University of Bologna

Image Segmentation and Blob Analysis

Luigi Di Stefano (luigi.distefano@unibo.it)

A Workflow for Industrial Vision

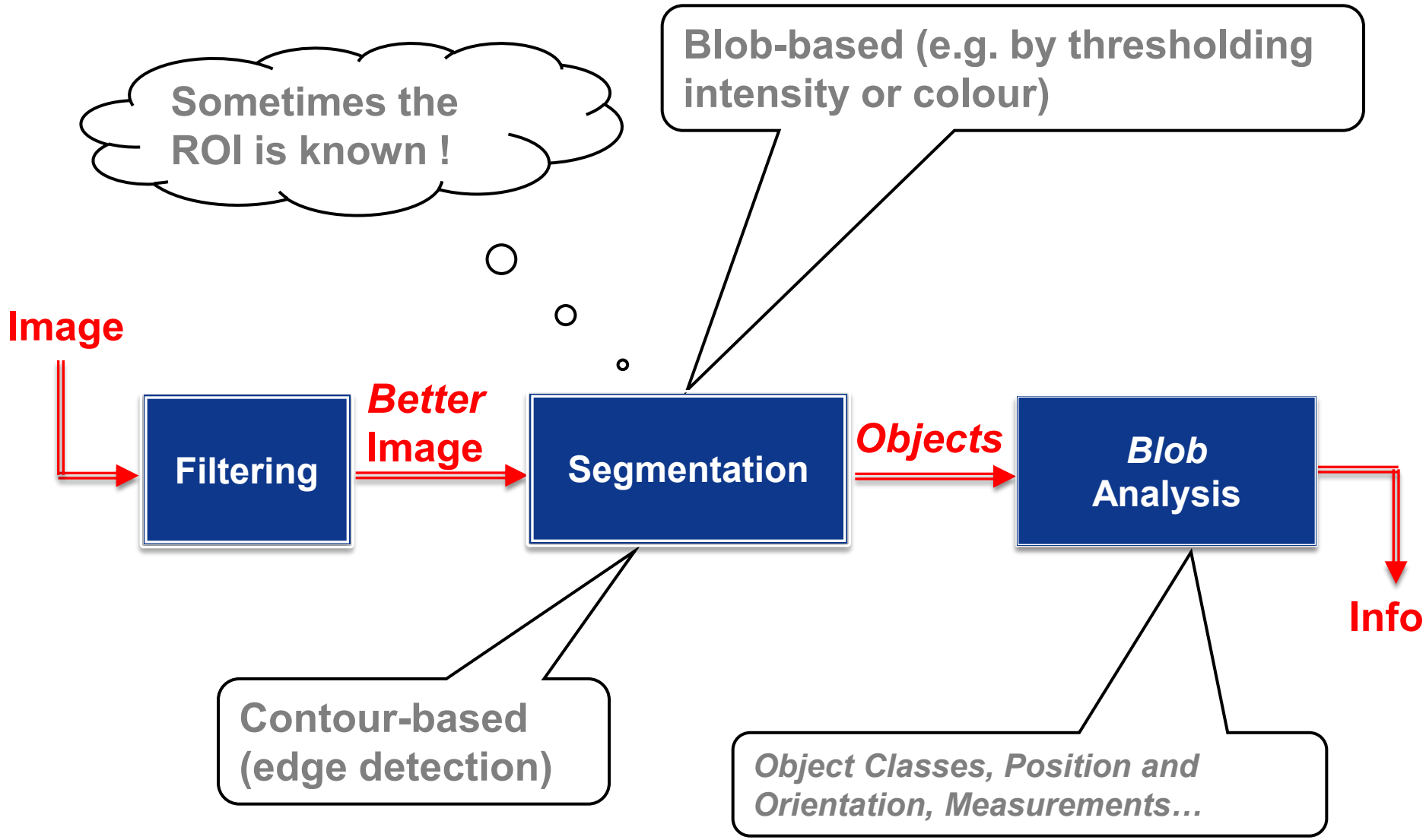
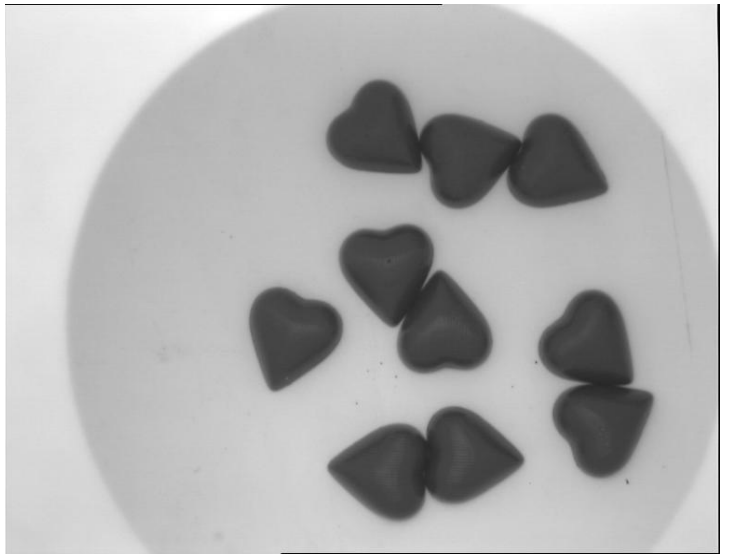


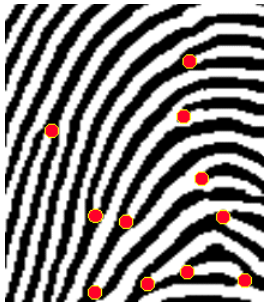
Image Binarization

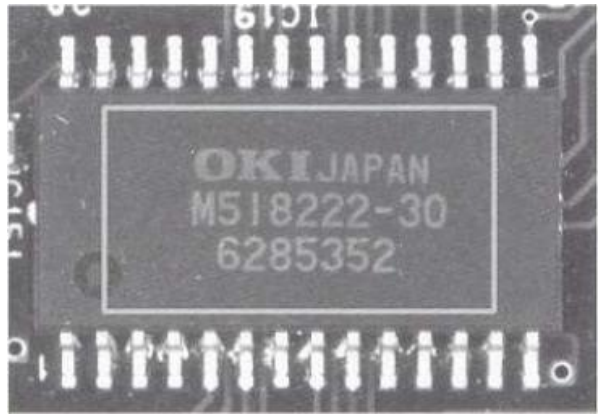
- In a variety of industrial applications the objects of interest (*foreground*) are neatly darker/brighter than the irrelevant areas of the scene (*background*). Oftentimes, this is achieved by *backlighting*, whereby the object is placed between the light source and the camera so as to cast onto the image a very dark shadow representing object's shape.
- In such circumstances, the first image analysis step consists typically in image *binarization*, i.e. segmentation of image pixels into two disjoint regions corresponding to foreground (dark/bright intensities) and background (bright/dark intensities).
- Should the application call for analysis of a single object per image, binarization would deliver all the required segmentation information.
- Conversely, the foreground region must be further split into sub-regions corresponding to individual objects (usually denoted as *Blobs*), which is usually achieved by a successive image analysis step referred to as *image labeling* (also *connected components labeling*) due to pixels belonging to different objects being given different labels.

Inherently-binary Images

Camera + Back-lighting by Basler
(Vision 2022- Stuttgart)







The Gray-Level Histogram (1)

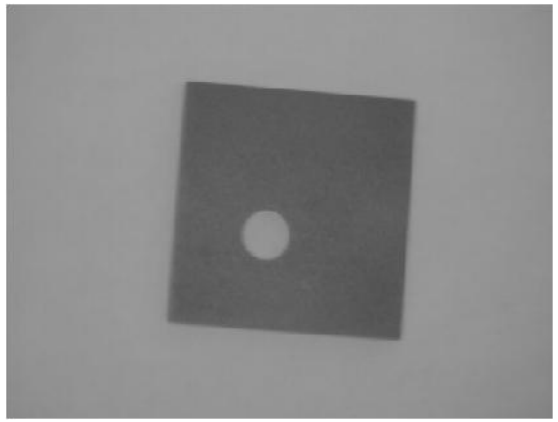
- **Given an inherently-binary image, its actual binarization is typically accomplished by analysing the gray-level histogram. Hence, we start by defining this useful and widespread function.**

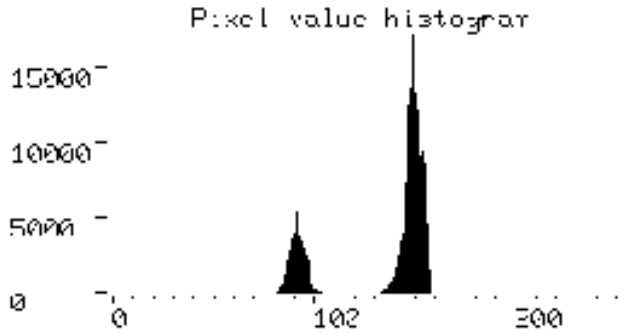
- **The gray-level histogram of an image is a function associating to each gray-level the number of pixels in the image taking that level.**

- **Computing the histogram is straightforward: we define a vector (H) having as many elements as the number of gray-levels, initialize all elements to zero and then scan the image (I) to increment the element of the vector corresponding to the gray-level of the current pixel (p):**

$$\mathbf{p} = \begin{bmatrix} u \\ v \end{bmatrix}, \qquad \forall \mathbf{p} \in \mathbf{I}: \quad \mathbf{H}[\mathbf{I}[\mathbf{p}]] + +$$

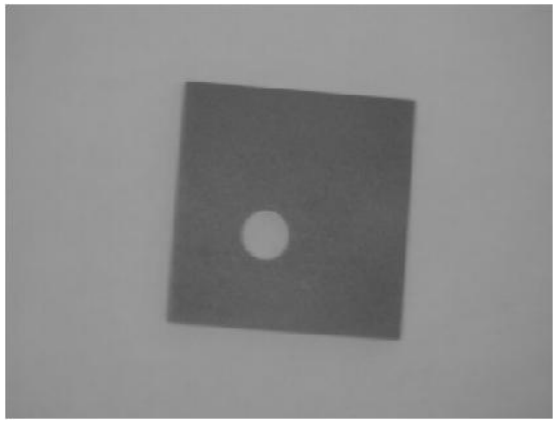
The gray-level histogram (2)

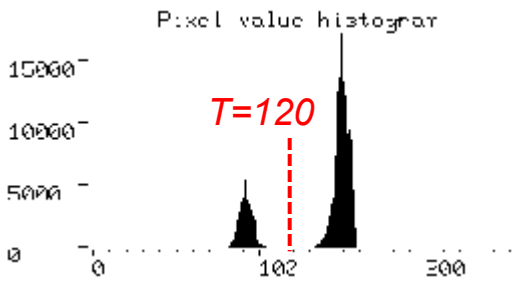


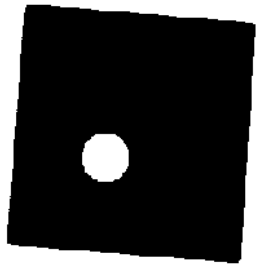


- **The histogram provides often useful information on image content, although it must be highlighted that it does not encode any information related to the spatial distribution of intensities, so that, e.g., if we shuffle randomly the pixels of an image we end up with a new image having exactly the same histogram as before.**
- **Normalization of histogram entries by the total number of pixels yields relative frequencies of occurrence of gray-levels, which can be interpreted as their probabilities. Accordingly, the normalized histogram can be thought of as the *pmf* (probability mass function) of the discrete random variable given by the gray-level of a randomly picked pixel in the image.**

Binarization by Intensity Thresholding







Inherently-binary images exhibit a clearly *bimodal* gray-level histogram, with two well-separated peaks corresponding to foreground and background pixels. Therefore, binarization can be achieved straightforwardly by a means of a *thresholding operator* deploying a suitably chosen threshold, *T*.

