Image Procesing and Computer Vision Notes

by Mattia Orlandi

5. Image Segmentation

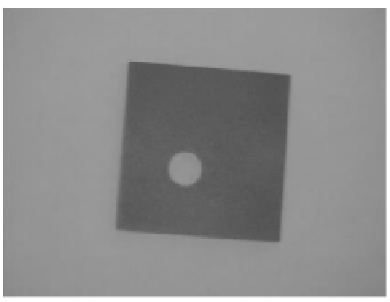
- Given a vector-valued function $P(x, y)$ encoding a set of image properties (e.g. intensity, colour, contrast, texture, etc.), **segmentation aims at partitioning the image into disjoint homogeneous regions** according to P.
- Segmentation **should preserve spatial proximity** (i.e. two nearby pixels must belong to same region, unless they exhibit significant different P values) and **provide large regions** featuring **few holes** and **well-localized smooth boundaries**.
- It splits the image into **semantically meaningful parts** (*semantic knowledge*) on which further analysis can be focused.
- Usually segmentation relies just on a single property, such as intensity ($P(x, y) = I(x, y)$) or colour ($P(x, y) = [\, I_r(x, y) \quad I_g(x, y) \quad I_b(x, y) \,]^T$).

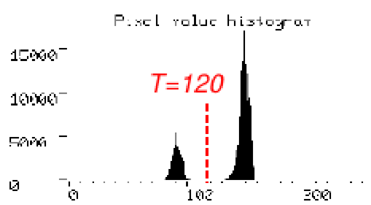
Image Binarization

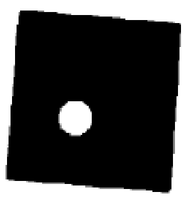
- Usually the **objects of interest** (*foreground*) are **neatly darker/brighter than irrelevant areas** of the scene (*background*).
- In industrial applications this is achieved by *backlighting*: the **object** is **placed between a light source and the camera**, so as to cast onto the image a very dark shadow representing object's shape.
- In this scenario, the first image analysis step consists in **image binarization**, i.e. **segmentation** of image pixels into two disjoint regions corresponding to **foreground and background**.
- If **foreground** must be **further split into sub-regions** corresponding to individual objects, another image analysis step called *image labeling* (*connected components labeling*) should be performed: **foreground pixels belonging to different objects are given different labels**.

5.1. Intensity Thresholding

- **Inherently-binary images** exhibit a clearly *bimodal* gray-level histogram with **two well-separated peaks** corresponding to background and foreground.







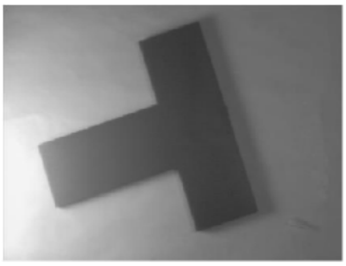
- Binarization can thus be achieved by applying the **thresholding operator** with a certain threshold T.

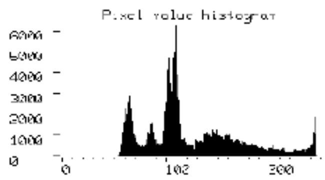
```
#define FOREGROUND 0
#define BACKGROUND 255
#define THRESHOLD 120

for(i = 0; i < N; i++)
  for(j = 0; j < M; j++)
    if(I[i][j] <= THRESHOLD)
      O[i][j] = FOREGROUND;
    else
      O[i][j] = BACKGROUND;
```

- Whenever the **histogram** is **not clearly bimodal** (e.g. due to illumination varying significantly across the scene), **binarization by intensity thresholding fails** to provide the correct segmentation (in these cases it is necessary to search for object's edges).

