In this course of two lectures we will discuss color histograms and their use for object recognition. This will lead us also to some related subjects, namely color normalization, color sensor calibration, and simple classification. We will introduce color histograms in various flavors, resolutions, and representations. We discuss different distance measures for color histograms and compare them. We learn how to compute histograms efficiently. Histogram intersection and histogram backprojection provide very simple and yet efficient ways to find objects in scenes. These methods can also be applied to image database queries.

Some of the computations require linear algebra. As the singular value decomposition can be used to compute many numerical solutions in linear algebra, we introduce this method and apply it so color calibration and color normalization problems.

In the exercises we implement the methods that are introduced in the lecture. These methods can be used to solve the task in the project work: recognize objects from a set of known objects in a database.

The task of the project will be solved in small groups (international members) of three students. The groups are free to choose their individual solution. On the final day of the summer school the groups will present their solution. In a contest we will determine the best solution.
## Contents

4 Color Histograms 5

5 Color Normalization 15

6 Color Calibration 21
  6.1 Color Constancy 21
  6.2 Color Checker 23
  6.3 Color Calibration 23

7 Classification 27
  7.1 General Notes on Classifiers 28
  7.2 Design of Classifiers 28
  7.3 Linear Discriminants 28
  7.4 Polynomial Classifiers 31
  7.5 Bayesian Classifiers 31
  7.6 From Bayesian to Geometric Classifiers 32
  7.7 Nearest Neighbor Classifier 34
  7.8 Testing a Classifier 35
    7.8.1 Learning and Testing 35

A Project Budapest 2006 37

B SVD 41

Annotated Bibliography 45
  B.1 Names and Symbols 47
This course material is a re-formatted version of the slides that are presented in the course. It is not a book. You will need additional notes or articles to understand the topics.
Chapter 4

Color Histograms

Object localization: common task in computer vision

Color as a cue for solving this problem presented by Swain and Ballard in [SB91]: histogram intersection and histogram backprojection

→ distributions of color information, for object localization
→ color histograms

General ideas:

• try to find that sub–image of the scene image, for which the histogram has the smallest distance from the object histogram.

Size $N_M \times M_M$ of subimage has to be determined somehow!

• color histogram backprojection

Example:
CHAPTER 4. COLOR HISTOGRAMS

- capture image \( f \) of some objects with known focal length
- add one object \( \kappa \) to scene
- capture image \( g \)
- \(|f - g|\), threshold, erode, filter to get mask \( h \)
- extract object as \( f \& h \), compute histogram \( T \)
- automatic detection of bounding box
- Sample: 20 objects in 3 illuminations

Color Histograms
Assume a close-up view of object given in the image

\[
\begin{align*}
\text{image } f &= [f_{ij}]_{i=1,...,M,j=1,...,N} \\
\text{where } f_{ij} \text{ is a color pixel, i.e. } f_{ij} &= (r_{ij}, g_{ij}, b_{ij})^T.
\end{align*}
\]

This object is to be found in the scene \( f' \).

We compute histograms \( S \) for a subimage of the scene and \( T \) for the object

\[
\begin{align*}
S &= [S_{l}]_{l=1,...,N_L} & T &= [T_{l}]_{l=1,...,N_L} \tag{4.1}
\end{align*}
\]

where the number of bins \( N_L \) depends of the chosen quantization and the color space.

Histogram computation
Function $\zeta$ maps color pixel to the index in histogram in any color space

**Example:** for a RGB histogram with $4 \times 4 \times 4$ bins ($N_L = 64$) and for color components in the range from 0 to 255

$$\zeta : \begin{cases} \mathbb{R}^3 & \rightarrow \{1, \ldots N_L\} \\ f_{ij} = (r_{ij}, g_{ij}, b_{ij})^T & \rightarrow \left[ \frac{r_{ij}}{64} \right] \ast 16 + \left[ \frac{g_{ij}}{64} \right] \ast 4 + \left[ \frac{b_{ij}}{64} \right] \end{cases} \quad (4.2)$$

Elements of the histogram $T$

$$T_l = |\{(i, j)| \zeta(f_{ij}) = l, i = 1 \ldots M_T, j = 1, \ldots N_T\}| \quad (4.3)$$

Analogously: $S, S_l$ for an image of size: $M_S \times N_S$

In order to compare the histograms we need distance measures

**Histogram Intersection** - a classical measure proposed in [SB91]. It is computationally inexpensive generalization of geometric $L_1$ Minkowski’s distance defined by:

$$\cap (S, T) = \sum_{l=1}^{N_L} \min \{S_l, T_l\} \quad (4.4)$$

Extended to focused intersection and active search in [VMH97]

**Active Search**

- sliding sums algorithm for histogram comparison (computation independent of size of window)
• for those distance measures fulfilling triangle equation: cancel comparison when current optimum cannot be met

(this can be predicted e.g. for histogram intersection, \[VMH97\])

\[\text{Sum of Squared Differences (SSD)} \text{ - defined by:}\]

\[
SSD(S, T) = \sum_{i=1}^{N_l} (T_i - S_i)^2 \tag{4.5}
\]

\[\text{Chi–square Test} \text{ - statistical method. Different versions in literature}\]

\[\text{[PBRT98, Sch97, PFTV88]}\]

\[\text{We use (like [Sch97])}\]

\[
\chi^2(S, T) = \sum_{i=1}^{N_l} \frac{(S_i - T_i)^2}{T_i} \tag{4.6}
\]

It was shown that using this definition, the best results were obtained with respect to noise, blur, and image plane rotation \[Sch97\].
This picture shows the desired properties of a histogram distance for another pair of comparisons.
This picture shows the desired properties of a histogram distance for another pair of comparisons.