Automatic Threshold Selection - Otsu's Algorithm

- **In many practical applications stability over time – especially long spans- of the lighting conditions cannot be guaranteed. Such applications mandate a robust, though computationally more demanding, approach whereby an algorithm computes automatically a suitable binarization threshold in each image under analysis.**
- **A principled and effective automatic threshold selection algorithm is due to Otsu. The key intuition is to segment the image into two maximally homogeneous regions. Accordingly, the *optimal* threshold is chosen so as to *minimize* across the gray-level range the so-called *within-group variance* of the resulting regions, such an indicator measuring how spread turn out region intensities upon binarization by a given gray-level.**

- Let us 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

  $p(i) = h(i)/N$: probability of gray-level i $\left(\sum_{i=1}^{L} p(i) = 1\right)$

- Accordingly, the mean, μ, and variance, σ^2, of the pmf associated with image gray-levels can be expressed as:

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

Otsu's Algorithm

- Any threshold value, t, would split pixels into two disjoint regions whose associated pmfs have mean and variance given by:

$$\mu_1(t) = \sum_{i=1}^{t} i\, p(i)/q_1(t) \qquad \sigma_1^2(t) = \sum_{i=1}^{t} (i - \mu_1(t))^2\, p(i)/q_1(t)$$

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

with

$$q_1(t) = \sum_{i=1}^{t} p(i) \qquad q_2(t) = \sum_{i=t+1}^{L} p(i)$$

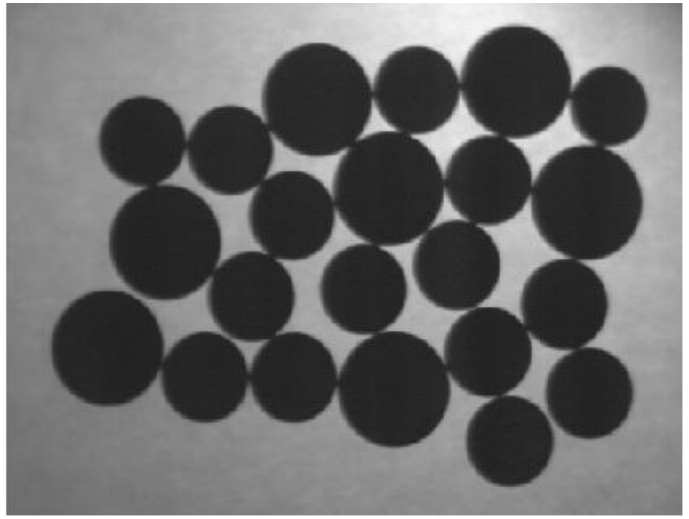
- The Within-group Variance of the two regions is defined as the weighted sum of their variances:

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

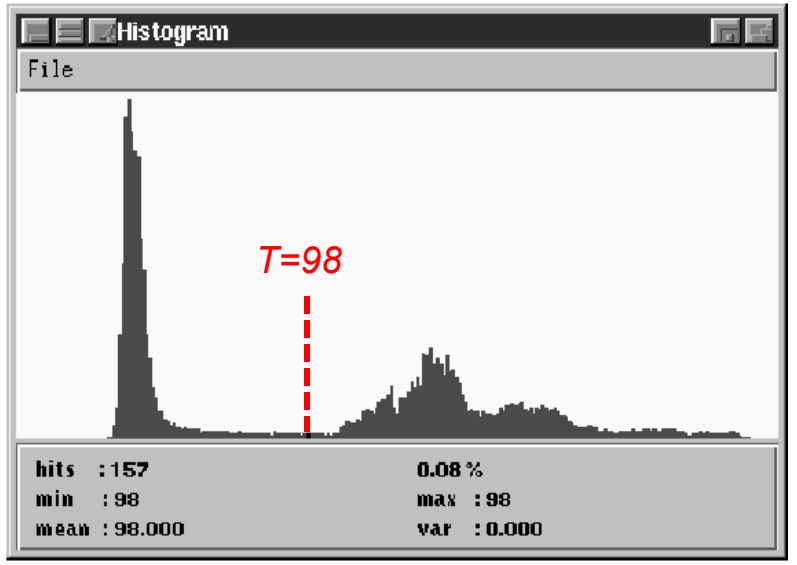
OpenCV: cv2.threshold(...cv2.THRESH_BINARY+cv2.THRESH_OTSU)

Examples (1)

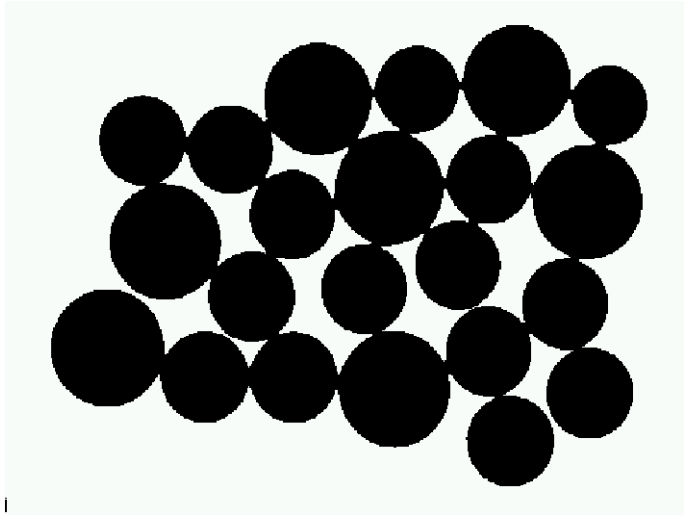
Input Image



Gray-level Histogram



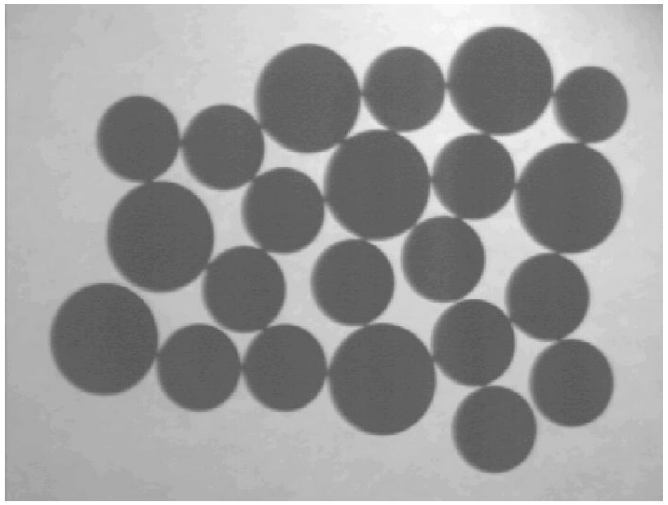
Binarized Image

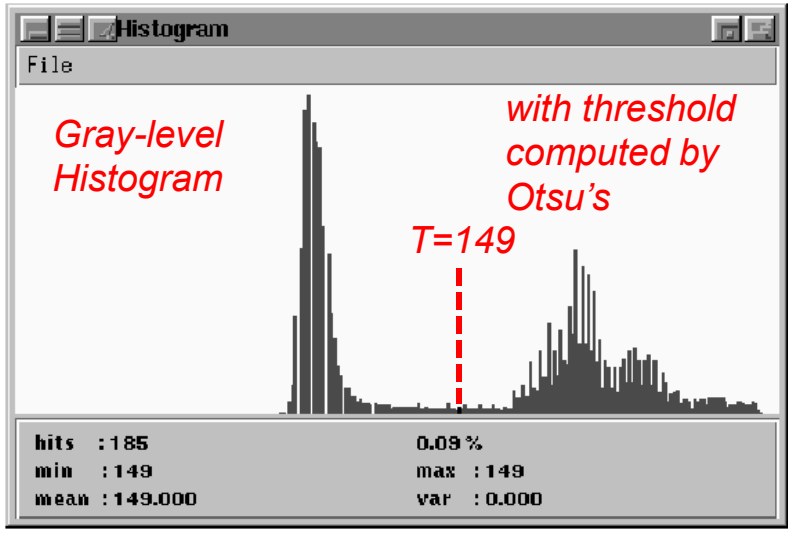


with threshold computed by Otsu's

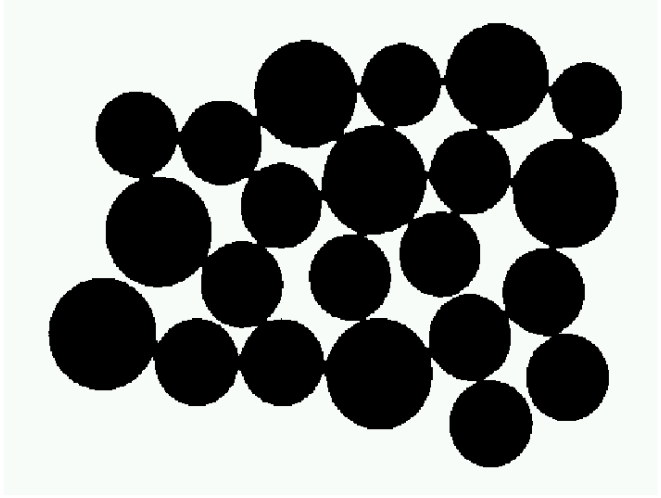
Examples (2)

Brighter Input Image

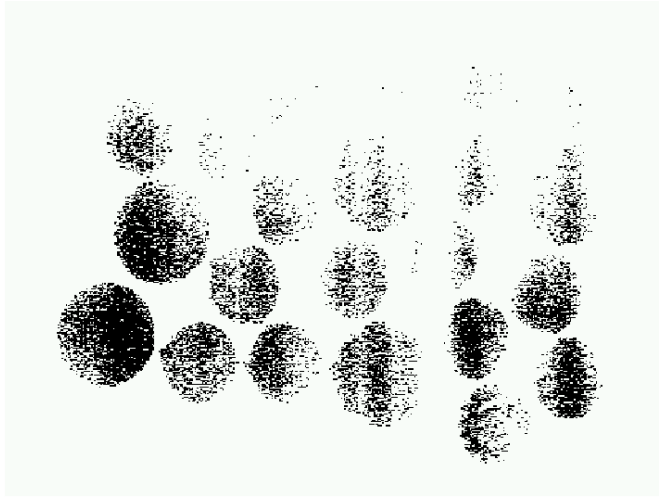




Binarized Image (T=149)

