Nonlinear Filters

Giacomo Boracchi

CVPR USI, March 31 2020

Book: GW chapters 3, 9, 10

Non Linear Filters

Non Linear Filters are such that the relation

$$H[\lambda\, f(t) + \mu\, g(t)] = \lambda H[f]\,(t) + \mu\, H[g]\,(t)$$

does not hold, at least for some value of λ, μ, f, g

Examples of nonlinear filter are

- Median Filter (Weighted Median)
- Ordered Statistics based Filters
- Threshold, Shrinkage

There are many others, such as data adaptive filtering procedures (e.g LPA-ICI)

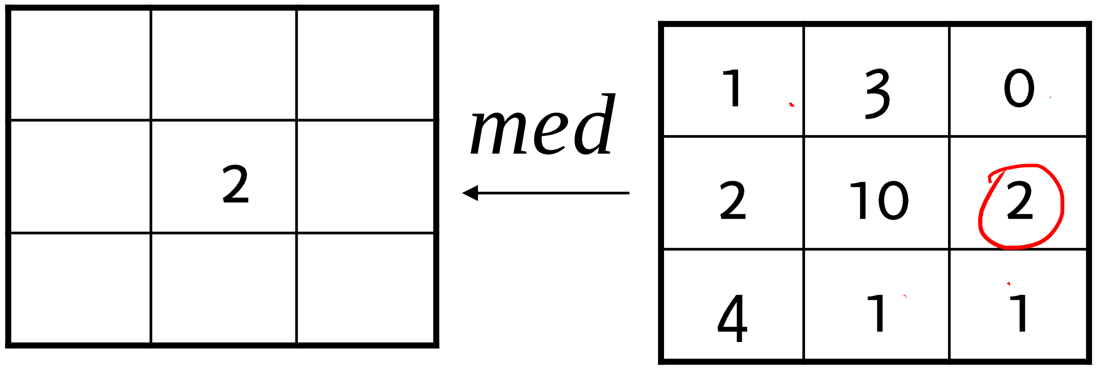
Blockwise Median

Local spatial transformations

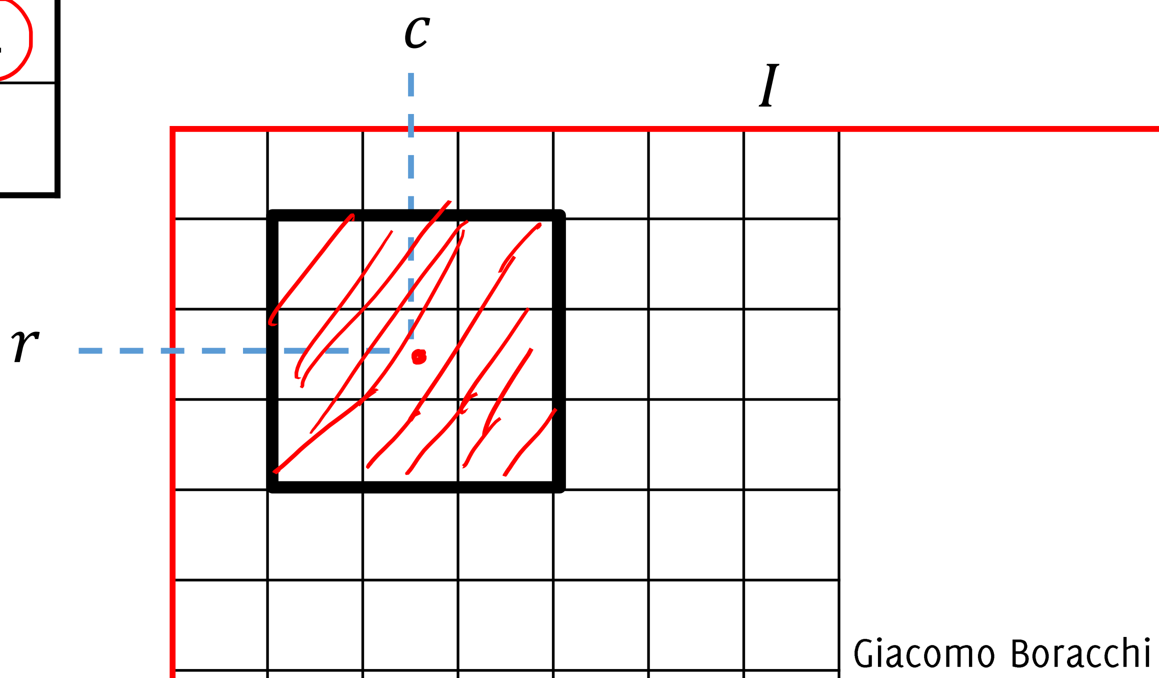
Block-wise median: replaces each pixel with the median of its neighborhood

U

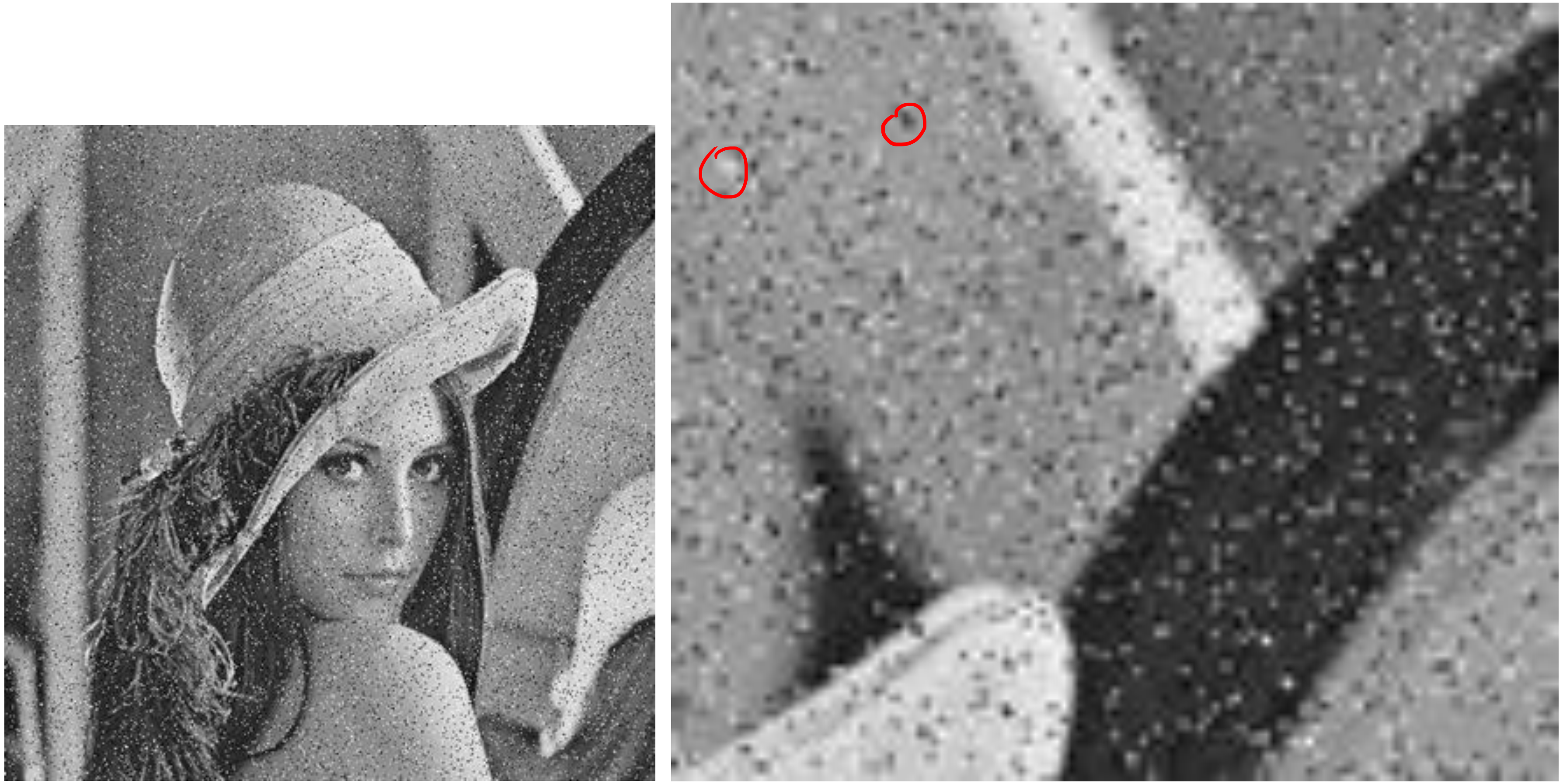
$\sum_{U} w_i \cdot$



$$m = median(1,3,0,2,10,2,4,1,1) = 2$$



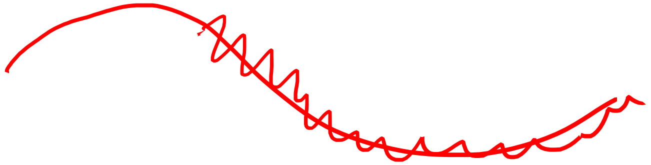
Salt-and-pepper noise

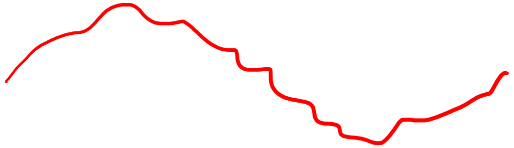


Salt and Pepper (Impulsive) noise

Denoisng using local smoothing 3x3

$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}/9$





Denoisng with median 3x3

$u \quad \begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$

Salt and Pepper (Impulsive) noise

Morphological Operations

Ordered Statitiscs and Blob Labeling

Giacomo Boracchi

An overview on morphological operations

Erosion, Dilation

Open, Closure

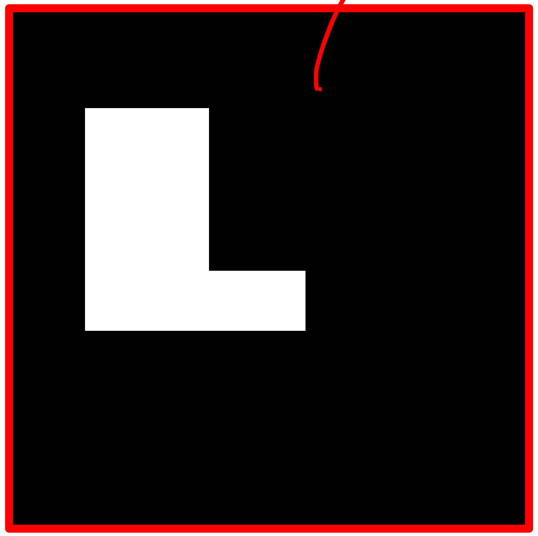
We assume the image being processed is binary, as these operators are typically meant for refining “mask” images.

Boolean operations on binary images

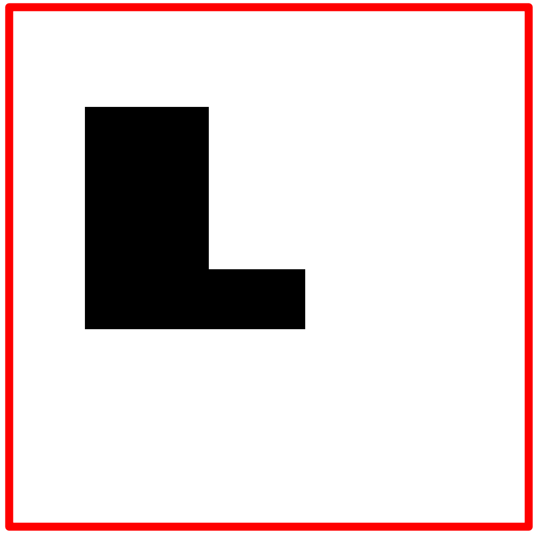
$I \in \{0,1\}^{R \times C}$

True / false
1 / 0

A

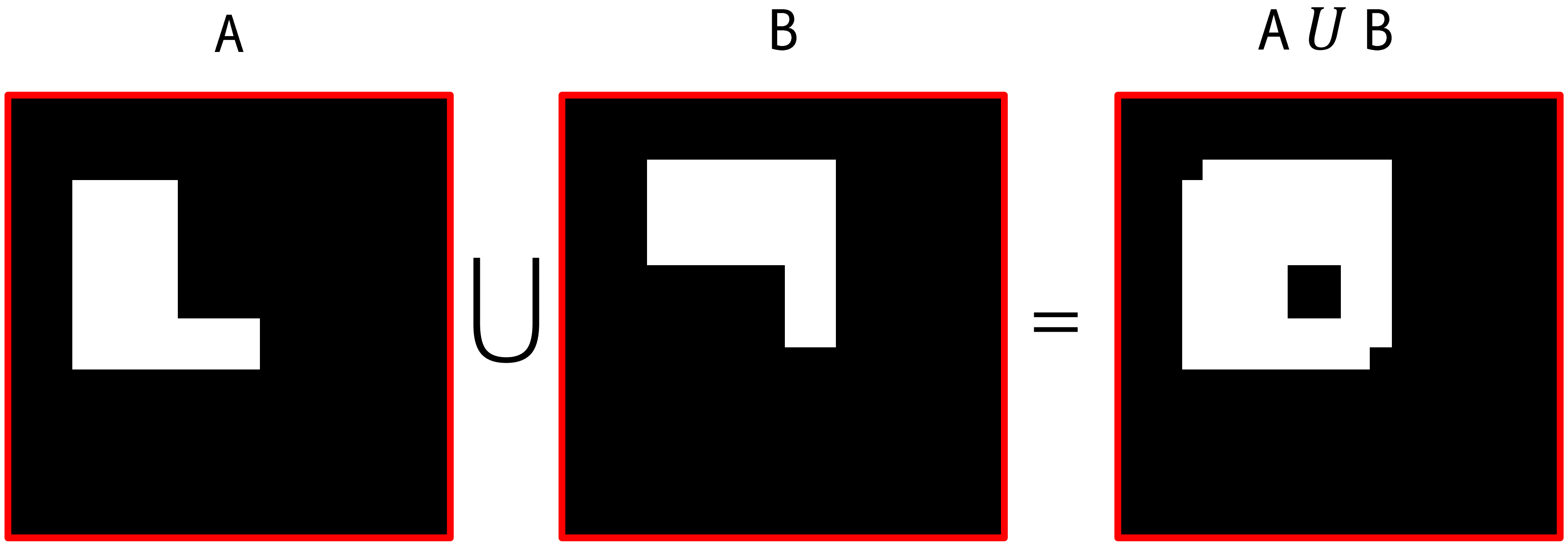


NOT(A)