1. **Minkowski-form distance** [KKIK03] The Minkowski-form distance \( L_p \) is a generalised metric distance. Depending on its order \( p \), it represents different distance functions. As the level of \( p \) decreases, the weight of large differences between single attribute values increases. The distance \( d \) between \( x \) and \( y \) is defined as

\[
L_p(x, y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}, \quad x, y \in \mathbb{R}^n
\]  

(4.7)

The computational complexity of the Minkowski-distances depends on the selection of \( p \). Specifications of the Minkowski-metric are the following distance measures:

- **absolute distance** (also referred to as City-Block or Manhattan distance)

\[
d(x, y) = L_1(x, y) = \sum_{i=1}^{n} |x_i - y_i|
\]  

(4.8)

- **Euclidean distance**

\[
d(x, y) = L_2(x, y) = |x - y| = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}
\]  

(4.9)

2. **\( \chi^2 \)-statistic** [Pap02]

\[
d(x, y) = \chi_1^2(x, y) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i}
\]  

(4.10)

\[
d(x, y) = \chi_2^2(x, y) = \sum_{i=1}^{n} \frac{(x_i - y_i)^2}{x_i + y_i}
\]  

(4.11)
3. **Kullback-Leibler divergence** \[KL51\]

\[ d(x, y) = KL(x, y) = \sum_{i=1}^{n} x_i \log \frac{x_i}{y_i} \quad (4.12) \]

4. **Jeffrey divergence** (also referred to as Jensen-Shannon divergence) \[PBRT99\]

\[ d(x, y) = JF(x, y) = \left( \sum_{i=1}^{n} x_i \log \frac{2x_i}{x_i + y_i} + y_i \log \frac{2y_i}{x_i + y_i} \right) \quad (4.13) \]

5. **Bhattacharyya distance** \[Kai67\]

\[ d(x, y) = BA(x, y) = \sqrt{1 - \rho[x, y]} \quad \text{where} \quad \rho[x, y] = \sum_{i=1}^{n} \sqrt{x_i y_i} \quad (4.14) \]

**Histogram backprojection**

**Histogram backprojection**

| Given: image histogram \( T = [T_k]_{k=1\ldots K} \) of an object, |
| Wanted: object position \((i_t, j_t)\) |
| Compute color histogram \( H = [H_k]_{k=1\ldots K} \) of given image |
| FOR Each bin \( k \in \{1, \ldots, K\} \) |
| \( \hat{R}_k = \min \left\{ \frac{T_k}{H_k}, 1 \right\} \) (compute ratio histogram \( R = [R_k]_{k=1\ldots K} \)) |
| FOR All positions \((i, j)\) in the image |
| \( A_{i,j} := \hat{R}_{h(f_{i,j})} \), where \( f_{i,j} \) denotes the color vector at position \((i, j)\) |
| \( B := D_r \ast A, \) where \( \ast \) denotes convolution |
| \((i_t, j_t) := \text{argmax}_{i,j}(B_{i,j})\) |
| \( D_r \) denotes a mask of the size of the object |

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

\[ p(O_i)p(f|O_i) = p(f)p(O_i|f) \]

in

\[ p(O_i|f) = \frac{p(f|O_i)p(O_i)}{p(f)} \]
CHAPTER 4. COLOR HISTOGRAMS

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

New Distance Measures

Computation of EMD involves the solution of the transportation problem, where a matrix $F = [F_{\mu,\nu}]_{1,...,N_L,1,...,N_L}$ represents the flow from bin $S_\mu$ to $T_\nu$

Constraints:
The flow cannot be negative ($0 \leq F_{\mu,\nu}$) and is further constrained by:

\begin{align*}
    \sum_{\mu=1}^{N_L} F_{\mu,\nu} &= T_\nu / (M_T N_T) \\
    \sum_{\nu=1}^{N_L} F_{\mu,\nu} &= S_\mu / (M_S N_S) \\
    \sum_{\mu=1}^{N_L} \sum_{\nu=1}^{N_L} F_{\mu,\nu} &= 1 
\end{align*}

(4.15)

It makes EMD the most computationally expensive among exploited measures. It can be implemented by means of linear programming methods, such as the simplex algorithm\(^1\) (requires $O(N_L^4)$ memory)

Feasible: $N_L \leq 64$

---

Left: scene, middle: histogram intersection, right: active search

Result of active search for object number 11. Only 13% of the total number of comparisons is required for active search

We tested:

- various color spaces
- various quantizations
- same scene under different illuminations

\(^1\)We used the Y. Rubner’s code for EMD found at
http://robotics.stanford.edu/~rubner/emd/default.htm
Figure 4.1: Fast computation of histograms. Top: Initialization. Left bottom: Advance by one pixel. Right bottom: Advance one line.