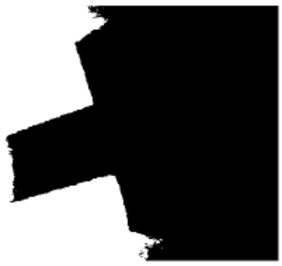
Original Image





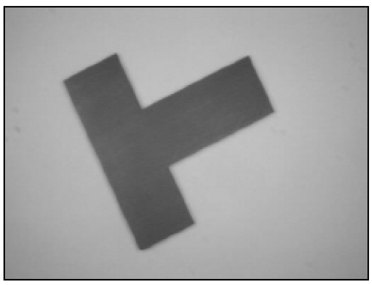
Gray-level Histogram

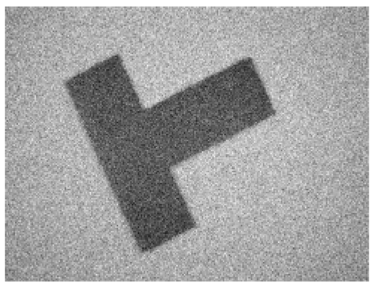
T=80



T=120

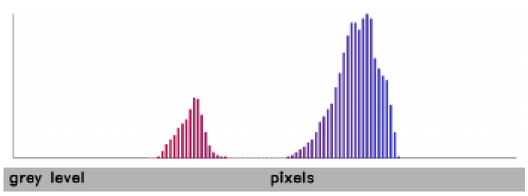
- When the **overlap between the two modes** is **due to noise**, **image smoothing** (e.g. Gaussian filter) **may improve the histogram** and thus the binarization.

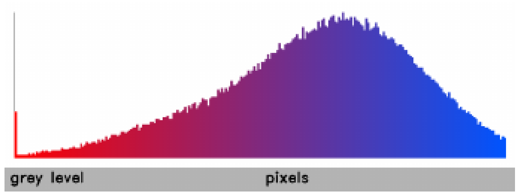


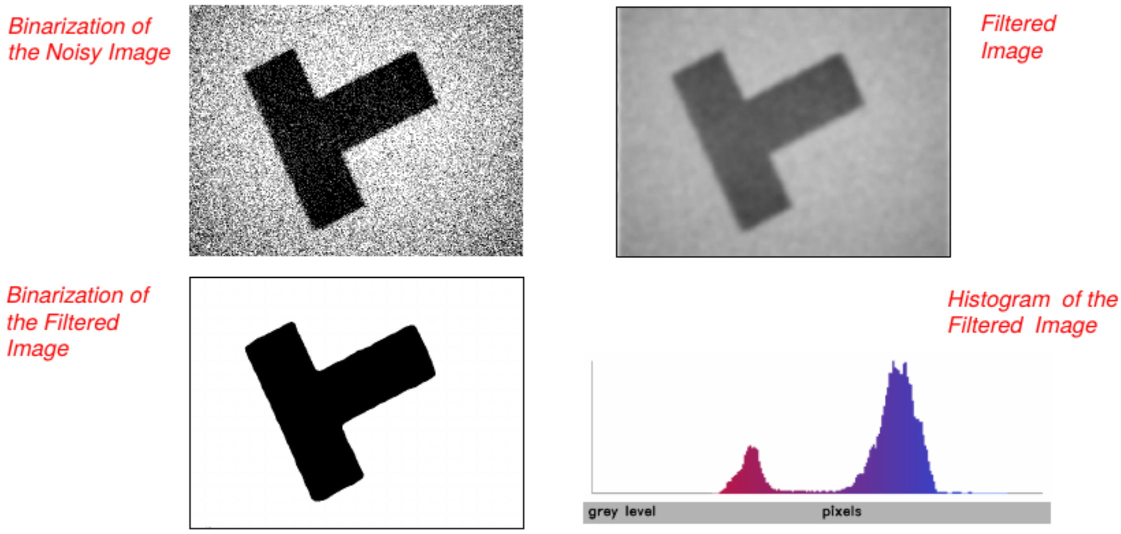


Noiseless Image and its Histogram

Noisy Image and its Histogram

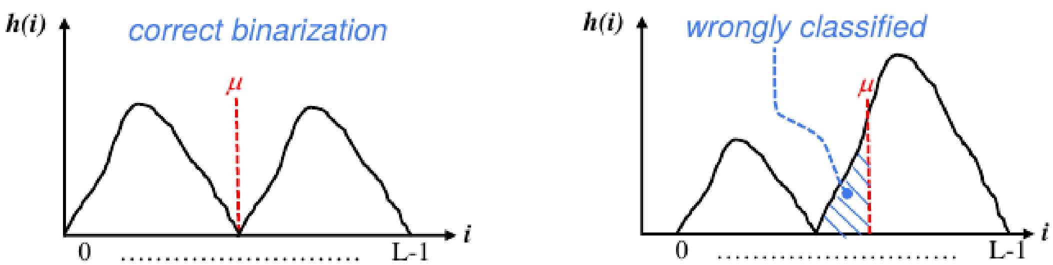






5.2. Automatic Threshold Selection

- Stability over time of the lighting conditions cannot be guaranteed $\Rightarrow$ **more robust** (though computationally more demanding) **approach** needed, whereby an **algorithm computes automatically a suitable binarization threshold** in each image under analysis.
- Simple heuristic approaches:
  - $T = \mu$: **works as long as pixels are equally distributed between the two classes** (if not, a certain properly estimated percentile may be chosen, e.g. 20^{th} percentile if objects cover 20 of image).