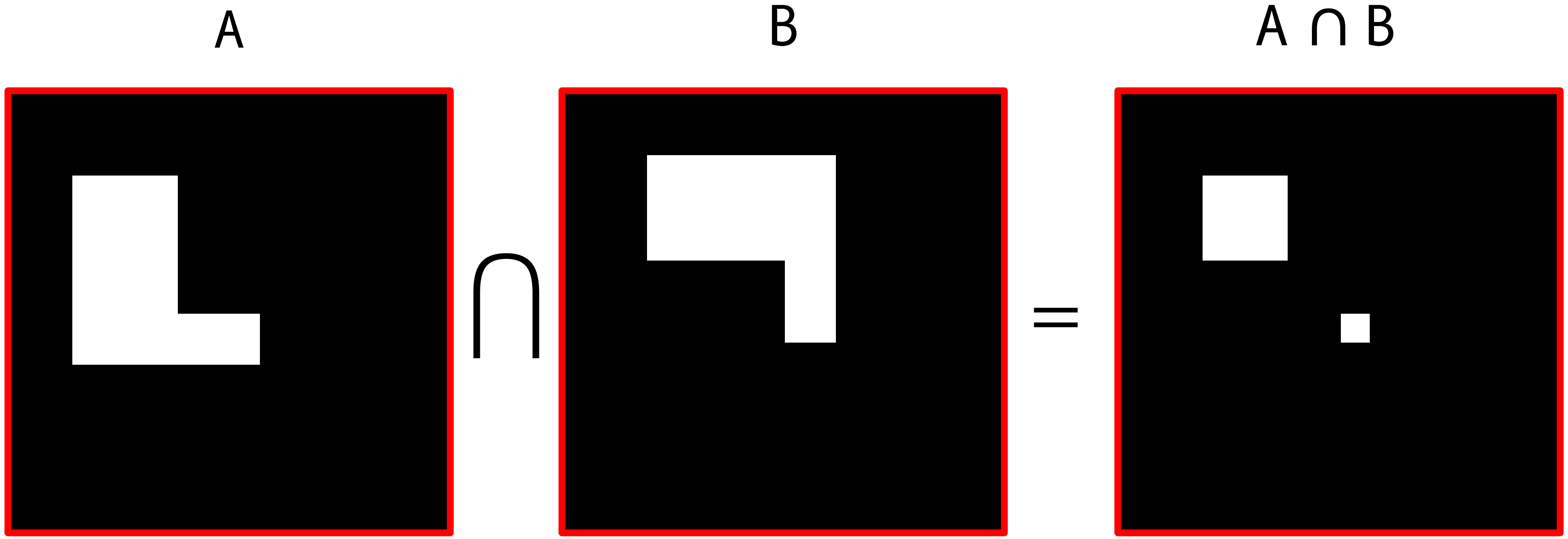
NOT_A = A == 0

UNION of binary images



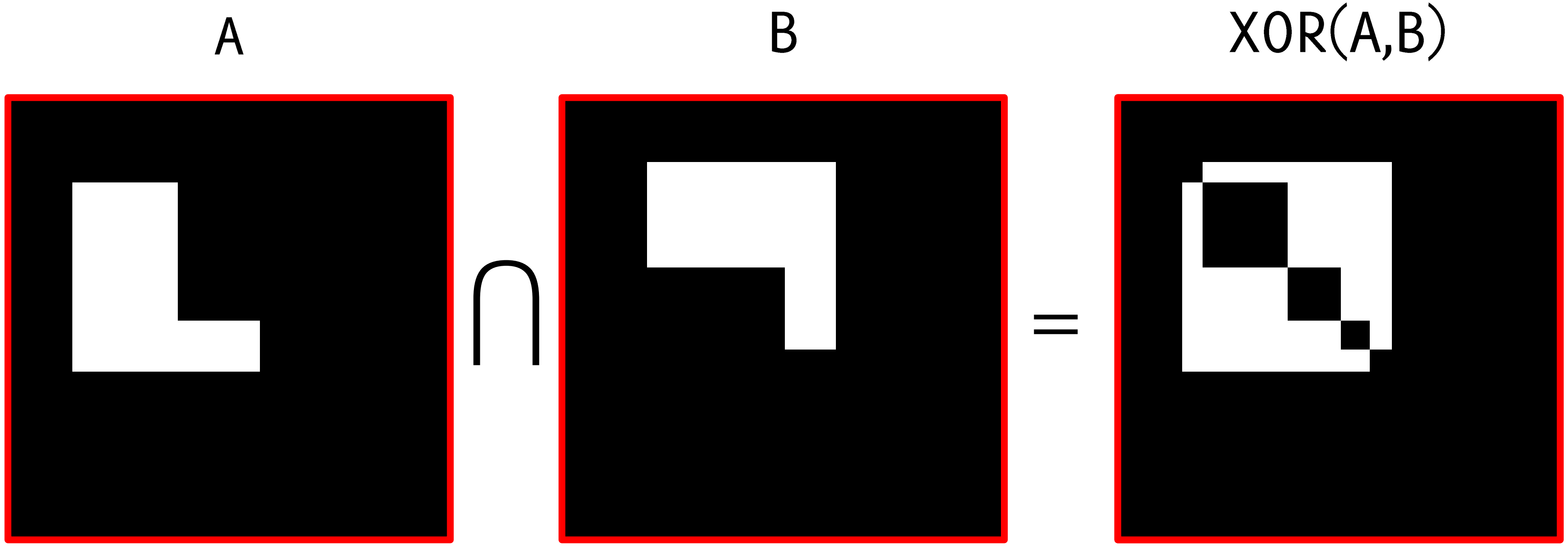
AUB = A + B > 0

INTERSECTION of binary images



A_AND_B = A + B > 0

On binary images it is possible to define XOR

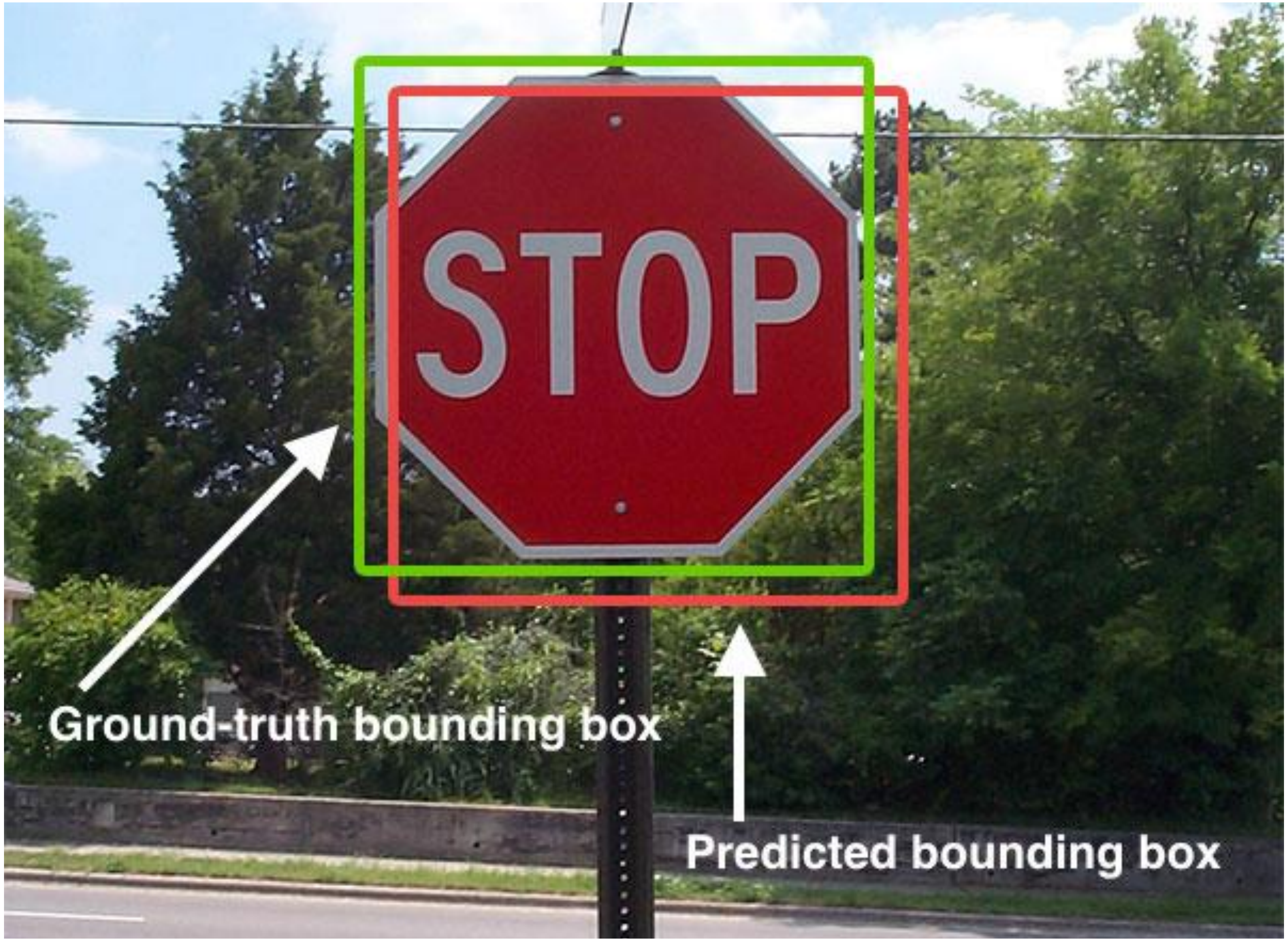


XOR(A,B) =

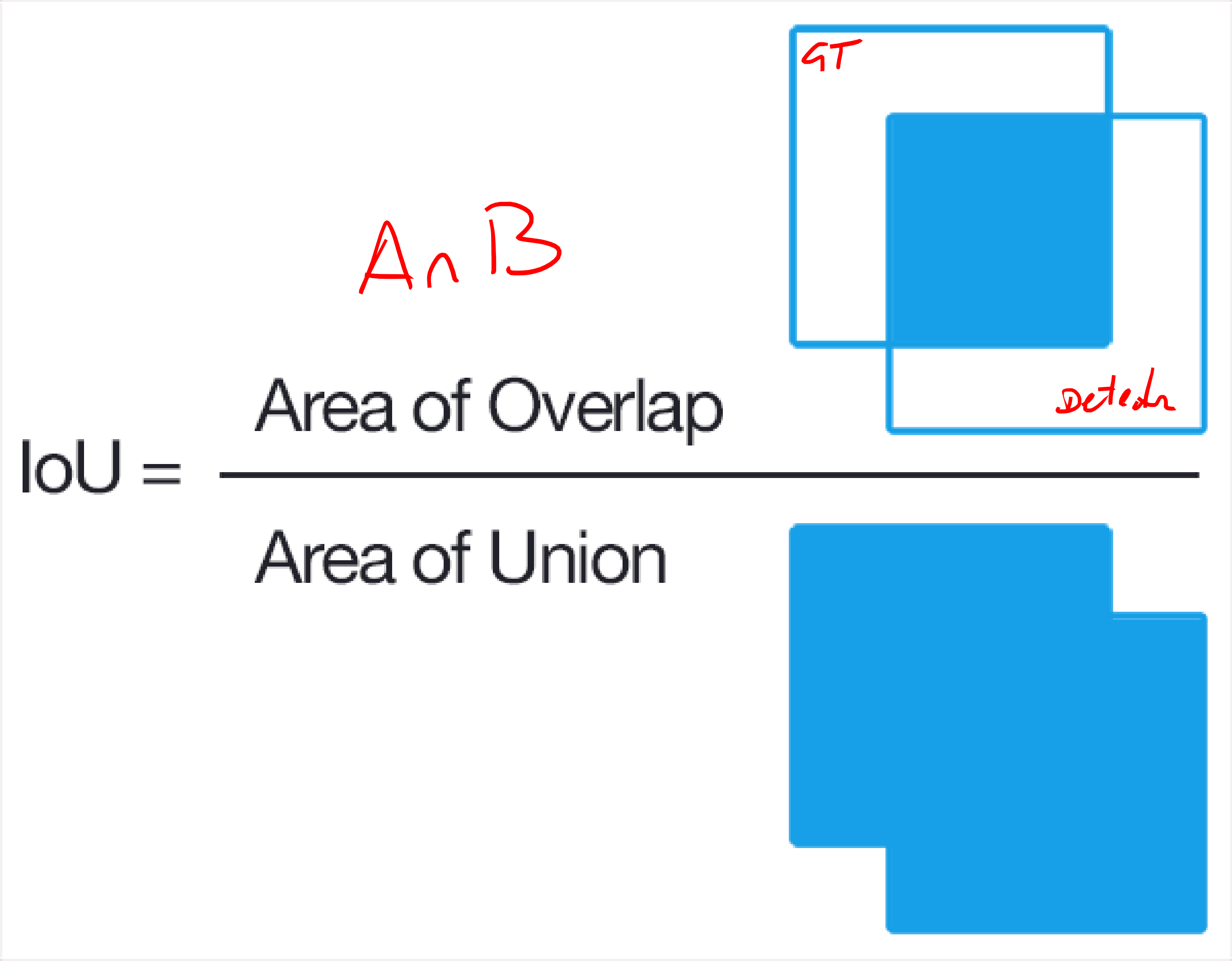
AUB - A_AND_B

$A \cap B$

Intersection over the Union (IoU, Jaccard Index)



Intersection over the Union (IoU, Jaccard Index)



Intersection over the Union (IoU, Jaccard Index)

Jaccard Index (IoU)

It is a statistical measure of similarity between two sets

- being in case of images the coordinates of the pixels set to true

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

It ranges between $[0,1]$ being $J(A, B) = 0$ when A and B are disjoint, and $J(A, B) = 1$, when the two sets coincides.

It is a standard reference measure for detection performance

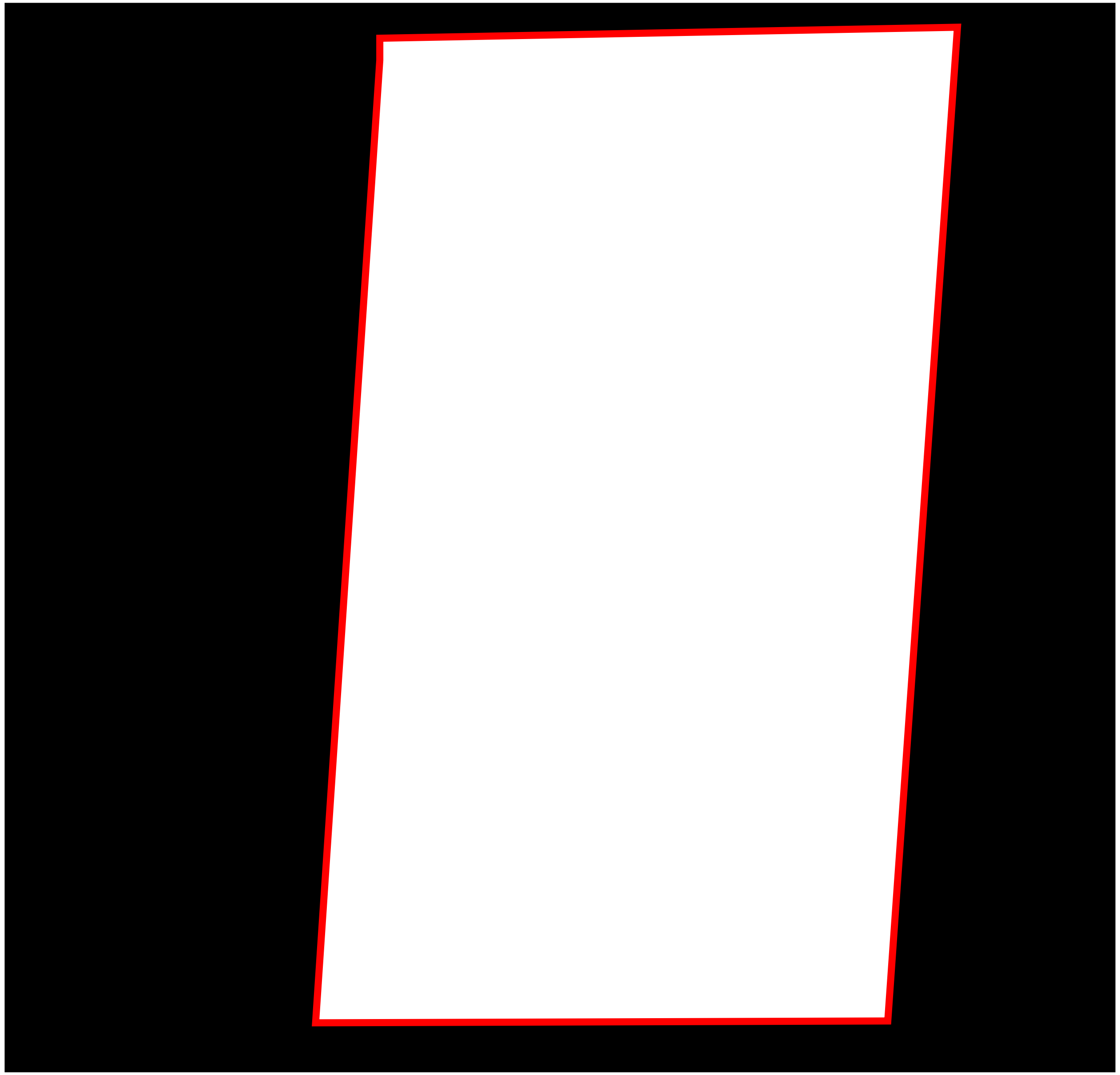
Jaccard Index (IoU)

It is not necessarily defined for bounding boxes (even though most of deep learning networks for detections provide bb as outputs)

Credits Barozzi – Guidi IACV project 2017

Jaccard Index (IoU)

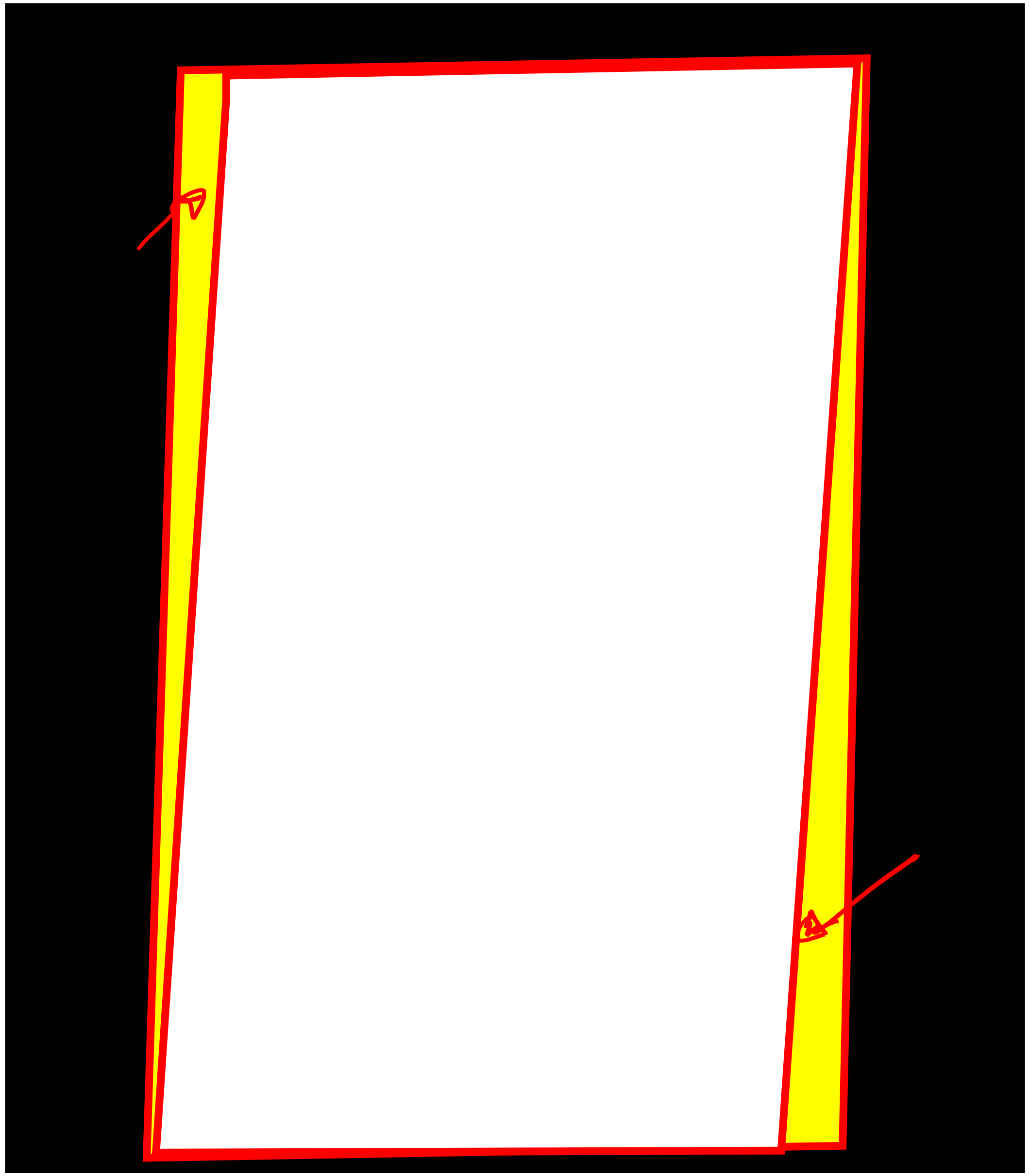
Jaccard Index (IoU)



Jaccard Index (IoU)

Jaccard Index (IoU)

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$



Filters on binary images

It is possible to define filtering operations between binary images

Now, assume that the filter weights are also binary.