- SSD, $\chi^2$, HI, EMD, DP, Kullbach-Divergenz, …
- $\approx$ 20 objects and 20 Scenes
- alltogether several thousands of experiments
Chapter 5

Color Normalization

Problem definition
Color image processing:
Color changes due to changes in lighting and position
Many algorithms are very sensitive to color changes
Goal 1: invariance of image processing results with respect to lighting changes by normalization
Goal 2: normalization without user interaction or knowledge about the problem domain data driven
Outline: "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.
Algorithm: do this transformation in a linear transformation of the $RGB$ space, than return to $RGB$

- **Given:** $\tilde{f} = (RG, BY, WB)^T \in \mathbb{R}^3$ (some color space)
  $A$: $(3 \times 3)$–transformation matrix $RGB - RG, BY, WB$ ($\tilde{f} = Af$)

- **Wanted:** Principal component (PC) of color vectors $\tilde{f}$

- **Principle:** eigenvector for greatest eigenvalue of
  $\tilde{C} = E\{\tilde{f}\tilde{f}^T\}$

- **Solution for PC analysis:** ANN [PG95] as proposed in [OP84]
CHAPTER 5. COLOR NORMALIZATION

Geometric interpretation

In $\mathbb{R}^2$:

In $\mathbb{R}^3$: $[0, \cdots, 255]^3 \subset \mathbb{R}^3$:

$\lambda_i, i \in 1, 2, 3$: eigenvalues

Scaling matrix

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

converts club to bowl

1. $\to$ 2 : PCA
2. $\to$ 3 : scaling and normalization
3. $\to$ 4 : pose normalization

How to compute the PCA of color vectors $f$

1. compute mean of the vectors $m = E\{f\}$ where $E\{\cdot\}$ denotes the expectation value
2. compute covariance matrix $C$ of the vectors by $C = E\{(f - m)(f - m)^T\}$

3. do an Eigenvalue analysis of $C$

4. read the math books how to do this

How to rotate around an axis $n$ by an angle $\phi$: $R_3(\phi, n) = I - \sin \phi U(n)$

use Rodrigues formula (see math books)

New idea

1. No change in color space:

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

2. $C$ is $(3 \times 3)$–matrix $\rightarrow$ eigenvalues (sorted) eigenvector[1] $(a, b, c)^T \rightarrow n' \quad n' = (a, b, c)^T \times \frac{1}{\sqrt{3}}(1, 1, 1)^T$

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

4. $R_3(\phi, n) = I - \sin \phi U(n) + (1 - \cos \phi) U^2(n)$

   $U^2(n) = nn^T - I$

   $$U(n') = \frac{1}{\sqrt{3}} \begin{pmatrix} 0 & b - a & c - a \\ a - b & 0 & c - b \\ a - c & b - c & 0 \end{pmatrix}$$

5. $m \rightarrow O$ (shift to origin)

   $R_3(\phi', n')$ (rotate)
• $O \rightarrow \frac{||m||}{\cos \phi} (1, 1, 1)^T$ (shift back on diagonal)

This algorithm explains quite a lot of small details:

1. How to rotate by an angle around an axis in 3-D
2. How to apply the Rodrigues Formula
3. How to interpret Eigenvalues in geometric context
4. How to use dot products to express angels

All this is very important for other problems in computer vision that deal with 3-D transformations, such as all the problems related to 3-D reconstruction.

**Comprehensive Color Normalization (CCN)**

Eliminate changes which are due to changes in light position and light color/intensity

Algorithm (below) works iteratively

Step 1 (local) eliminates intensity changes

Step 2 (global) eliminates color changes due to color changes of the light

Assumes that intensity at a point $(i, j)$ only depends on the object and the light source — and not on the other objects

Definitions:

$CompNorm(f)$ at time $t$, $t + 2$, $t + 4$, ... color image at time $t$ $f(t) = [f_{ij}^{(t)}]_{i=1...N,j=1...M}$ color vector at time $(t)$ $f_{ij}^{(t)} = \left(r_{ij}^{(t)}, g_{ij}^{(t)}, b_{ij}^{(t)}\right)^T$

Iterative algorithm:

1. (local) $S_{ij} := r_{ij}^{(t)} + g_{ij}^{(t)} + b_{ij}^{(t)}$
   
   $r_{ij}^{(t+1)} = \frac{r_{ij}^{(t)}}{S_{ij}}$
   
   $g_{ij}^{(t+1)} = \frac{g_{ij}^{(t)}}{S_{ij}}$
   
   $b_{ij}^{(t+1)} = \frac{b_{ij}^{(t)}}{S_{ij}}$

2. (global) $\hat{R} = \frac{3}{N \cdot M} \sum_{1}^{N} \sum_{1}^{M} r_{ij}^{(t+1)}$
   
   $\hat{G} = \frac{3}{N \cdot M} \sum_{1}^{N} \sum_{1}^{M} g_{ij}^{(t+1)}$
   
   $\hat{B} = \frac{3}{N \cdot M} \sum_{1}^{N} \sum_{1}^{M} b_{ij}^{(t+1)}$

3. (stop) when $\sum_{1}^{N} \sum_{1}^{M} \left( (r_{ij}^{(t+2)} - r_{ij}^{(t)})^2 + (g_{ij}^{(t+2)} - g_{ij}^{(t)})^2 + (b_{ij}^{(t+2)} - b_{ij}^{(t)})^2 \right) < \epsilon$
Properties

- Convergence proved in [FSC98]
- Uniqueness: \((\text{CompNorm}(f_1) = \text{CompNorm}(f_2)) \rightarrow (f_1 \sim f_2)\)
Chapter 6

Color Calibration

6.1 Color Constancy

Sensor model according to [JKS95, S. 284 ff.]:

Let the illumination have a spectral power distribution \( E(\lambda) \).

Prerequisite: Any point \( x \) in the image corresponds uniquely to a point in the scene.

Let the fraction of the light reflected in point \( x \) by a surface be \( S(x, \lambda) \).