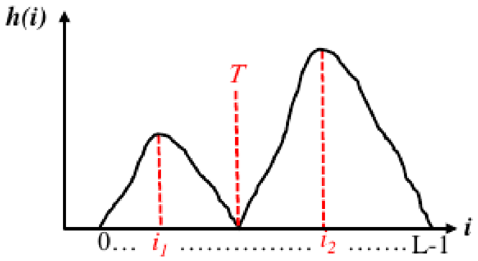


  - $T = \arg\min\{h(i) | i \in [i_1, i_2]\}$: requires **finding the two main peaks**, which often implies **smoothing the histogram** (e.g. Gaussi filter) beforehand in order **to avoid local maxima**.
    To find the two main peaks i_1, i_2 pixels neighbours are checked:

```
// 3-levels neighborhood
((h[i] > h[i-1]) && (h[i] > h[i+1]))

// 5-levels neighborhood
((h[i] > h[i-1]) && (h[i] > h[i+1]) && (h[i-1] > h[i-2]) && (h[i+1] >
h[i+2]))
```

    and as **threshold** the **valley between the two peaks** is chosen (i.e. the gray-level with minimum histogram count between the peaks).



Otsu's Algorithm

- **Automatic threshold selection algorithm**, which chooses the **optimal threshold by minimizing**, across the gray-level range, the so-called **Within-group Variance** of the resulting regions, an **indicator measuring how spread region intensities are** upon binarization by a given gray-level.
- Since the search space is small, the algorithm can try all thresholds and choose the one that minimizes the variance.
- Define:
  - $i = 1, \ldots, L$: gray-levels of the image
  - N : number of pixels of the image
  - $h(i)$: i^{th} entry of the image histogram
  - $p(i) = \frac{h(i)}{N}$: probability of gray-level i ($\sum_{i=1}^{L} p(i) = 1$)
- Mean μ and variance σ^2 of the PMF associated with image gray-levels can be expressed as:

$$\mu = \sum_{i=1}^{L} i\, p(i)$$
$$\sigma^2 = \sum_{i=1}^{L} (i-\mu)^2\, p(i)$$

- Any threshold value t will split pixels into two disjoint regions whose PMFs have the following mean and variance:

$$\mu_1(t) = \sum_{i=1}^{t} i\, p(i)/q_1(t), \quad \sigma_1^2(t) = \sum_{i=1}^{t} (i-\mu_1(t))^2 p(i)/q_1(t)$$
$$\mu_2(t) = \sum_{i=t+1}^{L} i\, p(i)/q_2(t), \quad \sigma_2^2(t) = \sum_{i=t+1}^{L} (i-\mu_2(t))^2 p(i)/q_2(t)$$

  where $q_1(t) = \sum_{i=1}^{t} p(i),\ q_2(t) = \sum_{i=t+1}^{L} p(i)$ are the total number of pixels belonging to the two classes, used for normalization.
- The **Within-group Variance** of the two regions is defined as the **weighted sum of their variances**:

$$\sigma_W^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)$$

- Minimizing $\sigma_W^2(t)$ would require computing $\mu_1(t)$, $\mu_2(t)$, $\sigma_1^2(t)$, $\sigma_2^2(t)$, $q_1(t)$ (since $q_2(t) = 1 - q_1(t)$) for each threshold value t (i.e. each gray-level value) $\Rightarrow$ **more efficient approach based on the Between-group Variance**:

$$
\begin{aligned}
\sigma^2 &= \sum_{i=1}^{L}(i-\mu)^2 p(i) \\
&= \sum_{i=1}^{t}(i-\mu_1(t)+\mu_1(t)-\mu)^2 p(i) + \sum_{i=t+1}^{L}(i-\mu_2(t)+\mu_2(t)-\mu)^2 p(i) \\
&= \sum_{i=1}^{t}\left[(i-\mu_1(t))^2 + 2(i-\mu_1(t))(\mu_1(t)-\mu) + (\mu_1(t)-\mu)^2\right] p(i) \\
&+ \sum_{i=t+1}^{L}\left[(i-\mu_2(t))^2 + 2(i-\mu_2(t))(\mu_2(t)-\mu) + (\mu_2(t)-\mu)^2\right] p(i)
\end{aligned}
$$

where:

$$
\begin{aligned}
&\sum_{i=1}^{t}(i-\mu_1(t))(\mu_1(t)-\mu)p(i) \\
&= (\mu_1(t)-\mu)\sum_{i=1}^{t}(i-\mu_1(t))p(i) \\
&= (\mu_1(t)-\mu)\left[\sum_{i=1}^{t} i\,p(i) - \mu_1 \sum_{i=1}^{t} p(i)\right] \\
&= (\mu_1(t)-\mu)\left[\mu_1(t)q_1(t) - \mu_1(t)q_1(t)\right] = 0, \\
&\sum_{i=t+1}^{L}(i-\mu_2(t))(\mu_2(t)-\mu)p(i) = 0
\end{aligned}
$$

therefore σ^2 can be expressed as:

$$
\begin{aligned}
\sigma^2 &= \sum_{i=1}^{t}(i-\mu_1(t))^2 p(i) + \sum_{i=t+1}^{L}(i-\mu_2(t))^2 p(i) \\
&+ (\mu_1(t)-\mu)^2 q_1(t) + (\mu_2(t)-\mu)^2 q_2(t) \\
&= \left[q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t)\right] + \left[(\mu_1(t)-\mu)^2 q_1(t) + (\mu_2(t)-\mu)^2 q_2(t)\right] \\
&\Rightarrow \sigma^2 = \sigma_W^2(t) + \sigma_B^2(t)
\end{aligned}
$$

where $\sigma_B^2(t)$ is the **Between-group Variance**, an **indicator of how well classes are separated** one to the other.

- As σ^2 is independent from the chosen t, **minimizing $\sigma_W^2(t)$ is equivalent to maximizing** $\sigma_B^2(t)$, which is **more efficient** since it does not require computing $\sigma_1^2(t)$ and $\sigma_2^2(t)$ but only $\mu_1(t)$ and $\mu_2(t)$.
- Further computational savings can be achieved as follows:

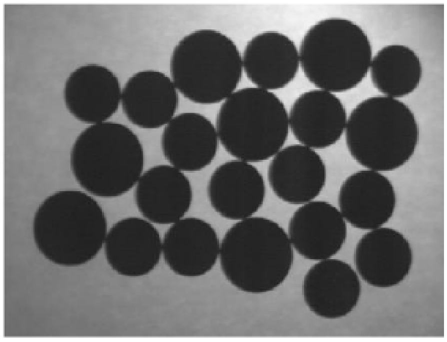
$$
\begin{aligned}
&\mu = q_1(t)\mu_1(t) + q_2(t)\mu_2(t), \quad q_2(t) = 1 - q_1(t) \\
&\Rightarrow \sigma_B^2(t) = (\mu_1(t)-\mu)^2 q_1(t) + (\mu_2(t)-\mu)^2 q_2(t) \\
&= \left[\mu_1(t) - q_1(t)\mu_1(t) - q_2(t)\mu_2(t)\right]^2 q_1(t) + \left[\mu_2(t) - q_1(t)\mu_1(t) - q_2(t)\mu_2(t)\right]^2 q_2(t) \\
&= \left[\mu_1(t)(1-q_1(t)) - \mu_2(t)(1-q_1(t))\right]^2 q_1(t) + \left[\mu_2(t) - q_1(t)\mu_1(t) - \mu_2(t)(1-q_1(t))\right]^2 (1-q_1(t)) \\
&= \left[(\mu_1(t)-\mu_2(t))((1-q_1(t)))\right]^2 q_1(t) + \left[\mu_2(t)q_1(t) - \mu_1(t)q_1(t)\right]^2 (1-q_1(t)) \\
&= \left[\mu_1(t)-\mu_2(t)\right]^2 \left[1-q_1(t)\right]^2 q_1(t) + \left[\mu_1(t)-\mu_2(t)\right]^2 q_1^2(t)\left[1-q_1(t)\right] \\
&= \left[\mu_1(t)-\mu_2(t)\right]^2 \left[1-q_1(t)\right] q_1(t) \left[q_1(t) + 1 - q_1(t)\right] \\
&\Rightarrow \sigma_B^2(t) = q_1(t)\left[1-q_1(t)\right]\left[\mu_1(t)-\mu_2(t)\right]^2
\end{aligned}
$$

in this way for every threshold t $\sigma_B^2(t)$ **can be computed using only 2 additions and 3 multiplications**.

