

Unterlagen  
zur  
Vorlesung

**IPCV 2002<sup>1</sup>**

Summer School on Computer Vision

**D. Paulus<sup>2</sup>**

Color Histogram Algorithms

31. Juli 2002

Universität Koblenz-Landau  
Computervisualistik  
Sommersemester / Summer 2002  
Version mit überwiegend deutschem Text

---

<sup>1</sup>[http://www.uni-koblenz.de/~lb/lb\\_activities/ipcv02/ipcv02.html](http://www.uni-koblenz.de/~lb/lb_activities/ipcv02/ipcv02.html)

<sup>2</sup><http://www.uni-koblenz.de/~paulus>



Dieses Dokument enthält die deutsche Fassung des Kursmaterials, das auf der Sommerschule im Kurs

Color Histogram Algorithms

verwendet wird.



# Einleitung

Auf diesen Folien sind einige Ergänzungen zum Material zu finden, das in der Vorlesung eingesetzt wird.

Für die meisten vorgestellten Algorithmen gibt es interaktive Demonstrationen.

Die Programme finden sich in der [PUMA Test Page](#)<sup>3</sup>. Weiter Informationen über PUMA und einige der Algorithmen, z B. zu [Histogram Backprojection](#), kann gefunden werden in [\[PH99\]](#).

Material zum Histogrammvergleich findet sich in [Aktives Bildverstehen](#)<sup>4</sup>. Dieses Buch [\[Pau01\]](#) ist als PDF-Datei verfügbar.

---

<sup>3</sup><http://www.bit.uni-koblenz.de/pumatest.php>

<sup>4</sup><http://www.der-andere-verlag>

## 3.1 Color Histograms

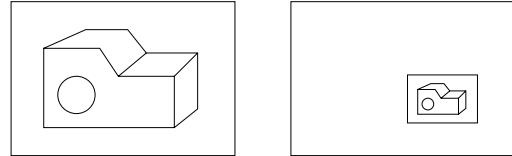
**Objektlokalisierung:** wichtige Aufgabe im Rechnersehen

Verwendung von Farbe zur Objekterkennung wurde vorgeschlagen in [SB91]: Verfahren: Histogrammschnitt und Histogrammrückprojektion

→ Verteilung der Farben

→ Farbhistogramme

Generelle Ideen:



- Versuch dasjenige Teilbild der Szene zu finden, für das das Histogramm den geringsten Abstand zum Objekthistogramm hat

Größe  $N_M \times M_M$  des Teilbilds muss bestimmt werden!

- Rückprojektion

### Farbhistogramme

Annahme: Großaufnahme eines Objekts ist gegeben als

Bild  $\mathbf{f} = [f_{ij}]_{i=1\dots M, j=1, \dots, N}$

wobei  $f_{ij}$  ein Pixel im RGB Farbraum ist, d. h.  $f_{ij} = (r_{ij}, g_{ij}, b_{ij})^T$ .

Dieses Objekt wird gesucht im Bild  $\mathbf{f}'$ .

Berechnung eines Histogramms  $\mathbf{S}$  für ein Teilbild der Szene und des Histogramms  $\mathbf{T}$  für das Objekt

$$\mathbf{S} = [S_l]_{l=1\dots N_L} \quad \mathbf{T} = [T_l]_{l=1\dots N_L} \quad (3.1)$$

wobei die Anzahl der Urnen  $N_L$  von der Quantisierung und vom Farbraum abhängt.

### Histogrammberechnung

Die Funktion  $\zeta$  bildet einen Farbpixel in den Index der Urne ab

**Beispiel:** für ein Histogramm im RGB mit  $4 \times 4 \times 4$  Urnen ( $N_L = 64$ ) und Farbkomponenten aus dem Bereich  $0 \dots 255$

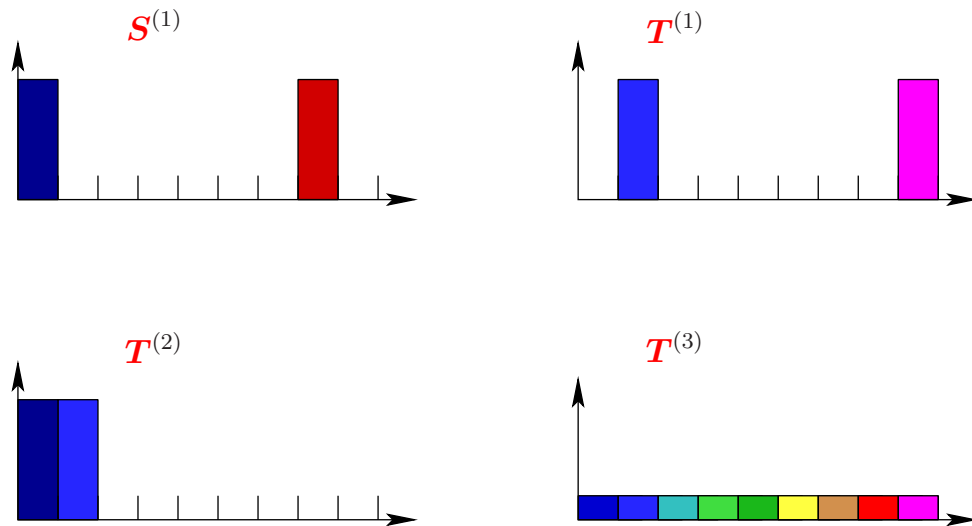
$$\zeta : \begin{cases} \mathbb{R}^3 & \rightarrow \{1, \dots, N_L\} \\ \mathbf{f}_{ij} = (r_{ij}, g_{ij}, b_{ij})^T & \rightarrow [r_{ij}/64] * 16 + [g_{ij}/64] * 4 + [b_{ij}/64] \end{cases} \quad (3.2)$$

Die Elemente des Histogramms  $\mathbf{T}$  sind

$$T_l = |\{(i, j) | \zeta(\mathbf{f}_{ij}) = l, i = 1 \dots M_T, j = 1, \dots N_T\}| \quad (3.3)$$

Analog:  $\mathbf{S}, S_l$  für Bild der Größe:  $M_S \times N_S$

Erforderlich: Vergleich von Histogrammen



Erforderlich: Vergleich von Histogrammen

**Histogrammschnitt** - vorgestellt in [SB91]. ist eine Verallgemeinerung der  $L_1$  Norm (*Minkowski Abstand*):

$$\cap(\mathbf{S}, \mathbf{T}) = \sum_{l=1}^{N_L} \min\{S_l, T_l\} \quad (3.4)$$

Erweitert auf die so genannte *focused intersection* mit *aktiver Suche* in [VMH97]

## Aktive Suche

- so genannte *sliding sum* zur Histogrammberechnung (Komplexität unabhängig von der Fenstergröße)
- für Vergleiche, die die Dreiecksungleichung erfüllen, können Abschätzungen für den Vergleich gemacht werden  
(z. B. für Histogrammschnitt in [VMH97])

**Summe der Abstandsquadrate (SSD)** - definiert als:

$$SSD(\mathbf{S}, \mathbf{T}) = \sum_{l=1}^{N_L} (T_l - S_l)^2 \quad (3.5)$$

*Chi-square Test* - aus der Statistik.

Verschiedene Verfahren in der Literatur, z. B. in [PBRT98, Sch97, PFTV88]

Hier (vgl. [Sch97])

$$\chi^2(\mathbf{S}, \mathbf{T}) = \sum_{l=1}^{N_L} \frac{(S_l - T_l)^2}{T_l} \quad (3.6)$$

Laut [Sch97] beste Ergebnisse.

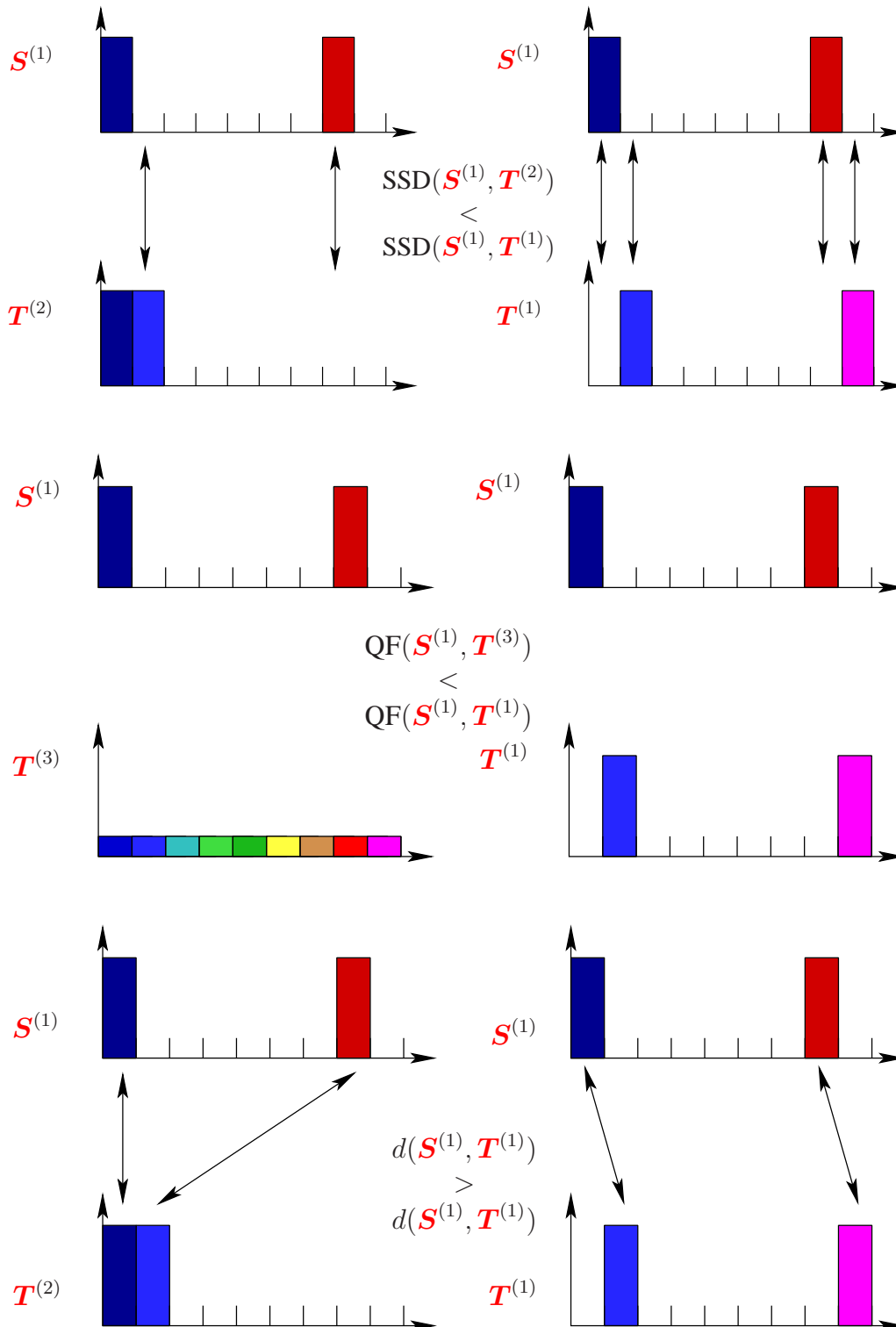


Bild zeigt gewünschte Eigenschaften von Histogrammabstand

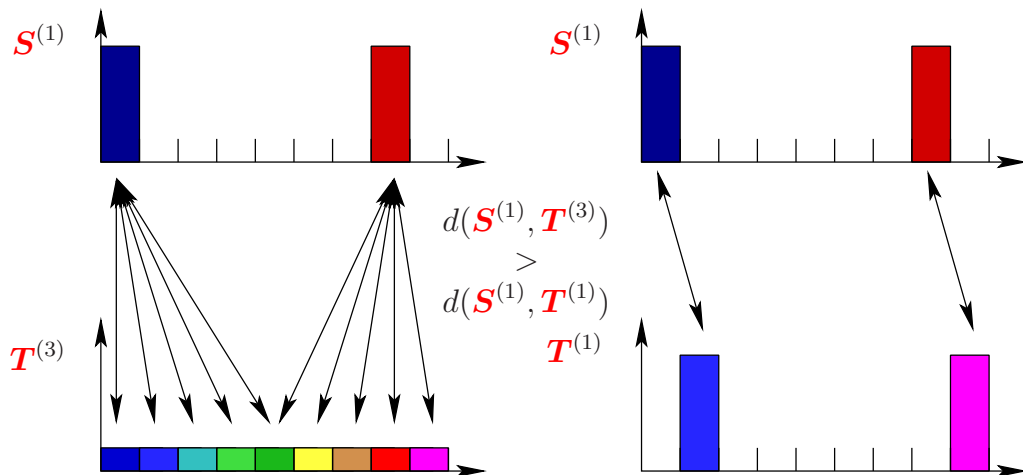


Bild zeigt gewünschte Eigenschaften von Histogrammabstand

**Farbräume**

**RGB** - Standard-Farbraum;  $r, g, b \in [0, 1]$

**(Normierter) RG** - ist intensitätsnormiert:

$$(r', g') = \left( \frac{r}{r + g + b}, \frac{g}{r + g + b} \right) \tag{3.7}$$

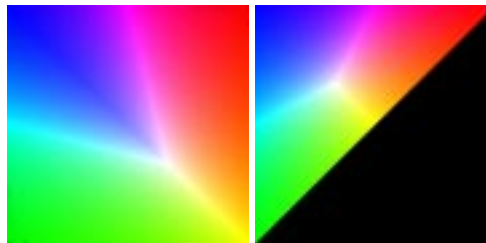


Abbildung 3.1: RG2-Farbraum

**YUV** Nach YCbCr **Standard**<sup>5</sup>

**CIElab**

$$r'' = \begin{cases} r' + g' & : \text{wenn } r' \geq g' \\ 2r' & : \text{sonst} \end{cases} \quad g'' = \begin{cases} 2g' & : \text{wenn } r' \geq g' \\ r' + g' & : \text{sonst} \end{cases} \tag{3.8}$$

Wir verwenden eine einfache Transformation des *RGB*-Raums:

Following the *ITU (International Telecommunication Union) R-601 Recommendation* , we used the YCbCr standard.

