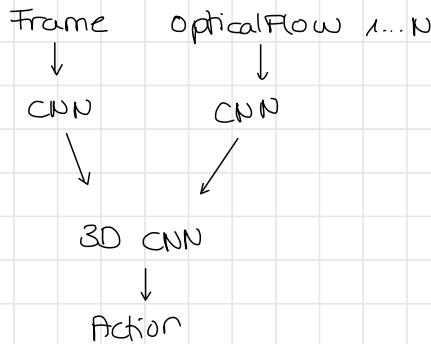


## 3D

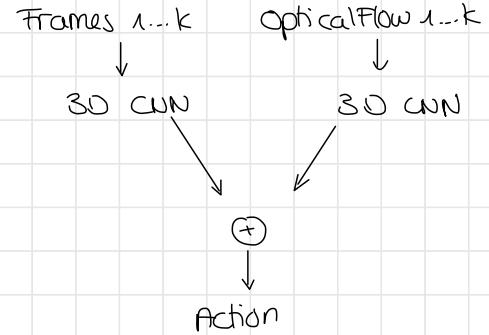
- two-stream inflated 3D Conv-Net

## 2 Architekturen

### fused CNNs



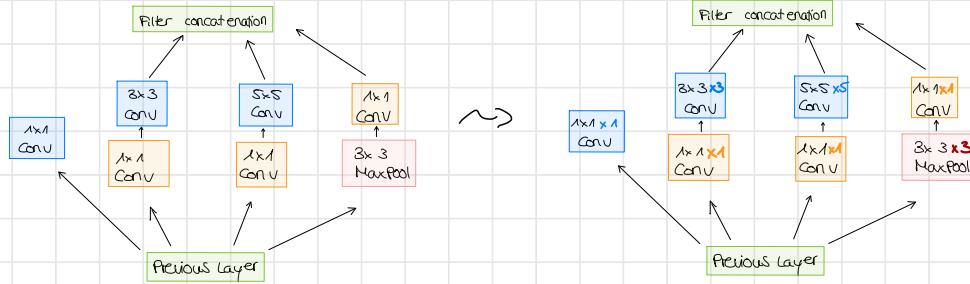
### E2E CNNs



## 3D Inception

- Inception Module mit 2D -Filtern pretrainen auf ImageNet
- Filter N mal entlang zeitachse kopieren

$$\rightarrow N \times N \rightsquigarrow N \times N \times N$$



- kein temporal Pooling bei 1., 2. Max Pooling

## Benchmarking

- Video datasets
- Vergleich schwierig (Pretraining, Trainingsdaten)

a) CNN - LSTM	b) C3D	c) 2-Stream	d) Fused 3D	e) 18D E2E
9 M Params	79 M	12 M	39 M	25M

$$e) > d) \approx c) > a) > b)$$

zu viele params, kein Pretr.

=> **Pretraining** wichtig!  
temporaler / langzeit kontext aktuell eher unwichtig

videos kurz

## Auswahl Architektur

- 2D vs. 3D (= Bildbasiert vs. Videobasiert)
- Input: RGB, optical flow, ...
- Fusion strategy: temporal Conn. / LSTM  $\rightarrow$  besser für longterm Activities / fine-grained

## Verbesserungen

- P-CNN: pose-based features mit benutzen