In the context of object detection, these can be used to refine the detection boundaries

Erosion

$$\boxed{\begin{matrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{matrix}}$$

General definition:
Nonlinear Filtering procedure that replace to each pixel value the minimum on a given neighbor

b/w
greyscale

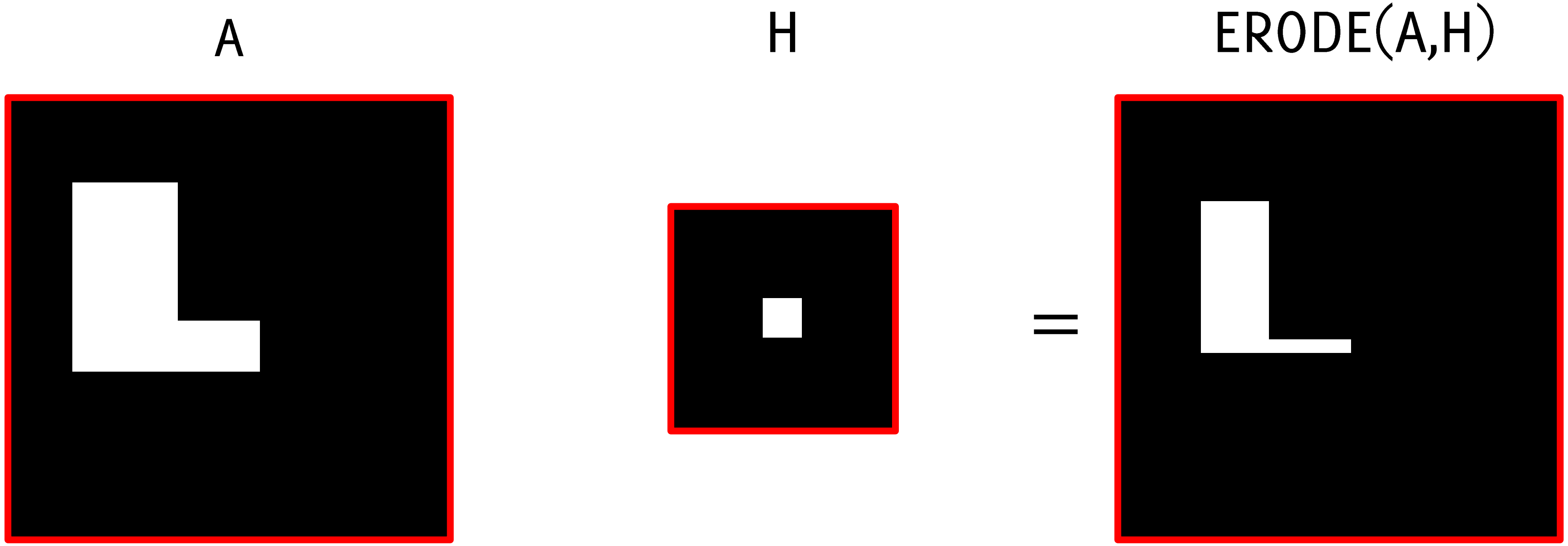
As a consequence on binary images, it is equivalent to the following rule:

E(x)=1 iff the image in the neighbor is constantly 1

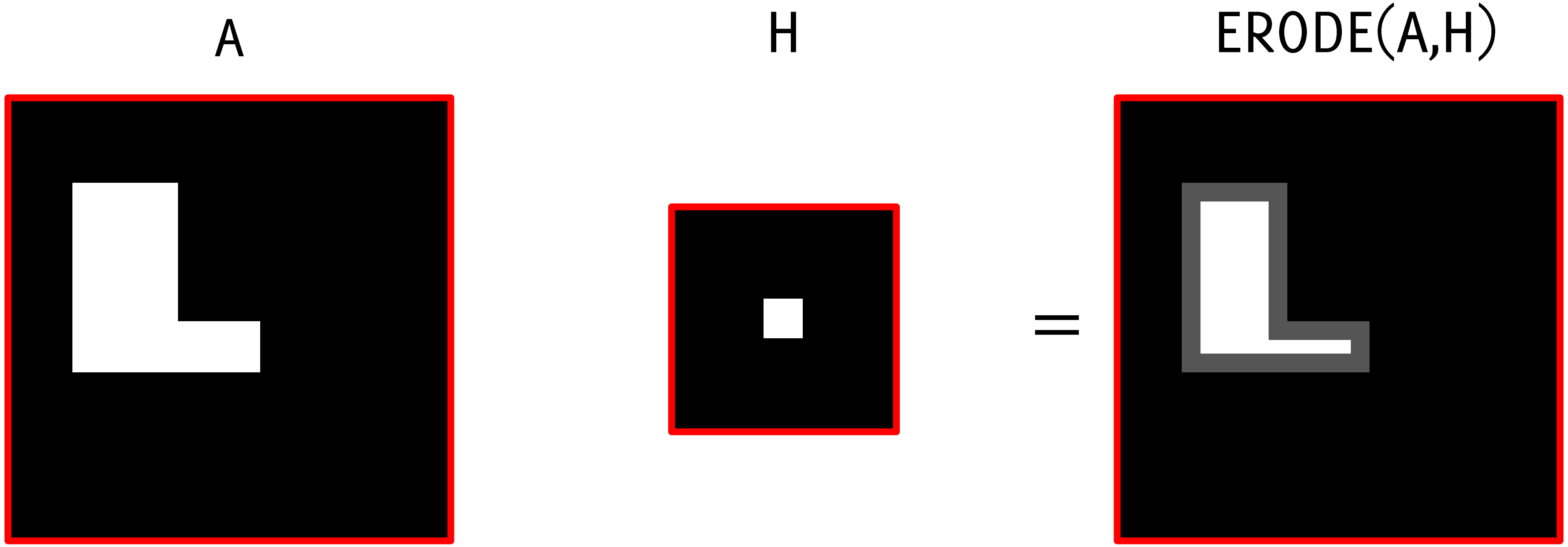
This operation reduces thus the boundaries of binary images

It can be interpreted as an AND operation of the image and the neighbour overlapped at each pixel

Erosion



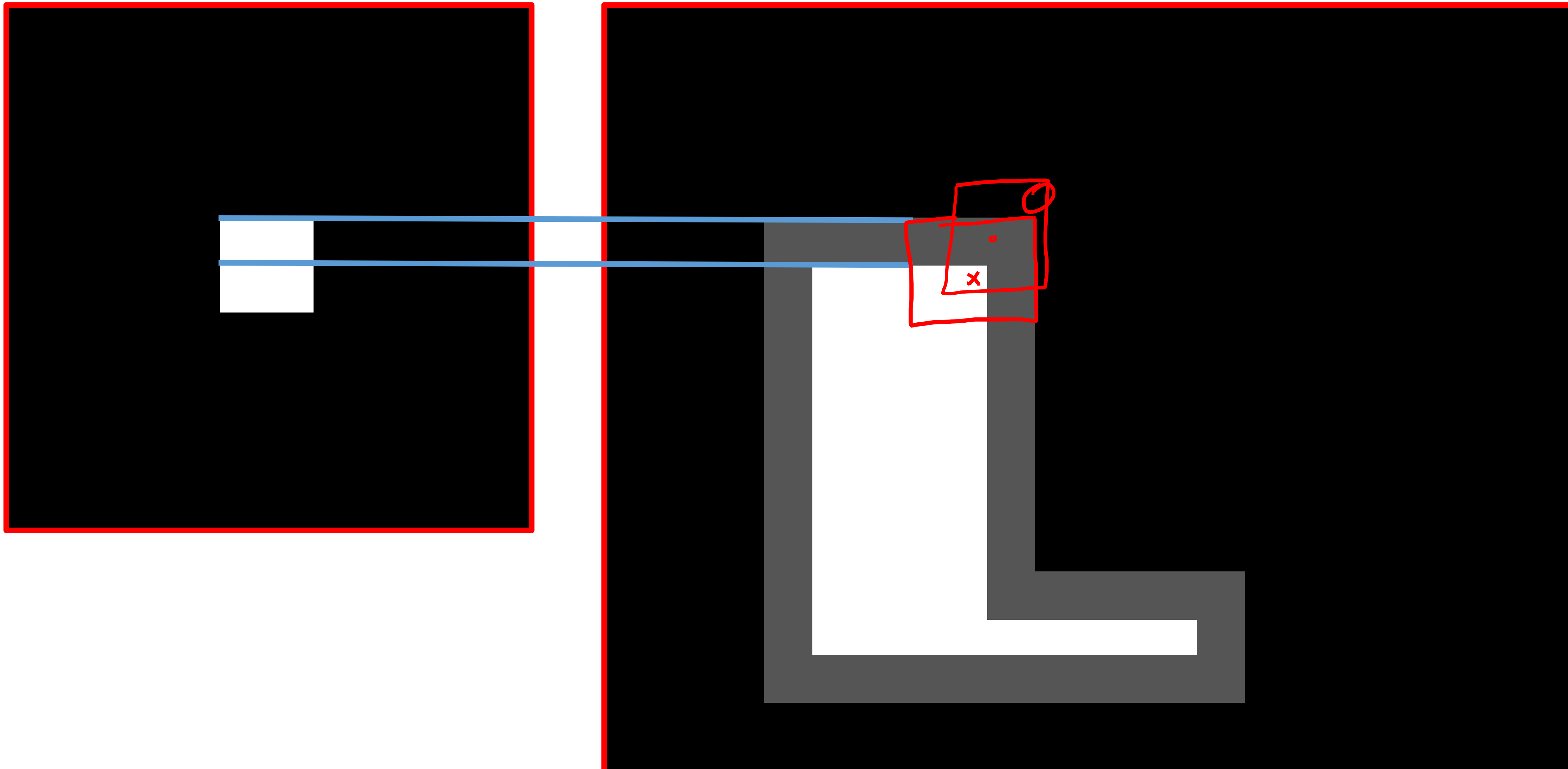
Erosion



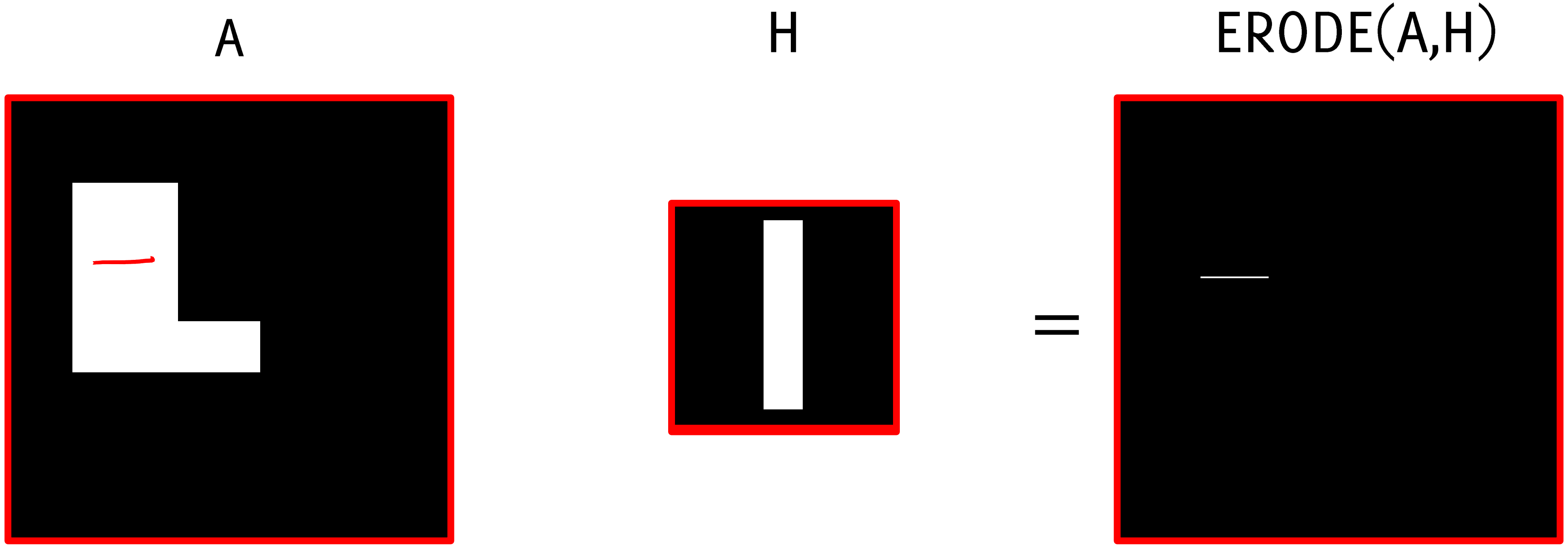
The gray area corresponds
to the input

Erosion

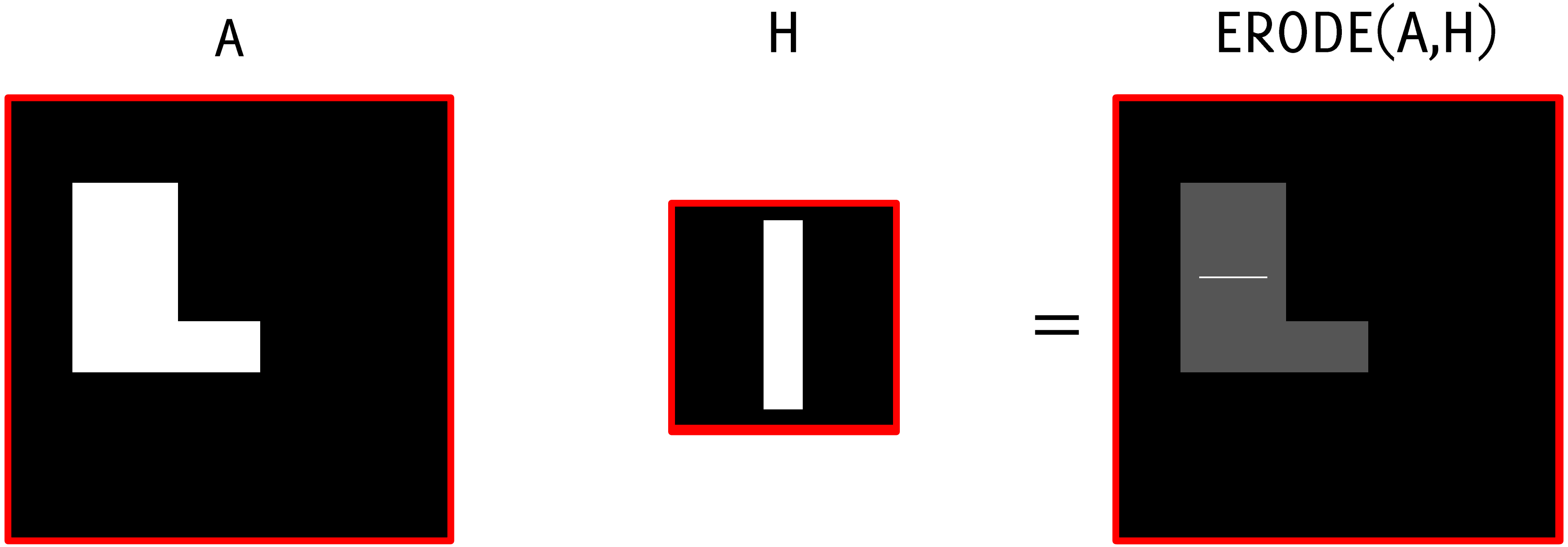
Erosion removes half size of the structuring element used as filter



Erosion



Erosion



Giacomo Boracchi

Dilation

General definition:
Nonlinear Filtering procedure that replace to each pixel value the maximum on a given neighbor

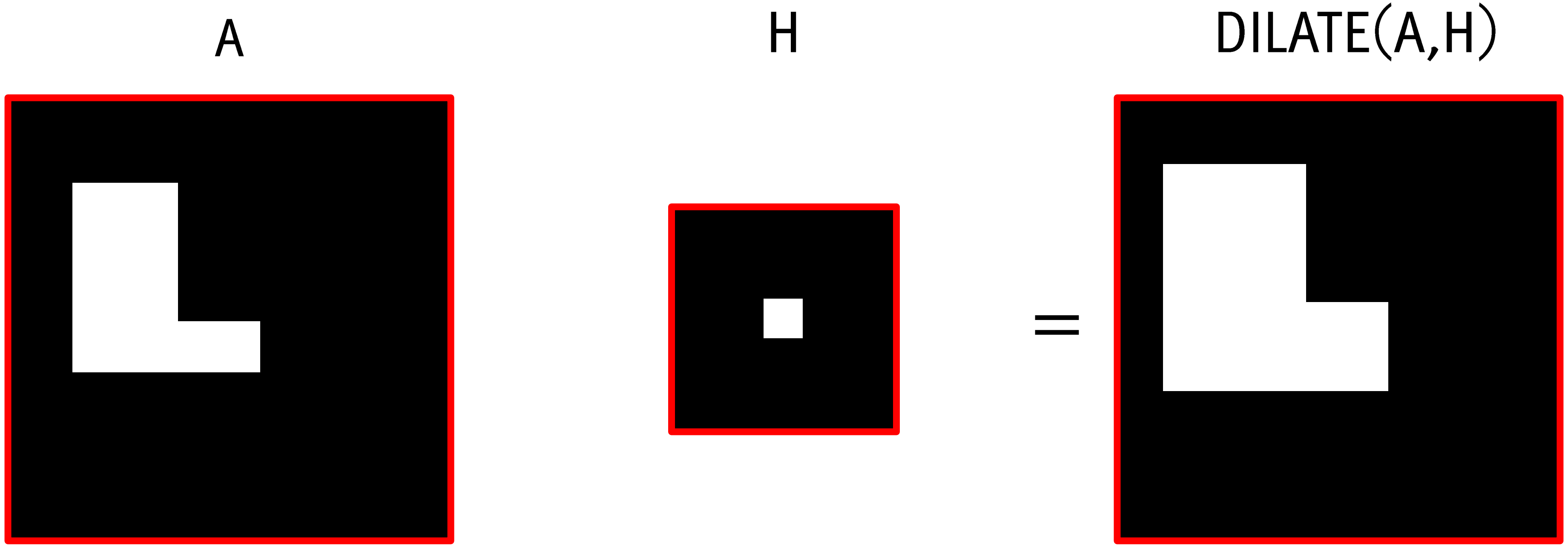
As a consequence on binary images, it is equivalent to the following rule:

E(x)=1 iff at least a pixel in the neighbor is 1

This operation grows fat the boundaries of binary images

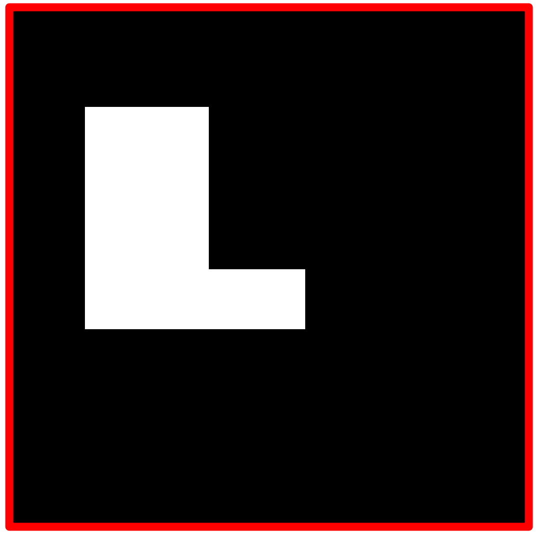
It can be interpreted as an OR operation of the image and the neighbour overlapped at each pixel

Dilation

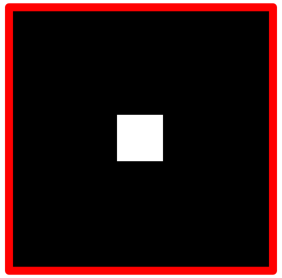


Dilation

A

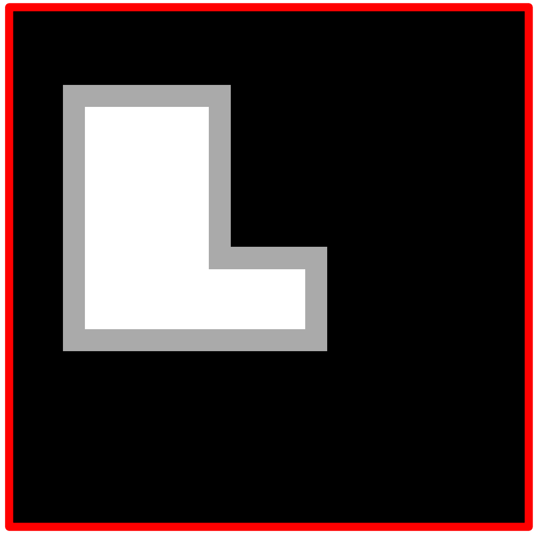


H



=

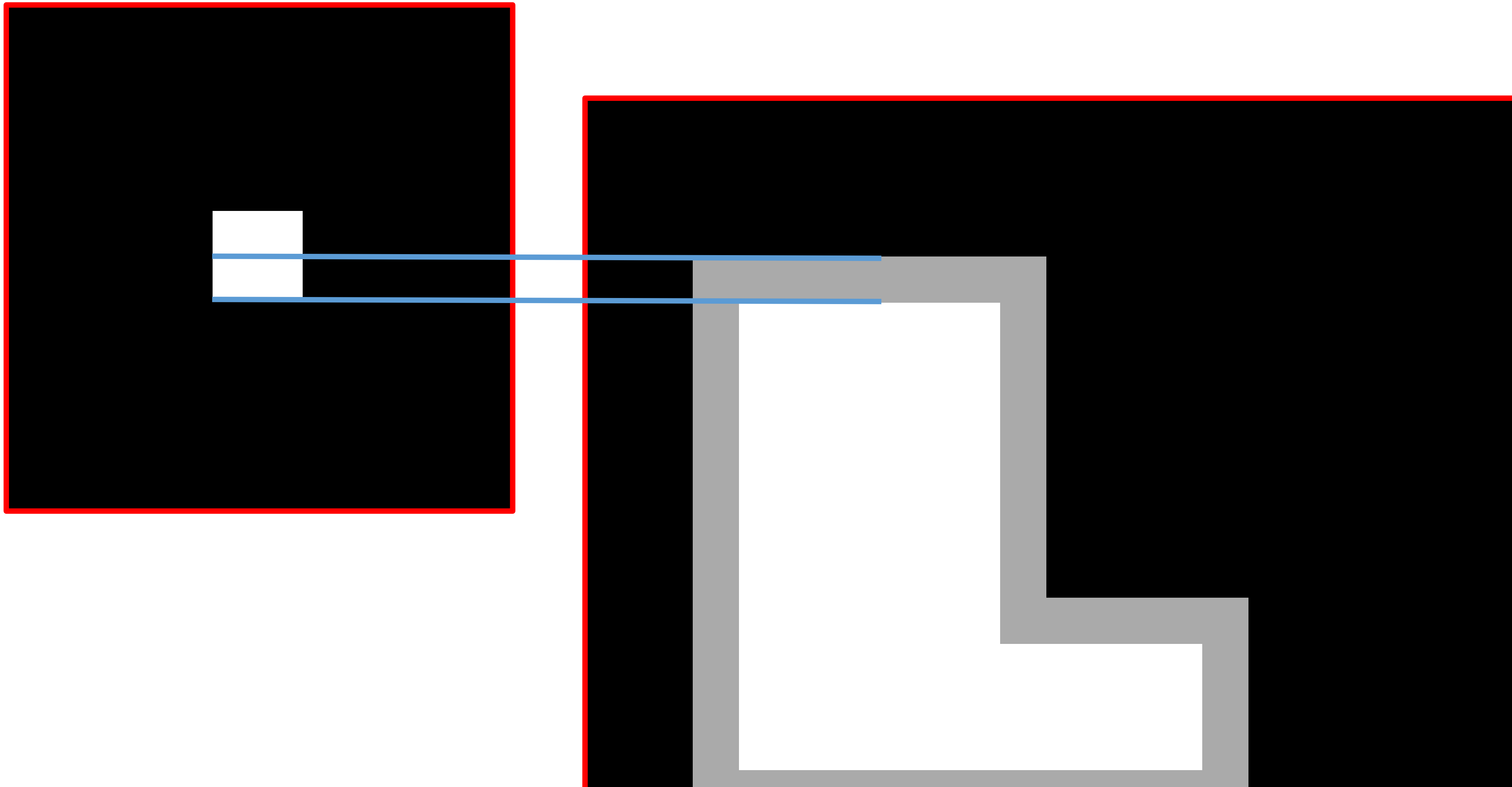
DILATE(A,H)



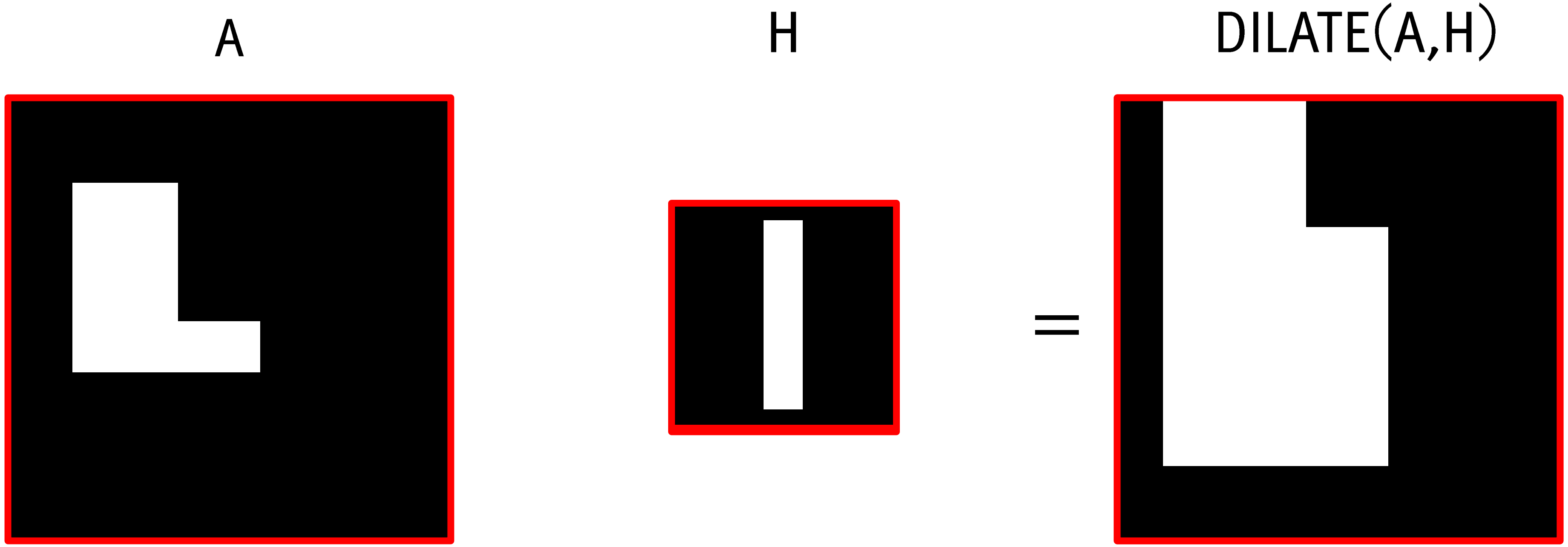
The brighter area now
corresponds to the input

Dilation

Dilation expands half size of the structuring element used as filter

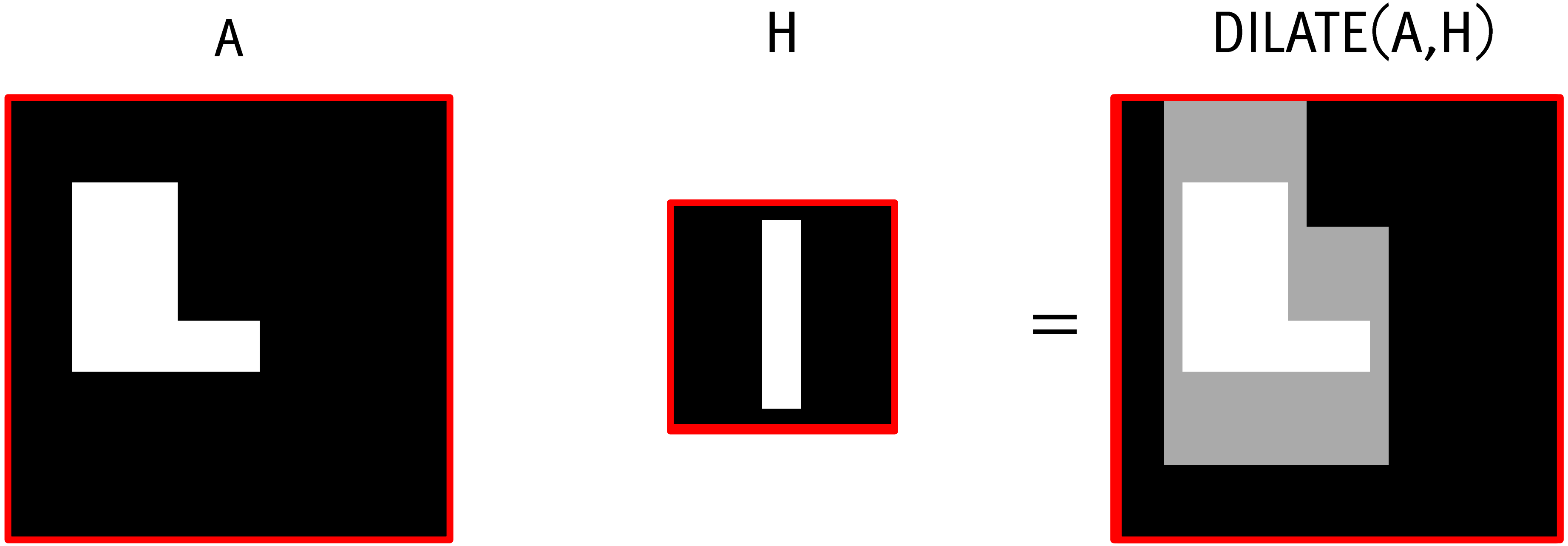


Dilation



Giacomo Boracchi

Dilation



Open and Closure

Open Erosion followed by a Dilation

Closure Dilation followed by an Erosion

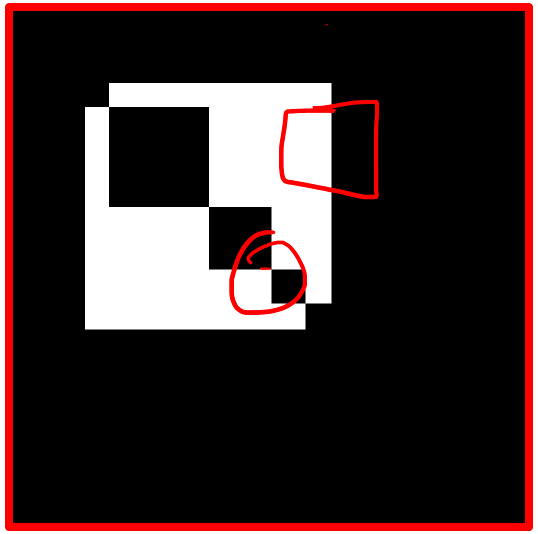
Open

Open Erosion followed by a Dilation

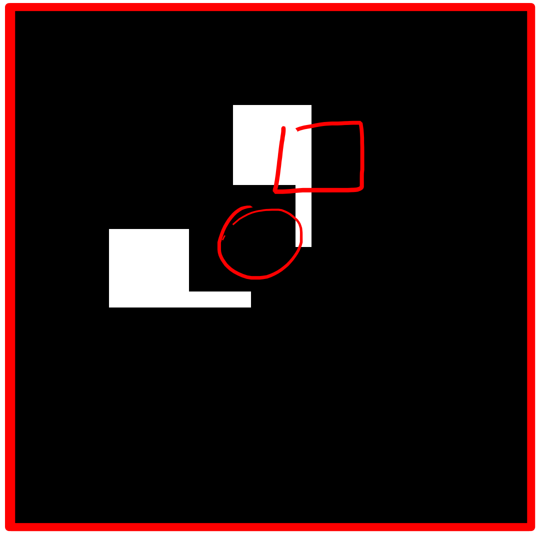
- Smooths the contours of an object
- Typically eliminates thin protrusions

Open

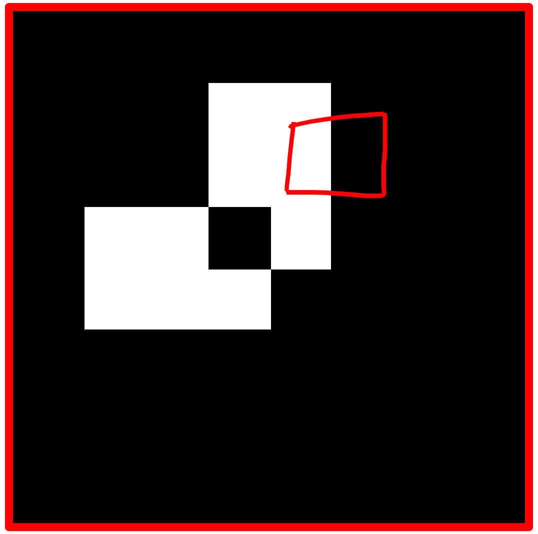
A



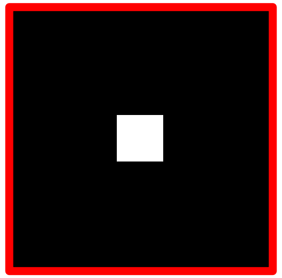
O = ERODE(A,H)



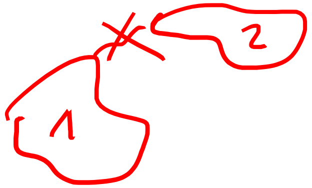
O = DILATE(O,H)



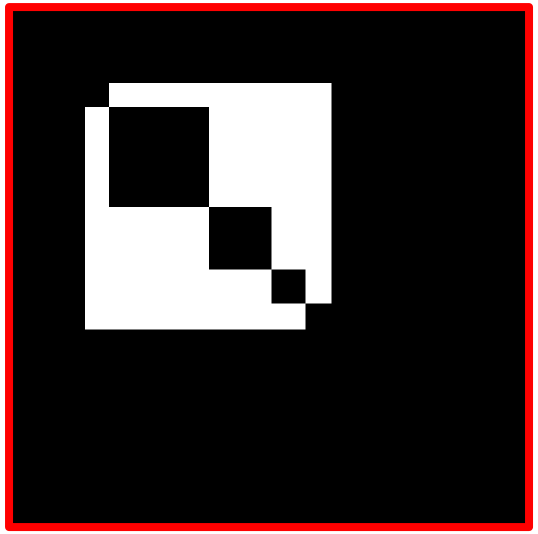
H



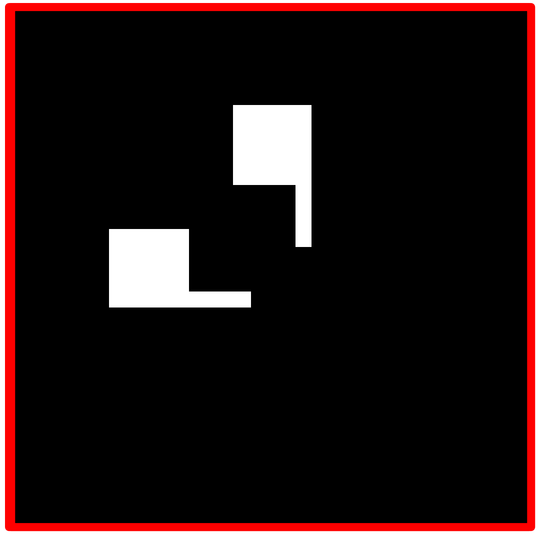
Open



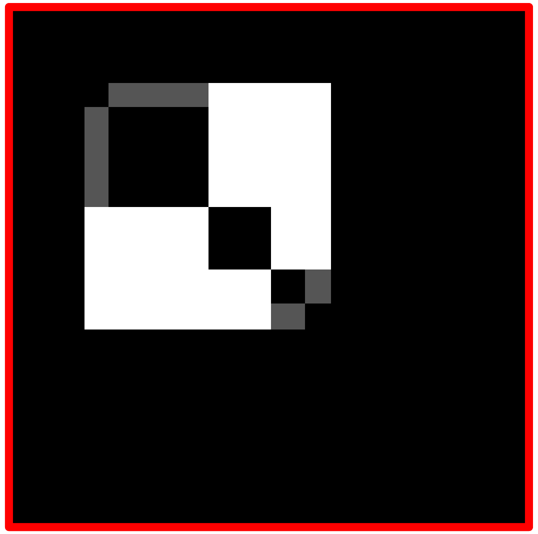
A



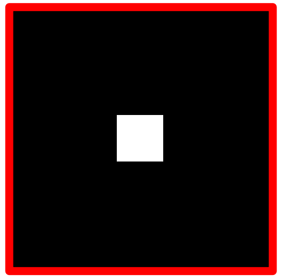
O = ERODE(A,H)