<sup>5</sup><http://www.inforamp.net/~poynton/ColorFAQ.html>

## Histogrammrückprojektion

### Histogrammrückprojektion

Gegeben: Farbbild $[f_{ij}]$ , Modellhistogramm $\mathbf{T} = [T_k]_{k=1\dots K}$ eines Objekts, Gesucht Position des Objekts $(i_t, j_t)$
Berechne Farbhistogramm $\mathbf{H} = [H_k]_{k=1\dots K}$ vom aktuellen Bild
FOR Jedes Histogramm "Bin" $k \in \{1, \dots, K\}$
$R_k = \min\{\frac{T_k}{H_k}, 1\}$ (Bilde Verhältnishistogramm $\mathbf{R} = [R_k]_{k=1\dots K}$ )
FOR Alle Positionen $(i, j)$ im Bild
$A_{i,j} := R_h(\mathbf{f}_{i,j})$ , wobei $\mathbf{f}_{i,j}$ den Farbvektor an der Stelle $(i, j)$ bezeichnet
$\mathbf{B} := \mathbf{D}_r \star \mathbf{A}$ , wobei $\star$ die Faltung bezeichnet
$(i_t, j_t) := \operatorname{argmax}_{i,j}(B_{i,j})$

Es seien  $N_o$  Objekte  $\{O_1, \dots, O_{N_o}\}$  gegeben, deren Auftretswahrscheinlichkeiten durch die a-priori Wahrscheinlichkeiten der Objekte  $p(O_i)$  beschrieben wird. Für jedes Objekt  $O_i$  gibt die bedingte Wahrscheinlichkeit  $p(\mathbf{f}|O_i)$  ( $i \in \{1, \dots, N_o\}$ ) die Wahrscheinlichkeiten der Merkmale — in diesem Fall der Farbvektoren  $\mathbf{f}$  — an, die durch Histogramme von Bildern der Objekte geschätzt werden. Gegeben ist nun ein Bild von Farbpixeln  $\mathbf{f}$ , bzw. die Verteilung der Farbwerte  $p(\mathbf{f})$  als Histogramm. Gesucht ist die Position des Objekts im Bild. Die Umformung der Formel für die bedingte Wahrscheinlichkeit

$$p(O_i)p(\mathbf{f}|O_i) = p(\mathbf{f})p(O_i|\mathbf{f})$$

in

$$p(O_i|\mathbf{f}) = p(O_i) \frac{p(\mathbf{f}|O_i)}{p(\mathbf{f})}$$

ergibt nun die Wahrscheinlichkeit eines Objekts bei beobachtetem Farbvektor  $\mathbf{f}$  als die a-priori Wahrscheinlichkeit des Objekts mal einem Quotienten. Der Quotient wird durch das Verhältnishistogramm geschätzt. Die a-priori Wahrscheinlichkeit  $p(O_i)$  des Objekts wird für jeden Ort im Bild als gleich angenommen und muss daher in der Maximierung nicht berücksichtigt werden.

Die Rückprojektion lässt sich als ein Wert proportional zur Schätzung einer Wahrscheinlichkeit interpretieren. Es seien  $N_o$  Objekte  $\{O_1, \dots, O_{N_o}\}$  gegeben, deren Auftretswahrscheinlichkeiten durch die a-priori Wahrscheinlichkeiten der Objekte  $p(O_i)$  beschrieben wird. Für jedes Objekt  $O_i$  gibt die bedingte Wahrscheinlichkeit  $p(\mathbf{f}|O_i)$  ( $i \in \{1, \dots, N_o\}$ ) die Wahrscheinlichkeiten der Merkmale — in diesem Fall der Farbvektoren  $\mathbf{f}$  — an, die durch Histogramme von Bildern der Objekte geschätzt werden. Gegeben ist nun ein Bild von Farbpixeln  $\mathbf{f}$ , bzw. die Verteilung der Farbwerte  $p(\mathbf{f})$  als Histogramm. Gesucht ist die Position des Objekts im Bild. Die Umformung der Formel für die bedingte Wahrscheinlichkeit

$$p(O_i)p(\mathbf{f}|O_i) = p(\mathbf{f})p(O_i|\mathbf{f})$$

in

$$p(O_i|\mathbf{f}) = p(O_i) \frac{p(\mathbf{f}|O_i)}{p(\mathbf{f})}$$

ergibt nun die Wahrscheinlichkeit eines Objekts bei beobachtetem Farbvektor  $\mathbf{f}$  als die a-priori Wahrscheinlichkeit des Objekts mal einem Quotienten. Der Quotient wird durch das Verhältnishistogramm geschätzt. Die a-priori Wahrscheinlichkeit  $p(O_i)$  des Objekts wird für jeden Ort im Bild als gleich angenommen und muss daher in der Maximierung nicht berücksichtigt werden.

## Neue Abstandsmaße

### Earth Mover's Distance (EMD)

Neues Maß, das in [RTG98] vorgestellt wurde; repräsentiert die minimalen Transportkosten von einem Histogramm in ein anderes.

$$\text{EMD}(\mathbf{S}, \mathbf{T}) = \min_{\mathbf{F}} \frac{\sum_{\mu=1}^{N_L} \sum_{\nu=1}^{N_L} F_{\mu\nu} d_{\mu\nu}}{\sum_{\mu=1}^{N_L} \sum_{\nu=1}^{N_L} F_{\mu\nu}} \quad (3.9)$$

EMD verwendet den Farbabstand  $d_{ij}$ , d. h. den Abstand zweier Urnen  $S_i$  und  $T_j$ ; hier wurde der geometrische Abstand der Farben gewählt

Zur Berechnung des EMD muss ein Transportproblem gelöst werden, wobei eine Matrix  $\mathbf{F} = [F_{\mu,\nu}]_{1,\dots,N_L,1,\dots,N_L}$  den Fluss von Urne  $S_\mu$  in die Urne  $T_\nu$  repräsentiert.

### EMD (Fortsetzung)

#### Constraints:

Der Fluss muss positiv sein ( $0 \leq F_{\mu,\nu}$ ) und weitere Einschränkungen sind:

$$\sum_{\mu=1}^{N_L} F_{\mu\nu} = T_\nu / (M_T N_T) \quad \sum_{\nu=1}^{N_L} F_{\mu\nu} = S_\mu / (M_S N_S) \quad \sum_{\mu=1}^{N_L} \sum_{\nu=1}^{N_L} F_{\mu\nu} = 1 \quad (3.10)$$

Damit ist EMD recht rechenaufwändig. Lösung mit linearer Optimierung, beispielsweise mit der Simplex Methode.<sup>6</sup> (erfordert  $\mathcal{O}(N_L^4)$  Speicher)

Machbar:  $N_L \leq 64$

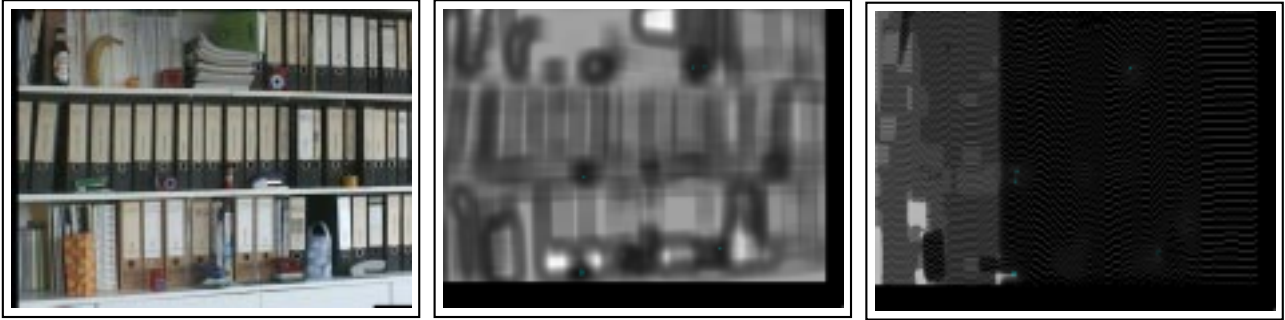
NB: Relative Häufigkeiten!

### Testobjekte



Bildstichprobe für Testobjekte

<sup>6</sup>Hier wurde der Code von Y. Rubner aus <http://robotics.stanford.edu/~rubner/emd/default.htm> verwendet



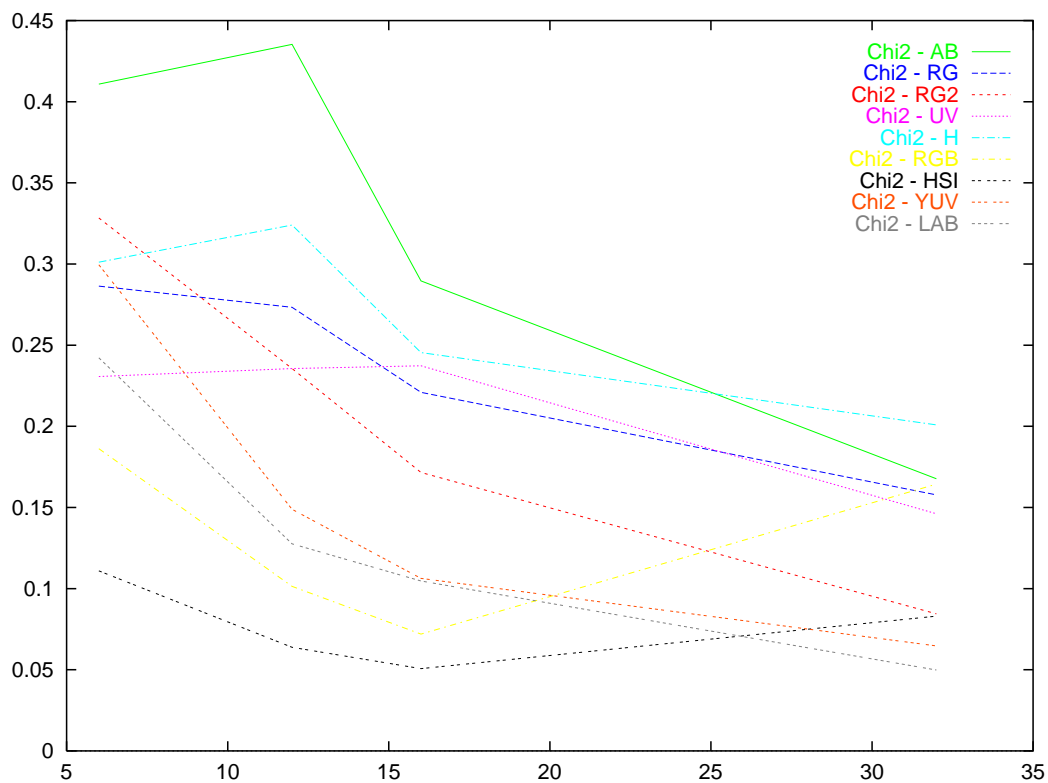
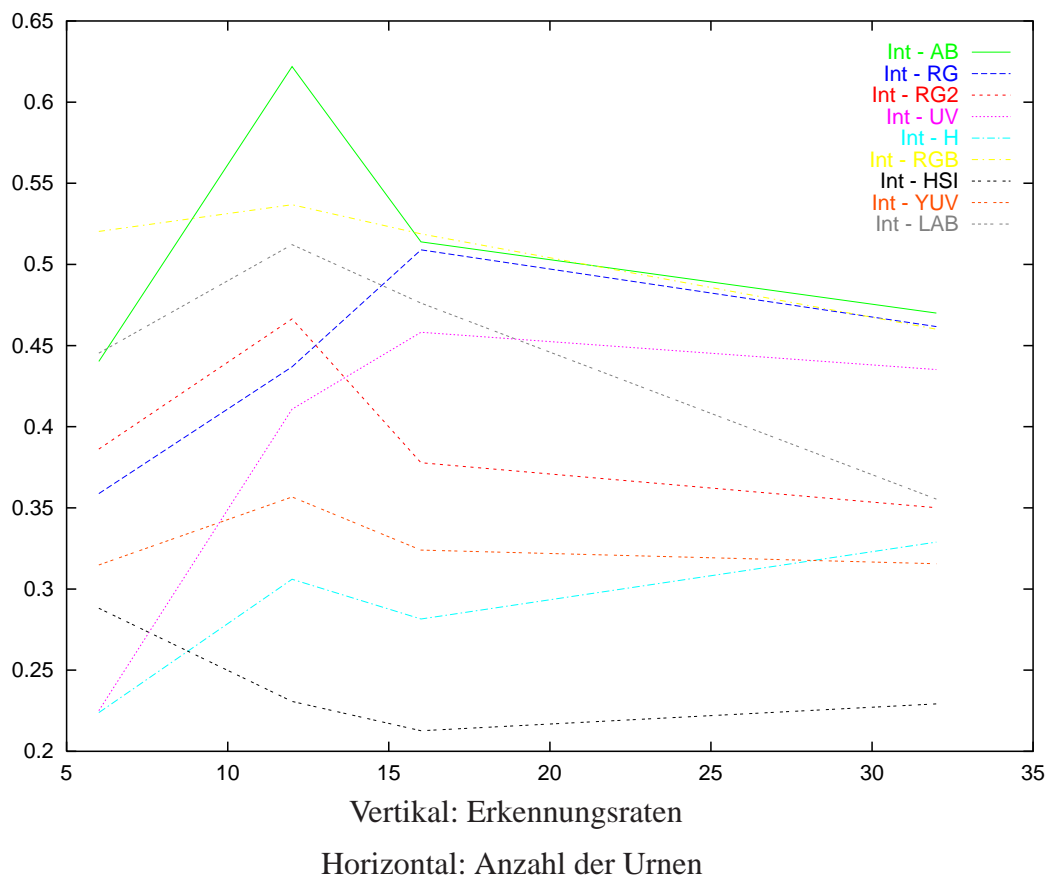
Links: Szene, mitte: **Histogrammschnitt**, rechts: **aktive Suche**

Ergebnisse der Suche nach Objekt **11**. Nur 13% der Vergleiche sind zur aktiven Suche erforderlich  
 Dunkel heißt: hier ist es gefunden. Mit Punkten sind die ersten 5 Hypothesen markiert!