Incident light in each image point

\[
S(x, \lambda) \cdot E(\lambda) .
\] (6.1)

K sensors per pixel, (e.g. one for red, green, and blue), have spectral sensitivity \( R_k(\lambda), (k = 1, \ldots, K) \).

Each sensor \( k \) samples at point \( x \) the following energy distribution:

\[
\rho_k(x) = \int_0^\infty R_k(\lambda) \cdot S(x, \lambda) \cdot E(\lambda) \, d\lambda
\] (6.2)

Surface reflection

\[
S(x, \lambda) = \sum_{j=1}^m \sigma_j(x) L_j(\lambda) ,
\] (6.3)

where \( L_j(\lambda) \) are basis functions and \( \sigma_j(x) \) the spatially varying reflection for one of the \( m \) components. Often \( m = 3 \).

Combining (6.2) with (6.3) yields for \( \rho(x) = (\rho_1, \ldots, \rho_K)^T \) and \( \sigma(x) = (\sigma, \ldots, \rho_m)^T \) the linear relation

\[
\rho(x) = A \sigma(x) ,
\] (6.4)

where the matrix \( A = [A_{ij}]_{1 \leq i \leq K, 1 \leq j \leq m} \) is defined as:

\[
A_{ij} = \int_0^\infty E(\lambda) L_i(\lambda) R_j(\lambda) \, d\lambda .
\] (6.5)
In many cases we choose \( m = K = 3 \).

For two different illuminations \( E_1 \) and \( E_2 \) of a planar surface we obtain two matrices \( A_1 \) and \( A_2 \). If \( A_2 \) is invertible, we obtain for the two corresponding images

\[
\rho_1(x) = A_1 A_2^{-1} \rho_2(x).
\] (6.6)

Argument for color rotation

With simplifying assumptions, such as highly band-limited sensitivity of sensors, we conclude from (6.2) for the sensor response \( k \) at location \( x \)

\[
\rho_k(x) = E_k(x) \cdot S_k(x).
\] (6.7)

for two adjacent pixels \( x \) and \( y \): (illumination assumed to be constant locally) i.e., \( E_k(x) = E_k(y) \):

\[
\frac{\rho_k(x)}{\rho_k(y)} = \frac{S_k(x) E_k(x)}{S_k(y) E_k(y)} = \frac{S_k(x)}{S_k(y)}
\] (6.8)

taking the logarithm of (6.8) yields

\[
\ln \rho_k(x) - \ln \rho_k(y) = \ln S_k(x) - \ln S_k(y).
\] (6.9)

→ independent of illumination!

where \( E_k(x) \) is the ambient light, \( S_k(x) \) is the reflectivity coefficient for sensor \( k \).

<table>
<thead>
<tr>
<th>Given: Color image ( f = [f_{ij} = (r_{ij}, g_{ij}, b_{ij})^T]_{i=1...M, j=1...N} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1) Take logarithm ( h_{ij} := (\ln r_{ij}, \ln g_{ij}, \ln b_{ij})^T )</td>
</tr>
<tr>
<td>2) Compute derivatives</td>
</tr>
<tr>
<td>a) ( h'<em>{ij} := \nabla^2 h</em>{ij} )</td>
</tr>
<tr>
<td>b) ( h'<em>{ij} := (\nabla^2 G) * h</em>{ij} )</td>
</tr>
<tr>
<td>c) ( h'<em>{m,ij} := \nabla_m h</em>{ij}, \ m = 1...4 )</td>
</tr>
<tr>
<td>3) Describe by histogram</td>
</tr>
<tr>
<td>a,b) ( T_{(l_1, l_2, l_3)} := \sum_{i,j} z, \ z = \begin{cases} 1, \text{ if } h'_{ij} = (l_1, l_2, l_3)^T \ 0, \text{ otherwise} \end{cases} )</td>
</tr>
<tr>
<td>c) ( T_{(l_1, l_2, l_3)} := \sum_{m=1}^4 \sum_{i,j} z, \ z = \begin{cases} 1, \text{ if } h'_{m,ij} = (l_1, l_2, l_3)^T \ 0, \text{ otherwise} \end{cases} )</td>
</tr>
<tr>
<td>4) Histogram–Backprojection</td>
</tr>
</tbody>
</table>

Figure 6.1: Color constant color indexing
6.2. COLOR CHECKER

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

Table 6.1: Names and RGB-values for areas on the color checker

[FF95] reports on experiments of different operators for computation of the derivatives. The laplacian of the Gaussian-filtered image \( (\nabla^2 G) \) as well as the discrete derivative of all for directions \( (\nabla^2) \) are used. Comparison uses the ratio histogram \( ?? \). The major steps of the algorithm are shown in Figure 6.1, where the function \( ?? \) is used.

\( m \) is the index for the direction.

Find McBeth Color Checker using color ratios:

1. Region segmentation
2. RAG, nodes contain color ratio
3. search for subgraph

### 6.2 Color Checker

Here we will see how to find the McBeth color checker using Hough transformation.

The following material has been taken from the student’s thesis of Matthias Grobe, Erlangen 2000

### 6.3 Color Calibration

Back to 6.2:

As noted in [AF02], the discrete version of (6.2) is often written as a sum of 31 samples
\[
\rho_k(x) = \sum_{\lambda=1}^{31} E_\lambda \cdot S_\lambda(x) \cdot R_{k,\lambda} \cdot \Delta \lambda ,
\]  
(6.10)

which can be written in Matrix notation as

\[ \rho = Cr . \]
Figure 6.4: Example for “5 lines on 3 lines”-criterion
Chapter 7

Classification

Sample:

- Train
- Test
- (Evaluation/verification)

Simple object recognition using color histograms:

Features: Histograms

Question: How to classify?