O = DILATE(O,H)



H



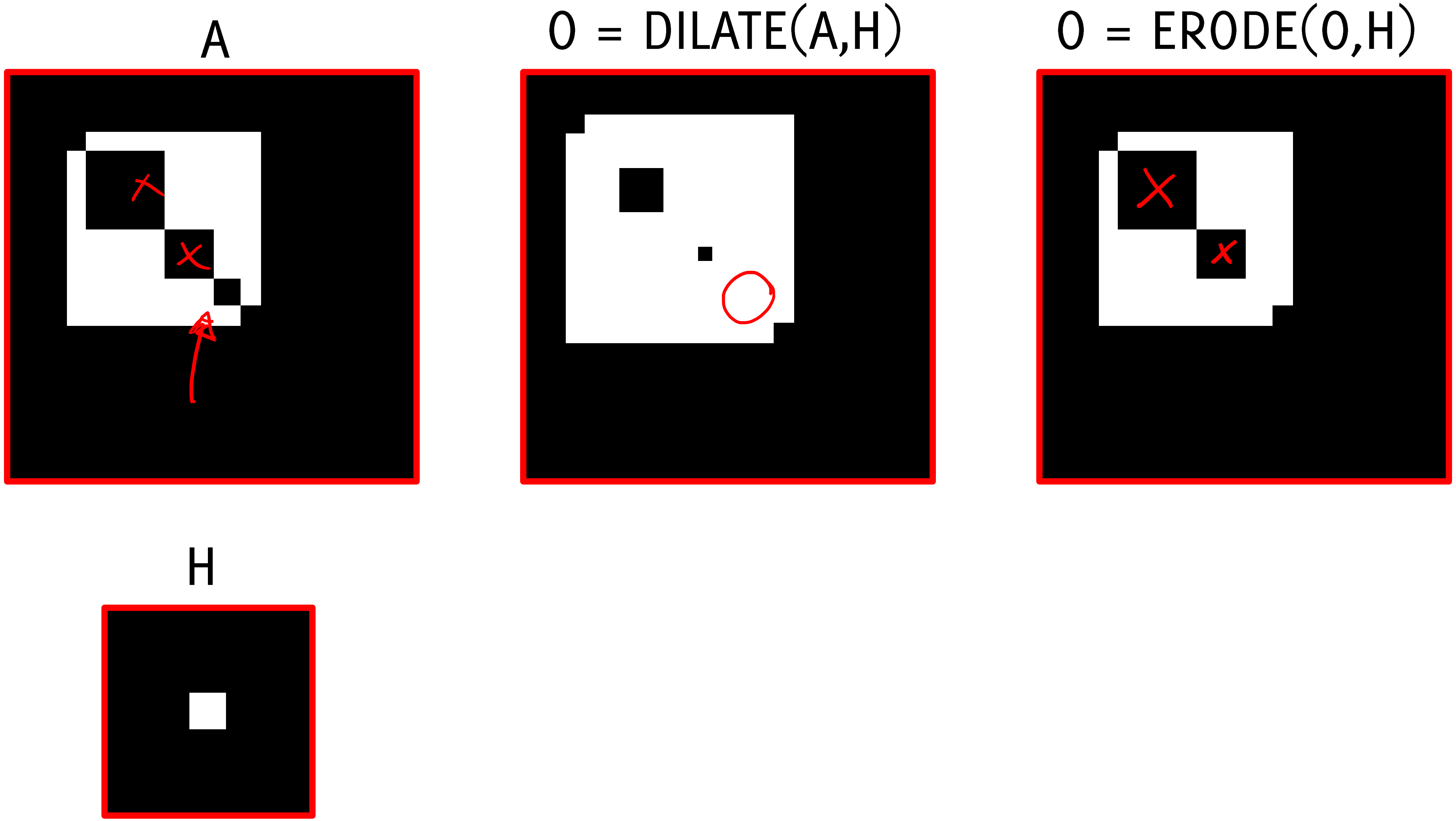
The gray area corresponds to the input

Closure

Closure Dilation followed by an Erosion

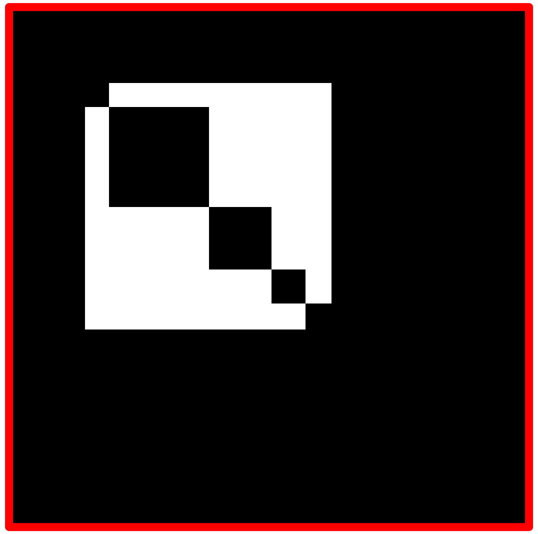
- Smooths the contours of an object, typically creates bridges
- Generally fuses narrow breaks

Close

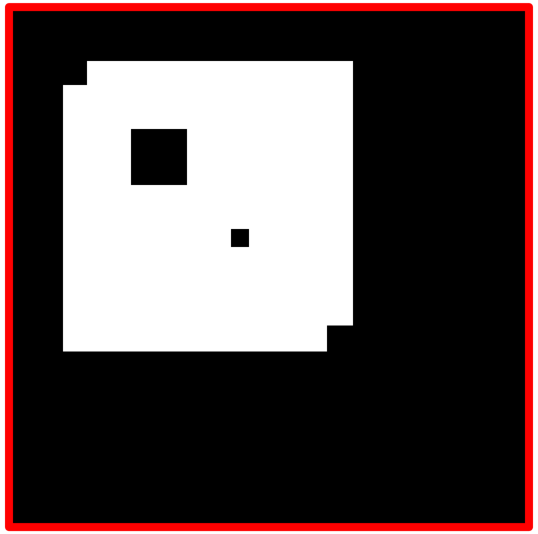


Close

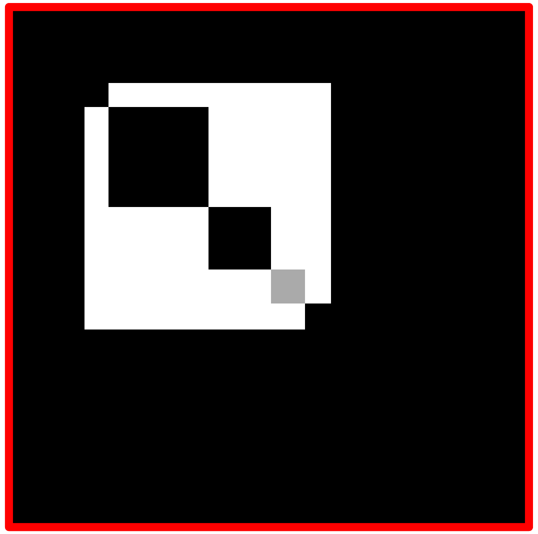
A



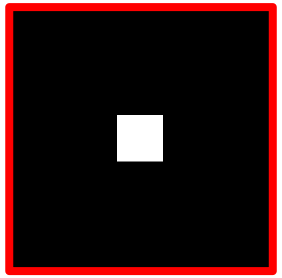
O = DILATE(A,H)



O = ERODE(O,H)



H

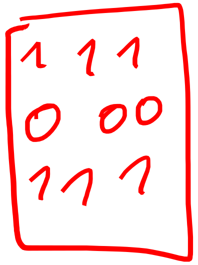


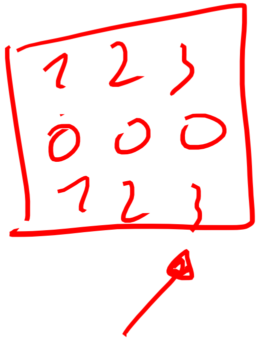
The gray spot was «false»
in the input

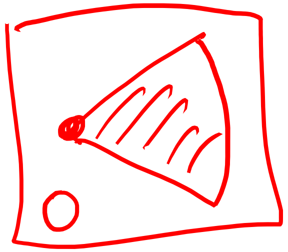
There are several other Non Linear Filters

Ordered Statistic based

- Median Filter
- Weight Ordered Statistic Filter
- Trimmed Mean
- Hybrid Median







In Python: **`skimage.morphology`**

Outline

A bit more on Nonlinear Filters

Edge Detection (Canny Edge Detector)

Going to Binary Images

Plenty morphological information can be extracted from binary images

- Binarization
- Blobs labeling, "morphological operations"

Edge detection (Canny)
Line detection (Hough)

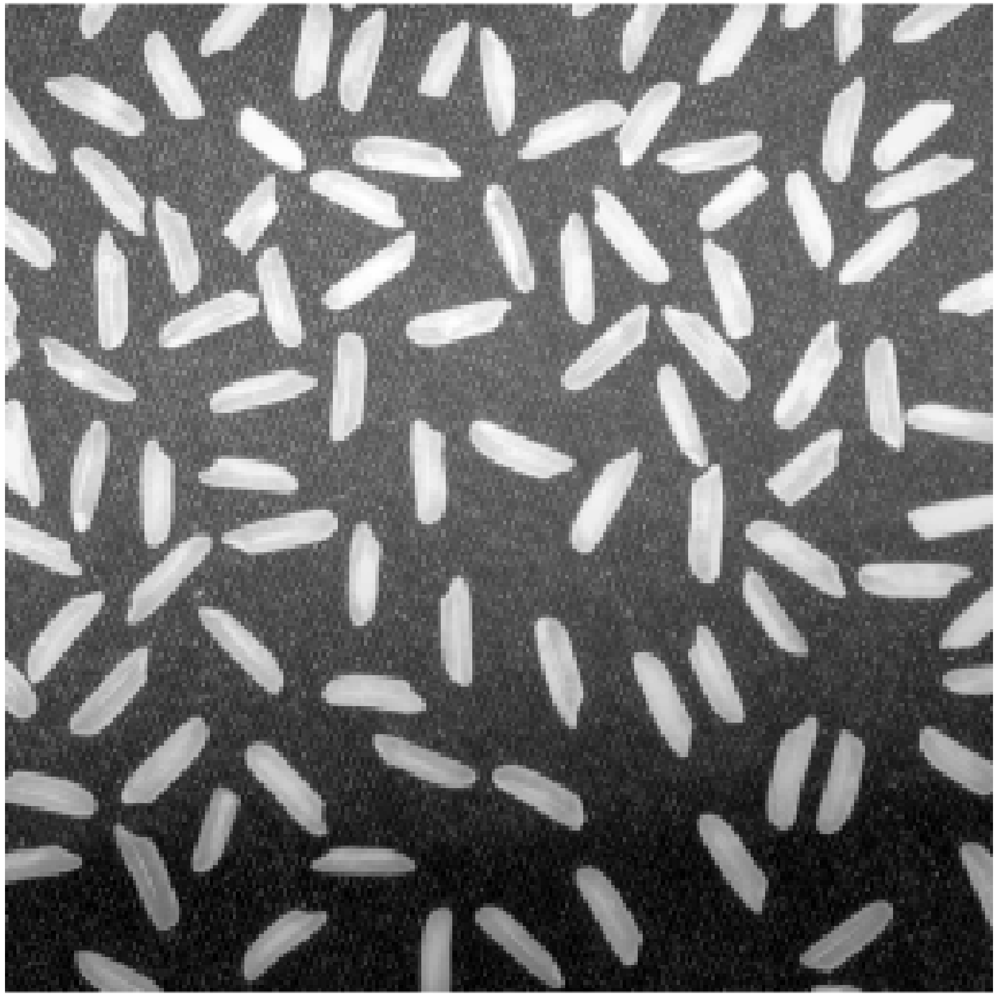
Binarization

$\in [0, 255]^{R\times C}$

I

$I > \Gamma$

$\in \{0,1\}^{R\times C}$



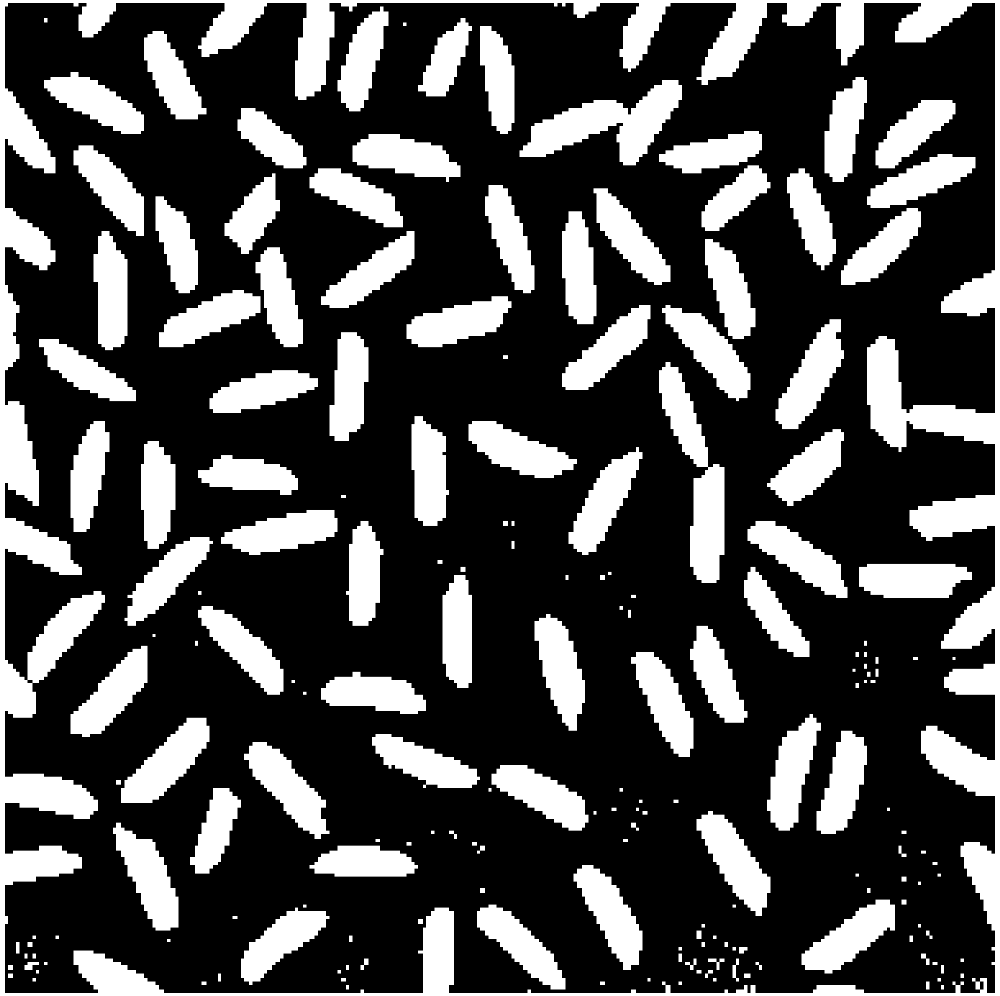
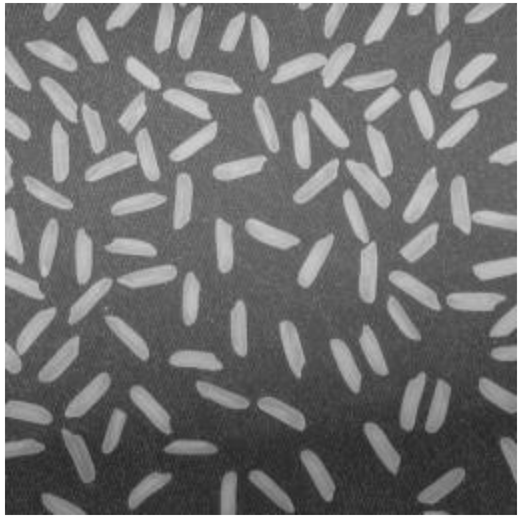