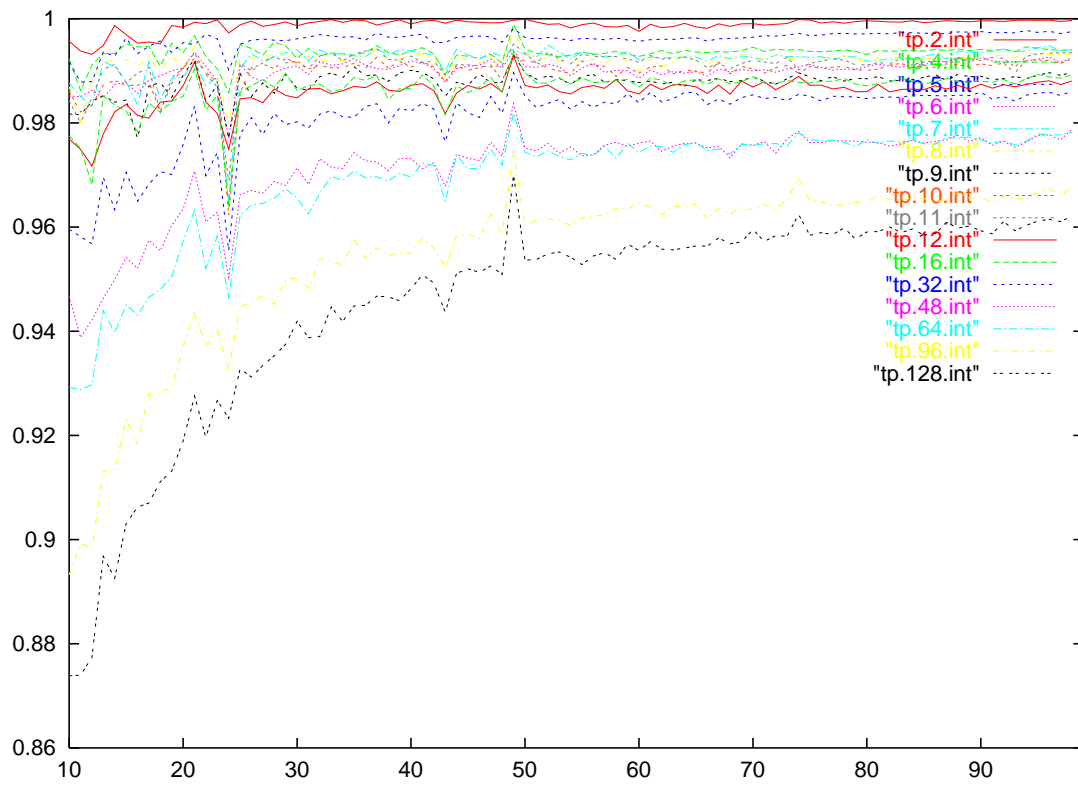
### **Testszene**

Getestet wurden:

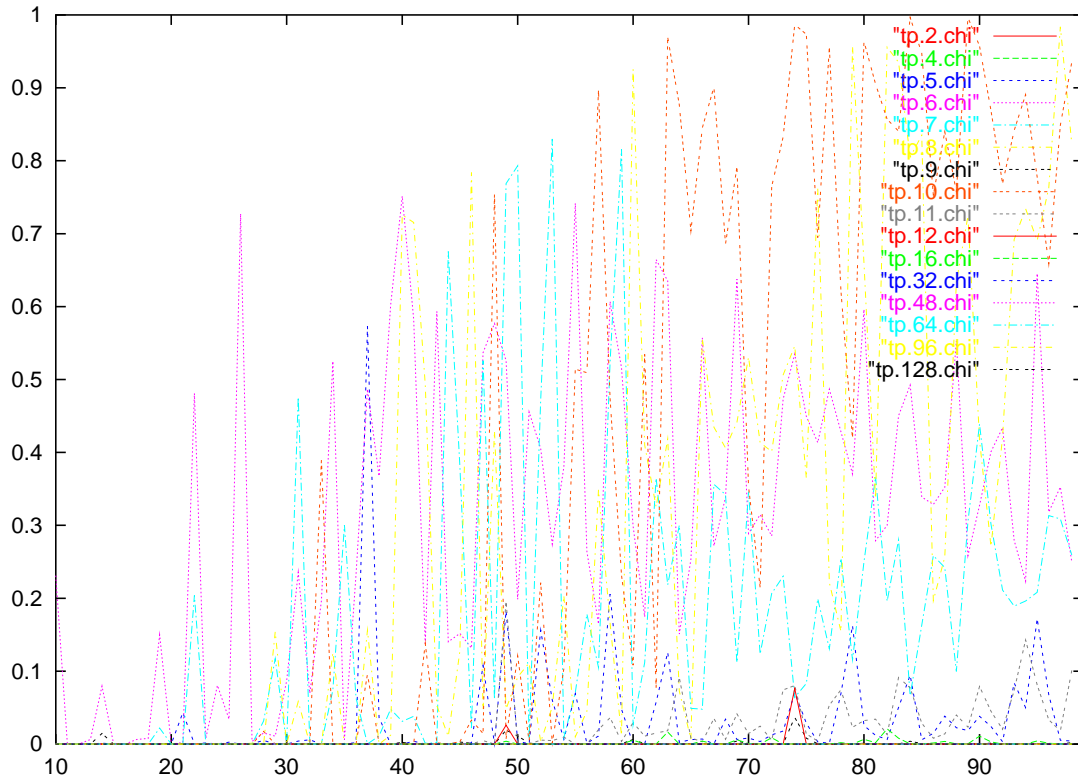
- verschiedene Farbräume
- verschiedene Quantisierungen
- dieselbe Szene mit verschiedenen Beleuchtungen
- SSD,  $\chi^2$ , HI, EMD, DP, Kullbach-Divergenz, ...
- $\approx$  20 Objekte und 20 Szenen
- zusammen 50000 Experimente



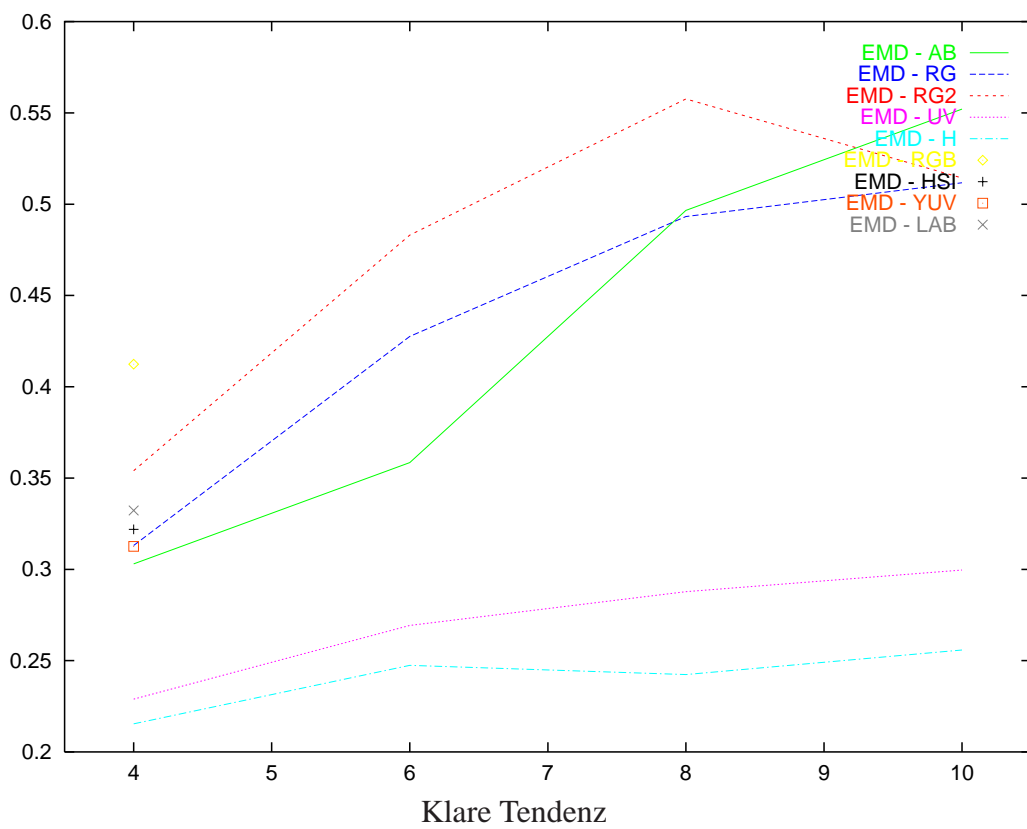
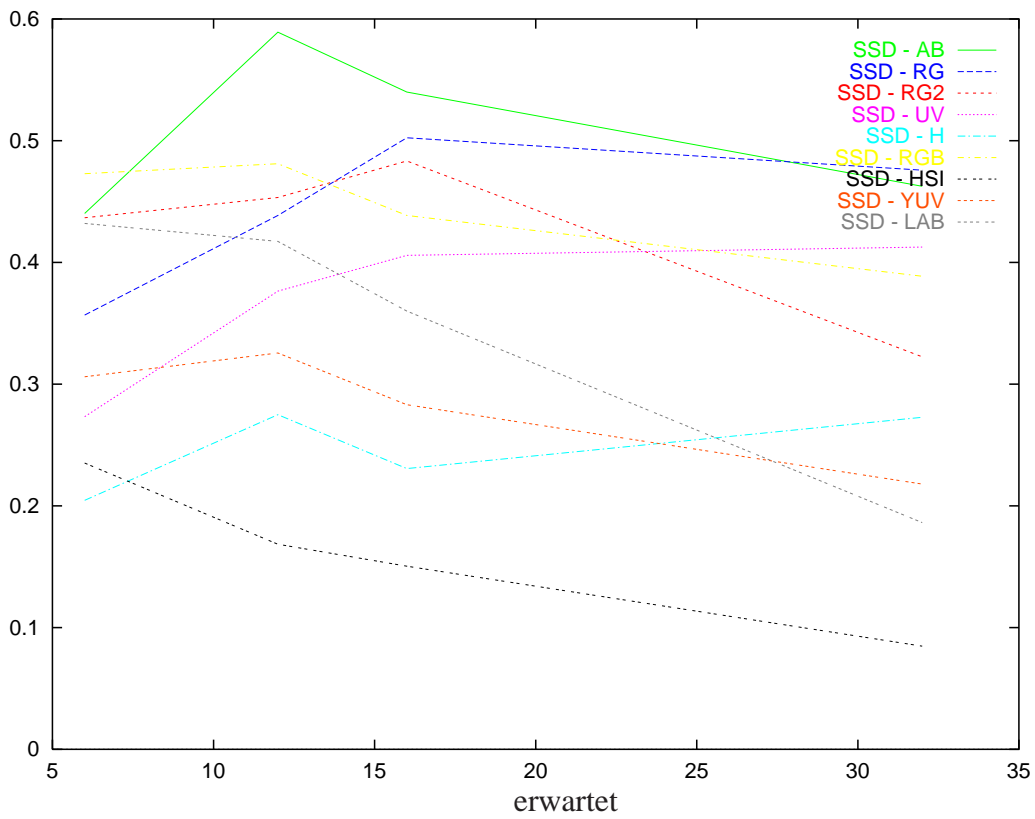
Gemittelte Erkennungsraten für  $\chi^2$  auf schwarzem Hintergrund  
Nicht so gut, wie erwartet



Histogrammschnitt mit sich selbst, bilinear skaliert, rg

 $\chi^2$  Test mit sich selbst, bilinear skaliert, uv Chi2 uv

Besser als



50% Erkennungsrate mit jedem Farbraum außer UV und H

Obj.	Exp.	found	1	2
1	4737	40.08%	25.64%	5.65%
2	4729	46.56%	30.53%	7.46%
3	4720	15.72%	7.41%	3.05%
4	4711	50.37%	28.06%	9.17%
5	4728	8.79%	2.41%	1.90%
6	4719	20.87%	8.58%	3.36%
7	4724	21.25%	9.39%	3.25%
8	4716	26.78%	12.53%	4.26%

Erste Spalte: Gesamtanzahl der Experimente. Zweite Spalte: Erkennungsrate

Obj.	Exp.	found	1	2
9	4594	50.28%	37.61%	4.65%
10	4697	34.46%	15.90%	6.77%
11	4597	32.67%	13.26%	8.15%
12	4701	46.05%	20.71%	10.21%
13	4599	19.15%	10.02%	3.30%
14	4703	24.21%	10.73%	4.27%
15	4568	46.71%	27.49%	8.12%
16	4699	8.17%	2.95%	1.57%
17	4696	60.43%	44.01%	7.21%

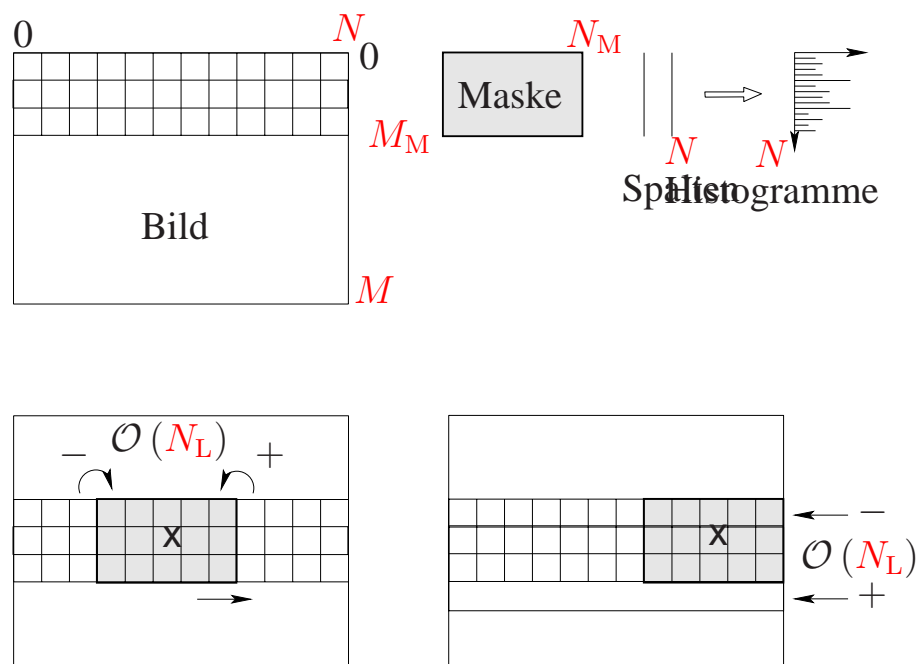


Abbildung 3.2: Schnelle Histogrammberechnung. Oben: Initialisierung der Hilfshistogramme, Maske, Histogramm einer Teilspalte. Unten links: Weiterschieben der Maske um einen Pixel. Unten rechts: Weiterschieben der Hilfshistogramme um eine Zeile.