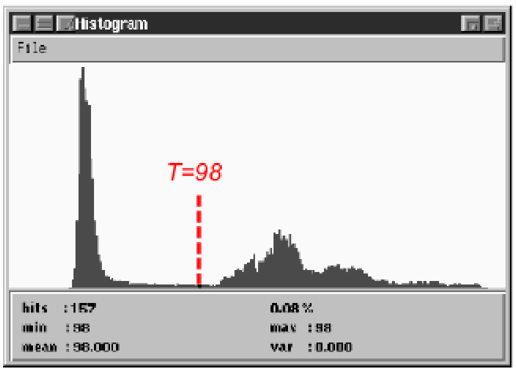
- Moreover, using a box-filtering-like approach, instead of computing every quantity involved from scratch, it is possible to apply **incremental computation** s.t., at every step, the results of the step before are used:

$$q_1(t+1) = q_1(t) + p(t+1)$$
$$\mu_1(t+1) = \frac{q_1(t)\mu_1(t) + (t+1)\cdot p(t+1)}{q_1(t+1)}$$
$$\mu_2(t+1) = \frac{\mu - q_1(t+1)\cdot \mu_1(t+1)}{1 - q_1(t+1)}$$

Input Image

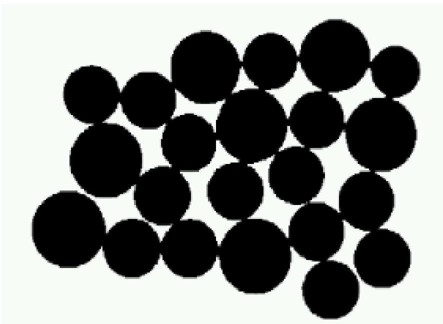


Gray-level Histogram



with threshold computed by Otsu's

Binarized Image



5.3. Adaptive Thresholding

- **Global thresholding** methods rely on the **assumption of uniform lighting**, and thus fail in case of **violations to that assumption** (e.g. shadows) $\Rightarrow$ **necessity of a spatially varying** (i.e. **adaptive**) **threshold**.

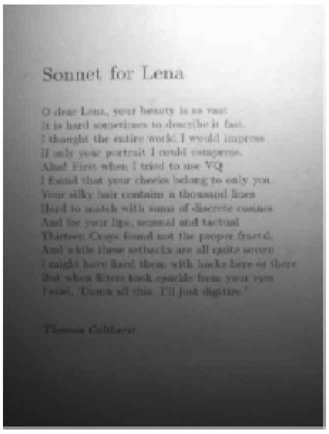
- Adaptive methods compute a **specific threshold at each pixel ($T = T(x, y)$)**, based on **intensities within a small neighborhood**.

- The reason behind the use of a **local window** is that, although, globally, **lighting** is not uniform, it is **uniform locally**.

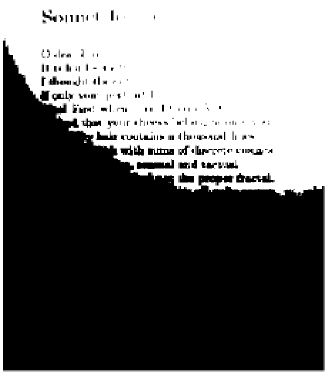
- For the sake of **efficiency**, **Mean or Median filters** are used.

- If the **neighborhood** is **too small**, it may contain either **only background pixels or only foreground pixels**, thus leading to a **wrong binarization**.