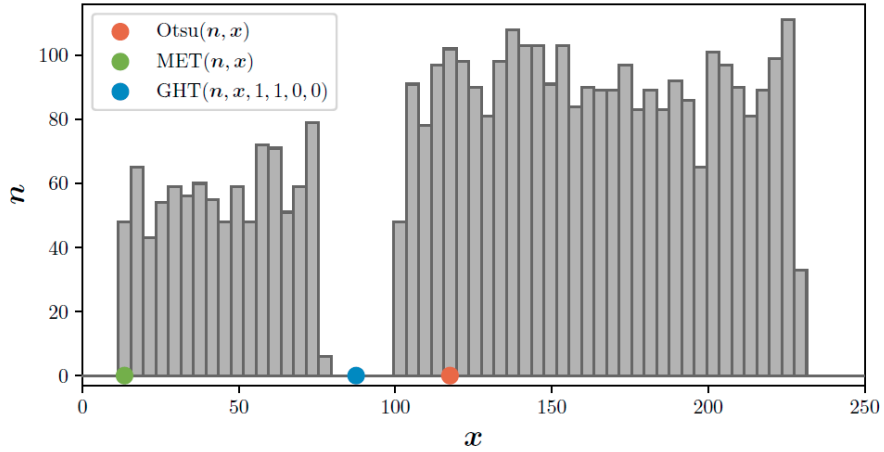


Binarized Image (T=98)



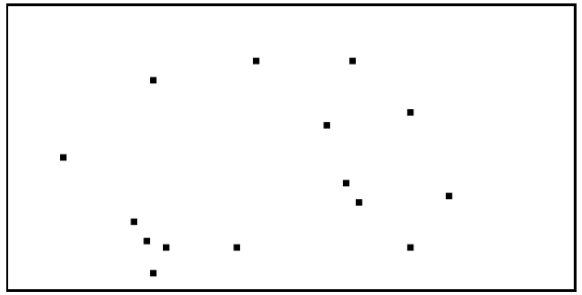
Revisiting Otsu's

(a) An input image I

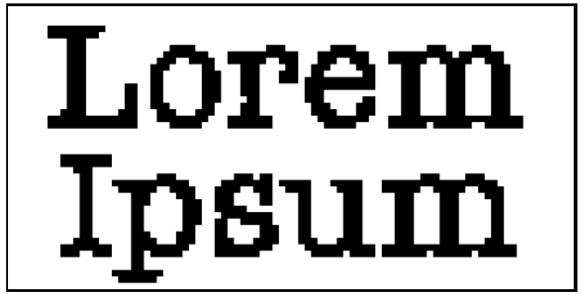


(b) $(\boldsymbol{n}, \boldsymbol{x}) = \text{hist}(\text{I})$

(c) $\text{I} > \text{Otsu}(\boldsymbol{n}, \boldsymbol{x})$



(d) $\text{I} > \text{MET}(\boldsymbol{n}, \boldsymbol{x})$



(e) $\text{I} > \text{GHT}(\boldsymbol{n}, \boldsymbol{x}; 1, 1, 0, \cdot)$

Jonathan T. Barron "A Generalization of Otsu's Method and Minimum Error Thresholding", ECCV 2020.

Adaptive Thresholding

- **Any global thresholding method relies on the assumption of uniform lighting across the scene. Should this assumption be violated, e.g. due to shadows, an adaptive (i.e. *spatially varying*) threshold ought be employed in case one wishes to binarize the image.**

- **Usually, adaptive thresholding methods compute a binarization threshold at each image pixel ($T \rightarrow T(x,y)$) based on the intensities within a small neighbourhood (such as e.g. 5×5, 7×7, 9×9..). However, too small a neighbourhood might lack either background of foreground pixels, which would imply segmentation errors unless the issue is dealt with explicitly.**

- **For the sake of efficiency, rather simple operators, such as the *Mean, Gaussian Mean* or the *Median*, are typically adopted to compute the threshold value at each pixel.**

OpenCV: cv2.adaptiveThreshold

Colour-based Segmentation

- In several applications, the sought-for objects exhibit a known colour quite different from that of background structures (e.g. "*blue screen*"). Hence, it is reasonable to try segmenting-out foreground from background based on colour information.

- Let us denote a pixel's colour as: $\mathbf{I}(p) = \begin{bmatrix} I_r(p) \\ I_g(p) \\ I_b(p) \end{bmatrix}$

- Then, segmentation can be achieved by computing and thresholding the distance (e.g. Euclidean) between each pixel's colour and the reference background (foreground) colour μ :

$$\Rightarrow \quad \forall p \in \mathbf{I}: \quad \begin{cases} d(\mathbf{I}(p), \boldsymbol{\mu}) \le T \rightarrow O(p) = B \quad (F) \\ d(\mathbf{I}(p), \boldsymbol{\mu}) > T \rightarrow O(p) = F \quad (B) \end{cases}$$

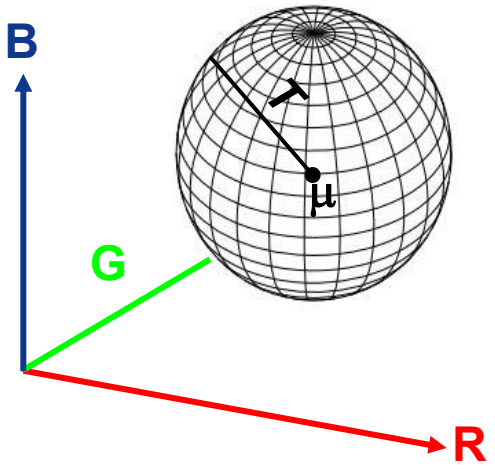
$$d(\mathbf{I}(p), \boldsymbol{\mu}) = \left((I_r(p) - \mu_r)^2 + (I_g(p) - \mu_g)^2 + (I_b(p) - \mu_b)^2 \right)^{\frac{1}{2}}$$

Estimating the Reference Colour

• It is thus necessary to know the reference colour, μ , which can be conveniently estimated ("learned") from one or more *training images*.

• For example, modelling the colour of a background (foreground) pixel as a multivariate random variable (i.e. a 3D random vector), the reference colour can be taken to be the mean (expected value) over the available training samples ($\mathbf{I}(p_k)$, $k=1..N$):

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



Accordingly, segmentation consists in classifying as background (foreground) all pixels lying within a 3D sphere of the RGB colour space centred at μ and having radius as large as T.

Mahalanobis Distance (1)

- **A far richer probabilistic characterization of colour distribution among foreground pixels can be obtained estimating from training samples not only the mean but also the covariance matrix:**

$$\Sigma = \begin{pmatrix} \sigma^2_{rr} & \sigma^2_{rg} & \sigma^2_{rb} \\ \sigma^2_{gr} & \sigma^2_{gg} & \sigma^2_{gb} \\ \sigma^2_{br} & \sigma^2_{bg} & \sigma^2_{bb} \end{pmatrix} \quad \Rightarrow \quad \begin{cases} \sigma^2_{ij} = \dfrac{1}{N}\displaystyle\sum_{k=1}^{N}\left(I_i(p_k)-\mu_i\right)\left(I_j(p_k)-\mu_j\right) \\ i,j \in \{r,g,b\} \end{cases}$$

- **Recalling that the Euclidean can also be written as :**

$$d\left(\mathbf{I}(p),\boldsymbol{\mu}\right) = \left(\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)^T\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)\right)^{\frac{1}{2}}$$

the *Mahalanobis Distance* is given by :

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right) = \left(\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)^T\Sigma^{-1}\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)\right)^{\frac{1}{2}}$$

Mahalanobis Distance (2)

- **To understand the difference between the two distances, it is worth considering the case of diagonal covariance matrix (i.e. independent components in *I(p)*)**

$$\Sigma = \begin{pmatrix} \sigma^2_{rr} & 0 & 0 \\ 0 & \sigma^2_{gg} & 0 \\ 0 & 0 & \sigma^2_{bb} \end{pmatrix} \rightarrow \Sigma^{-1} = \begin{pmatrix} 1/\sigma^2_{rr} & 0 & 0 \\ 0 & 1/\sigma^2_{gg} & 0 \\ 0 & 0 & 1/\sigma^2_{bb} \end{pmatrix}$$

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right) = \left(\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)^T \Sigma^{-1}\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)\right)^{\frac{1}{2}}$$

$$\begin{bmatrix} \left(I_r(p)-\mu_r\right)/\sigma^2_{rr} \\ \left(I_g(p)-\mu_g\right)/\sigma^2_{gg} \\ \left(I_b(p)-\mu_b\right)/\sigma^2_{bb} \end{bmatrix}$$

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right) = \left(\frac{\left(I_r(p)-\mu_r\right)^2}{\sigma^2_{rr}} + \frac{\left(I_g(p)-\mu_g\right)^2}{\sigma^2_{gg}} + \frac{\left(I_b(p)-\mu_b\right)^2}{\sigma^2_{bb}}\right)^{\frac{1}{2}}$$

Mahalanobis Distance (3)

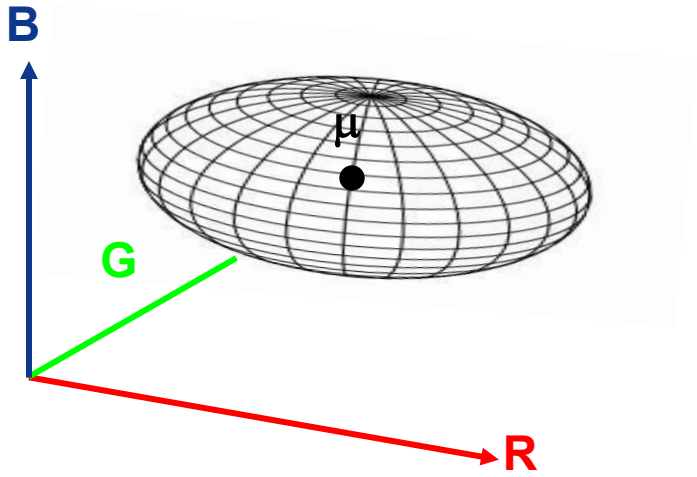
- **Contrary to the Euclidean distance, the Mahalanobis distance weighs unequally the differences along the components of the random vector, in particular according to inverse proportionality to the learned variances. As such, the more spread has been learned to be a component, the less the difference along it will contribute to the overall distance.**

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

Mahalanobis Distance (4)

Which region of the RGB space determines now the segmentation ?