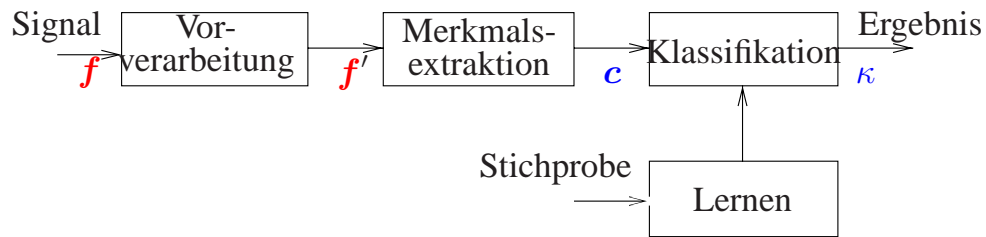


Abbildung 3.3: Verarbeitung einfacher Muster nach [Nie83] zusammen mit der Notation für Signale, Merkmale und Klassen

## 3.2 Klassifikation

Sample:

- Training
- Test
- (Evaluation / Verifikation)

Einfache Objekterkennung mittels Farb-Histogrammen:

Merkmale: Histogramme

Frage: Wie wird klassifiziert?

Frage: Wie wird gelernt?

Der folgende Text konnte ist derzeit nur auf Englisch verfügbar  
 The following material is taken from [PH99, Chapter 21ff]

### 3.3 Numerical Pattern Classification

Pattern classification:

- features are computed using image or speech signals
- features of the same class occupy a compact area
- features of the different classes can be separated from each other
- assignment of features to a discrete class

Formal definition:

- numerical feature vector:  $\mathbf{c} \in \mathbb{R}^d$
- assignment function (discrete):

$$\zeta : \begin{cases} \mathbb{R}^d & \rightarrow & \{1, 2, \dots, K\} \\ \mathbf{c} & \mapsto & \kappa \end{cases} \quad (3.11)$$

#### Examples

- speech recognition
  - *word recognition*: map a signal to a single word (lexicon)
  - *identification of languages*: assign a signal to a language class
- image recognition
  - *binarization*: assign each pixel to a binary value
  - *face recognition*: assign each pixel to a binary value
  - *quality control*: assign each image to a binary value
  - *object recognition*: assign an image to an object
  - *object localization*: assign an image to an object and

**Problem:** How to select the decision rule???

### 3.3.1 General Notes on Classifiers

Basic concepts of decision theory:

- construction of classifiers is based on:
  - observable data
  - experience
  - domain knowledge
- learning stage, generalization property
- types of learning:
  - supervised learning
  - unsupervised learning

Judgment of classifiers:

- probability of misclassification
- computational complexity
- minimization of costs

Classifier minimizing the probability of a incorrect assignment with forced desicion: Bayes classifyer

The error probability  $p_B$  of this optimal classifier is the so-called *Bayesian error probability*.

A discrimination rule is called *consistent*, if an increase of the sample data size implies the approximation of the Bayesian error  $p_B$

Of theoretical interest is the following question:

- Which decision rule optimizes the quality criterion of an classifier???

#### Basic Assumptions:

- feature vectors of the same class: small distance
- feature vectors of different classes: large distance
- distance measures: Euclidean distance, city block metric, probability density functions

### 3.3.2 Design of Classifiers

Classifier

- discriminating functions
- statistics (either parametric, or non-parametric)

### 3.3.3 Linear Discriminants

The discussion of linear discriminants is first restricted to two classes, i.e., we deal only with binary classes  $\Omega_1$  and  $\Omega_2$ . This simplifies the understanding of linear classifiers, and allows an easier and clearer insight into basic concepts.

- restriction to two classes:  $\Omega_1$  and  $\Omega_2$
- two classes partition the feature space
- assumption:
  - a hyperplane defines the partition
  - hyperplanes are defined by linear functions

Decision rule of linear discriminant classifiers:

$$\zeta(\mathbf{c}) = \begin{cases} 1, & \text{if } q(\mathbf{c}) > 0 \\ 2, & \text{otherwise} \end{cases}, \quad (3.12)$$

$$q(\mathbf{c}) = q_0 + \sum_{i=1}^d q_i c_i = q_0 + (q_1, q_2, \dots, q_d) \cdot (c_1, c_2, \dots, c_d)^T, \quad (3.13)$$

where  $q_i \in \mathbb{R}$ .

**Examples:**

- linear discriminants for one- and two-dimensional features
- one-dimensional case: discriminant is a simple threshold (see binarization of gray-level images)
- two-dimensional case:  $q(\mathbf{c}) = q_0 + q_1 c_1$ .
- optimal classifier: define the splitting line such that the number of misclassifications is minimal

**Extension to multiple classes:**

- define for each  $\Omega_1$  and  $\Omega_2$  linear functions  $q_1(\mathbf{c})$  and  $q_2(\mathbf{c})$
- splitting function:  $q(\mathbf{c}) := q_1(\mathbf{c}) - q_2(\mathbf{c})$

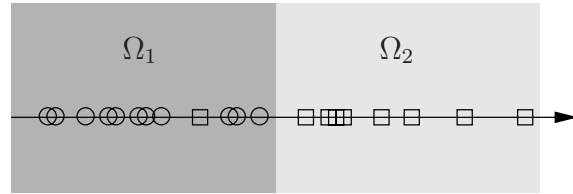


Abbildung 3.4: Two classes in a one-dimensional feature space

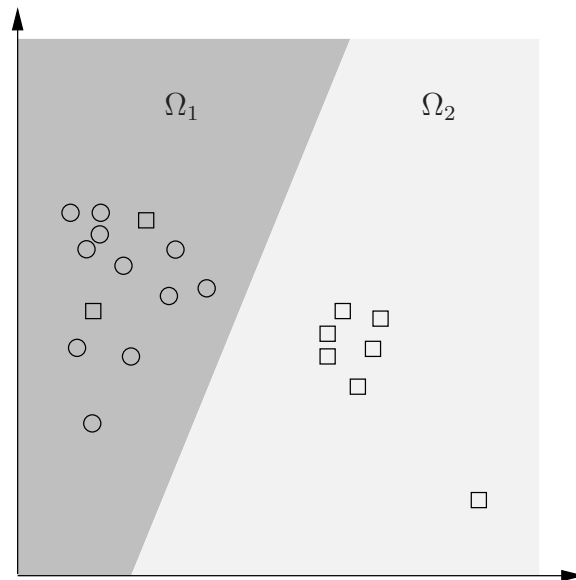


Abbildung 3.5: Two classes in a two-dimensional feature space

- decide for class  $\Omega_1$ , if  $q(\mathbf{c}) > 0$
- $q(\mathbf{c}) > 0 \leftrightarrow q_1(\mathbf{c}) > q_2(\mathbf{c})$
- decision rule for many classes:

$$\zeta(\mathbf{c}) = \operatorname{argmax}_{\lambda} \tilde{q}_{\lambda}(\mathbf{c}) = \operatorname{argmax}_{\lambda} \left( \tilde{q}_{\lambda,0} + \sum_{i=1}^d \tilde{q}_{\lambda,i} c_i \right) , \quad (3.14)$$

where  $\tilde{q}_{\lambda}$  denotes the *splitting polynomial* of class  $\Omega_{\lambda}$  which is defined by the  $(d+1)$ -dimensional vector  $\tilde{\mathbf{q}}_{\lambda}^T = (\tilde{q}_{\lambda,0}, \tilde{q}_{\lambda,1}, \dots, \tilde{q}_{\lambda,d})$ .

NB: the polynomial does *not* separate the samples geometrically

Question: how to find the coefficient vectors  $\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_K$  using a sample set of classified feature vectors.

→ Linear optimization problem

Let

$$\omega_{\kappa} = \{^1\mathbf{c}_{\kappa}, ^2\mathbf{c}_{\kappa}, \dots, ^{N_{\kappa}}\mathbf{c}_{\kappa}\}$$

be the set of sample data assigned to class  $\Omega_{\kappa}$ , i.e., we observe  $N_{\kappa}$  samples for classes  $\Omega_{\kappa}$ ,  $\kappa = 1, 2, \dots, K$ . The complete set of samples is

$$\omega = \bigcup_{\kappa=1}^K \omega_{\kappa} . \quad (3.15)$$

We define (unique) *indicator functions*  $\chi_{\kappa}$ , where  $\kappa = 1, 2, \dots, K$ :

$$\chi_{\kappa}(\mathbf{c}) = \begin{cases} 1 & , \text{ if } \mathbf{c} \text{ belongs to } \Omega_{\kappa} \\ 0 & , \text{ otherwise } . \end{cases} \quad (3.16)$$

Let  $^j\mathbf{c}_{\lambda}$  be the  $j$ -th sample feature of  $\omega_{\lambda}$  which is known to belong to class  $\Omega_{\lambda}$ .

Using (3.12) we decide for the class  $\Omega_{\kappa}$  just depending of *differences* of values in polynomials

Without loss of generality we expect the function value to be *one* for the correct class, and *zero* otherwise.

*Ideal splitting function* (3.16)

For each  $d$ -dimensional feature vector  $^j\mathbf{c}_{\lambda} = (^j c_{\lambda,1}, ^j c_{\lambda,2}, \dots, ^j c_{\lambda,d})^T$  ( $\lambda = 1, 2, \dots, K$ , and  $j = 1, 2, \dots, N_{\lambda}$ ) of the training set we get  $K$  equations ( $\kappa = 1, 2, \dots, K$ ):

$$q_{\kappa}(^j\mathbf{c}_{\lambda}) = q_{\kappa,0} + \sum_{i=1}^d q_{\kappa,i} ^j c_{\lambda,i} = \chi_{\kappa}(^j\mathbf{c}_{\lambda}) , \quad (3.17)$$

which are *linear* in the coefficients of splitting functions. This system of linear equations can be written in matrix notation. For that purpose, we define extended features by  $^j\tilde{\mathbf{c}}_{\lambda} = (1, ^j c_{\lambda,1}, ^j c_{\lambda,2}, \dots, ^j c_{\lambda,d})^T$  by just adding the component 1. This trick allows us to introduce the a matrix  $\mathbf{A} \in \mathbb{R}^{D \times K(d+1)}$  which is shown below in (3.19) where

$$D = K \cdot \sum_{\kappa=1}^K N_{\kappa} . \quad (3.18)$$

$$\mathbf{A} = \begin{pmatrix}
{}^1\tilde{\mathbf{c}}_1^T & \mathbf{0}^T & \dots & \mathbf{0}^T \\
\mathbf{0}^T & {}^1\tilde{\mathbf{c}}_1^T & & \mathbf{0}^T \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}^T & \dots & \mathbf{0}^T & {}^1\tilde{\mathbf{c}}_1^T \\
{}^2\tilde{\mathbf{c}}_1^T & \mathbf{0}^T & \dots & \mathbf{0}^T \\
\mathbf{0}^T & {}^2\tilde{\mathbf{c}}_1^T & & \mathbf{0}^T \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}^T & \dots & \mathbf{0}^T & {}^2\tilde{\mathbf{c}}_1^T \\
\vdots & \vdots & \ddots & \vdots \\
{}^{N_1}\tilde{\mathbf{c}}_1^T & \mathbf{0}^T & \dots & \mathbf{0}^T \\
\mathbf{0}^T & {}^{N_1}\tilde{\mathbf{c}}_1^T & & \mathbf{0}^T \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}^T & \dots & \mathbf{0}^T & {}^{N_1}\tilde{\mathbf{c}}_1^T \\
\vdots & \vdots & \ddots & \vdots \\
{}^1\tilde{\mathbf{c}}_K^T & \mathbf{0}^T & \dots & \mathbf{0}^T \\
\mathbf{0}^T & {}^1\tilde{\mathbf{c}}_K^T & & \mathbf{0}^T \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}^T & \dots & \mathbf{0}^T & {}^1\tilde{\mathbf{c}}_K^T \\
\vdots & \vdots & \ddots & \vdots \\
{}^{N_K}\tilde{\mathbf{c}}_1^T & \mathbf{0}^T & \dots & \mathbf{0}^T \\
\mathbf{0}^T & {}^{N_K}\tilde{\mathbf{c}}_1^T & & \mathbf{0}^T \\
\vdots & \vdots & \ddots & \vdots \\
\mathbf{0}^T & \dots & \mathbf{0}^T & {}^{N_K}\tilde{\mathbf{c}}_1^T
\end{pmatrix}
\quad
\mathbf{b} = \begin{pmatrix}
\chi_1({}^1\mathbf{c}_1) \\
\chi_2({}^1\mathbf{c}_1) \\
\vdots \\
\chi_K({}^1\mathbf{c}_1) \\
\chi_1({}^2\mathbf{c}_1) \\
\chi_2({}^2\mathbf{c}_1) \\
\vdots \\
\chi_K({}^2\mathbf{c}_1) \\
\vdots \\
\vdots \\
\chi_1({}^{N_K}\mathbf{c}_K) \\
\chi_2({}^{N_K}\mathbf{c}_K) \\
\vdots \\
\chi_K({}^{N_K}\mathbf{c}_K)
\end{pmatrix}$$