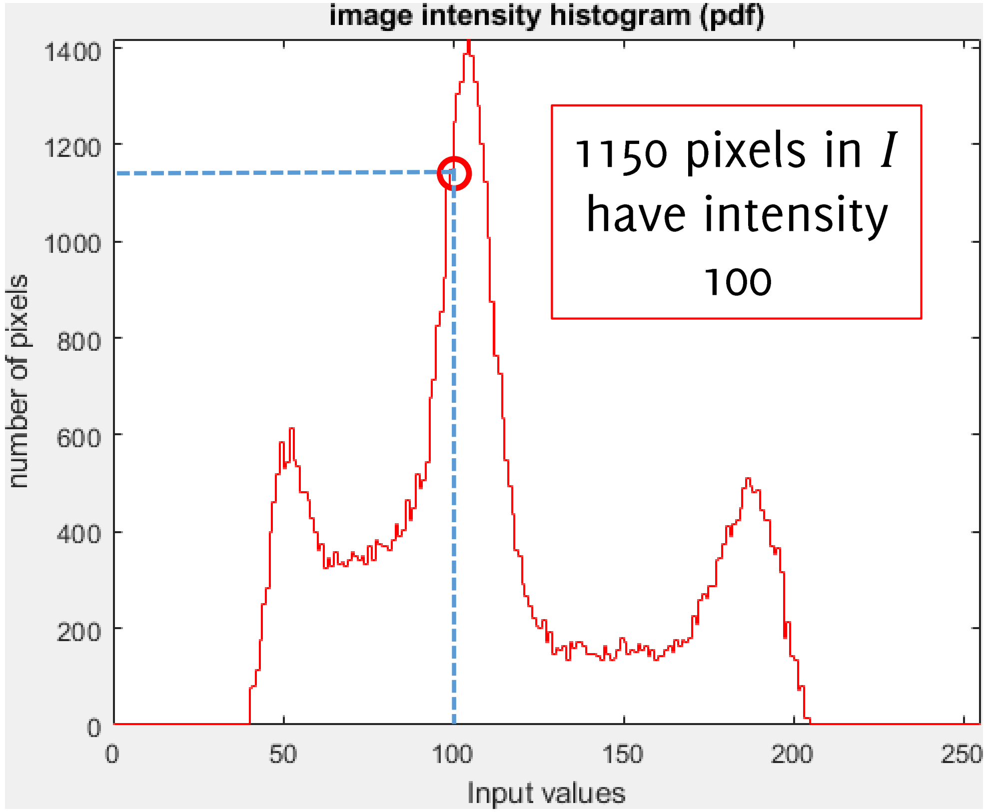
Image histograms

Histogram of pixel intensities can be used to define intensity transformation



img

$h_{100} = 1150$

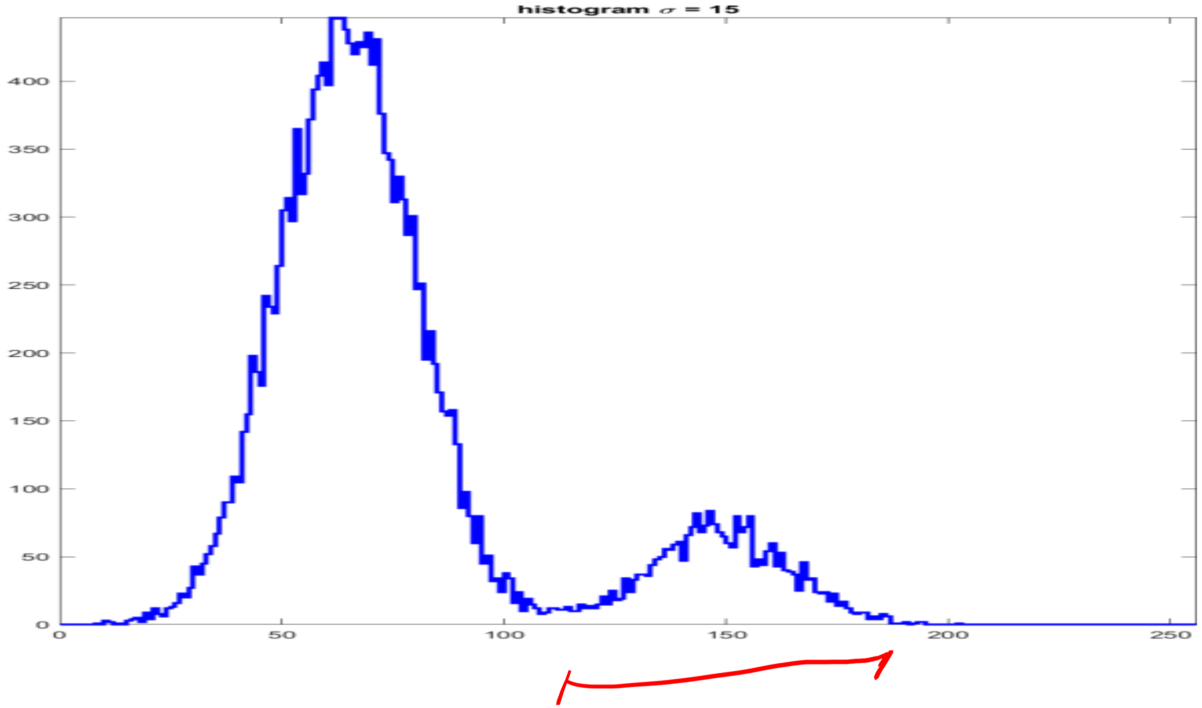


$i = 100$

Histogram

Threshold definition by Otsu Method

Assume your (grayscale) image has a bi-modal histogram



Threshold definition by Otsu Method

Assume your (grayscale) image has a bi-modal histogram

T

bi-modal histogram

narize your image.
iciently since this operation might

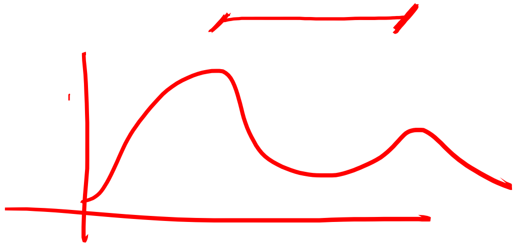
Goal: Find a threshold T to suitably binarize your image.

This threshold has to be computed efficiently since this operation might be repeated in different image regions

Thresholding

Different options:

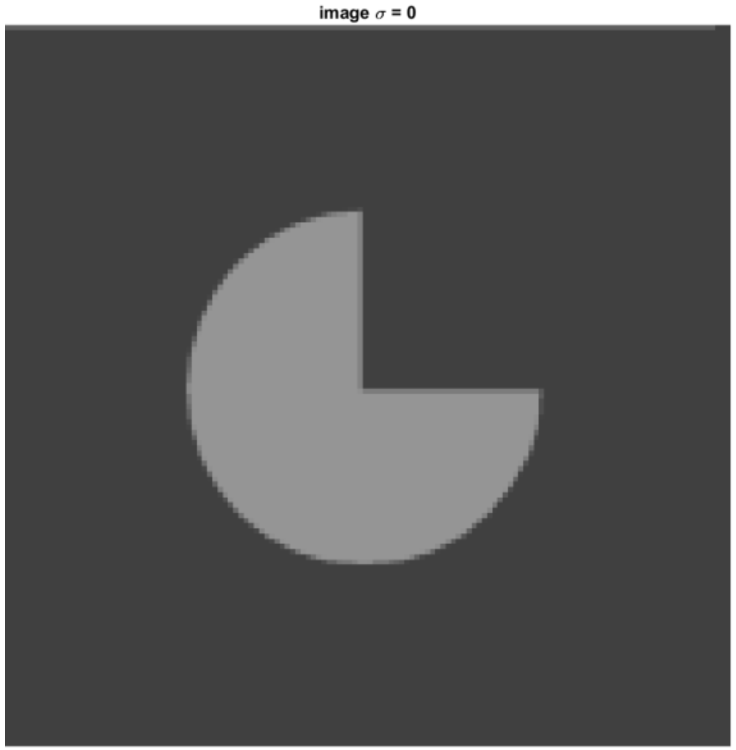
- Global (a fixed threshold for the whole image) T
- Variable (threshold defined locally)
- Adaptive (the way threshold is set varies within the image)

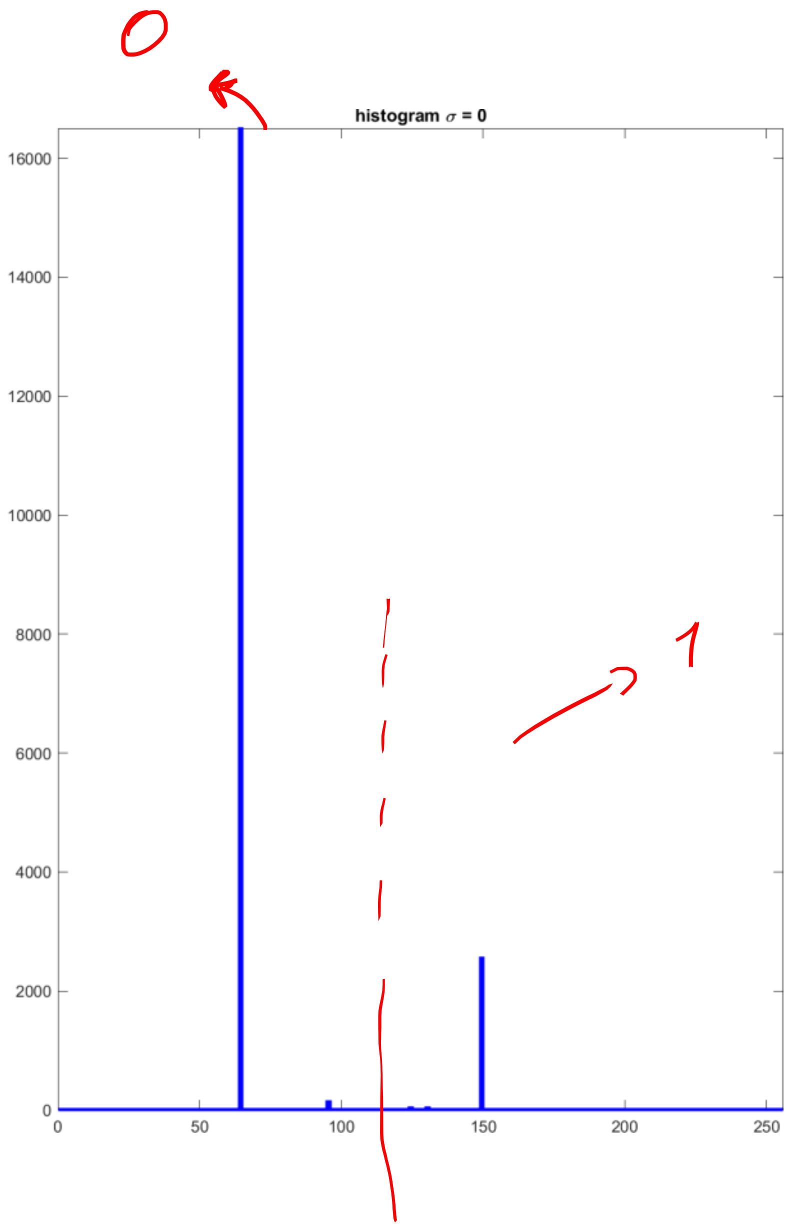


The success of thresholding depends on:

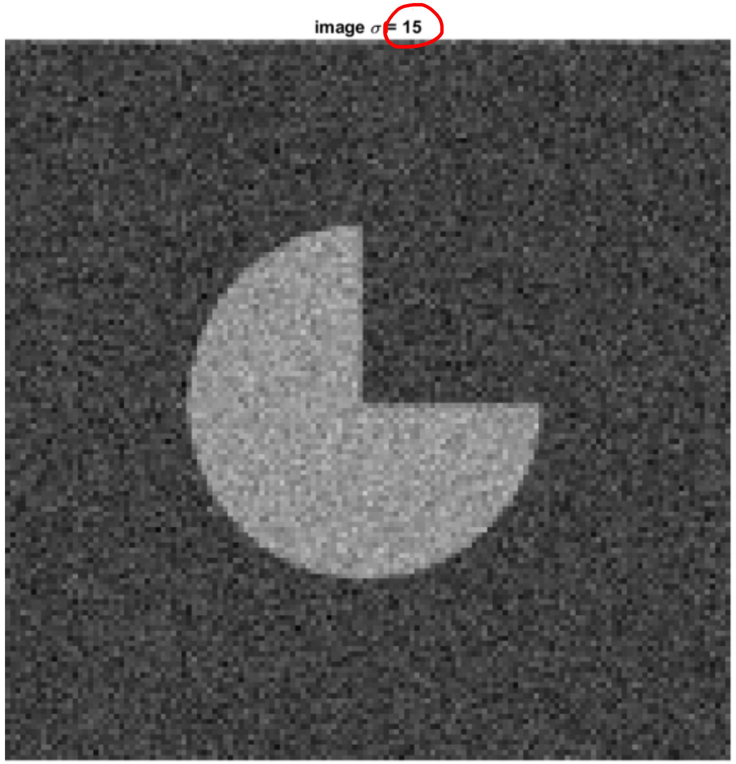
- Peaks separation in the grayscale image
- Noise
- Relative size of the two peaks
- Uniformity of scene illumination / reflectance