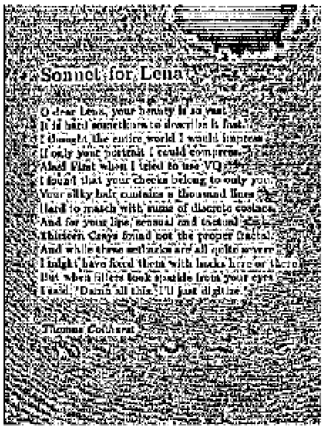
Image of an unevenly lit scene



Binarization by global thresholding



Binarization by adaptive thresholding

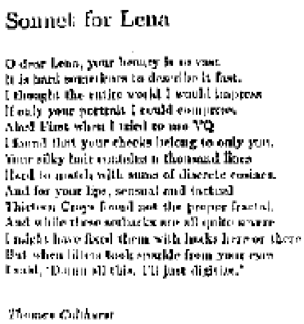


T(x,y) =μ(x,y), 7×7 neighbourhood

- To avoid that, a suitable **constant is subtracted from the mean**, s.t. the **wrongly classified background pixels are pushed above the threshold**:

$$T(x,y) = \mu(x,y) - C$$

where C **must be sufficiently big to compensate intensity oscillations** in background pixels, and **sufficiently small to avoid classification errors** for foreground pixels.



T(x,y) =μ(x,y), C=10
7×7 neighbourhood

5.4. Colour-based Thresholding

- If the sought-for object exhibits a **known color different from background one**, segmentation can be performed relying on color information.
- Given pixel's color:

$$\mathbf{I}(p) = \begin{bmatrix} I_r(p) \\ I_g(p) \\ I_b(p) \end{bmatrix}$$

segmentation can be **achieved by computing and thresholding the distance** (e.g. Euclidian) **between each pixel's colour and the reference foreground color $\boldsymbol{\mu}$** (which depends on the domain problem):

$$\forall p \in \mathbf{I} : \begin{cases} \operatorname{dist}(\mathbf{I}(p), \boldsymbol{\mu}) \leq T \Rightarrow O(p) = F \\ \operatorname{dist}(\mathbf{I}(p), \boldsymbol{\mu}) > T \Rightarrow O(p) = B \end{cases}$$

where F, B represent foreground and background, and:

$$\text{dist}(\mathbf{I}(p), \boldsymbol{\mu}) = \left[(\mathbf{I}(p) - \boldsymbol{\mu})^T \cdot (\mathbf{I}(p) - \boldsymbol{\mu})\right]^{\frac{1}{2}}$$
$$= \left[(I_r(p) - \mu_r)^2 + (I_g(p) - \mu_g)^2 + (I_b(p) - \mu_b)^2\right]^{\frac{1}{2}}$$

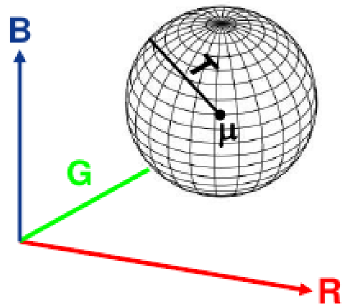
Estimating reference colour

- **Reference colour $\boldsymbol{\mu}$** can be **estimated** ("learned") from some **training images**.

- The **colour of a foreground pixel** can be **modeled as a multivariate random variable** (i.e. 3D random vector), **reference colour** can be taken as the **mean** (expected value) over **available training samples** $(\mathbf{I}(p_k), k = 1 \dots N)$:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_r \\ \mu_g \\ \mu_b \end{bmatrix} = \frac{1}{N} \sum_{k=1}^{N} \mathbf{I}(p_k)$$

- Segmentation **consists in classifying as foreground all pixels lying within a 3D sphere** of RGB colour space centered at $\boldsymbol{\mu}$ and having radius T.



- If the **foreground pixels** of an image are **"squeezed" along one particular direction**, a sphere is not suited, since a **small radius** T would produce many **false negative**, whereas a **high radius** T would produce many **false positives** $\Rightarrow$ necessity to use a surface (and thus a **distance**) able to **capture pixels variability** along every direction.