Let  $\mathbf{x} \in \mathbb{R}^{K(d+1)}$  be (3.19)

$$\mathbf{x} = (q_{1,0}, q_{1,1}, \dots, q_{1,d}, q_{2,0}, q_{2,1}, \dots, q_{2,d}, \dots, q_{K,0}, q_{K,1}, \dots, q_{K,d})^T \quad (3.20)$$

Computation of linear splitting functions now corresponds to solving the system of linear equations:

$$\mathbf{A} \mathbf{x} = \mathbf{b} \quad (3.21)$$

For real data, the probability that the vector  $\mathbf{b}$  is not in the range of the matrix  $\mathbf{A}$ , is one. Therefore, we compute the solution  $\mathbf{x}$  which minimizes the residual  $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|$ .

$$\mathbf{x} = \mathbf{A}^+ \mathbf{b} \quad (3.22)$$

The pseudo-inverse  $\mathbf{A}^+$  is usually computed using **SVD**

### 3.3.4 Polynomial Classifiers

A multivariate polynomial attached to each class  $\Omega_\lambda$

$$q_\lambda(\mathbf{c}) = \sum_{i_1, i_2, \dots, i_d=1}^m q_{\lambda, i_1, i_2, \dots, i_d} c_1^{i_1} c_2^{i_2} \dots c_d^{i_d} \quad , \quad (3.23)$$

decision rule (3.14) remains unchanged

Several degrees of freedom:

- degree  $m$
- coefficients  $q_{\lambda, i_1, i_2, \dots, i_d} \in \mathbb{R}$

Complex optimization problem with respect to the recognition rate

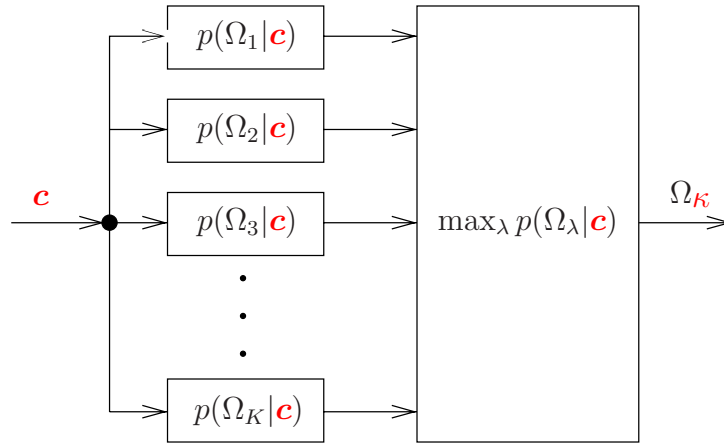


Abbildung 3.6: The principle of the Bayesian classifier

### 3.3.5 Bayesian Classifiers

Statistical characterization of classes: need

- a priori probability  $p(\Omega_{\kappa})$
- class-specific probability density function  $p(\mathbf{c} | \Omega_{\kappa})$  of each class  $\Omega_{\kappa} \in \Omega$ .  
(could be e.g. Gaussian or in the discrete case a histogram)

Example:

If all classes appear with the same probability:  $p(\Omega_{\kappa}) = p(\Omega_{\lambda})$  for  $\kappa, \lambda \in \{1, 2, \dots, K\}$ .

A posteriori probability

$$p(\Omega_{\kappa} | \mathbf{c}) = \frac{p(\Omega_{\kappa}) p(\mathbf{c} | \Omega_{\kappa})}{p(\mathbf{c})} = \frac{p(\Omega_{\kappa}) p(\mathbf{c} | \Omega_{\kappa})}{\sum_{\lambda=1}^K p(\Omega_{\lambda}) p(\mathbf{c} | \Omega_{\lambda})} . \quad (3.24)$$

Bayes classifier solves maximization problem:

$$\zeta(\mathbf{c}) = \operatorname{argmax}_{\lambda} p(\Omega_{\lambda} | \mathbf{c}) = \operatorname{argmax}_{\lambda} p(\Omega_{\lambda}) p(\mathbf{c} | \Omega_{\lambda}) . \quad (3.25)$$

For uniformly distributed classes ( $p(\Omega_{\kappa}) = p(\Omega_{\lambda})$  for all classes), the Bayesian decision rule (3.25) reduces to the maximum likelihood decision:

$$\begin{aligned} \zeta(\mathbf{c}) &= \operatorname{argmax}_{\lambda} p(\Omega_{\lambda} | \mathbf{c}) = \operatorname{argmax}_{\lambda} p(\Omega_{\lambda}) p(\mathbf{c} | \Omega_{\lambda}) \\ &= \operatorname{argmax}_{\lambda} p(\mathbf{c} | \Omega_{\lambda}) . \end{aligned} \quad (3.26)$$

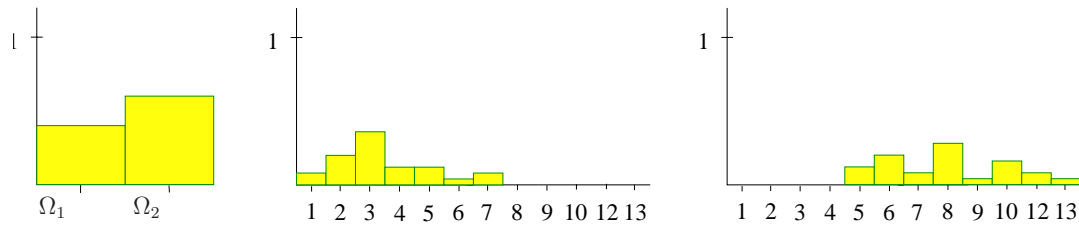


Abbildung 3.7: A priori probabilities (left), probabilities of scalar discrete features corresponding to class  $\Omega_1$  (middle), and class  $\Omega_2$  (right)

### 3.3.6 Properties of Bayesian Classifiers

Error probability of Bayesian classifiers  $p_B$

$p_B$ : lower bound for misclassifications of *all* classifiers

cost function with correct decision  $\rightarrow$  zero and i wrong decision  $\rightarrow$  one:

A priori probabilities  $p(\Omega_{\kappa})$ ,  $1 \leq \kappa \leq K$  can easily be computed from relative frequencies and classified sample data

Models for probability density functions: more difficult

- non-parametric: apply discrete probabilities (e.g. Histograms)
- parametric: use parametric densities instead of histograms  
most commonly used parametric density concerning statistical classification:

Histogram for non-parametric modeling

For  $d$ -dimensional feature vectors, normally distributed:

use multivariate Gaussian density function:

$$p(\mathbf{c}|\Omega_{\kappa}) = \frac{1}{\sqrt{|\det 2\pi \Sigma_{\kappa}|}} \exp\left(-\frac{(\mathbf{c} - \boldsymbol{\mu}_{\kappa})^T \Sigma_{\kappa}^{-1} (\mathbf{c} - \boldsymbol{\mu}_{\kappa})}{2}\right) . \quad (3.27)$$

Parameters  $\boldsymbol{\mu}_{\kappa}$  and  $\Sigma_{\kappa}$  can be estimated applying the maximum likelihood method, if a classified sample is available

### 3.3.7 From Bayesian to Geometric Classifiers

Specialization of Bayesian classifier of Gaussians to Euclidean distances

Let us consider two classes  $\Omega_1$  and  $\Omega_2$ . The discrete prior probabilities  $p(\Omega_1)$  and  $p(\Omega_2)$  as well as the Gaussian densities  $p(\mathbf{c}|\Omega_1)$  and  $p(\mathbf{c}|\Omega_2)$  are assumed to be known. then the Bayesian classifier decides for class  $\Omega_1$  if

$$p(\Omega_1)p(\mathbf{c}|\Omega_1) > p(\Omega_2)p(\mathbf{c}|\Omega_2) . \quad (3.28)$$

Now we specialize this decision rule by setting

$$\Sigma_1 = \Sigma_2 = \Sigma ; \quad (3.29)$$

taking the logarithm of both sides in (3.28), we get

$$\log p(\Omega_1) + \boldsymbol{\mu}_1^T \Sigma^{-1} \mathbf{c} - \frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 > \log p(\Omega_2) + \boldsymbol{\mu}_2^T \Sigma^{-1} \mathbf{c} - \frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2. \quad (3.30)$$

Now we specialize this decision rule for uniform priors

$$p(\Omega_1) = p(\Omega_2) . \quad (3.31)$$

to get the discriminant

$$\boldsymbol{\mu}_1^T \Sigma^{-1} \mathbf{c} - \frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 > \boldsymbol{\mu}_2^T \Sigma^{-1} \mathbf{c} - \frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 \quad (3.32)$$

which is a simplified version of the

$$(\mathbf{c} - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{c} - \boldsymbol{\mu}_1) < (\mathbf{c} - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{c} - \boldsymbol{\mu}_2) . \quad (3.33)$$

Assume  $\Sigma = \mathbf{1}$

$$(\mathbf{c} - \boldsymbol{\mu}_1)^T (\mathbf{c} - \boldsymbol{\mu}_1) < (\mathbf{c} - \boldsymbol{\mu}_2)^T (\mathbf{c} - \boldsymbol{\mu}_2) . \quad (3.34)$$

→ decision rule compares the quadratic Euclidean distances

$$\|\mathbf{c} - \boldsymbol{\mu}_1\|^2 < \|\mathbf{c} - \boldsymbol{\mu}_2\|^2 , \quad (3.35)$$

### 3.3.8 Nearest Neighbor Classifier

The nearest neighbor classifier requires a set of classified sample data:

For each element  $\mathbf{c}_i$  of  $\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_n\}$ : the class  $\zeta(\mathbf{c}_i)$  number is known.

Decision rule

$$\zeta(\mathbf{c}) = \operatorname{argmin}_{\zeta(\mathbf{c}_i)} \{ \|\mathbf{c} - \mathbf{c}_i\| \mid i = 1, 2, \dots, n \} . \quad (3.36)$$

Set of points whose nearest neighbor is  $\mathbf{c}_i$  is called the Voronoi cell of  $\mathbf{c}_i$ .

→ partition of the feature space: the Voronoi partition

$$p_B \leq p_{NN} \leq 2p_B , \quad (3.37)$$

$k$ -nearest neighbor decision rule:

decide for that class, which is the class of the majority of the  $k$  nearest neighbors.

Probability of misclassifications for the  $k$ -nearest neighbor classifier converges against the Bayesian error probability with increasing training data.

### 3.3.9 Learning and Testing

- Have two sets: training and test set (need to be disjoint!)
- leave one out
- n-fold cross validation

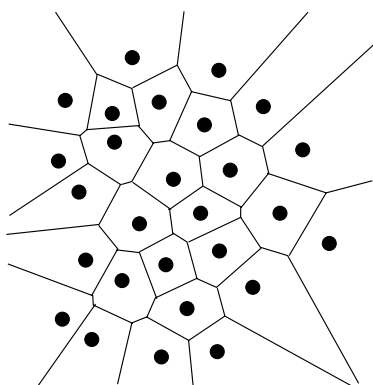


Abbildung 3.8: Voronoi diagram with 27 two-dimensional reference vectors

## 3.4 Color Normalization

Problemstellung

Farbbildverarbeitung:

Farbveränderungen durch Beleuchtungsänderung / Position

Manche Verfahren sind wenig robust gegenüber Farbänderungen

Ziel 1: Invarianz der Ergebnisse durch Normalisierung

Ziel 2: Normalisierung ohne Wissen über den Problembereich (datengetrieben)

Idee: "gray-world assumption": the world is gray (in average). If the reality (the image) is not gray, we make it gray. In  $RGB$ , this means we produce a color distribution which is clustered around the main diagonal of the  $RGB$  cube.