Question: How to learn?

The following material is taken from [PH03, Chapter 21ff]

Let us assume that the numerical feature \( c \) is a \( d \)-dimensional real-valued vector. A classifier is

![Image of classification system]

Figure 7.1: The architecture of a simple classification system [Nie83] together with the notation for signals, feature vectors, and class index
mathematically defined by the discrete mapping
\[
\zeta : \mathbb{R}^d \rightarrow \{1, 2, \ldots, K\}
\]
which assigns a class index \( \kappa \), i.e., a discrete value, to a feature vector \( c \). The assignment \( \zeta \) is the so-called decision function also called the decision rule.

### 7.1 General Notes on Classifiers

Before we start with various definitions of the decision function \( \zeta \), we have to explain some basic concepts of classification theory.

The training of classifiers based on unlabeled feature vectors is incomparably harder than the supervised case of learning [Rip96]. The classification of previously unobserved features is called the generalization property of classifiers.

The error probability \( p_0 \) of this optimal classifier is the so-called Bayesian error probability. The major problem is the computation of a decision rule that results in an optimal classifier.

### 7.2 Design of Classifiers

The design of classifiers is based on the fundamental assumption that feature vectors of the same class have a small distance with respect to a suitably defined distance function. This measure might be a geometric distance function (e.g., the Euclidean distance), some probability measure (e.g., a posteriori probability) or others.

### 7.3 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.

For a formal definition, we set \( c = (c_1, c_2, \ldots, c_d)^T \in \mathbb{R}^d \) to be the observed, \( d \)-dimensional feature vector. A linear discriminant classifier applies the decision rule
\[
\zeta(c) = \begin{cases} 
1, & \text{if } q(c) > 0 \\
2, & \text{otherwise}
\end{cases},
\]
with a splitting function
\[
q(c) = q_0 + \sum_{i=1}^{d} q_i c_i = q_0 + (q_1, q_2, \ldots, q_d) \cdot (c_1, c_2, \ldots, c_d)^T,
\]
7.3. LINEAR DISCRIMINANTS

Figure 7.2: Two classes in a one-dimensional feature space

Figure 7.3: Two classes in a two-dimensional feature space

and \( q_t \in \mathbb{R} \).

\[
\zeta(e) = \arg \max_{\lambda} \tilde{q}_\lambda(e) = \arg \max_{\lambda} \left( \tilde{q}_{\lambda,0} + \sum_{i=1}^{d} \tilde{q}_{\lambda,i}c_i \right), \tag{7.4}
\]

where \( \tilde{q}_\lambda \) denotes the splitting polynomial of class \( \Omega_\lambda \) which is defined by the \((d+1)\)–dimensional vector \( \tilde{q}^T_\lambda = (\tilde{q}_{\lambda,0}, \tilde{q}_{\lambda,1}, \ldots, \tilde{q}_{\lambda,d}) \).

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

\[
\omega = \bigcup_{\kappa=1}^{K} \omega_\kappa \tag{7.5}
\]
\( \chi_\kappa(c) = \begin{cases} 1, & \text{if } c \text{ belongs to } \Omega_\kappa \\ 0, & \text{otherwise} \end{cases} \) \quad (7.6)

Let \( j c_\lambda \) be the \( j \)-th sample feature of \( \omega_\lambda \) which is known to belong to class \( \Omega_\lambda \). For each \( d \)-dimensional feature vector \( j c_\lambda = (j c_{\lambda,1}, j c_{\lambda,2}, \ldots, j c_{\lambda,d})^T \) \( (\lambda = 1, 2, \ldots, K, \text{ and } j = 1, 2, \ldots, N_\lambda) \) of the training set we get \( K \) equations \( (\kappa = 1, 2, \ldots, K) \):

\[
q_\kappa(j c_\lambda) = q_{\kappa,0} + \sum_{i=1}^{d} q_{\kappa,i} j c_{\lambda,i} = \chi_\kappa(j c_\lambda), \quad (7.7)
\]

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

\[
\tilde{j c_\lambda} = (1, j c_{\lambda,1}, j c_{\lambda,2}, \ldots, j c_{\lambda,d})^T
\]

by just adding the component 1. This trick allows us to introduce the a matrix \( A \in \mathbb{R}^{D \times K(d+1)} \) which is shown below in (7.12) where

\[
D = K \cdot \sum_{\kappa=1}^{K} N_\kappa. \quad (7.8)
\]

Furthermore we set the vector \( x \in \mathbb{R}^{K(d+1)} \) to

\[
x = (q_{1,0}, q_{1,1}, \ldots, q_{1,d}, q_{2,0}, q_{2,1}, \ldots, q_{2,d}, \ldots, q_{K,0}, q_{K,1}, \ldots, q_{K,d})^T \quad (7.9)
\]

Using the above notation, the computation of linear splitting functions now corresponds to solving the system of linear equations:

\[
A x = b. \quad (7.10)
\]

For real data, the probability that the vector \( b \) is not in the range of the matrix \( A \), is one. Therefore, we compute the solution \( x \) which minimizes the residual \( \| A x - b \| \). The coefficients \( x \) of the linear splitting functions are thus given by

\[
x = A^+ b. \quad (7.11)
\]

The pseudo-inverse \( A^+ \) is usually computed using SVD (see Sect. ??).
7.4 Polynomial Classifiers

An obvious generalization of linear discriminant classifiers is possible, if multivariate polynomials of higher degrees are used instead of linear functions. A multivariate polynomial

$$q_{\lambda}(c) = \sum_{i_1, i_2, \ldots, i_d=1}^{m} q_{i_1, i_2, \ldots, i_d} c_1^{i_1} c_2^{i_2} \cdots c_d^{i_d}, \quad (7.13)$$

is attached to each class \(\Omega_{\lambda}\), and the decision rule \((7.4)\) remains unchanged.