Mahalanobis Distance

- It is **"aware" of data variability** since it is based on the **covariance matrix**, which is **estimated** together with the mean **from training samples**:

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{rr}^2 & \sigma_{rg}^2 & \sigma_{rb}^2 \\ \sigma_{gr}^2 & \sigma_{gg}^2 & \sigma_{gb}^2 \\ \sigma_{br}^2 & \sigma_{bg}^2 & \sigma_{bb}^2 \end{pmatrix} \Rightarrow \begin{cases} \sigma_{i,j}^2 = \frac{1}{N} \sum_{k=1}^{N} (I_i(p_k) - \mu_i)(I_j(p_k) - \mu_j) \\ i, j \in \{r, g, b\} \end{cases}$$

- The Mahalanobis Distance is defined as:

$$\text{dist}_M(\mathbf{I}(p), \boldsymbol{\mu}) = \left[(\mathbf{I}(p) - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{I}(p) - \boldsymbol{\mu})\right]^{\frac{1}{2}}$$

- In case of independent components in $\mathbf{I}(p)$ (i.e. diagonal covariance matrix):

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_{rr}^2 & 0 & 0 \\ 0 & \sigma_{gg}^2 & 0 \\ 0 & 0 & \sigma_{bb}^2 \end{pmatrix} \Rightarrow \boldsymbol{\Sigma}^{-1} = \begin{pmatrix} 1/\sigma_{rr}^2 & 0 & 0 \\ 0 & 1/\sigma_{gg}^2 & 0 \\ 0 & 0 & 1/\sigma_{bb}^2 \end{pmatrix}$$

$$\Rightarrow \mathrm{dist}_M(\mathbf{I}(p), \boldsymbol{\mu}) = \left[(\mathbf{I}(p) - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{I}(p) - \boldsymbol{\mu})\right]^{\frac{1}{2}}$$

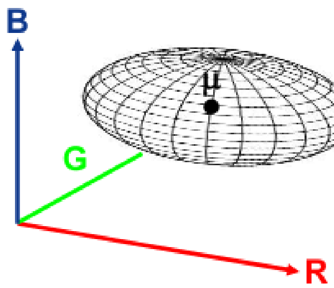
$$= \left((\mathbf{I}(p) - \boldsymbol{\mu})^T \begin{bmatrix} 1/\sigma_{rr}^2 & 0 & 0 \\ 0 & 1/\sigma_{gg}^2 & 0 \\ 0 & 0 & 1/\sigma_{bb}^2 \end{bmatrix} (\mathbf{I}(p) - \boldsymbol{\mu}) \right)^{\frac{1}{2}}$$

$$= \left((\mathbf{I}(p) - \boldsymbol{\mu})^T \begin{bmatrix} (I_r(p) - \mu_r)/\sigma_{rr}^2 \\ (I_g(p) - \mu_g)/\sigma_{gg}^2 \\ (I_b(p) - \mu_b)/\sigma_{bb}^2 \end{bmatrix} \right)^{\frac{1}{2}}$$

$$\Rightarrow \mathrm{dist}_M(\mathbf{I}(p), \boldsymbol{\mu}) = \left(\frac{(I_r(p) - \mu_r)^2}{\sigma_{rr}^2} + \frac{(I_g(p) - \mu_g)^2}{\sigma_{gg}^2} + \frac{(I_b(p) - \mu_b)^2}{\sigma_{bb}^2} \right)^{\frac{1}{2}}$$

- Contrary to the Euclidian distance, the **Mahalanobis distance weights** unequally the **differences along components of the random vector according to inverse proportionality to learned variances**.

- The **more spread** is a component, the **higher** the **variance** and thus the **lower** its **contribution to the overall distance**, resulting in a ellipsoid-like shape.



- Since the **covariance matrix** is **symmetric and real-valued**, it can **always be diagonalized** by a rotation of coordinate axes (**EigenValue Decomposition**, EVD):

$$\boldsymbol{\Sigma} = \mathbf{R}\mathbf{D}\mathbf{R}^T : \mathbf{R} = \begin{pmatrix} \mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3 \end{pmatrix}, \ \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

where $\mathbf{e}_i$ are the **orthonormal eigenvectors** of $\boldsymbol{\Sigma}$, λ_i the corresponding **eigenvalues** and $\mathbf{R}^T$ the **rotation matrix to transform data into a new coordinate system** with **axes aligned to the eigenvectors**.

- Upon rotation by $\mathbf{R}^T$ the new covariance matrix becomes D, s.t. eigenvalues of $\boldsymbol{\Sigma}$ represent the variances along eigenvectors direction.

- In the new coordinate system the **Mahalanobis distance** is a **weighted sum of contributions along the new axes**, with **weights** being **inversely proportional to variances** along new axes.