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right)^2 = \left(\frac{\left(I_r(p)-\mu_r\right)^2}{\sigma_{rr}^2} + \frac{\left(I_g(p)-\mu_g\right)^2}{\sigma_{gg}^2} + \frac{\left(I_b(p)-\mu_b\right)^2}{\sigma_{bb}^2} \right) \leq T^2$$



Ellipsoid centred at : $\boldsymbol{\mu} = \begin{bmatrix} \mu_r \\ \mu_g \\ \mu_b \end{bmatrix}$

With axes aligned to coordinate axes and semi-axes lengths given by:

$$\mathbf{L} = \begin{bmatrix} L_r \\ L_b \\ L_g \end{bmatrix} = T \begin{bmatrix} \sigma_{rr} \\ \sigma_{bb} \\ \sigma_{gg} \end{bmatrix}$$

Mahalanobis Distance (4)

- The interpretation of the Mahalanobis distance which has been given in case of diagonal covariance matrix is of general validity.
- Indeed, the covariance matrix can always been diagonalized by a rotation of the coordinate axes. Thus, in the new coordinate system the Mahalanobis distance is a weighted sum of the contributions along the new axes, with weights being inversely proportional to the variances along the new axes.

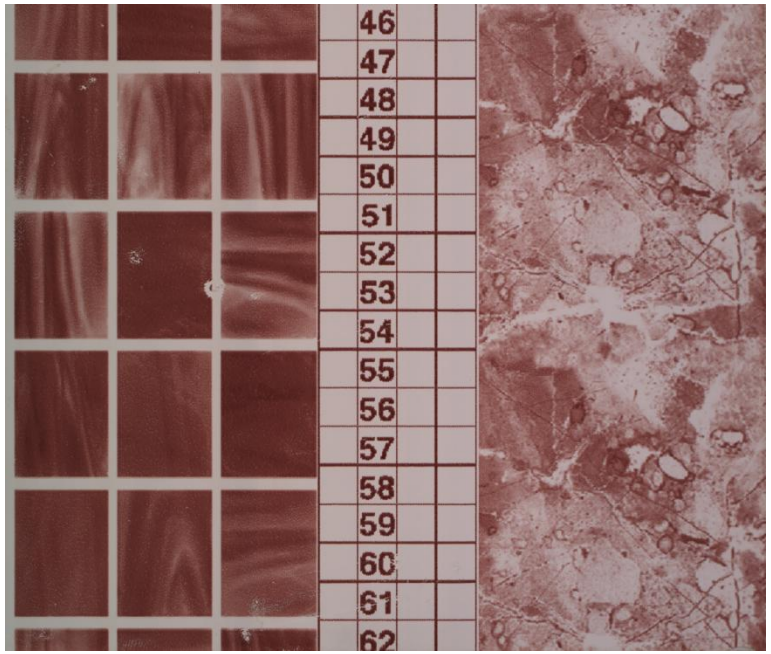
- This is due to the *EigenValue (Spectral) Decomposition* of any real and symmetric matrix (Σ):

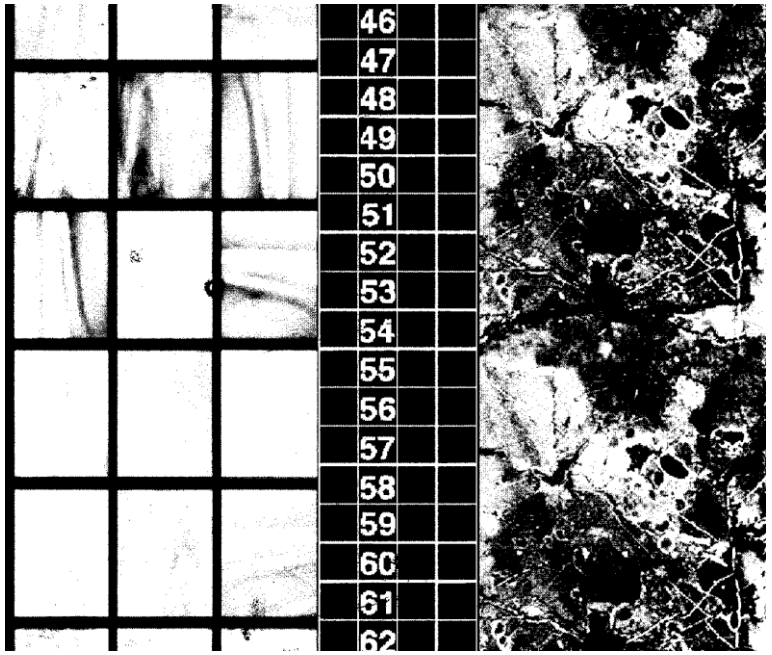
$$\boldsymbol{\Sigma} = \mathbf{R}\mathbf{D}\mathbf{R}^T : \quad \mathbf{R} = \left(\mathbf{e}_1\ \mathbf{e}_2\ \mathbf{e}_3\right), \quad \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

Where $\mathbf{e}_i$ are the (orthonormal) *eigenvectors* of Σ, λ_i the corresponding *eigenvalues* and R^T the rotation matrix to transform the data into a new coordinate system having axes aligned to the eigenvectors.
Upon rotation by R^T the new covariance matrix becomes D, so that the eigenvalues of Σ represent the variances along the direction of the eigenvectors.

(More details concerning the case of general covariance matrix can be found in Appendix 1)

Example: Tile Inspection





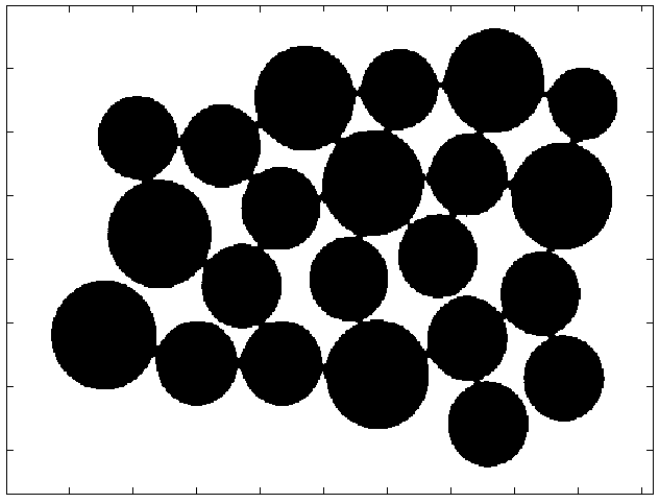
$$\mathbf{I}(p) \quad \rightarrow \quad O(p) = \begin{cases} 255, & d_M\left(\mathbf{I}(p), \boldsymbol{\mu}\right) \le T \\ 0, & d_M\left(\mathbf{I}(p), \boldsymbol{\mu}\right) > T \end{cases}$$

Example: Fruit Inspection

$$\mathbf{I}(p) \quad \rightarrow \quad O(p) = \begin{cases} \mathbf{I}(p), & d_M\left(\mathbf{I}(p), \boldsymbol{\mu}\right) \leq T \\ 0, & d_M\left(\mathbf{I}(p), \boldsymbol{\mu}\right) > T \end{cases}$$

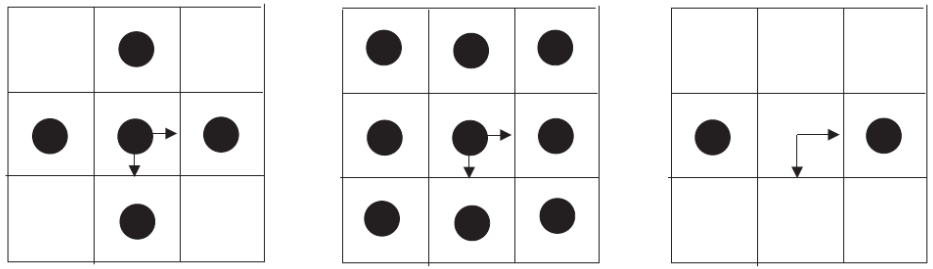
Binary Morphology

- **Binary Morphology operators are simple though effective tools to improve or analyse binary images, in particular those achieved by any kind of foreground/background segmentation (e.g. based on intensity or colour).**



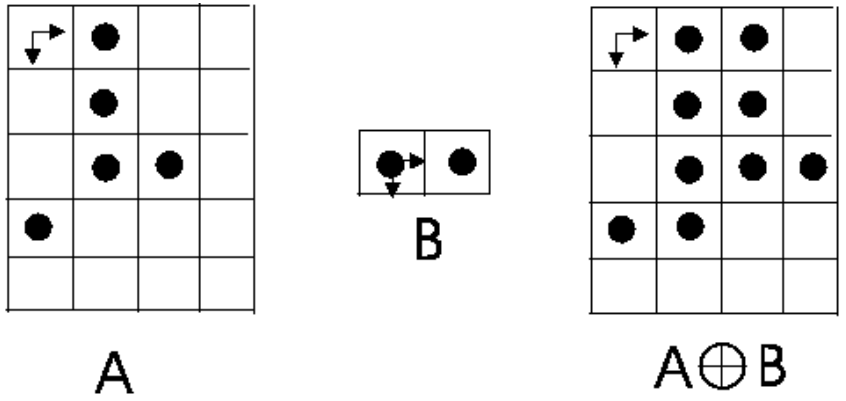
- **These operators manipulate the *binary image (A)* by a small "kernel" referred to as *structuring element (B)***

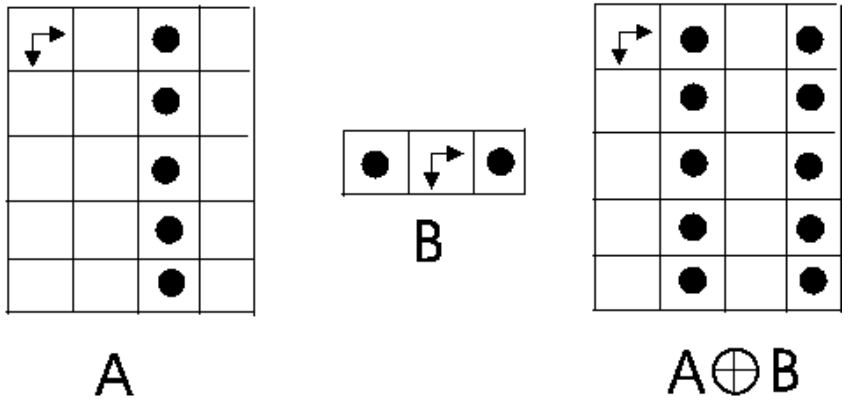
e.g.:



Dilation (Minkowsky Sum)

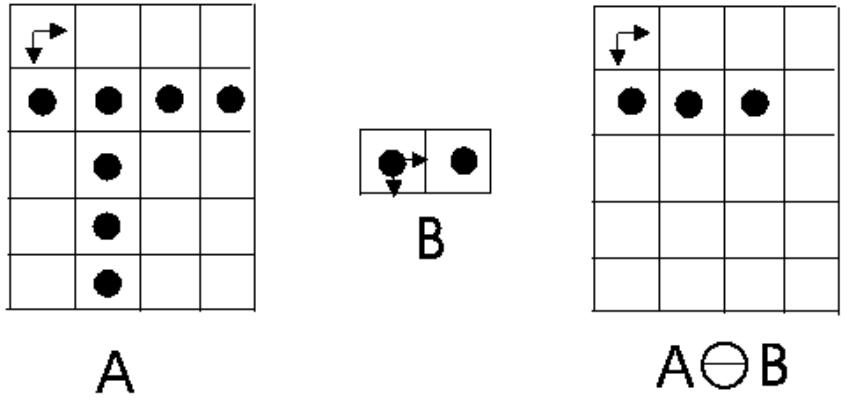
The output image is obtained by sliding *B* at each black (i.e. foreground) pixel in *A*.