Algorithmus: Transformation in einen anderen Farbraum, Drehung, Rücktransformation in den  $RGB$

Gegeben:

$$\tilde{\mathbf{f}} = (RG, BY, WB)^T \in \mathbb{R}^3 \text{ (noch ein Farbraum)}$$

$$\mathbf{A}: (3 \times 3)\text{-Transformationsmatrix } RGB - RG, BY, WB (\tilde{\mathbf{f}} = \mathbf{A}\mathbf{f})$$

Gesucht:

Hauptachse von Farbvektoren  $\tilde{\mathbf{f}}$

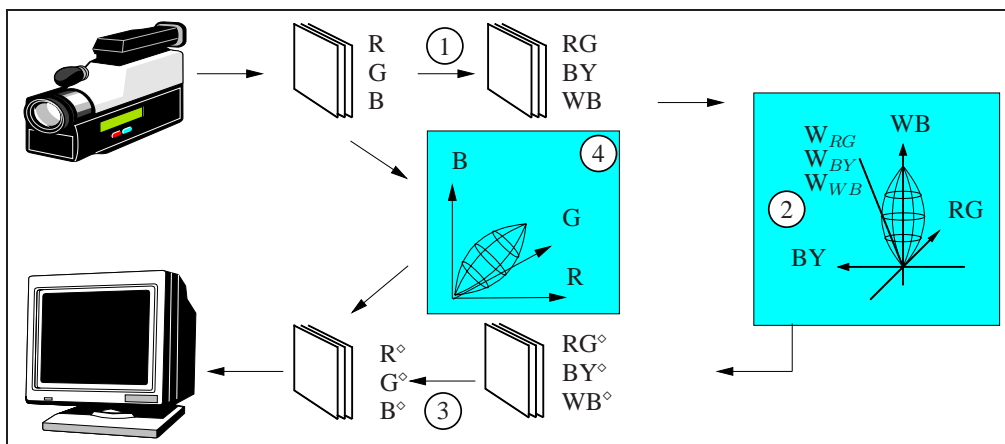
Lösungsprinzip:

Eigenvektor zum größten Eigenwert von

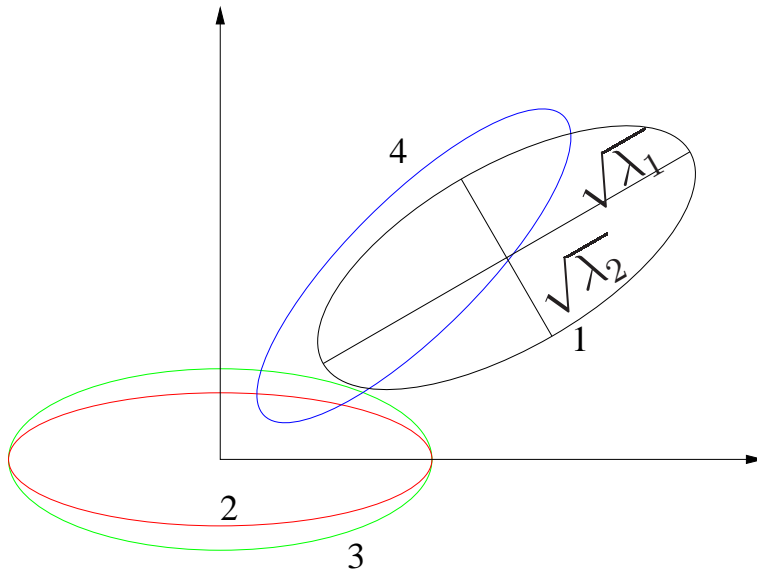
$$\tilde{\mathbf{C}} = E\{\tilde{\mathbf{f}}\tilde{\mathbf{f}}^T\}$$

Lösungsvorschlag:

Neuronales Netz [PG95] nach [OP84]



Geometrische Interpretation

Im  $\mathbb{R}^2$ :Im  $\mathbb{R}^3$ :  $[0, \dots, 255]^3 \subset \mathbb{R}^3$ : $\lambda_i, i \in 1, 2, 3$ : Eigenwerte

Skalierungsmatrix

$$\Lambda = \begin{pmatrix} \frac{255}{\sqrt{\lambda_1}} & 0 & 0 \\ 0 & \frac{255}{\sqrt{\lambda_2}} & 0 \\ 0 & 0 & \frac{255}{\sqrt{\lambda_3}} \end{pmatrix}$$

macht Keule zur Kugel

- 1  $\rightarrow$  2 : Hauptachsentransformation  
 2  $\rightarrow$  3 : Skalierung und Normierung : Varianz Wie wird die PCA von Farbknoten  $\mathbf{f}$  be-  
 3  $\rightarrow$  4 : Normierung : Lage  
 rechnet?

1. Berechne Mittelwertvektor  $\mathbf{m} = E\{\mathbf{f}\}$  wobei  $E\{\{\cdot\}\}$  den Erwartungswert bezeichnet
2. Berechne Kovarianzmatrix  $\mathbf{C}$  der Vektoren  $\mathbf{C} = E\{(\mathbf{f} - \mathbf{m})(\mathbf{f} - \mathbf{m})^T\}$
3. Eigenwertanalyse von  $\mathbf{C}$
4. Siehe lineare Algebra

Wie rotieren wir um  $\mathbf{n}$  um einen Winkel  $\phi$ :  $\mathbf{R}_3(\phi, \mathbf{n}) = \mathbf{I} - \sin \phi \mathbf{U}(\mathbf{n})$ 

Formel von Rodrigues

Neuer Lösungsweg

1. Keine Farbraumtransformation:

$$\mathbf{m} = E\{\mathbf{f}\} \quad \text{und} \quad \mathbf{C} = E\{(\mathbf{f} - \mathbf{m})(\mathbf{f} - \mathbf{m})^T\}$$

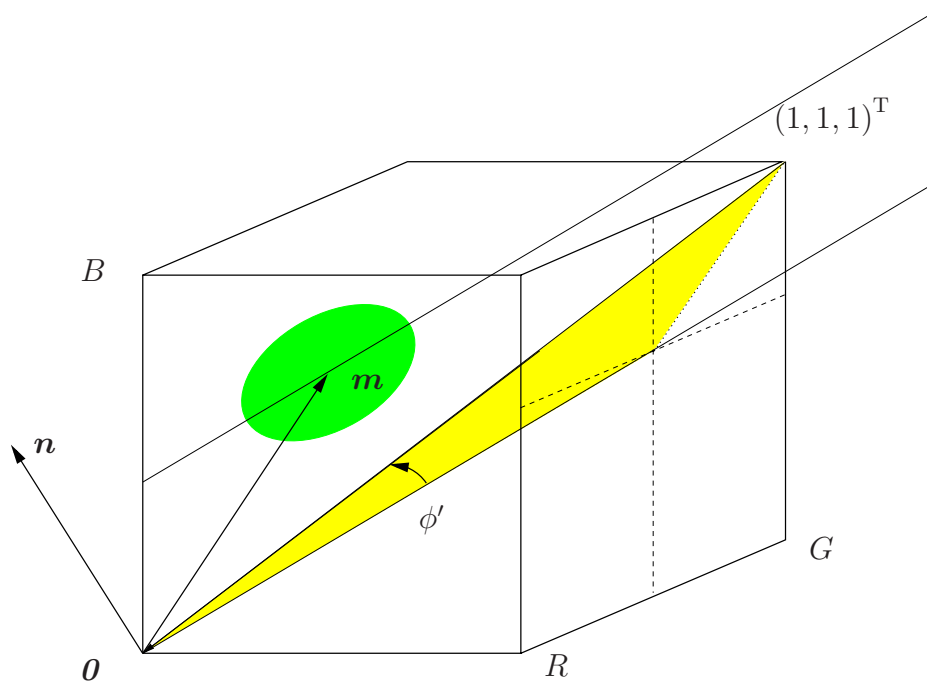


Abbildung 3.9: Farbrotaion

2.  $C$  ist  $(3 \times 3)$ -Matrix  $\rightarrow$  Eigenwerte (sortiert)

Eigenvektor[1]  $(a, b, c)^T \rightarrow \mathbf{n}'$

$$\mathbf{n}' = (a, b, c)^T \times \frac{1}{\sqrt{3}}(1, 1, 1)^T$$

3.  $\cos \phi' = (a, b, c)^T \cdot \frac{1}{\sqrt{3}}(1, 1, 1)^T$

4.  $\mathbf{R}_3(\phi, \mathbf{n}) = \mathbf{I} - \sin \phi \mathbf{U}(\mathbf{n}) + (1 - \cos \phi) \mathbf{U}^2(\mathbf{n})$

$$\mathbf{U}^2(\mathbf{n}) = \mathbf{n}\mathbf{n}^T - \mathbf{I}$$

$$\mathbf{U}(\mathbf{n}') = \frac{\sqrt{3}}{3} \begin{pmatrix} 0 & b-a & c-a \\ a-b & 0 & c-b \\ a-c & b-c & 0 \end{pmatrix}$$

5. •  $\mathbf{m} \rightarrow O$  (Verschiebung in den Ursprung)

•  $\mathbf{R}_3(\phi', \mathbf{n}')$  (Rotation)

•  $O \rightarrow \frac{\|\mathbf{m}\|}{\cos \phi'}(1, 1, 1)^T$  (Rückverschiebung)

**Comprehensive Color Normalization (CCN)**

Eliminiert Änderungen durch Beleuchtungsfarbe und Position der Lichtquelle

Algorithmus ist iterativ

Schritt 1 (lokal) eliminiert Positionsänderung der Beleuchtung

Schritt 2 (global) eliminiert Farbänderung der Beleuchtung

Annahme: keine Beeinflussung durch andere Objekte

Definitionene:

$CompNorm(\mathbf{f})$  zum Zeitpunkt  $t, t + 2, t + 4, \dots$

Farbbild(t)  $\mathbf{f}^{(t)} = [\mathbf{f}_{ij}^{(t)}]_{i=1\dots N, j=1\dots M}$

Farbvektor(t)  $\mathbf{f}_{ij}^{(t)} = \left( r_{ij}^{(t)}, g_{ij}^{(t)}, b_{ij}^{(t)} \right)^T$

Iteratives Verfahren:

1. (local)

$$S_{ij} := r_{ij}^{(t)} + g_{ij}^{(t)} + b_{ij}^{(t)}$$

$$r_{ij}^{(t+1)} = r_{ij}^{(t)} / S_{ij}$$

$$g_{ij}^{(t+1)} = g_{ij}^{(t)} / S_{ij}$$

$$b_{ij}^{(t+1)} = b_{ij}^{(t)} / S_{ij}$$

2. (global)

$$\hat{R} = \frac{3}{N * M} * \sum_1^N \sum_1^M r_{ij}^{(t+1)}$$

$$r_{ij}^{(t+2)} = r_{ij}^{(t+1)} / \hat{R}$$

$$\hat{G} = \frac{3}{N * M} * \sum_1^N \sum_1^M g_{ij}^{(t+1)}$$

$$g_{ij}^{(t+2)} = g_{ij}^{(t+1)} / \hat{G}$$

$$\hat{B} = \frac{3}{N * M} * \sum_1^N \sum_1^M b_{ij}^{(t+1)}$$

$$b_{ij}^{(t+2)} = b_{ij}^{(t+1)} / \hat{B}$$

3. (stop)

$$\text{wenn } \sum_1^N \sum_1^M \left( (r_{ij}^{(t+2)} - r_{ij}^{(t)})^2 + (g_{ij}^{(t+2)} - g_{ij}^{(t)})^2 + (b_{ij}^{(t+2)} - b_{ij}^{(t)})^2 \right) < \epsilon$$

**Eigenschaften**

- Konvergenz bewiesen [FSC98]
- Eindeutigkeit:  $(CompNorm(\mathbf{f}_1) = CompNorm(\mathbf{f}_2)) \rightarrow (\mathbf{f}_1 \sim \mathbf{f}_2)$

## 3.5 Color Calibration

### 3.5.1 Color Constancy

Sensormodell nach [JKS95, S. 284 ff.]:

Beleuchtung habe eine spektrale Energieverteilung  $E(\lambda)$ .

Voraussetzung: jeder Punkt  $\mathbf{x}$  des Bilds entspricht eindeutig einem Punkt der Szene.

Anteil des im Punkt  $\mathbf{x}$  von einer Oberfläche reflektierten Lichts sei  $S(\mathbf{x}, \lambda)$ .

In jedem Bildpunkt eintreffendes Licht

$$S(\mathbf{x}, \lambda) \cdot E(\lambda) \quad . \quad (3.38)$$

Das Spektrum des Tageslichts ändert sich in Abhängigkeit beispielsweise von der Tageszeit, der Bewölkung, der Jahreszeit und der geographischen Breite. Der Mensch hat die Fähigkeit, die Farbe eines Objekts weitgehend unabhängig von dem sich ständig ändernden Umgebungslicht als dieselbe zu empfinden. Dieses Phänomen wird Farbkonstanz genannt.

Um eine physikalische Beschreibung dieser Phänomene zu erhalten, wird nach [JKS95, S. 284 ff.] ein Sensormodell eingeführt

$K$  Sensoren pro Bildpunkt, (beispielsweise je ein Sensor für Rot, Grün und Blau), haben eine spektrale Empfindlichkeit von  $R_k(\lambda)$ , ( $k = 1, \dots, K$ ).

Dadurch tastet jeder Sensor  $k$  im Punkt  $\mathbf{x}$  folgende Energieverteilung des Lichts ab:

$$\rho_k(\mathbf{x}) = \int_0^\infty R_k(\lambda) \cdot S(\mathbf{x}, \lambda) \cdot E(\lambda) d\lambda \quad (3.39)$$

Die  $K$  Sensoren pro Bildpunkt, bei einem Farbbild typischerweise je ein Sensor für Rot, Grün und Blau, haben eine spektrale Empfindlichkeit von  $R_k(\lambda)$ , ( $k = 1, \dots, K$ ). Dadurch tastet jeder Sensor  $k$  im Punkt  $\mathbf{x}$  folgende Energieverteilung des Lichts ab:

Farbkonstanz bedeutet nun, aus den  $K$  Abtastwerten  $\rho_1(\mathbf{x}) \dots \rho_K(\mathbf{x})$  der Sensoren auf die Oberflächenreflexion  $S(\mathbf{x}, \lambda)$  zu schließen, unabhängig von der Spektralverteilung der Beleuchtung  $E(\lambda)$ .

Die Oberflächenreflektion kann nun durch eine gewichtete Summe von Basisfunktionen approximiert werden

$$S(\mathbf{x}, \lambda) = \sum_{j=1}^m \sigma_j(\mathbf{x}) L_j(\lambda) \quad , \quad (3.40)$$

wobei  $L_j(\lambda)$  die Basisfunktionen und  $\sigma_j(\mathbf{x})$  die ortsabhängige Reflektion für einen der  $m$  Komponenten ist. Häufig wird  $m = 3$  gewählt. Die Kombination von (3.39) mit (3.40) ergibt für  $\boldsymbol{\rho}(\mathbf{x}) = (\rho_1, \dots, \rho_K)^\top$  und  $\boldsymbol{\sigma}(\mathbf{x}) = (\sigma, \dots, \sigma_m)^\top$  den linearen Zusammenhang

$$\boldsymbol{\rho}(\mathbf{x}) = \mathbf{A} \boldsymbol{\sigma}(\mathbf{x}) \quad , \quad (3.41)$$

wobei die Matrix  $\mathbf{A} = [A_{ij}]_{1 \leq i \leq K, 1 \leq j \leq m}$  wie folgt definiert ist:

$$A_{ij} = \int_0^\infty E(\lambda) L_i(\lambda) R_j(\lambda) d\lambda \quad . \quad (3.42)$$

In vielen Fällen wird  $m = K = 3$  gewählt.