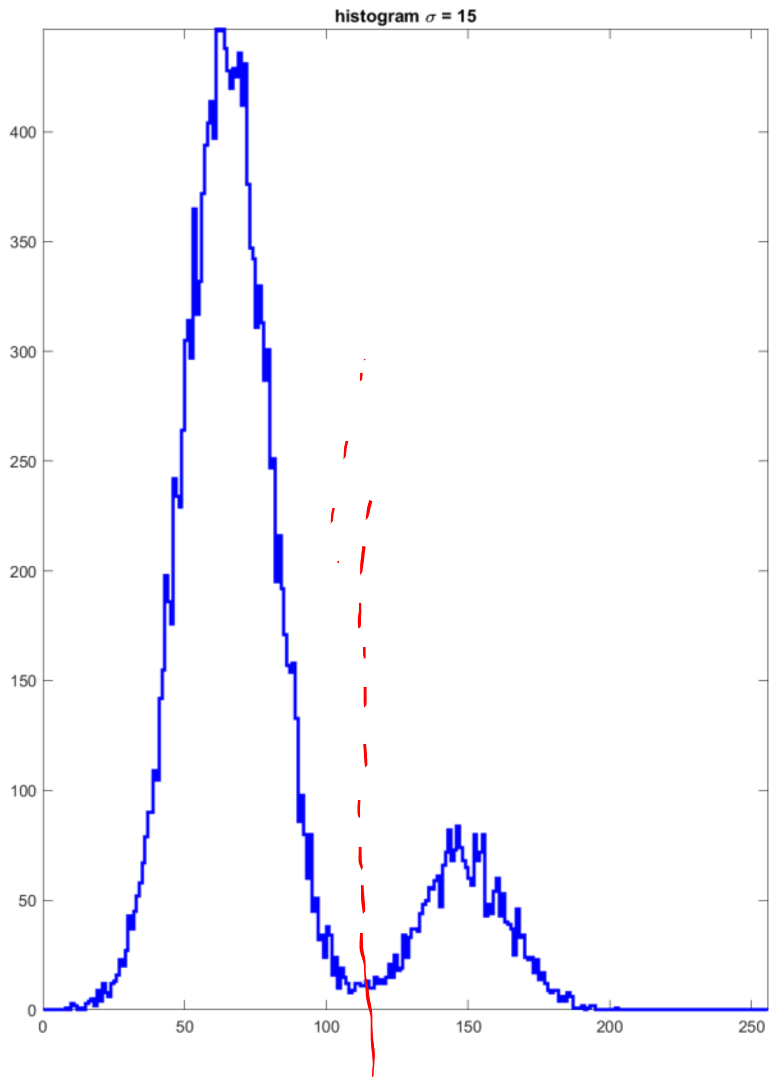
Binary Image



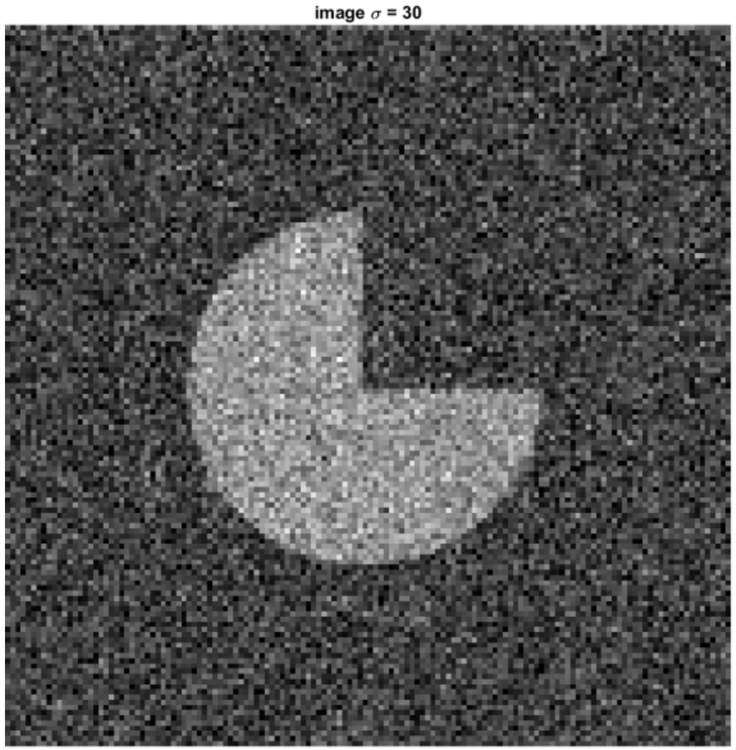


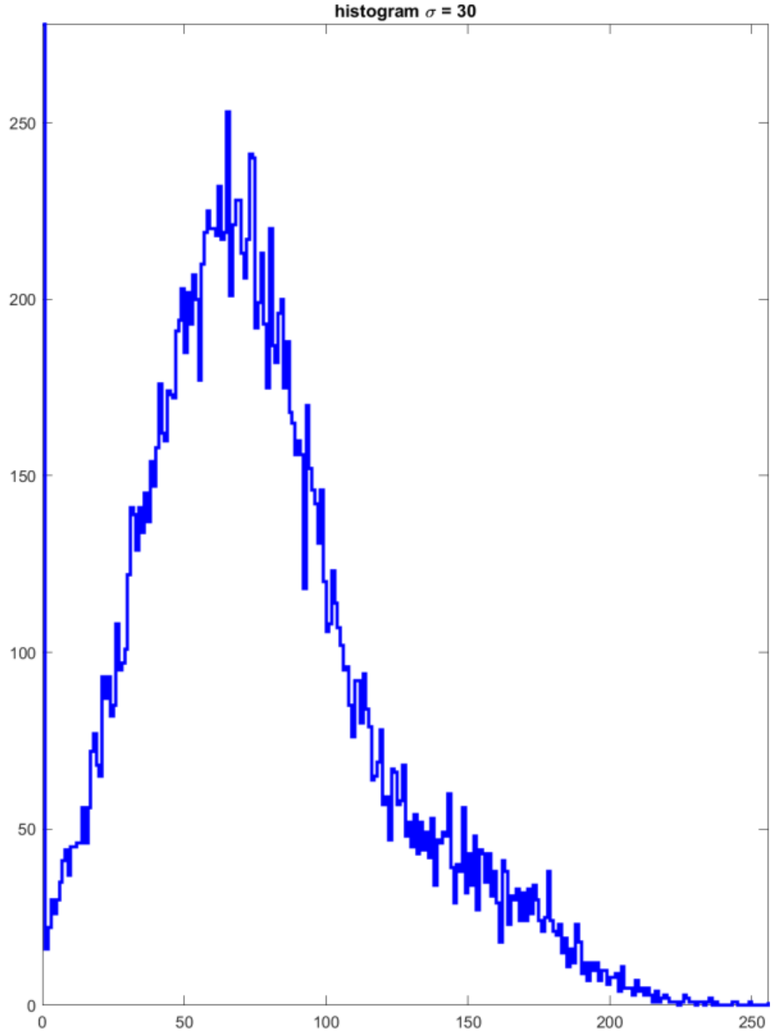
Noisy Binary Image



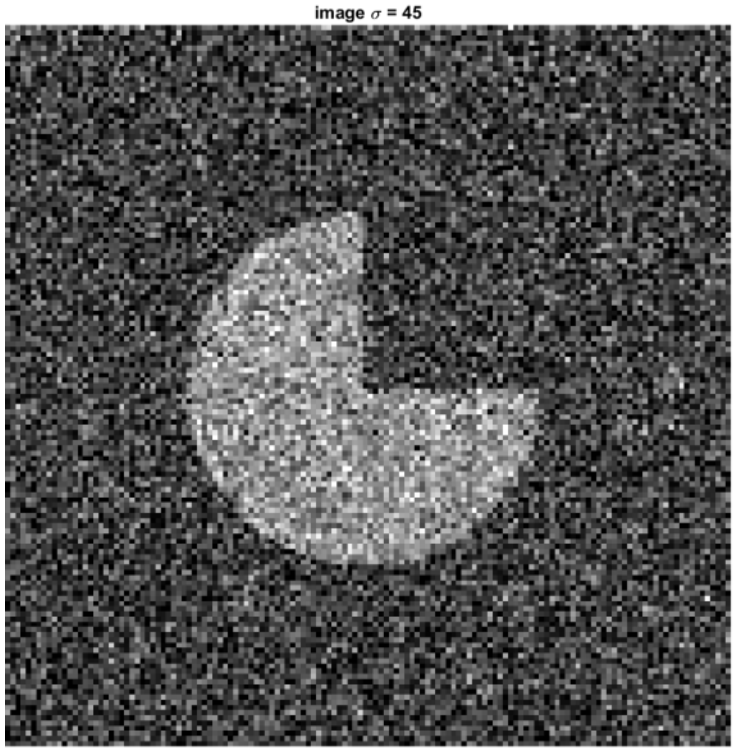


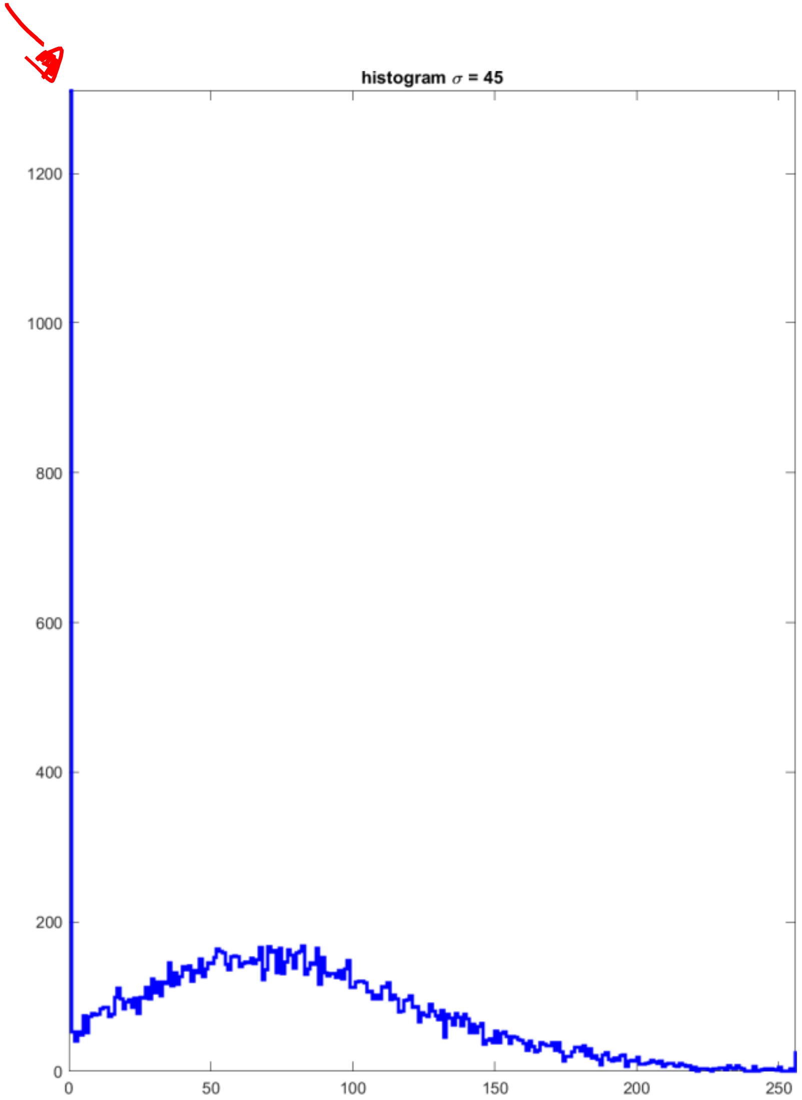
Noisy Binary Image





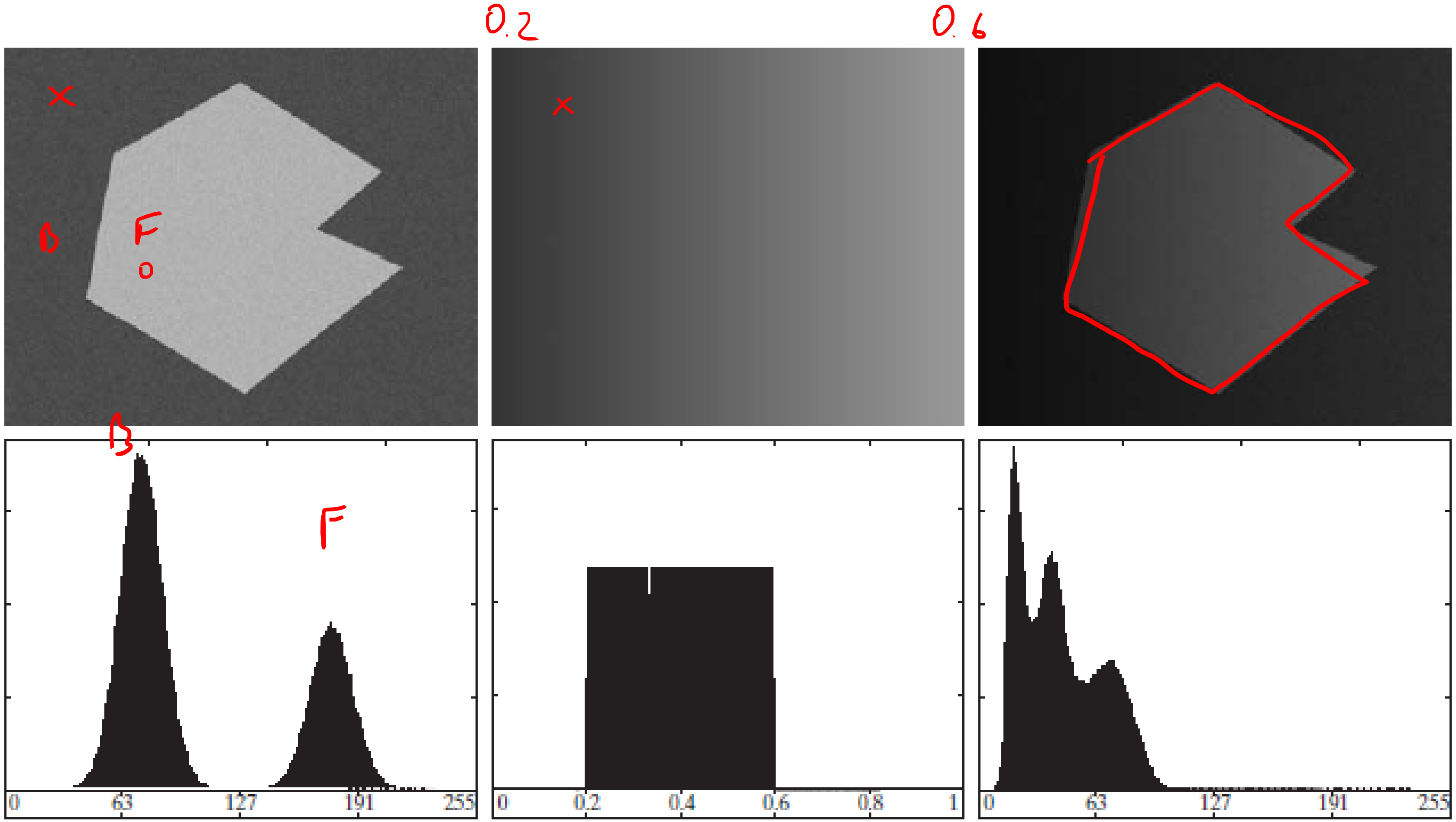
Noisy Binary Image





The impact of light shades on image binarization

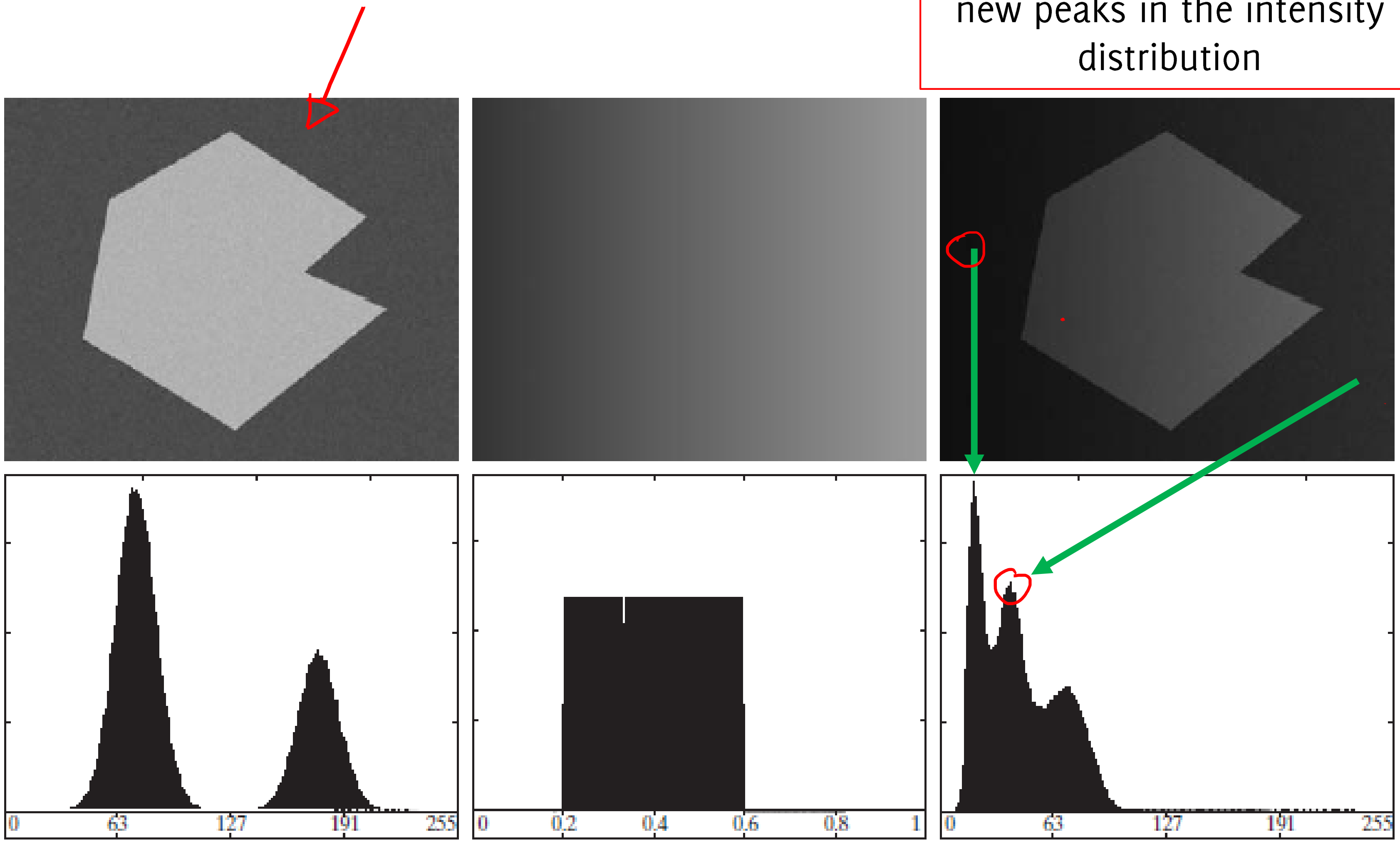
Light shades introduced by point-wise multipliaction



Gonzalez and Woods «Digital image Processing», Prentice Hall;, 3° edition

The impact of light shades on image binarization

Light shades can introduce new peaks in the intensity distribution



Gonzalez and Woods «Digital image Processing», Prentice Hall;, 3° edition

Otsu Metdod

A principled method to **compute thresholds very efficiently**

The threshold estimation problem is formulated as the problem of producing **groups that are "as tight as possible among themselves"**

The criteria to select the threshold is: **minimizing the intra-class distribution**

This is equivalent to maximizing the inter-class variance, which can be done more efficiently using iterative expressions and histogram representation of the image.

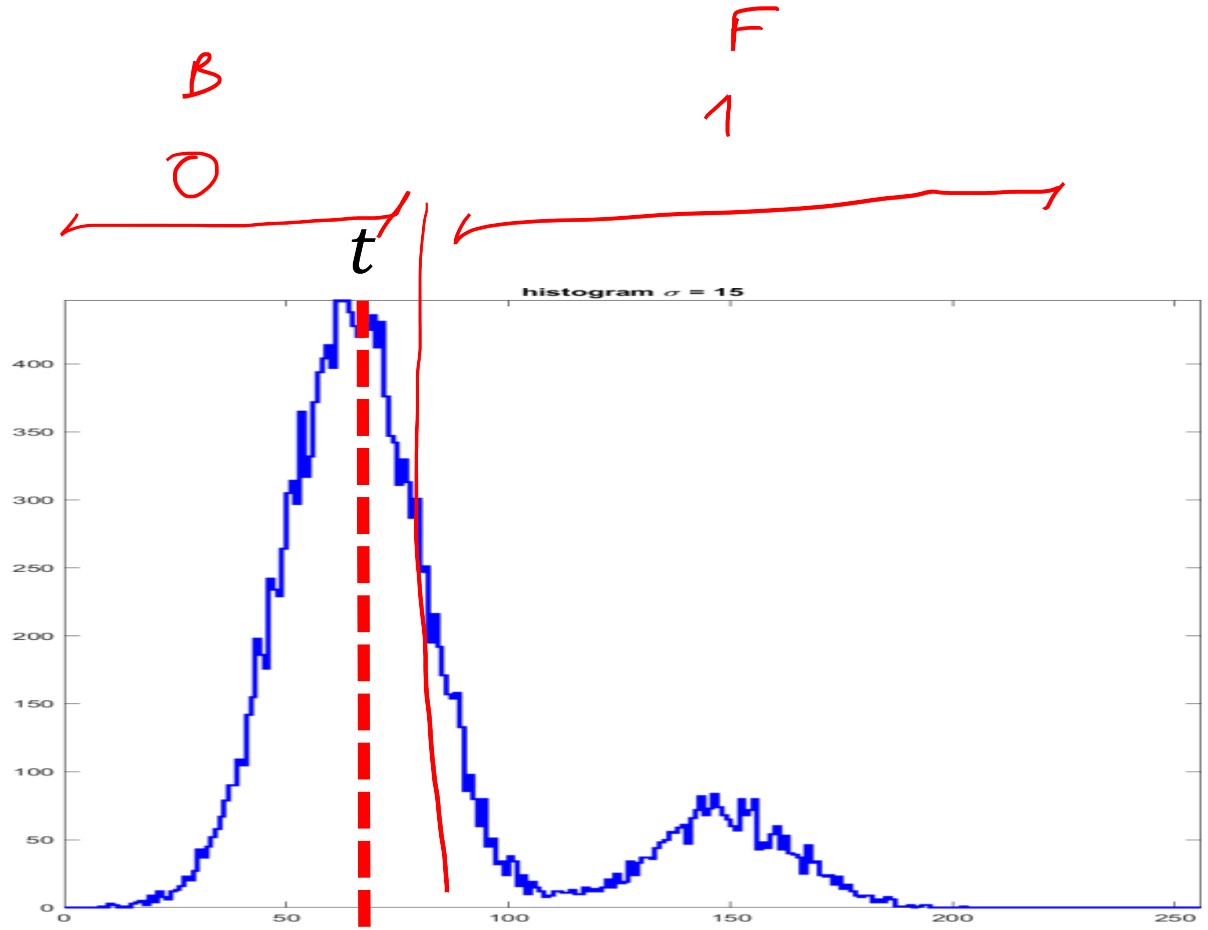
In Python `skimage.filters.threshold_otsu`