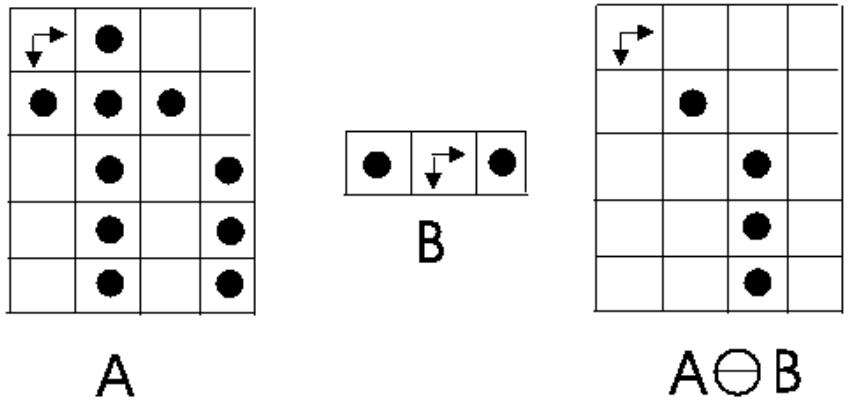




Erosion (Minkowsky Subtraction)

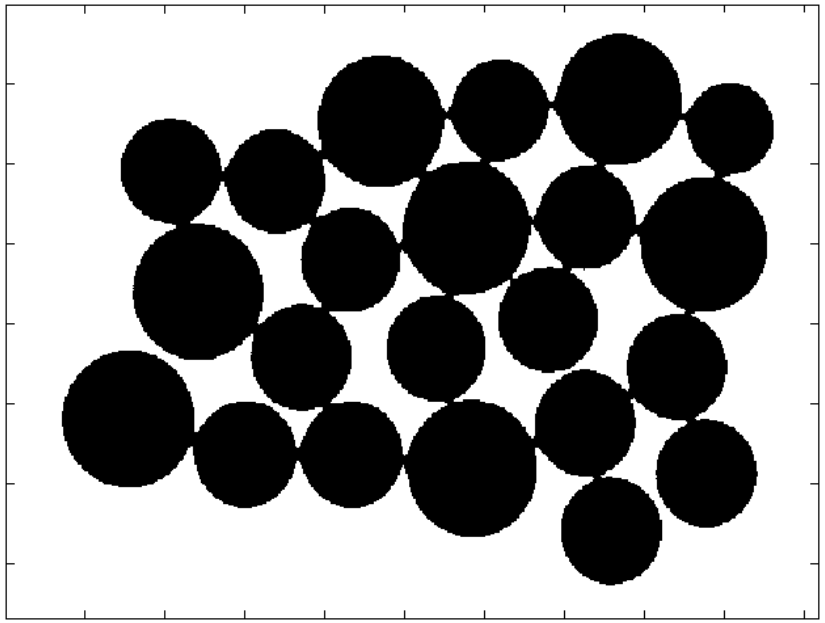
The output image is obtained by sliding *B* across the whole *A* and keeping only those pixels where *B* fits into the foreground of *A*.



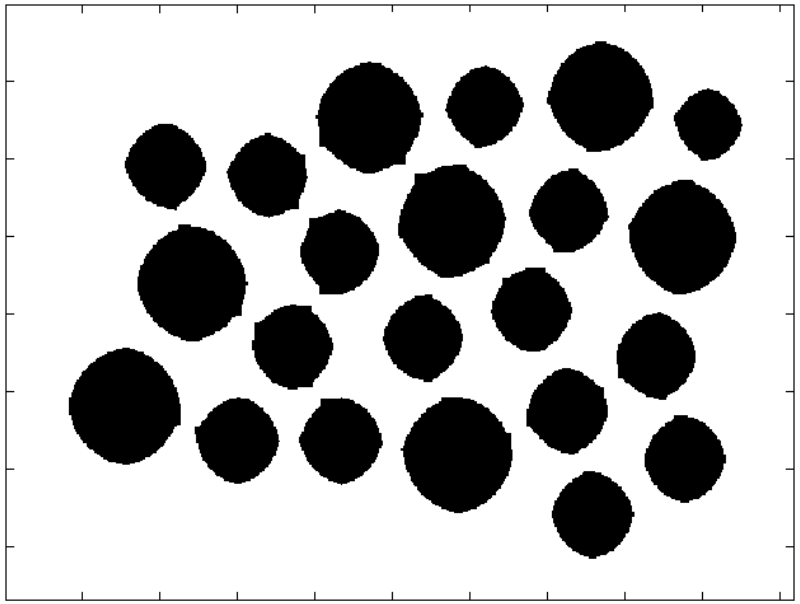


Erosion – Example (1)

Wrongly connected objects yielded by segmentation

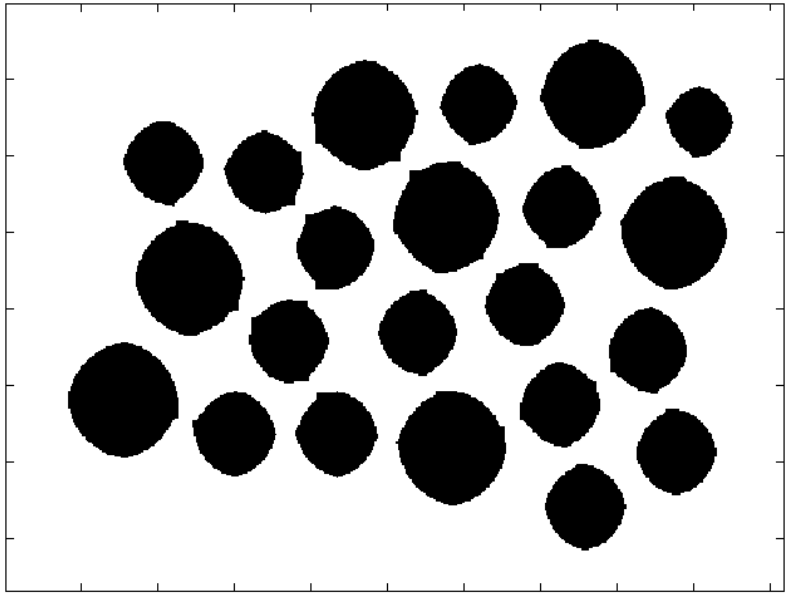


Objects can be split (e.g. to allow counting them correctly) by 5 successive erosions with 3×3 square

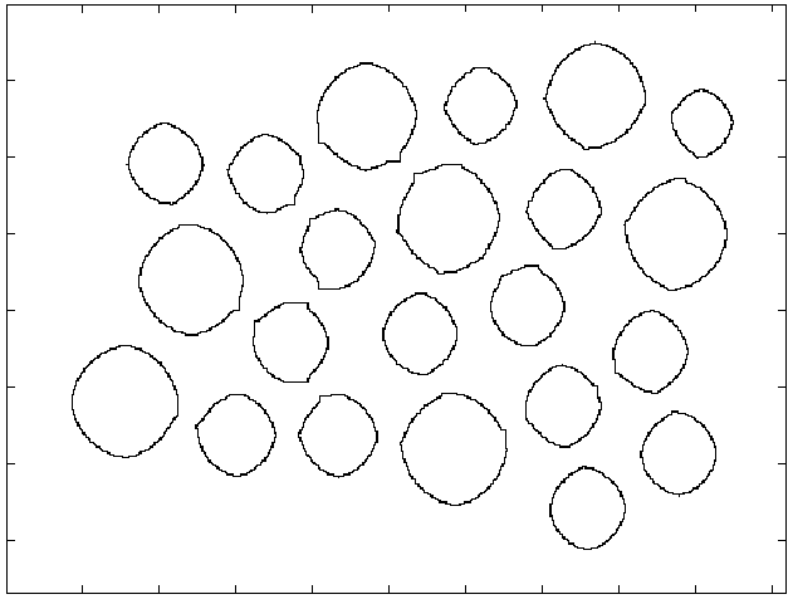


Erosion - Example (2)

Erosion by a 3x3 square followed by subtraction of the eroded image from the original one yields the contours of the foreground regions, i.e. foreground pixels adjacent to background ones.



Input binary image

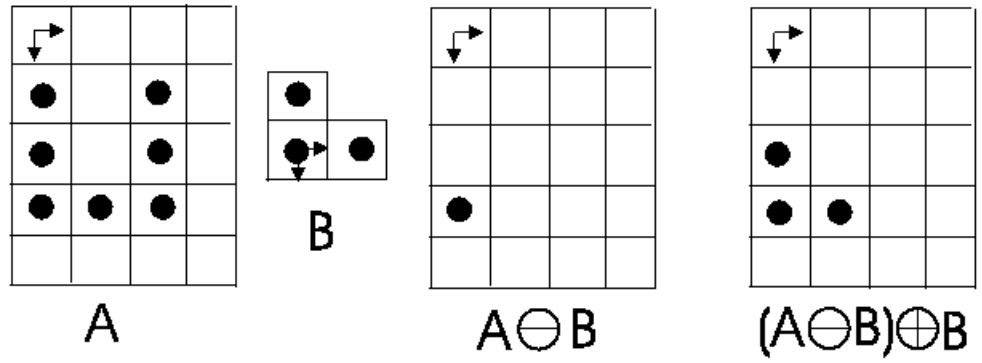


Contours extracted by erosion followed by subtraction

Opening and Closing

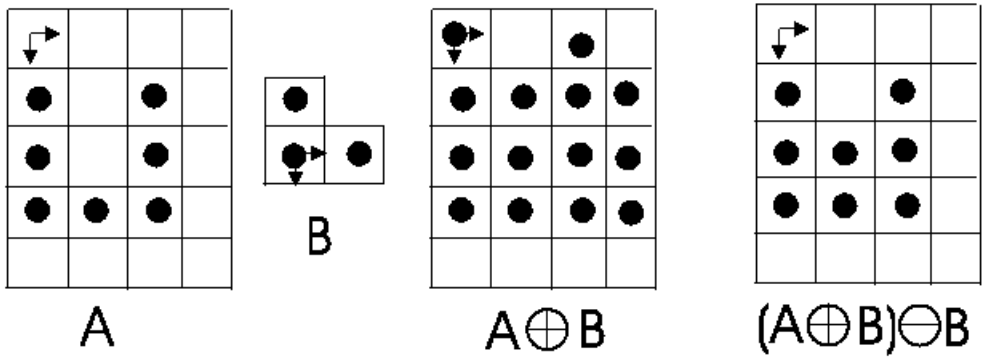
- **Erosion followed by Dilation is known as Opening:**

$$A \circ B = (A \ominus B) \oplus B$$



- **Dilation followed by Erosion is known as Closing:**

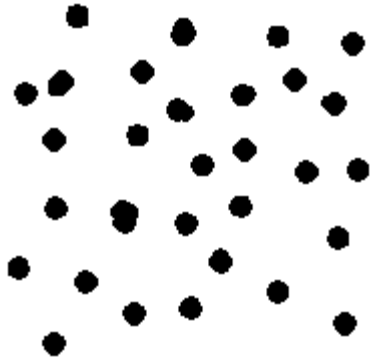
$$A \bullet B = (A \oplus B) \ominus B$$



"Smarter" operators to remove selectively from either foreground (opening) or background (closing) those parts that do not match exactly the structuring element. Other parts are left unchanged. Indeed, closing is equivalent to opening the complement of the image by a reflected structured element.