7.5 Bayesian Classifiers

The a posteriori probability

$$p(\Omega_{\lambda} | c) = \frac{p(\Omega_{\lambda}) p(c | \Omega_{\lambda})}{p(c)} = \frac{p(\Omega_{\lambda}) p(c | \Omega_{\lambda})}{\sum_{\lambda=1}^{K} p(\Omega_{\lambda}) p(c | \Omega_{\lambda})} . \quad (7.14)$$

summarizes the probability that the class \(\Omega_{\lambda}\) is present, if the feature vector \(c\) is observed. This discrete measure is the basic component of the Bayesian classifier and its decision rule.

$$\zeta(c) = \arg\max_{\lambda} p(\Omega_{\lambda} | c) = \arg\max_{\lambda} p(\Omega_{\lambda}) p(c | \Omega_{\lambda}) . \quad (7.15)$$

The principle of Bayesian classifiers is illustrated in Figure \(7.4\).
32

CHAPTER 7. CLASSIFICATION

\[ p(\Omega_3|c) \]
\[ p(\Omega_1|c) \]
\[ p(\Omega_2|c) \]
\[ \max_\lambda p(\Omega_\lambda|c) \]
\[ \Omega_\kappa \]

Figure 7.4: The principle of the Bayesian classifier

\[ \zeta(c) = \arg\max_\lambda p(\Omega_\lambda|c) = \arg\max_\lambda p(\Omega_\lambda) p(c|\Omega_\lambda) \]
\[ = \arg\max_\lambda p(c|\Omega_\lambda). \quad (7.16) \]

\[ p(c|\Omega_\kappa) = \frac{1}{\sqrt{|\det 2\pi \Sigma_\kappa|}} e^{-\frac{(c - \mu_\kappa)^T \Sigma_\kappa^{-1} (c - \mu_\kappa)}{2}}. \quad (7.17) \]

7.6 From Bayesian to Geometric Classifiers

By specialization, the Bayesian classifier which uses normally distributed features can be reduced to a simple classifier. In this special classifier, decision making depends on Euclidean distances only. 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 \( p(c|\Omega_1) \) and \( p(c|\Omega_2) \) are assumed to be known. In this situation the Bayesian classifier decides for class \( \Omega_1 \) if
\[
p(\Omega_1)p(c|\Omega_1) > p(\Omega_2)p(c|\Omega_2) .
\] (7.18)

Now we specialize this decision rule by setting
\[
\Sigma_1 = \Sigma_2 = \Sigma ;
\] (7.19)

Taking the logarithm of both sides in (7.18), we get
\[
\log p(\Omega_1) + \mu_1^T \Sigma^{-1} c - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 > \log p(\Omega_2) + \mu_2^T \Sigma^{-1} c - \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 .
\] (7.20)

This decision rule proves the following important result: both splitting functions are linear in the components of the feature vector, and thus the Bayesian classifier reduces to a linear splitting function if the features are normally distributed and have the same covariance matrix. The coefficients of the polynomials are defined by functions of prior probabilities, of the mean vectors, and of the covariance matrix. For practical applications, this observation has an important consequence. For \( d \)-dimensional feature vectors, the linear splitting functions require the estimation of \( K(d + 1) \) parameters in the presence of \( K \) pattern classes. In contrast, the use of normal distributions expects the computation of \( d(d + 1)/2 + K(d + 1) \) parameters. Therefore, for high-dimensional feature vectors and small values of \( K \), the learning of linear splitting function parameters might lead to more robust estimates.

A further specialization results from uniform priors:
\[
p(\Omega_1) = p(\Omega_2) .
\] (7.21)

Using this constraint we obtain the discriminant
\[
\mu_1^T \Sigma^{-1} c - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 > \mu_2^T \Sigma^{-1} c - \frac{1}{2} \mu_2^T \Sigma^{-1} \mu_2 .
\] (7.22)

which is a simplified version of the
\[
(c - \mu_1)^T \Sigma^{-1} (c - \mu_1) < (c - \mu_2)^T \Sigma^{-1} (c - \mu_2) .
\] (7.23)

For implementation purposes, of course, we prefer (7.22) to (7.23) because these splitting functions are linear in the components of \( c \).

If we additionally assume that the covariance matrix \( \Sigma \) is the identity matrix, the classification is based on the inequality
\[
(c - \mu_1)^T (c - \mu_1) < (c - \mu_2)^T (c - \mu_2) .
\] (7.24)
CHAPTER 7. CLASSIFICATION

This decision rule compares the squared Euclidean distances

\[ ||c - \mu_1||^2 < ||c - \mu_2||^2, \quad (7.25) \]

between feature and mean vectors, i.e., class centers.

The final specialization has shown that statistical classifiers lead to a simple distance measure for restricted statistical assumptions. The minimum distance classifier with respect to class dependent mean vectors is optimal if the feature vectors used are normally distributed with covariance matrices $\Sigma$ that are equal to the identity matrix.