Für zwei verschiedene Beleuchtungen  $E_1$  und  $E_2$  einer ebenen Oberfläche ergeben sich nun zwei Matrizen  $A_1$  und  $A_2$ . Falls  $A_2$  invertierbar ist, so ergibt sich für die entsprechenden Bilder unter diesen beiden Beleuchtungen

$$\rho_1(\mathbf{x}) = A_1 A_2^{-1} \rho_2(\mathbf{x}). \quad (3.43)$$

→ Argument für (die erfolgreiche) Farbrotaion!

Dabei stellt  $E_k(\mathbf{x})$  das Umgebungslicht und  $S_k(\mathbf{x})$  den Reflexionskoeffizienten jeweils im Empfindlichkeitsbereich des Sensors  $k$  dar.

Das Verhältnis zweier benachbarter Farbpixel ist unter diesen Voraussetzungen also unabhängig von der Beleuchtung und wird in (??) zu einem konstanten Faktor. Dabei stellt  $E_k(\mathbf{x})$  das Umgebungslicht und  $S_k(\mathbf{x})$  den Reflexionskoeffizienten jeweils im Empfindlichkeitsbereich des Sensors  $k$  dar.

Das Verhältnis zweier benachbarter Farbpixel ist unter diesen Voraussetzungen also unabhängig von der Beleuchtung und wird in (??) zu einem konstanten Faktor. Bei benachbarten Bildpunkten  $\mathbf{x}$  und  $\mathbf{y}$  entspricht diese Differenz der diskreten Vorwärtsableitung der Logarithmen der Farbwerte, die komponentenweise berechnet wird.

Gegeben: Farbbild $\mathbf{f} = \left[ \mathbf{f}_{ij} = (r_{ij}, g_{ij}, b_{ij})^T \right]_{i=1 \dots M, j=1 \dots N}$
1) Logarithmieren $\mathbf{h}_{ij} := (\ln r_{ij}, \ln g_{ij}, \ln b_{ij})^T$
2) Differentiation a) $\mathbf{h}'_{ij} := \nabla^2 \mathbf{h}_{ij}$ b) $\mathbf{h}'_{ij} := (\nabla^2 \mathbf{G}) * \mathbf{h}_{ij}$ c) $\mathbf{h}'_{m,ij} := \nabla_m \mathbf{h}_{ij}, \quad m = 1 \dots 4$
3) Histogrammdarstellung a,b) $T_{\zeta(l_1, l_2, l_3)} := \sum_{i,j} z, \quad z = \begin{cases} 1, & \text{falls } \mathbf{h}'_{ij} = (l_1, l_2, l_3)^T \\ 0, & \text{sonst} \end{cases}$ c) $T_{\zeta(l_1, l_2, l_3)} := \sum_{m=1}^4 \sum_{i,j} z, \quad z = \begin{cases} 1, & \text{falls } \mathbf{h}'_{m,ij} = (l_1, l_2, l_3)^T \\ 0, & \text{sonst} \end{cases}$
4) Histogramm-Rückprojektion

Abbildung 3.10: Farbkonstanter Farbhistogrammschnitt

In [FF95] wird mit unterschiedlichen Operatoren experimentiert zur Berechnung dieser Ableitung experimentiert. Der Laplace-Operator des Gauss-gefilterten Bilds ( $\nabla^2 \mathbf{G}$ ) sowie die diskrete Ableitung in vier Richtungen ( $\nabla^2$ ) kommen zum Einsatz. Dann folgt die je nach Ableitungsoperator eine unterschiedliche Histogrammerstellung. Ein Vergleich von Histogrammen erfolgt über ein Verhältnishistogramm (??) Die wichtigsten Schritte dieses Algorithmus sind in Abbildung 3.10 zusammengefasst, wobei die Funktion (??) verwendet wird.

$m$  indiziert die Richtung

Nr	Name	Rot	Grün	Blau	Nr	Name	Rot	Grün	Blau
1	dark skin	94	28	13	13	blue	0	0	142
2	light skin	241	149	108	14	green	64	173	38
3	blue sky	97	119	171	15	red	203	0	0
4	foliage	90	103	39	16	yellow	255	217	0
5	blue flower	164	131	196	17	magenta	207	3	124
6	bluish green	140	253	153	18	cyane	0	148	189
7	orange	255	116	21	19	white	255	255	255
8	purplish blue	7	47	122	20	light gray	249	249	249
9	moderate red	222	29	42	21	light-medium gray	180	180	180
10	purple	69	0	68	22	medium gray	117	117	117
11	yellow green	187	255	19	23	dark gray	53	53	53
12	orange yellow	255	142	0	24	black	0	0	0

Tabelle 3.1: Namen und RGB-Farbwerte der Farbfelder des ColorCheckers

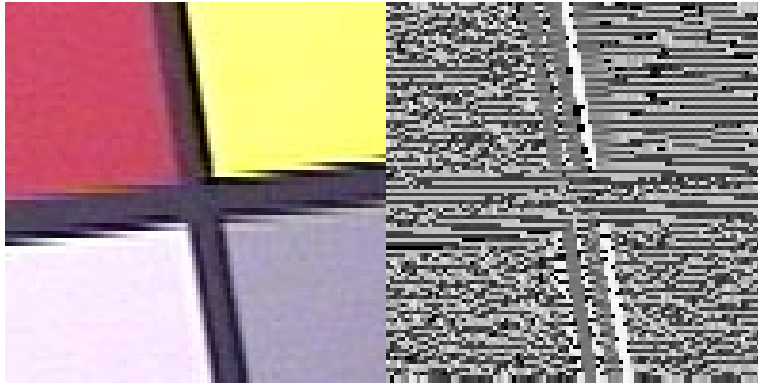


Abbildung 3.11: Ausgefranzte Farbränder und verfälschte Kantenrichtungen

Finde McBeth Color Checker mit Farbverhältnissen:

1. Regionensegmentierung
2. RAG, Knoten enthalten Farbverhältnis
3. Suche Subgraph

### 3.5.2 Color Checker

Hier wird dargestellt, wie sich der McBeth Color Checker mittels Houghtransformation im Bild finden lässt.

Die folgenden Bilder entstammen der Studienarbeit von Matthias Grobe, Erlangen 2000

### 3.5.3 Color Calibration

Zurück zu (3.39):



Abbildung 3.12: Parameterraum einer Houghtransformation (invertiert)

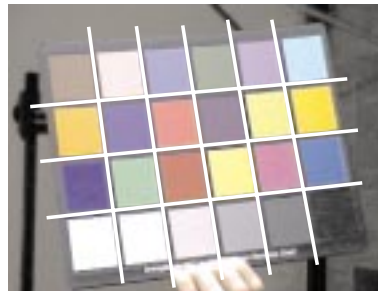


Abbildung 3.13: Beispiel für das “5 Stege auf 3 Stegen”-Kriterium

Wie in [?, ?] festgestellt wird, lässt sich eine andere diskrete version von (3.39) oft wie folgt ausdrücken

$$\rho_k(\mathbf{x}) = \sum_{\lambda=1}^{31} E_{\lambda} \cdot S_{\lambda}(\mathbf{x}) \cdot R_{k,\lambda} \cdot \Delta\lambda \quad , \quad (3.47)$$

was als Matrix-Multiplikation geschrieben werden kann als

$$\boldsymbol{\rho} = \mathbf{C} \mathbf{r}.$$

Der Vektor  $\mathbf{E} = [E_{\lambda}]_{\lambda=1\dots 31}$  bezeichnet die spektrale Energie des Lichts,  $\mathbf{S}(\mathbf{x}) = [S_{\lambda}(\mathbf{x})]_{\lambda=1\dots 31}$  denotes reflectance at position  $\mathbf{x}$ , and the matrix  $\mathbf{R} = (R_{k,\lambda})_{k=1,\dots,K,\lambda=1\dots 31}$  denotes the spectral sensitivity curves of the sensors. Using  $\Delta\lambda = 10\text{nm}$ , the whole visible range of light can be covered. In the following we omit the location  $\mathbf{x}$  and we also discard the scalar constant  $\Delta\lambda$  in the equations for simplicity. The independency of the position  $\mathbf{x}$  is justified by ????

We now arrange (3.47) into a matrix equation, as we re-write matrix  $\mathbf{R}$  as a vector  $\mathbf{r} = (R_{1,1} \dots R_{1,31}, R_{2,1} \dots R_{2,31}, \dots)$  and define a  $(3N \times 31K)$  - matrix  $\mathbf{C}$  consisting of sensor responses measured at  $N$  points of either zeros or products  $E_\lambda \cdot S_\lambda$  to get

$$\boldsymbol{\rho} = \mathbf{C}\mathbf{r} \quad . \quad (3.48)$$

Assuming standard illumination, i.e.,  $E_\lambda$  is known, and knowledge on the reflectivity and surface of the object,

Empirisches Ergebnis:  $\mathbf{C}$  hat Rang 6 bis 8;

1. Sensor Positivität:

$$R_{k,\lambda} > 0 \text{ where } k \in \{1, \dots, K\}, \lambda \in \{1, \dots, 31\}$$

2. Sensor-Glattheit:

$$|R_{k,\lambda} - R_{k,\lambda+1}| < T \text{ for some threshold } T \text{ and } \lambda \in \{1, \dots, 30\}$$

3. Unimodalität:

$$\forall k \exists \kappa : R_{k,\lambda} < R_{k,\lambda+1} \text{ für } \lambda \leq \kappa \text{ und } R_{k,\lambda} > R_{k,\lambda+1} \text{ für } \lambda > \kappa$$

4. beschränkter Vorhersagefehler:

$$|\rho_i - \sum_{j=1}^{31} C_{i,j} \cdot r_j| \leq \epsilon \text{ for } i \in \{1, \dots, N\}.$$

The rank constraint is solved by **SVD**<sup>7</sup>.

---

<sup>7</sup>h: SVD

## **3.6 Übungen**

### **3.6.1 Homework / Project**

GOAL: implement a simple classifier for object recognition

TOOLS: any (may use **matlab!**)

PROCEDURE: contest / competition

SCHEDULE: showdown on Friday, Aug. 2<sup>nd</sup> in the morning

REGULATIONS: loser as well as winner present their approach using 2 slides

Today:

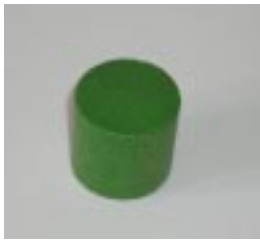
- capture images (/lab/as/images/ipcv)
- write one program to
  - read a color image
  - compute a color histogram
  - save the color histogram
- write another program to
  - read a histogram
- iterate over all the images in the test set (as a loop in matlab)

**Final Competition**

1. Your program has to
  - (a) Classify all files in the directory `/lab/as/images/ipcv/contest`
  - (b) Create a file `$HOME/result.dat` with the following format:  
filename : class number  
filename : class number  
...
2. Prepare two slides (pdf) to explain how your program does it, which parameters you chose, which strategy, etc.

**How to capture the images**

- arrange objects
- capture
- transfer to Linux computer (via network or diskette)
- manually select and crop area (using `xv` or `gimp` or `display`)
- store to `/lab/as/images/ipcv/NUMBER/NUMBER.jpg`



1



2



3



4



5



6



7



8

```

function H=manipulate(FILENAME);
%loads the image, does some operations on it. save the results and load them
again.
%*** DEMO1
%read an color image. Get a m by n by 3 Matrix; where m and n are the dimension
of the image.
%try without figure
%show the image
%---> NOW YOU CAN LOOK AT THE IMAGE.
img=imread(FILENAME);, figure; , imshow(img);
%perform histogram equalization, using MATLAB image toolbox.
I2=histeq(img);
%save the equalized image
imwrite(I2,'output.png');
I3=imread('output.png');
figure
imshow(I3);

%*** DEMO2
%---> NOW YOU CAN COMPUTE WITH THE IMAGE.
dimg=double(imread(FILENAME));

%check the dimension and the type of the image.
whos

img=double(imread(FILENAME));

```

```

R=0,G=0,B=0;
[n,m,d]=size(img);
for i=1:n
    for j=1:m
        R=R+img(i,j,1); G=G+img(i,j,2); B=B+img(i,j,3);
    end
end

R , G , B

M=[R G B];
save 'output2.dat' -ascii M
MEAN=load('output2.dat')

```

### 3.6.2 Lab-Session

- Implement color distance measures, such as the histogram intersection method, in matlab and do experiments using our data base
- Using matlab, find a matrix  $A$  and a vector  $b$  for which the result computed by a pseudo inverse  $A^+ = (A^T A)^{-1} A^T$  differs from the result computed by SVD for an over-determined equation  $b = Ax$ .
- Implement comprehensive color normalization in matlab.
- The code below for color rotation is written in Maple, another symbolic tool for mathematics. Do the same thing in matlab for some fixed values instead of symbolic names which you compute for a real image.

```

#####
# input: eigenvector of cluster (a,b,c);
#       : matrix A (3x3): color transform
#####

with(linalg); readlib(C);      # load required packages
ev0 := vector([a,b,c]);      # define vector from input variables
Idmat := matrix ([[1,0,0],[0,1,0],[0,0,1]]); # identity mat
A := Idmat;                  # default: transformation matrix is Id
sqrt3 := sqrt(3);
wbaxis := 1/sqrt3 * vector([1,1,1]);
nvec := crossprod(ev0,wbaxis); # rotation axis
cosphi := innerprod(ev0,wbaxis); #
sinphi := norm(nvec,frobenius); # rotation angle
nvec := scalarmul(nvec,1/sinphi); # normalize
#####
# Rodrigues formula
#####
nz := nvec[3]; ny := nvec[2]; nx := nvec[1];
U := matrix ([[ 0, -nz , ny], [ nz , 0 , -nx], [ -ny , nx , 0]]);
U2 := multiply (U,U);

```

```
RphiN1 := Idmat + scalarmul(U,sinphi) + scalarmul(U2,1-cosphi);  
  
n := 1/norm(A,frobenius); A:= scalarmul(A,n);  
A_T := transpose(A); A_i := inverse(A);  
Rphi := multiply(A_i,multiply(RphiN1,A));
```

Please return to us:

1. listing of matlab code, well commented
2. explanations, if required

# SVD

Kurze Einführung der SVD

## Singular Value Decomposition (SVD)

- Methode aus der linearen Algebra,
- Hier: Lösung von überbestimmten Gleichungssystemen,
- Numerische Konditionierung viel besser als z. B. Gauss-Eliminierung,
- Einfach: Erzwingen des Rangs einer Matrix.