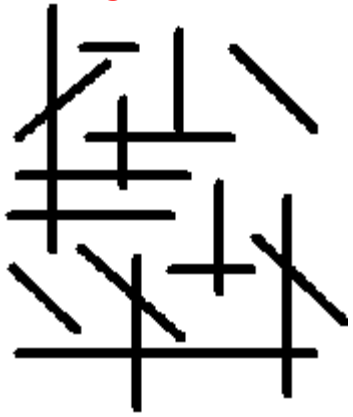
Examples (1)

Input binary image



Opening by a circular structuring element (diameter=11 pixels) allows detecting circular objects.

Input binary image



Opening by a vertical structuring element (9×3 pixels) …

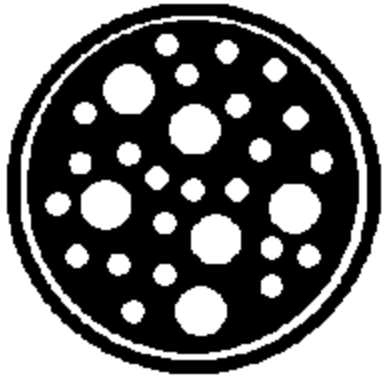
.. allows detecting vertical bars.

Opening by an horizontal structuring element (3×9 pixels)…

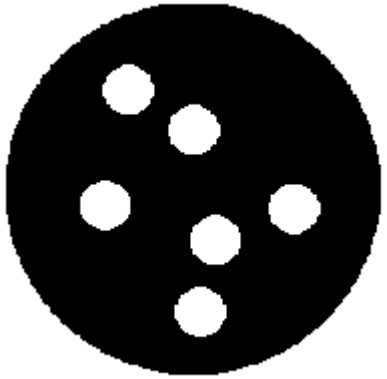
…allows detecting horizontal bars.

Examples (2)

Input binary image



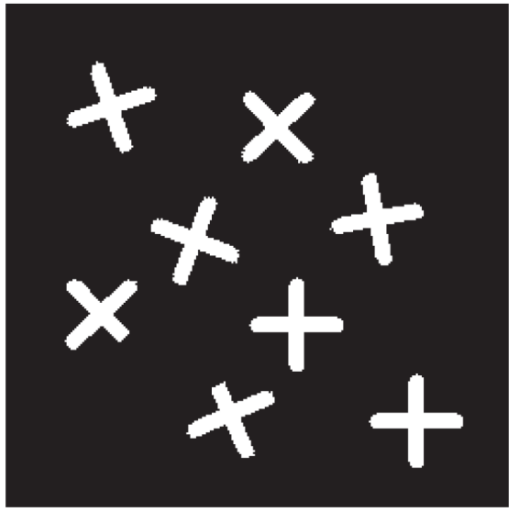
Closing by a circle smaller than the big holes and larger than the small ones …

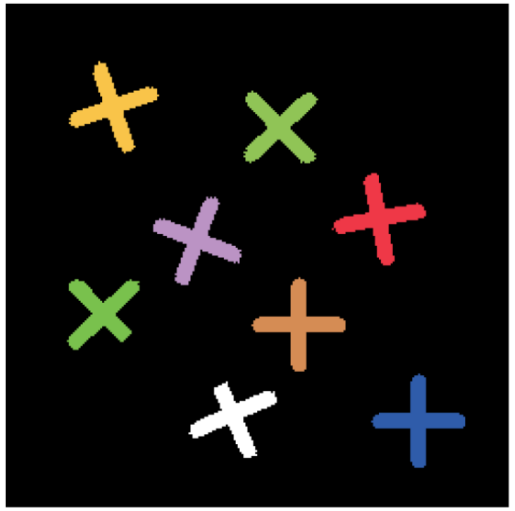


…removes the small holes (and the external thin circular contour alike).

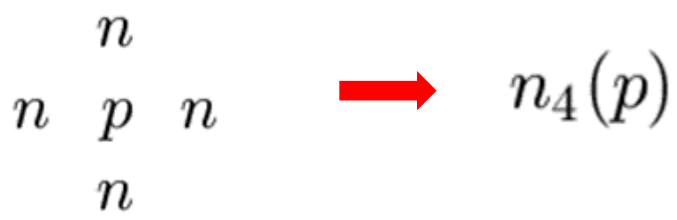
Connected Components Labeling

Computation whereby foreground pixels belonging to different *connected components* (aka Blobs) are given different labels, while background pixels are left unaffected (e.g. they may keep the previous label or a new one).



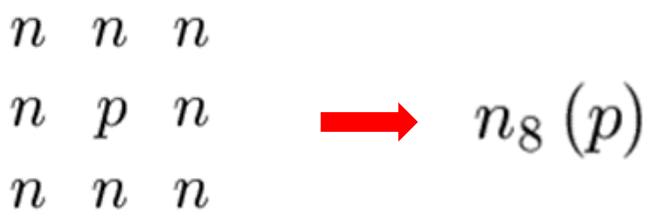


Connectivity & Connected Components (1)



$$n_4(p)$$

***4-neighbourhood* of *p*, i.e. the set of neighbours of *p* according to the notion of *4-way connectivity*.**



$$n_8(p)$$

***8-neighbourhood* of *p*, i.e. the set of neighbours of *p* according to the notion of *8-way connectivity*.**

- **A *path* from pixel *p* to pixel *q* is a sequence of pixels $p=p_1, p_2,\ldots,p_n=q$ such that p_i and p_{i+1} are neighbours according to the chosen connectivity.**

Connectivity & Connected Components (2)

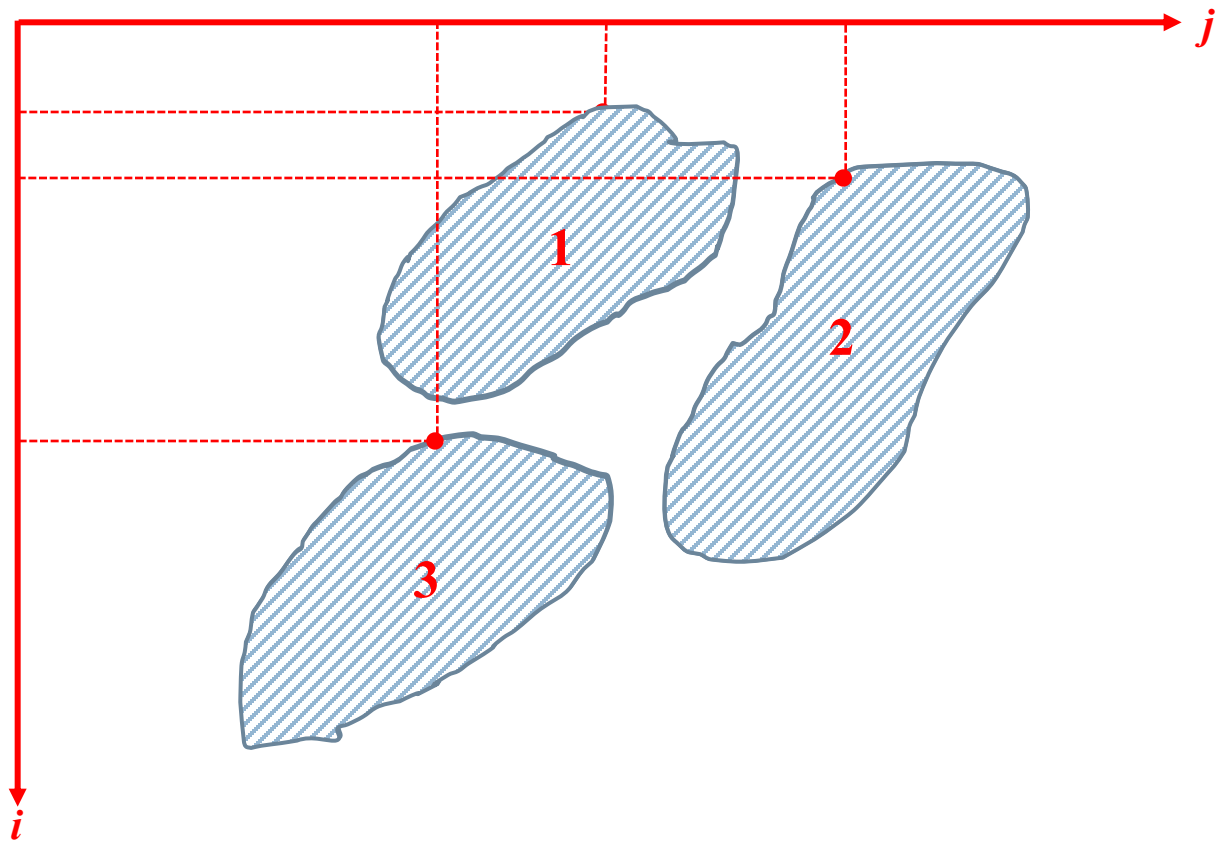
- **A set of pixels, *R*, is said to be a *connected region* if for any two pixels *p,q* in *R* there exists a path contained in *R* between *p* and *q*. Depending on the notion of connectivity, *R* is said to be a *4-connected* or *8-connected* region.**

- **A set of pixels is said to be a *connected foreground (background) region* if it is a connected region and includes *foreground* (*background*) pixels only.**

A *connected component* of a binary image is a <u>maximal</u> connected foreground region.

Labeling by Flood-fill

Simple and intuitive approach, though "real" algorithms are quite more complex and vastly more efficient (see, *e.g.*, Appendix 2).



A new label is propagated throughout a connected component starting from a "seed" pixel. Propagation is typically carried out by a iterative procedure which requires scanning the image multiple times.

Recent Algorithms and Benchmarks

C. Grana, D. Borghesani, R. Cucchiara “Optimized Block-based Connected Components Labeling with Decision Trees”. IEEE Transactions on Image Processing, 2010.

F. Bolelli, S. Allegretti, L. Baraldi, C. Grana “Spaghetti Labeling: Directed Acyclic Graphs for Block-Based Connected Components Labeling”, IEEE Transactions on Image Processing, 2019.