Classifiers based on parametric densities have the disadvantage that a parametric distribution of the feature vectors must be known. This causes some problems, especially for features that are not normally distributed. In general, there exist three possibilities to verify a density assumption:

1. the distribution of features is known by construction,
2. the hypothesized parametric density is proven by statistical tests or
3. the recognition rate of the resulting classifier suggests the correctness.

The use of the Euclidean distance to a reference vector — the mean vector of each class — motivates the introduction of nearest neighbor classifiers. Instead of computing mean vectors for each class and using a distance measure to mean vectors for discrimination, we utilize all observed feature vectors of the training set as a reference. The resulting classifier is the nearest neighbor classifier.

7.7 Nearest Neighbor Classifier

The nearest neighbor classifier requires a set of classified sample data, i.e., for each element $e_i$ of $C = \{e_1, e_2, \ldots, e_n\}$ the class number $\zeta(e_i)$ is known. For a new feature vector, the class that points to the reference vector with the closest distance to the new vector is chosen. Thus, the decision rule is defined by

\[ \zeta(e) = \arg \min \zeta(e_i) \quad \{||c - e_i|| \mid i = 1, 2, \ldots, n\} \quad (7.26) \]

\[ p_B \leq p_{NN} \leq 2p_B \quad (7.27) \]

There exist various modifications of the nearest neighbor classifier.
7.8. TESTING A CLASSIFIER

Figure 7.6: Voronoi diagram with 27 two-dimensional reference vectors

<table>
<thead>
<tr>
<th>FOR $\forall \omega \in \Omega_p \cup \Omega_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Train classifier with $\Omega_p \setminus \omega$ and $\Omega_n \setminus \omega$</td>
</tr>
<tr>
<td>Classify $\omega$ and record recognition rates</td>
</tr>
<tr>
<td>Average recognition of the previous tests</td>
</tr>
</tbody>
</table>

Figure 7.7: Leave one out Strategy

7.8 Testing a Classifier

Now as we have explained our classifier, how do we examine its properties? Measures characterizing a classifier are recognition rates or probabilities of mis-classification. In a two-class problem for positive and negative samples often four numbers are given to describe a classification result:

1. the rate or probability of correctly classifying positive samples (true positive)
2. the rate or probability of incorrectly classifying negative samples as positive (false positive)
3. the rate or probability of correctly recognizing negative samples (true negative)
4. the rate or probability of incorrectly classifying positive samples as negative (false positive)

N-Fold cross validation

7.8.1 Learning and Testing

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

Given an image, the goal is to recognize objects. The contest focuses on:

- A small set of possible objects (approx. 12)
- Only planar objects (2D processing)
- Homogeneous background

However, there are several issues to consider:

- Size / orientation changes
- Illumination changes
- Different background
Some Details on the Strategy

Given an image . . .

- try to isolate image from background
- compute so-called features (e.g. mean of the object region)

Given several images of one object type (a so-called class) . . .

- compute features of each image
- store set of features somehow for that class

Given one images and a set of known classes

- compute features for the image
- compute „distance” of features to all known feature sets
- select „closest” set

Details will be given in the lectures

Organization

- Work in small groups (3 students)
- Groups should be multi-national
- Approach is up to you – we only give hints
- During the two weeks
  - you will get to know nice features in the lectures
  - implement them in the exercises and project work
  - create a tool box for feature extraction
- Contest on last friday

Example images will be provided.

- size, orientation may vary
- image size may be different
- background may be different or textured

Hint: start working on the simple cases!
Schedule

- Mo: build groups – view sample images
- Tu-Fr: start implementing feature extractors
- Fr: create image database
- Mo-Tu: work on the database
- We: Define test framework
- Th: learn about color histograms, integrate them
- Fr: contest

Hints:

- Start to work today!
- Images will be in color
- First tasks:
  - separate object from background or identify background
  - compute simple features to test your system (e.g. mean color)
  - plot the feature vectors (e.g. with gnuplot) to see whether
    - match for object of one class
    - they differ for different classes
Appendix B

SVD

- Powerful normal form for matrices
- Method of numerical linear algebra
  - invented in the 19th century,
  - rediscovered and pushed for practical application by Gene Golub,
  - established in computer vision by Tomasi’s factorization algorithm [TK92]
- Advantage: SVD can be applied to solve most problems in linear algebra (not always in an optimal manner)
  - solution of overdetermined of linear equations
  - computation of condition numbers
  - enforcing rank criterion, etc.

SVD known to image processors since [TK92]

POOR MAN’s SVD

Every $M \times N$ matrix $A$ can be decomposed into a product

$$ A = U D V^T, $$

where

- $U \in \mathbb{R}^{M \times M}$, $V \in \mathbb{R}^{N \times N}$ square and orthogonal,
- $D \in \mathbb{R}^{M \times N}$ diagonal matrix.
APPENDIX B. SVD

With $D$:

\[
D = \begin{pmatrix}
\sigma_1 & 0 & 0 & 0 \\
0 & \sigma_2 & 0 & 0 \\
0 & 0 & \ddots & 0 \\
0 & 0 & \cdots & \sigma_{\min(m,n)}
\end{pmatrix}
\]