Jede  $m \times n$  Matrix  $A$  kann zerlegt werden in ein Produkt

$$A = U \Sigma V^T,$$

wobei

- $U \in \mathbb{R}^{m \times m}$ ,  $V \in \mathbb{R}^{n \times n}$  quadratisch und orthogonal,
- $\Sigma \in \mathbb{R}^{n \times n}$  Diagonalmatrix.

Wobei  $\Sigma$ :

$$\Sigma = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \sigma_{\min(m,n)} \end{pmatrix} \quad \begin{array}{l} \sigma_i: \text{ singular values} \\ \sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)} \geq 0 \end{array}$$

- Rang von  $A$ :  $\#(\sigma_i \neq 0)$
- Nullspace of  $A$   
Aufgespannt durch die Spaltenvektoren  $v_i$  of  $V$ , mit  $\sigma_i = 0$ .

- Bild von  $A$   
Aufgespannt von den Spaltenvektoren  $\mathbf{v}_i$  of  $\mathbf{V}$ , mit  $\sigma_i \neq 0$ .
- Orthogonalisierung von  $A$ 
  - Sei  $A = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T$  numerisch orthogonal,
  - Setze alle  $\sigma_i = 1$ ,
  - Berechne  $A' = \mathbf{U} \mathbf{\Sigma}' \mathbf{V}^T$ ,
  - Dann ist  $A'$  exakt diagonal.

Der folgende Text ist gegenwärtig nur in Englisch verfügbar

### SVD of a Matrix

every  $M$  (rows)  $\times$   $N$  (columns) matrix  $\mathbf{M}^{M \times N}$  can be decomposed as

$$\mathbf{M}^{M \times N} = \mathbf{U}^{M \times N} \mathbf{D}^{N \times N} \mathbf{V}^{T N \times N} = \sum_{i=1}^r \sigma_i \mathbf{u}_i \mathbf{v}_i^T$$

- the columns of the  $M \times N$  matrix  $\mathbf{U}$  are the *eigenvectors*  $\mathbf{u}_i$  of  $\mathbf{M} \mathbf{M}^T$  (hence,  $\mathbf{U}^T \mathbf{U} = \mathbf{I}$ );  
i.e. the column vector  $\mathbf{u}_i$  is obtained from

$$\lambda_i \mathbf{u}_i = (\mathbf{M} \mathbf{M}^T) \mathbf{u}_i$$

and  $\lambda_i$  is the  $i$ -th *eigenvalue* of  $\mathbf{M} \mathbf{M}^T$

- matrix  $\mathbf{D}$  is diagonal with elements  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r > \sigma_{r+1} = \dots = \sigma_N = 0$ ,  $r \leq N$ ;  
the elements  $\sigma_i$  of  $\mathbf{D}$  are the *singular values*;  
they are the square-roots of the *eigenvalues* of  $\mathbf{M} \mathbf{M}^T$ , i.e.  $\sigma_i = \sqrt{\lambda_i}$ ;
- the columns of the  $N \times N$  matrix  $\mathbf{V}$  are the *eigenvectors*  $\mathbf{v}_i$  of  $\mathbf{M}^T \mathbf{M}$  (hence,  $\mathbf{V}^T \mathbf{V} = \mathbf{I}$ );
- if  $M < N$ ,  $\sigma_i = 0$ ,  $i = M + 1, \dots, N$ ; corresponding columns of  $\mathbf{U}$  are zero;

### Some Properties of SVD

- the *condition number*  $c = \frac{\sigma_1}{\sigma_N}$  measures the “degree of singularity” of  $\mathbf{M}$ ;  
if  $1/c$  is comparable to the arithmetic precision of the computer,  $\mathbf{M}$  is *ill-conditioned* and “singular for practical purposes”;
- columns of  $\mathbf{U}$  corresponding to non-zero singular values span the range of  $\mathbf{M}$ ;  
columns of  $\mathbf{V}$  corresponding to zero singular values span the nullspace of  $\mathbf{M}$ ;
- denoting the columns of  $\mathbf{U}$  and  $\mathbf{V}$  by  $\mathbf{u}_i$  and  $\mathbf{v}_i$ , respectively, we have  
 $\mathbf{M}^T \mathbf{u}_i = \sigma_i \mathbf{v}_i$  and also  $\mathbf{M} \mathbf{v}_i = \sigma_i \mathbf{u}_i$ ;

- the *pseudo-inverse*  $M^+$  of  $M$  is

$$M^+ = VD'^{-1}U^T$$

with entries of  $D'^{-1}$  equal to  $D^{-1}$  for nonzero singular values and zero otherwise

- the Frobenius–norm of a matrix is

$$\|M\|_F = \sum_{i,j} m_{i,j}^2 = \sum_i \sigma_i^2$$

### Solving Linear Equations With SVD (1)

consider the linear equation  $Ax = a$  with  $A$  a square matrix;

1. if  $A$  is *nonsingular*, we get from SVD

$$x = A^{-1}a = VD^{-1}U^T a \quad \text{with } D^{-1} = \text{diag}(1/\sigma_i)$$

2. if  $A$  is *singular and*  $a$  is in the range of  $A$ , we get from SVD the solution of minimal norm

$$x = VD'^{-1}U^T a \quad (*)$$

with  $D'^{-1} = \text{diag}(d'_i)$ , with  $d'_i = 1/\sigma_i$ , if  $\sigma_i \neq 0$ , and  $d'_i = 0$ , if  $\sigma_i = 0$ ;

3. if  $A$  is *singular and*  $a$  is *not* in the range of  $A$ , we get from (\*) the solution with minimal mean square error  $\varepsilon = |Ax - a|$

4. if  $A$  is *ill-conditioned*, then it is often advisable to obtain a solution from

$$x = VD''^{-1}U^T a \quad (**)$$

with  $D''^{-1} = \text{diag}(d''_i)$ , with  $d''_i = 1/\sigma_i$ , if  $\sigma_i \geq \Delta$ , and  $d''_i = 0$ , if  $\sigma_i < \Delta$ ;

### Solving Linear Equations With SVD (2)

now we consider a linear equation with an  $M \times N$  matrix  $A$ :

- 5 if  $A$  is an  $M \times N$  matrix with  $M < N$  (underdetermined system), there is no unique solution; in the SVD of  $A$  there will be  $N - M$  zero singular values  $\sigma_i$ ; and there may be additional zero (or close to zero) singular values; one solution is obtained from (\*\*), the solution space is obtained by adding the linear combination of the nullspace of  $A$  (see above: a basis of this nullspace are the columns of  $V$  corresponding to zero or zeroed  $\sigma_i$ );
- 6 if  $A$  is an  $M \times N$  matrix with  $M > N$  (more equations than unknowns, i.e. overdetermined system), we want the solution which minimizes the mean square error; again it is obtained from (\*);

conclusion: SVD is a kind of panacea for linear equations — it cannot fail (but in ‘easy’ cases there may be faster solutions);

for numerical computation of SVD see “Numerical Recipes in C” (<http://www.nr.com>);



# Kommentiertes Literaturverzeichnis

- [BN98] H. Burkhard and B. Neumann, editors. *Computer Vision — ECCV '98*, number 1406 in Lecture Notes in Computer Science, Heidelberg, 1998. Springer. 47
- [FF95] B. V. Funt and G. D. Finlayson. Color constant color indexing. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 17(5):522–529, 1995. 35
- [FSC98] G.D. Finlayson, B. Schiele, and J.L. Crowley. Comprehensive colour image normalization. In Burkhard and Neumann [BN98], pages I/475–490. 33
- [JKS95] R. Jain, R. Kasturi, and B. G. Schunk. *Machine Vision*. McGraw-Hill, Inc., 1 edition, 1995. 34
- [Nie83] H. Niemann. *Klassifikation von Mustern*. Springer, Heidelberg, 1983. 18
- [OP84] E. Oja and J. Parkkinen. On Subspace Clustering. In *Proc. Int. Conf. on Acoustics, Speech, and Signal Processing*, pages 692–695. San Diego, 1984. 30
- [Pau01] D. Paulus. *Aktives Bildverstehen*. Der andere Verlag, Osnabrück, 2001. Habilitationsschrift in der Praktischen Informatik, Universität Erlangen-Nürnberg, Mai 2000.

Verfügbar als PDF-Datei online unter

[http://www.der-andere-verlag.de/buecher/paulus\\_xlink.html](http://www.der-andere-verlag.de/buecher/paulus_xlink.html)

5

- [PBRT98] J. Puzicha, J.M. Buhmann, Y. Rubner, and C. Tomasi. Empirical evaluation of dissimilarity measures for color and texture. In Burkhard and Neumann [BN98], pages 563–577. 8
- [PFTV88] W.H. Press, B.P. Flannery, S. Teukolsky, and W.T. Vetterling. *Numerical Recipes - the Art of Numerical Computing, C Version*. Cambridge University Press, Cambridge, 1988. 8
- [PG95] T. Pomierski and H. M. Gross. Biological neural architecture for chromatic adaption resulting in constant color sensations. Berlin, 1995. Springer. 30
- [PH99] D. Paulus and J. Hornegger. *Applied pattern recognition: A practical introduction to image and speech processing in C++*. Advanced Studies in Computer Science. Vieweg, Braunschweig, 3 edition, 1999.

The basic description of the programming environment and its principles which are applied in the projects of the LME. Contains image filtering, segmentation, and principles of image analysis. Speech filtering, spectral features for speech, speech understanding principles.

5, 19

- [RTG98] Y. Rubner, C. Tomasi, and L.J. Guibas. A metric for distributions with applications to image databases. In *Proceedings of the 6<sup>th</sup> International Conference on Computer Vision (ICCV)*, pages 59–66, Bombay, Januar 1998. IEEE Computer Society Press. 11
- [SB91] M. J. Swain and D. H. Ballard. Color indexing. *International Journal of Computer Vision*, 7(1):11–32, November 1991. 6, 7
- [Sch97] B. Schiele. *Object Recognition using Multidimensional Receptive Field Histograms (English translation)*. PhD thesis, Institut National Polytechnique de Grenoble, Grenoble Cedex, 1997. 8
- [VMH97] V.V. Vinod, H. Murase, and C. Hashizume. Focussed color intersection with efficient searching for object extraction. In *Pattern Recognition*, volume 30, pages 1787–1797, 1997. 7

# Inhaltsverzeichnis

3.1	Color Histograms	6
3.2	Klassifikation	18
3.3	Numerical Pattern Classification	19
3.3.1	General Notes on Classifiers	20
3.3.2	Design of Classifiers	21
3.3.3	Linear Discriminants	21
3.3.4	Polynomial Classifiers	24
3.3.5	Bayesian Classifiers	26
3.3.6	Properties of Bayesian Classifiers	27
3.3.7	From Bayesian to Geometric Classifiers	27
3.3.8	Nearest Neighbor Classifier	28
3.3.9	Lerning and Testing	28
3.4	Color Normalization	30
3.5	Color Calibration	34
3.5.1	Color Constancy	34
3.5.2	Color Checker	36
3.5.3	Color Calibration	36
3.6	Übungen	39
3.6.1	Homework / Project	39
3.6.2	Lab-Session	41
	<b>Kommentiertes Literaturverzeichnis</b>	<b>46</b>
3.7	Verzeichnis der Symbole und Namen	49

## 3.7 Verzeichnis der Symbole und Namen

Die folgende Aufstellung listet die Symbole und Namen in der Reihenfolge ihrer Verwendung auf..

$N_M$   
 Maskenröße vertikal 6, 16  
 $N_M$   
 Teilbildgröße vertikal 6, 16  
 $M_M$   
 Maskenröße horizontal 6, 16  
 $M_M$   
 Teilbildgröße horizontal 6, 16  
 $f$   
 Bildfunktion 6, 10, 18, 35  
 $M$   
 Bildgröße vertikal 6, 7, 11, 16, 35  
 $N$   
 Bildgröße horizontal 6, 7, 11, 16, 35  
 $S$   
 Szenenhistogramm 6, 7, 8, 11  
 $T$   
 Objekthistogramm 6, 7, 8, 11, 35  
 $l$   
 Index für Urnen 6, 7  
 $N_L$   
 Anzahl Urnen 6, 7, 11, 16  
 $\zeta$

Farbvektorabbildung auf Urne 6, 35  
 $f$   
 Farbbildfunktion 10  
 $H$   
 Histogramm 10  
 $N_o$   
 Anzahl Objekte 10  
 $O$   
 Segmentierungsobjekte 10  
 $F$   
 Fluss 11  
 $d$   
 Geometrischer Abstand 11  
 $c$   
 Merkmal 18, 19, 21, 23, 26  
 $\kappa$   
 Klassenindex 18, 19, 23, 26, 27  
 $E$   
 Spektrale Energieverteilung der  
 Lichtquelle 34, 35, 37, 38  
 $S$   
 Reflexions-Koeffizient 34, 35, 37  
 $G$   
 Gauß-Maske 35