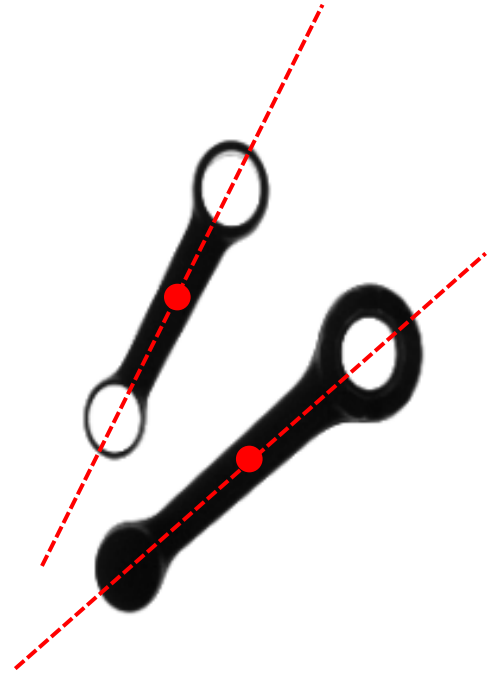
Used in OpenCV

F. Bolelli, M. Cancilla, L. Baraldi, C. Grana “Toward reliable experiments on the performance of connected components labeling algorithms”, Journal of Real-Time Image Processing 2020.

F. Bolelli, S. Allegretti, L. Lumetti, C. Grana “A State-of-the-Art Review with Code about Connected Components Labeling on GPUs”, IEEE Transactions on Parallel and Distributed Systems, 2024.

Blob Analysis – Area, Barycentre and Orientation

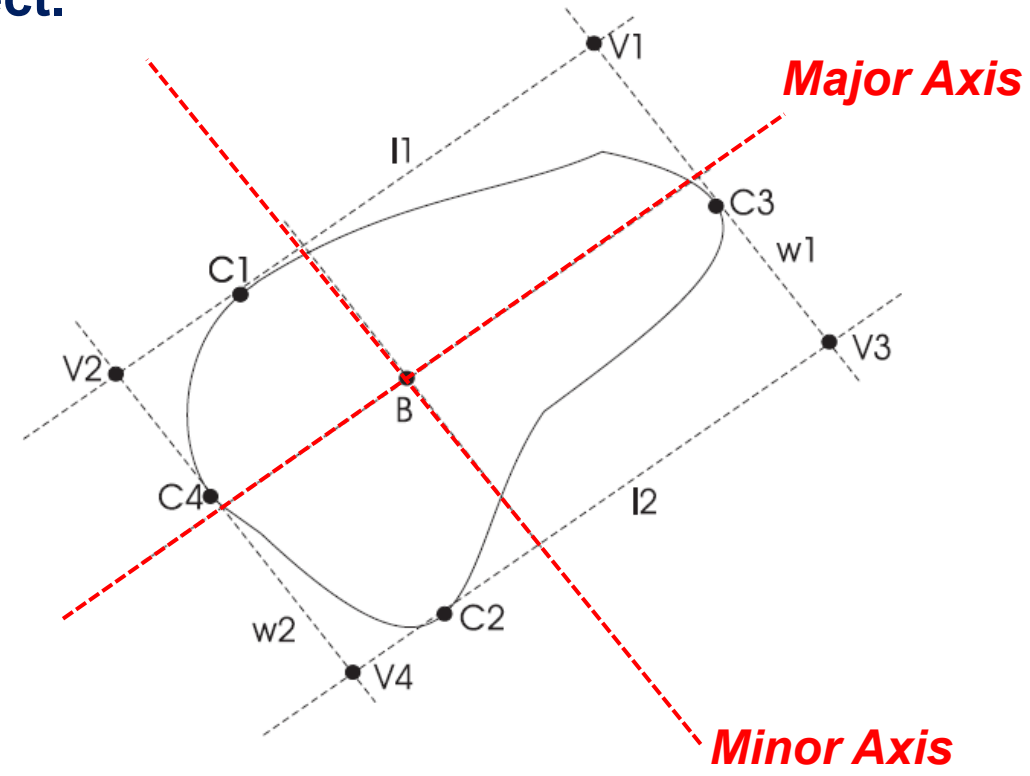
As already pointed out, once *connected components*, aka *blobs*, have been identified, the next step consists in computing *features* associated with each such region. This step is usually referred to as *blob analysis* The main computer vision libraries provide the user with blob analysis tools to extract rich sets of features. The simplest and most commonly deployed features are *area, barycentre* and orientation.



Blob Analysis - Oriented Bounding Box

Given the *major and minor axes* one can compute the *Oriented Bounding Box* (aka MER: *Minimum Enclosing Rectangle*) aligned to the object.

Purposely, the four points laying at maximum distance on opposite sides of the two axes need to be determined (*C1,C2,C3,C4*). Then, one finds the lines through *C1,C2* parallel to the major axis and those through *C3,C4* parallel to the minor axis.



Eventually, the vertexes of the oriented bounding box (*V1,V2,V3,V4*) are given by the intersections between these lines.

Blob Analysis - Features from OBB

- **Length (*L*) and width (*W*):**

L: extent of the object along the major axis

$$L = d_{V_1V_2} = d_{V_3V_4} = \sqrt{(i_{V_1} - i_{V_2})^2 + (j_{V_1} - j_{V_2})^2} = \sqrt{(i_{V_3} - i_{V_4})^2 + (j_{V_3} - j_{V_4})^2}$$

W: extent of the object along the minor axis

$$W = d_{V_1V_3} = d_{V_2V_4} = \sqrt{(i_{V_1} - i_{V_3})^2 + (j_{V_1} - j_{V_3})^2} = \sqrt{(i_{V_2} - i_{V_4})^2 + (j_{V_2} - j_{V_4})^2}$$

- **Elongatedness:** $E = \dfrac{L}{W}$

- **Rectangularity:** $R = \dfrac{A}{LW}$ A: object's area, LW: area of the oriented MER

- **Ellipticity:** $E = \dfrac{A}{A_{LW}}, \qquad A_{LW} = \dfrac{\pi}{4}LW$

A: object's area, A_{LW}: area of the ellipse oriented exactly as the object and having major and minor axis lengths equal to L and W respectively.

OpenCV: cv2.connectedComponentsWithStats

Appendix 1 - Mahalanobis distance with general covariance matrix (1)

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right)^2=\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)^T\Sigma^{-1}\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)$$

$$\boldsymbol{\Sigma}=\mathbf{RDR}^T\ \rightarrow\ \boldsymbol{\Sigma}^{-1}=\left(\mathbf{RDR}^T\right)^{-1}=\left(\mathbf{R}^T\right)^{-1}\left(\mathbf{RD}\right)^{-1}=\mathbf{RD}^{-1}\mathbf{R}^T$$

Rotation bringing I(p) and μ into the new coordinate system with axes aligned to the eigenvectors of Σ

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right)^2=\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)^T\mathbf{RD}^{-1}\mathbf{R}^T\left(\mathbf{I}(p)-\boldsymbol{\mu}\right)$$

$$\left[\mathbf{I}^T\mathbf{e}_1-\boldsymbol{\mu}^T\mathbf{e}_1\quad\mathbf{I}^T\mathbf{e}_2-\boldsymbol{\mu}^T\mathbf{e}_2\quad\mathbf{I}^T\mathbf{e}_3-\boldsymbol{\mu}^T\mathbf{e}_3\right]$$

$$=\left[\mathbf{e}_1{}^T\mathbf{I}-\mathbf{e}_1{}^T\boldsymbol{\mu}\quad\mathbf{e}_2{}^T\mathbf{I}-\mathbf{e}_2{}^T\boldsymbol{\mu}\quad\mathbf{e}_3{}^T\mathbf{I}-\mathbf{e}_3{}^T\boldsymbol{\mu}\right]$$

$$\left(\mathbf{I}'(p)-\boldsymbol{\mu}'\right)^T$$

$$\left[I_{r'}(p)-\mu_{r'}\quad I_{g'}(p)-\mu_{g'}\quad I_{b'}(p)-\mu_{b'}\right]$$

$$(\mathbf{I}'(p)-\boldsymbol{\mu}')=\begin{bmatrix}\mathbf{e}_1{}^T\mathbf{I}\\ \mathbf{e}_2{}^T\mathbf{I}\\ \mathbf{e}_3{}^T\mathbf{I}\end{bmatrix}-\begin{bmatrix}\mathbf{e}_1{}^T\boldsymbol{\mu}\\ \mathbf{e}_2{}^T\boldsymbol{\mu}\\ \mathbf{e}_3{}^T\boldsymbol{\mu}\end{bmatrix}$$

$$\mathbf{D}^{-1}\left(\mathbf{I}'(p)-\boldsymbol{\mu}'\right)=\begin{bmatrix}\frac{I_{r'}(p)-\mu_{r'}}{\lambda_1}\\ \frac{I_{g'}(p)-\mu_{g'}}{\lambda_2}\\ \frac{I_{b'}(p)-\mu_{b'}}{\lambda_3}\end{bmatrix}$$

Appendix 1 - Mahalanobis distance with general covariance matrix (2)

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right)^2 = \begin{bmatrix} I_{r'}(p)-\mu_{r'} & I_{g'}(p)-\mu_{g'} & I_{b'}(p)-\mu_{b'} \end{bmatrix} \begin{bmatrix} \frac{I_{r'}(p)-\mu_{r'}}{\lambda_1} \\ \frac{I_{g'}(p)-\mu_{g'}}{\lambda_2} \\ \frac{I_{b'}(p)-\mu_{b'}}{\lambda_3} \end{bmatrix}$$

$$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right)^2 = \frac{\left(I_{r'}(p)-\mu_{r'}\right)^2}{\lambda_1} + \frac{\left(I_{g'}(p)-\mu_{g'}\right)^2}{\lambda_2} + \frac{\left(I_{b'}(p)-\mu_{b'}\right)^2}{\lambda_3}$$

$d_M\left(\mathbf{I}(p),\boldsymbol{\mu}\right)^2 = T^2$ → ***Ellipsoid with semi-axes of length $T\sqrt{\lambda_i}$ and axes aligned to the eigenvectors of Σ***

Hence, in the general case, the contributions to the overall distance are inversely proportional to the eigenvalues of Σ, which in turn represent the variances of the data along the directions of the eigenvectors.

Appendix 1 - Mahalanobis distance with general covariance matrix (3)

Eventually, we verify here that in the new coordinate system obtained by a rotation through $\mathbf{R}^T$, the new covariance matrix, Σ', is diagonal and its non-null elements are the eigenvalues of Σ.