$\sigma_i$: singular values

$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min(m,n)} \geq 0$

- Rank of $A$: $\text{rank}(A) = \# \{ \sigma_i > 0 \}$
- Numerical $\epsilon$-rank of $A$: $\text{rank}_{\epsilon}(A) = \# \{ \sigma_i > \epsilon \}$
- Kernel of $A$
  - Spanned by the column vectors $v_i$ of $V$, where $\sigma_i = 0$.
  - Condition number
    \[
    \kappa = \frac{\sigma_1}{\sigma_k}
    \] (B.1)
    where $\sigma_k$ is smallest non-zero singular value.
- Range of $A$ spanned by the column vectors $u_i$ of $U$, where $\sigma_i$ are non-zero singular values.
- Orthogonalization of $A$
  - Let $A = UDV^T$ numerically orthogonal,
  - Set all $\sigma_i = 1$,
  - Compute $A' = UD'V^T$,
  - Then $A'$ is exactly orthogonal and minimizes
    \[
    ||A - A'||_F
    \] (B.2)
    where $||.||_F$ denotes the Frobenius norm.

RICH MAN’s SVD

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

\[
A^{M \times N} = U^{M \times N}D^{N \times N}V^{T N \times N} = \sum_{i=1}^{N} \sigma_i u_i v_i^T
\]

where the three matrices $U, D, V$ have the following properties
• the columns of the $M \times N$ matrix $U$ are the eigenvectors $u_i$ of $AA^T$ (hence, $U^T U = I$); i.e. the column vector $u_i$ is obtained from
\[ \lambda_i u_i = (AA^T) u_i \]
and $\lambda_i$ is the $i$-th eigenvalue of $AA^T$
• matrix $D$ is diagonal with elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_N = 0$, $r \leq N$; the elements $\sigma_i$ of $D$ are the singular values; they are the square–roots of the eigenvalues of $AA^T$, i.e. $\sigma_i = \sqrt{\lambda_i}$;
• the columns of the $N \times N$ matrix $V$ are the eigenvectors $v_i$ of $A^T A$ (hence, $V^T V = I$);
• if $M < N$, $\sigma_i = 0$, $i = M + 1, \ldots, N$; corresponding columns of $U$ are zero;

• the condition number $c = \frac{\sigma_1}{\sigma_N}$ measures the “degree of singularity” of $A$; if $1/c$ is comparable to the arithmetic precision of the computer, $A$ is ill-conditioned and “singular for practical purposes”;
• columns of $U$ corresponding to non–zero singular values span the range of $A$; columns of $V$ corresponding to zero singular values span the kernel of $A$;
• denoting the columns of $U$ and $V$ by $u_i$ and $v_i$, respectively, we have
\[ A^T u_i = \sigma_i v_i \]
and also
\[ A v_i = \sigma_i u_i \]
• the pseudo–inverse $A^+$ of $A$ is
\[ A^+ = V D^{-1} U^T \] (B.3)
with entries of $D^{-1}$ equal to $D^{-1}$ for nonzero singular values and zero otherwise
• the Frobenius–norm of a matrix is
\[ ||A||_F = \sum_{i,j} \sqrt{a_{i,j}^2} = \sum_i \sqrt{\sigma_i^2} \]
consider the linear equation $Ax = a$ with $A$ a square matrix;

1. if $A$ is nonsingular, we get from SVD
\[ x = A^{-1} a = V D^{-1} U^T a \quad \text{with} \quad D^{-1} = \text{diag}(1/\sigma_i) \]
2. if \( A \) is singular and \( a \) is in the range of \( A \), we get from SVD the solution of minimal norm
\[
x = V D^{-1} U^T a \quad (*)
\]
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
\[
x = V D''^{-1} U^T a \quad (B.4)
\]
with \( D''^{-1} = \text{diag}(d''_i) \), with \( d''_i = 1/\sigma_i \), if \( \sigma_i \geq \Delta \), and \( d''_i = 0 \), if \( \sigma_i < \Delta \);

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 \( (B.4) \),
   the solution space is obtained by adding the linear combination of the kernel of \( A \) (see above: a basis of this kernel 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”\[1\]

Annotated Bibliography


B.1 Names and Symbols

The following list shows all important symbols in the sequence of their definition in the text.

\[ N_M \]
Mask size \( Y \) \[ 5, 13 \]

\[ N_M \]
Sub-image size vertically \[ 5, 13 \]

\[ M_M \]
Mask size \( X \) \[ 5, 13 \]

\[ M_M \]
Sub-image size horizontally \[ 5, 13 \]

\[ M \]
Image size \( Y \) \[ 6, 7, 12, 13, 22 \]

\[ N \]
Image size \( X \) \[ 6, 7, 12, 13, 22 \]

\[ l \]
Index of Bins \[ 6, 7, 8 \]

\[ N_L \]
Number of Bins \[ 6, 7, 8, 12, 13 \]

\[ \zeta \]
Color mapping function to bin \[ 6, 7, 22 \]

\[ d \]
Geometric distance measure \[ 10, 11 \]

\[ R \]
Ratio histogram \[ 11 \]

\[ f \]
Color image function \[ 11 \]

\[ D_r \]
Object mask \[ 11 \]

\[ N_o \]
Number of objects \[ 11 \]

\[ O \]
Segmentierungsobjekte \[ 11 \]

\[ E \]
Spectral energy distribution of illumination \[ 21, 22, 23 \]

\[ S \]
Reflection coefficient \[ 21, 22, 23 \]

\[ G \]
Gauß-Maske \[ 22 \]

\[ \kappa \]
Class Index \[ 27, 28, 29, 30, 31, 32 \]