Given an *n*-dimensional random vector x, with mean and covariance denoted as m_x and C_x respectively, let us consider a generic linear transformation defined through the $n \times n$ matrix A:

$$\mathbf{y} = \mathbf{A}\mathbf{x}$$

It can be shown that the mean and covariance of the transformed data are given by:

$$\mathbf{m}_y = \mathbf{A}\mathbf{m}_x, \quad \mathbf{C}_y = \mathbf{A}\mathbf{C}_x\mathbf{A}^T$$

Therefore, in the considered case :

$$\boldsymbol{\Sigma}' = \mathbf{R}^T\boldsymbol{\Sigma}\mathbf{R} = \mathbf{R}^T\left(\mathbf{R}\mathbf{D}\mathbf{R}^T\right)\mathbf{R} = \left(\mathbf{R}^T\mathbf{R}\right)\mathbf{D}\left(\mathbf{R}^T\mathbf{R}\right) = \mathbf{I}\mathbf{D}\mathbf{I} = \mathbf{D} = \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{pmatrix}$$

with I denoting here the 3×3 identity matrix: $\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$

Appendix 2 - The classical 2-scans labeling algorithm

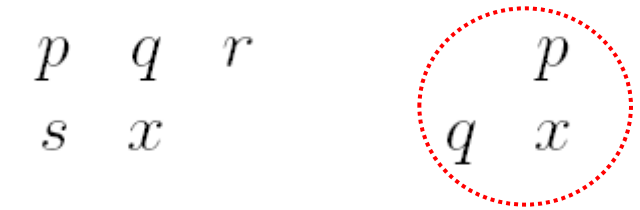
- **By the first scan foreground pixels take temporary labels based on those given to already visited neighbours, which depend on both the chosen connectivity, i.e. *4-way* or *8-way*, and the scan order, e.g. *left-right, top-down*.**

- **Upon the first scan, different blobs have certainly been given different labels, though, depending on shape, this may be the case for connected parts of a single blob too.**

- **Hence, the second scan allows a unique final label to be assigned to those parts belonging to the same blob that had been given different temporary labels by the first scan.**

- **Purposely, *equivalent* temporary labels need to be found between the two scans, so as to assign a unique final label to each of the *equivalence classes* among temporary labels.**

Appendix 2 - First scan (1)



4-way connectivity !

$l_q = l_p = B \rightarrow l_x = Newlabel$

$l_q \neq B,\ l_p = B \rightarrow l_x = l_q$

$l_q = B,\ l_p \neq B \rightarrow l_x = l_p$

$l_q = l_p \neq B \rightarrow l_x = l_q \text{ or } l_x = l_p$

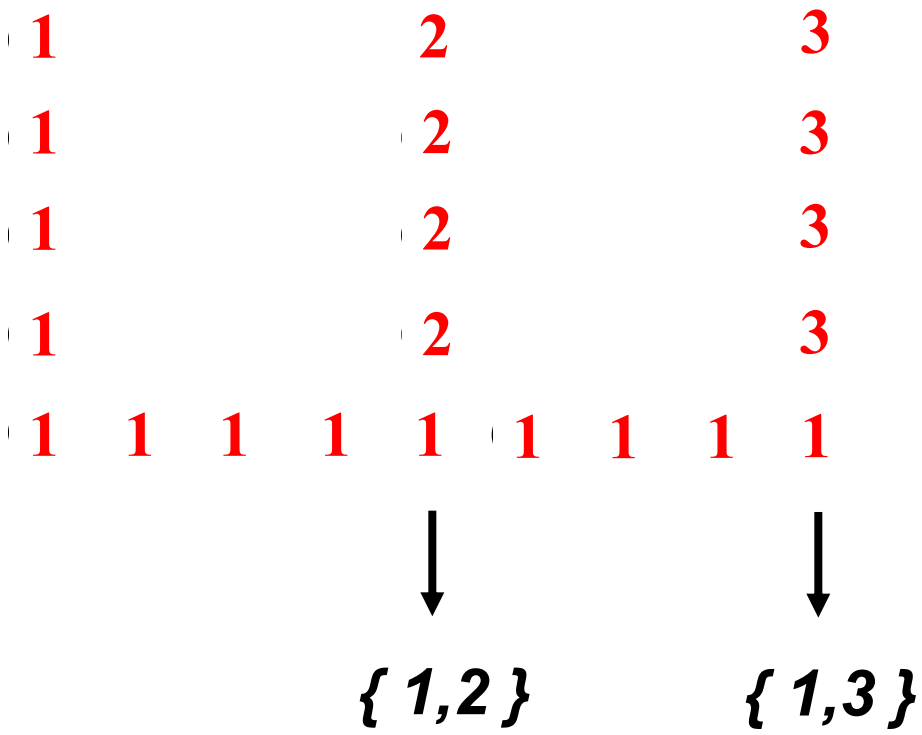
$l_q \neq B,\ l_p \neq B,\ l_q \neq l_p \rightarrow l_x = l_q \text{ or } l_x = l_p,\ \{l_q, l_p\}$

lq and *lp* are equivalent labels !

Appendix 2 - First scan (2)

```
// lp,lq,lx: labels assigned to p,q,x
// B:background, F:foreground.
// FIRST SCAN:
for(i=1; i<NROWS-1; i++)
for(j=1; j<NCOLS-1; j++) {
if (I[i,j]==F) {
lp = I[i-1,j];
lq = I[i,j-1];
if(lp == B && lq == B) {
NewLabel++;
lx = NewLabel;}
else if((lp != lq)&&(lp != B)&&(lq != B)){
// REGISTER EQUIVALENCE (lp,lq)
lx = lq;}
else if(lq != B) lx = lq;
else if(lp != B) lx = lp;
I[i,j] = lx;} }
// FIND EQUIVALENCE CLASSES
// SECOND SCAN
```

Appendix 2 - Example



After the first scan: $\{1,2\}, \{1,3\} \longrightarrow \{1,2,3\} \longrightarrow \{1,2,3\}$

Appendix 2 - Handling equivalences (1)

1. **During the first scan the equivalences found between temporary label pairs are recorded into an *N* × *N* matrix (*N*: number of temporary labels):**

$$B = \begin{array}{c} \\ 1 \\ i \\ N \end{array} \begin{array}{c} \begin{array}{ccccc} 1 & \cdots\cdots & j & \cdots\cdots & N \end{array} \\ \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & 1 \end{bmatrix} \end{array} \rightarrow B[i,j] = \begin{cases} 1, \; if \; \{i,j\} \\ 0, \; otherwise \end{cases}$$

Because label equivalence is an equivalence relation:

$$i = j \rightarrow B[i,j] = 1$$

$$B[i,j] = B[j,i] \;\; \forall \, i,j = 1 \ldots N \quad \rightarrow \quad \textbf{B is symmetric}$$

Appendix 2 - Handling equivalences (2)

2. **After the first scan, B is processed so as to elicit the equivalences among temporary labels implied by virtue of the transitive property:**

$$\{i,j\} \rightarrow \{i,k\} \ \forall \ \{k,j\}$$

$$B[i,k] = B[i,k] \ OR \ B[j,k] \ \forall \ k = 1...N$$

Thus, through a scan by columns, all the equivalences implied by the transitive property can be recorded into B :

```
for (j=1,....N)
 for (i=1....N)
  if (B[i,j]=1)AND(i≠j)
   for(k=1....N)
     B[i,k]=B[i,k] OR B[j,k];
```

Appendix 2 - Handling equivalences (3)

For the sake of example, let us assume the following equivalences to have been found during the first scan:

$$\{1,2\}, \{4,5\}, \{2,6\}$$

	1	2	3	4	5	6
1	1	1				
2	1	1				1
3			1			
4				1	1	
5				1	1	
6		1				1

	1	2	3	4	5	6
1	1	1				1
2	1	1				1
3			1			
4				1	1	
5				1	1	
6	1	1				1

After the computation, B contains the information related to equivalence classes, row recording a "1" in each column associated to a label belonging to the same equivalence class as the row.

Appendix 2 - Handling equivalences (4)

3. **Having found equivalence classes, it is now necessary to assign a unique final label to each of them. To this purpose, a simple table initialized by the identity function may be conveniently deployed:**

$$\mathbf{LUT} = \begin{array}{c} 1 \\ \vdots \\ i \\ \vdots \\ N \end{array} \begin{bmatrix} 1 \\ \vdots \\ i \\ \vdots \\ N \end{bmatrix}$$

```
for (i=1,....N)
 for (j=1....N)
  if (B[i,j]=1)AND(i≠j){
      LUT[j]=i;
      for(k=1....N)
      B[j,k]=0;}
```

4. **Accordingly, the second scan boils down to an image look-up operation by the table:**

```
for (i=1....NROWS-1)
 for (j=1....NCOLS-1)
     I[i,j]=LUT[I[i,j]];
```

Appendix 2 - Example

Back to the previous example:

B

	1	2	3	4	5	6
1	1	1				1
2	1	1				1
3			1			
4				1	1	
5				1	1	
6	1	1				1

LUT

1	1
2	2
3	3
4	4
5	5
6	6

Appendix 2 - Example

Back to the previous example:

B

	1	2	3	4	5	6
1	1	1				1
2						
3			1			
4				1	1	
5				1	1	
6	1	1				1

LUT

1	1
2	1
3	3
4	4
5	5
6	6

Appendix 2 - Example

Back to the previous example:

B

	1	2	3	4	5	6
1	1	1				1
2						
3			1			
4				1	1	
5				1	1	
6						

LUT

1	1
2	1
3	3
4	4
5	5
6	1

Appendix 2 - Example

Back to the previous example:

B

	1	2	3	4	5	6
1	1	1				1
2						
3			1			
4				1	1	
5						
6						

LUT

1	1
2	1
3	3
4	4
5	4
6	1

Further straightforward manipulation of the LUT before the second scan would allow final labels to be consecutive numbers (i.e. 1,2,3…)

Main References

1) R. Gonzales, R. Woods, “Digital Image Processing – Third Edition”, Pearson Prentice-Hall, 2008.
2) R. Fisher, S. Perkins, A. Walker, E. Wolfart, “Hypermedia Image Processing Reference”, Wiley, 1996 (http://homepages.inf.ed.ac.uk/rbf/HIPR2/)
3) N. Otsu, “A threshold selection method from gray-level histograms”, IEEE Transactions on Systems, Man, and Cybernetics, 1979.
4) R. Haralick, L. Shapiro “Computer and Robot Vision – Vol. I, Addison-Wesley Publishing Company, 1993.
5) C. Steger, M. Ulrich, C. Wiedemann, “Machine Vision Algorithms and Applications – 2nd Edition”, Wiley-VCH, Weinheim, 2018.
6) L Di Stefano, A Bulgarelli, “A simple and efficient connected components labeling algorithm”, ICIAP 1999.
7) C. Grana, D. Borghesani, R. Cucchiara “Optimized Block-based Connected Components Labeling with Decision Trees”. IEEE Transactions on Image Processing, 2010.
8) F Bolelli, S Allegretti, L Baraldi, C Grana “Spaghetti Labeling: Directed Acyclic Graphs for Block-Based Connected Components Labeling”, IEEE Transactions on Image Processing, 2019.
9) F Bolelli, M Cancilla, L Baraldi, C Grana “Toward reliable experiments on the performance of connected components labeling algorithms”, Journal of Real-Time Image Processing 2020.
10) Jonathan T. Barron “A Generalization of Otsu's Method and Minimum Error Thresholding”, ECCV 2020
11) F. Bolelli, S. Allegretti, L. Lumetti, C. Grana “A State-of-the-Art Review with Code about Connected Components Labeling on GPUs”, IEEE Transactions on Parallel and Distributed Systems, 2024.