N. Otsu, *"A threshold selection method from gray-level histograms", IEEE Trans. Sys., Man., Cyber.*, vol. 9, 1979

Otsu Method

Goal: find the value of t that minimizes the intra-class variance, where

- class "0" means intensity below t
- class "1" means intensity above t



Otsu Method

Otsu computes its threshold as

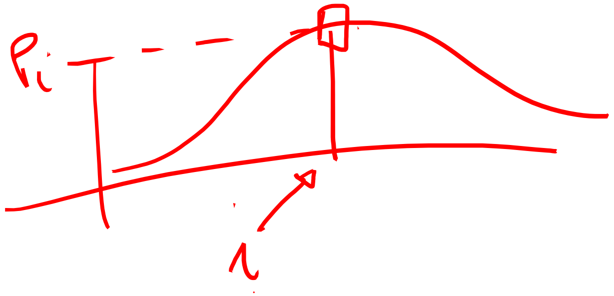
$$t^* = \operatorname*{argmin}_t \boxed{\sigma_w^2(t)}$$

where the intra-class variance is

$$\sigma_w^2(t) = \boxed{w_0}(t)\boxed{\sigma_0^2(t)} + \boxed{w_1}(t)\boxed{\sigma_1^2(t)}$$

And w_0, w_1 are the class proportions

$$\boxed{w_0(t)} = \sum_{i<t} p_i \text{ and } w_1(t) = \sum_{i\geq t} p_i$$



Otsu Method

This is shown to be equivalent to

$$t^* = \operatorname*{argmax}_t \sigma_b^2(t)$$

where

$$\sigma_b^2(t) = \underline{w_0(t)}w_1(t)[\underline{\mu_0(t)} - \underline{\mu_1(t)}]^2$$

$\sigma_b^2(t-1) \mapsto \sigma_b^2(t)$

and

$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} ip_i}{w_0(t)} \text{ and } \mu_1(t) = \frac{\sum_{i=t}^{L-1} ip_i}{w_1(t)}$$

This is much more convenient to compute, as these quantities for different values of t can be quickly computed directly from the histogram

Otsu Method

$w_0(t-1)$, $w_1(t-1)$

$w_0(t) = w_0(t-1) + p_{t-1}$

These quantities can be computed in an incremental manner w.r.t t

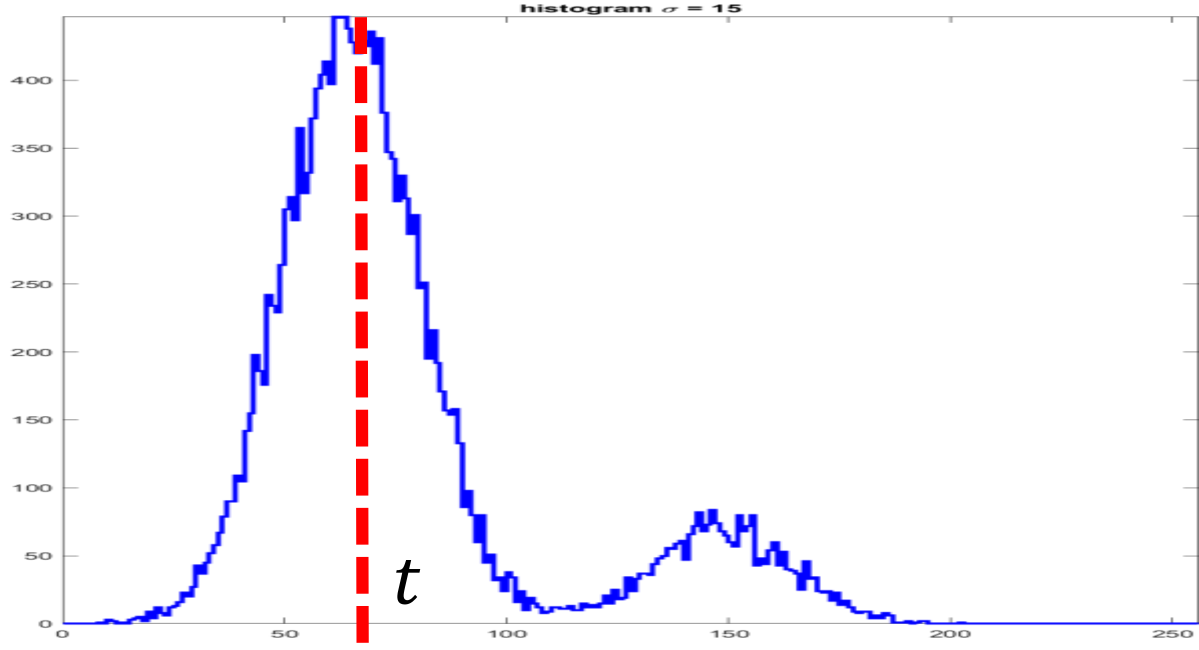
$w_1(t) = w_1(t-1) - p_{t-1}$

$$w_0(t) = \sum_{i<t} p_i \text{ and } w_1(t) = \sum_{i \geq t} p_i$$

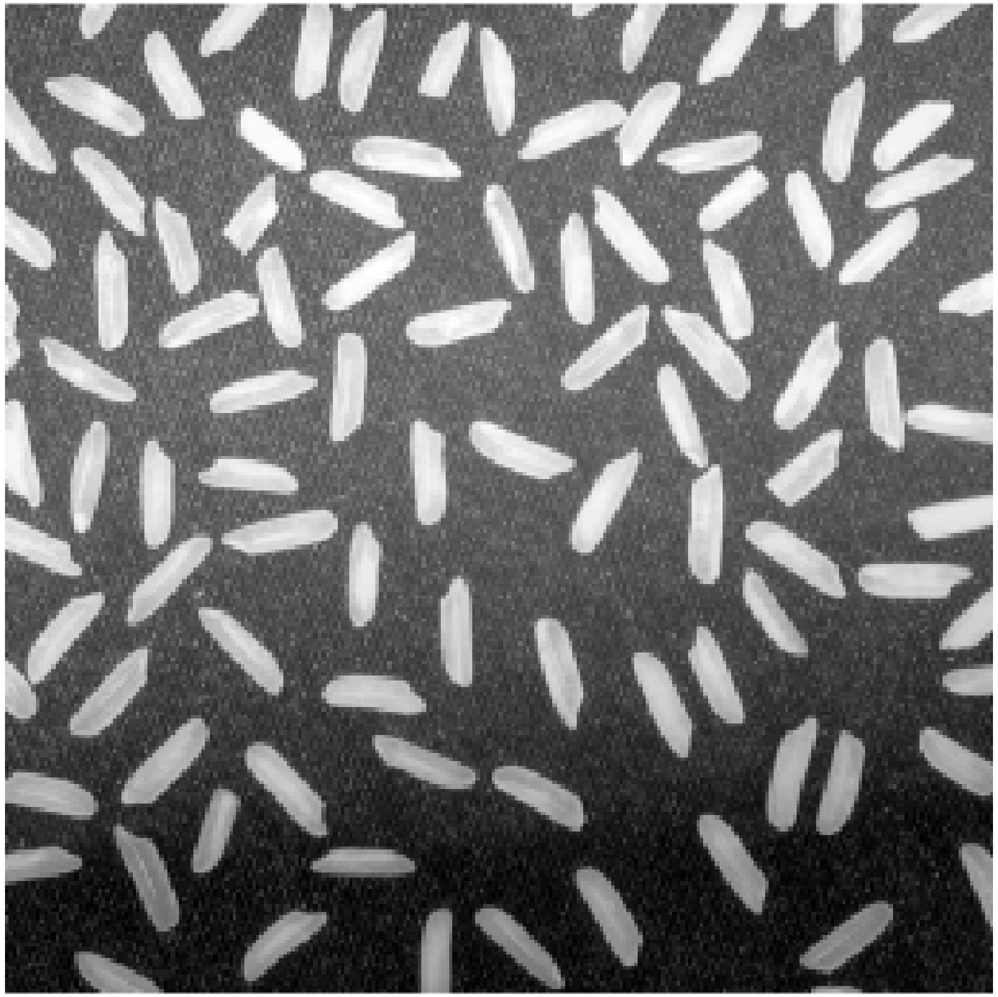
$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} i p_i}{w_0(t)} \text{ and } \mu_1(t) = \frac{\sum_{i=t}^{L-1} i p_i}{w_1(t)}$$

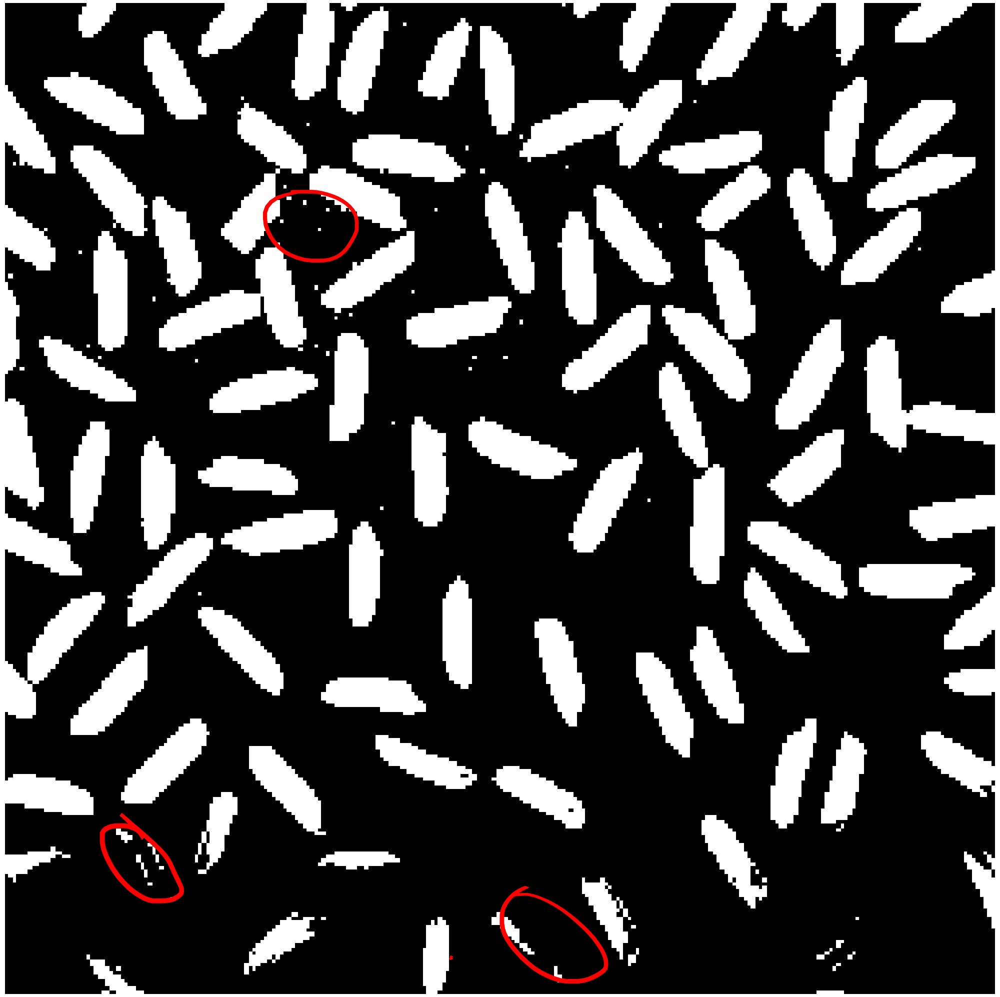
Thus $\sigma_b^2(t)$ can be easily maximized

$T = \arg\max_t \sigma_b^2(t)$



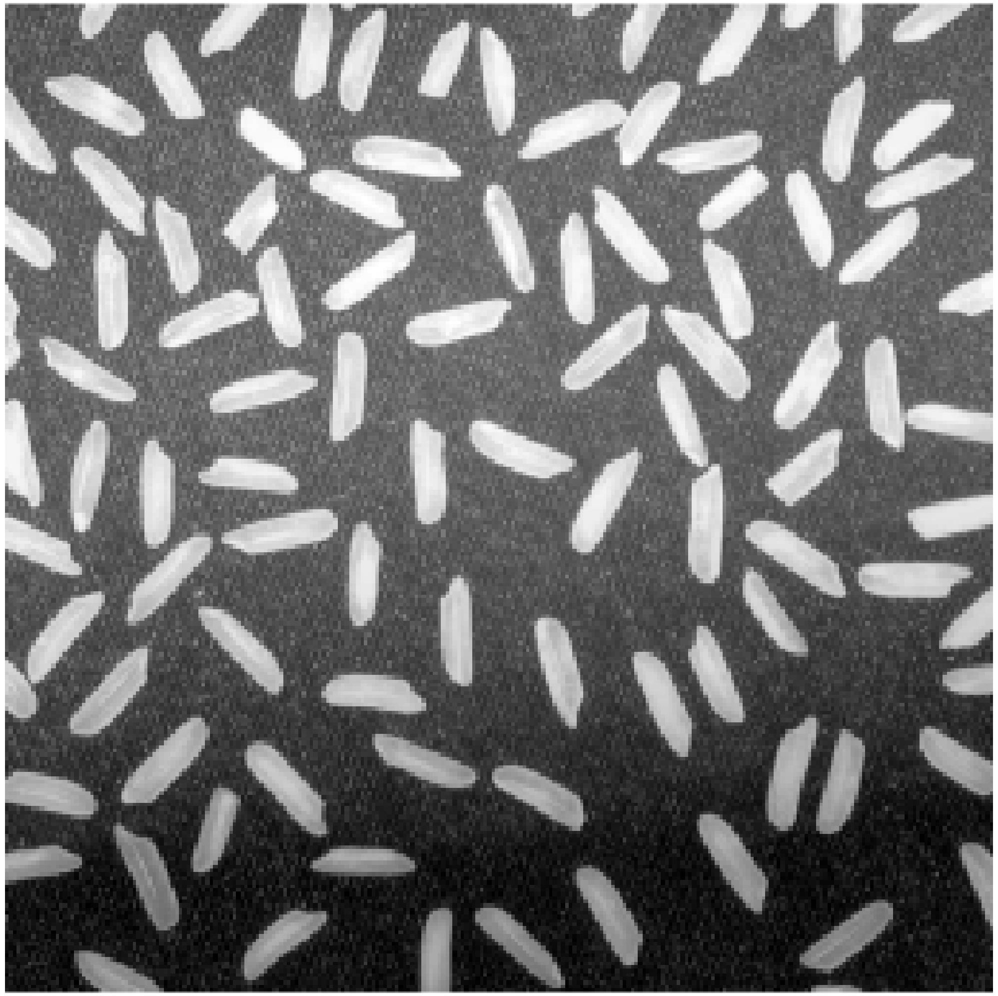
Binarization using a global threshold

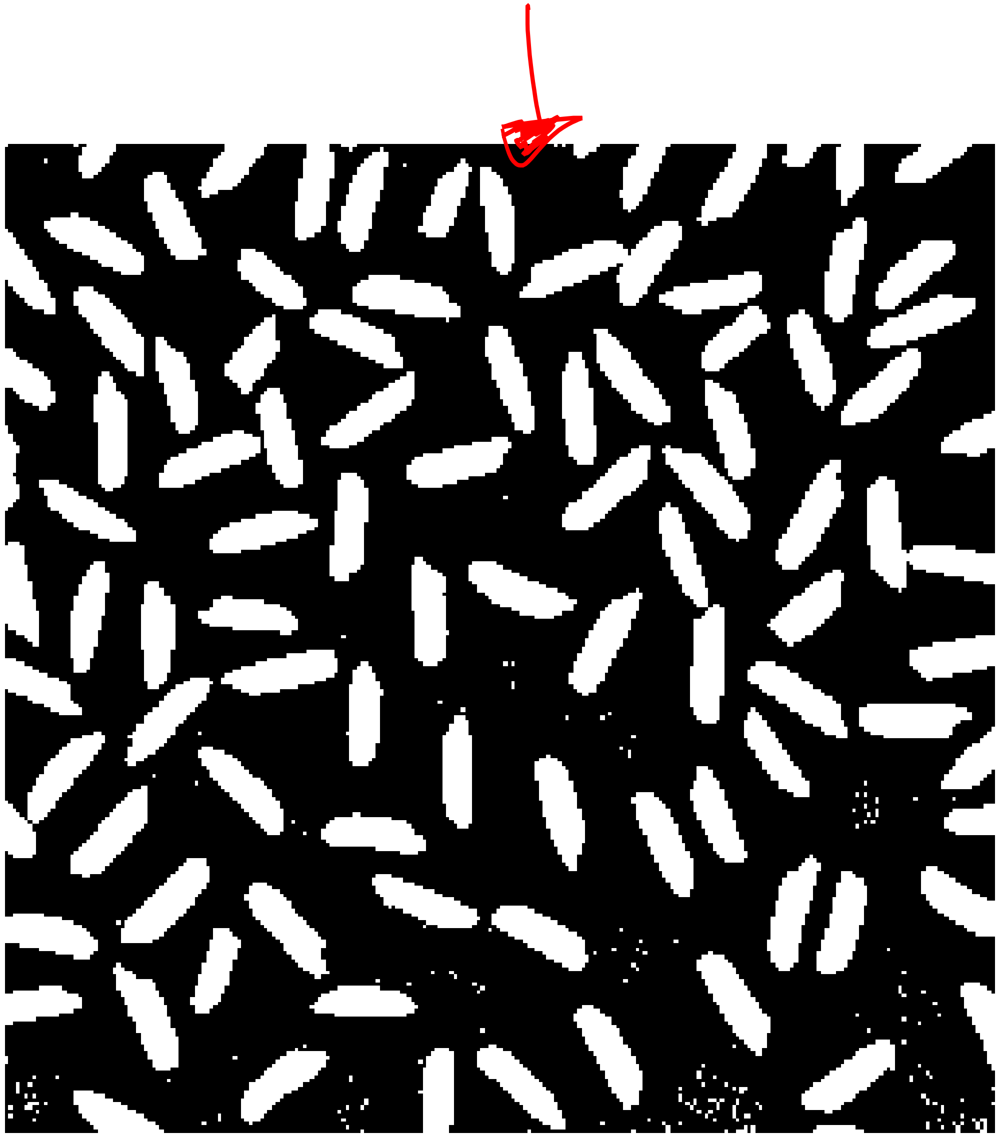




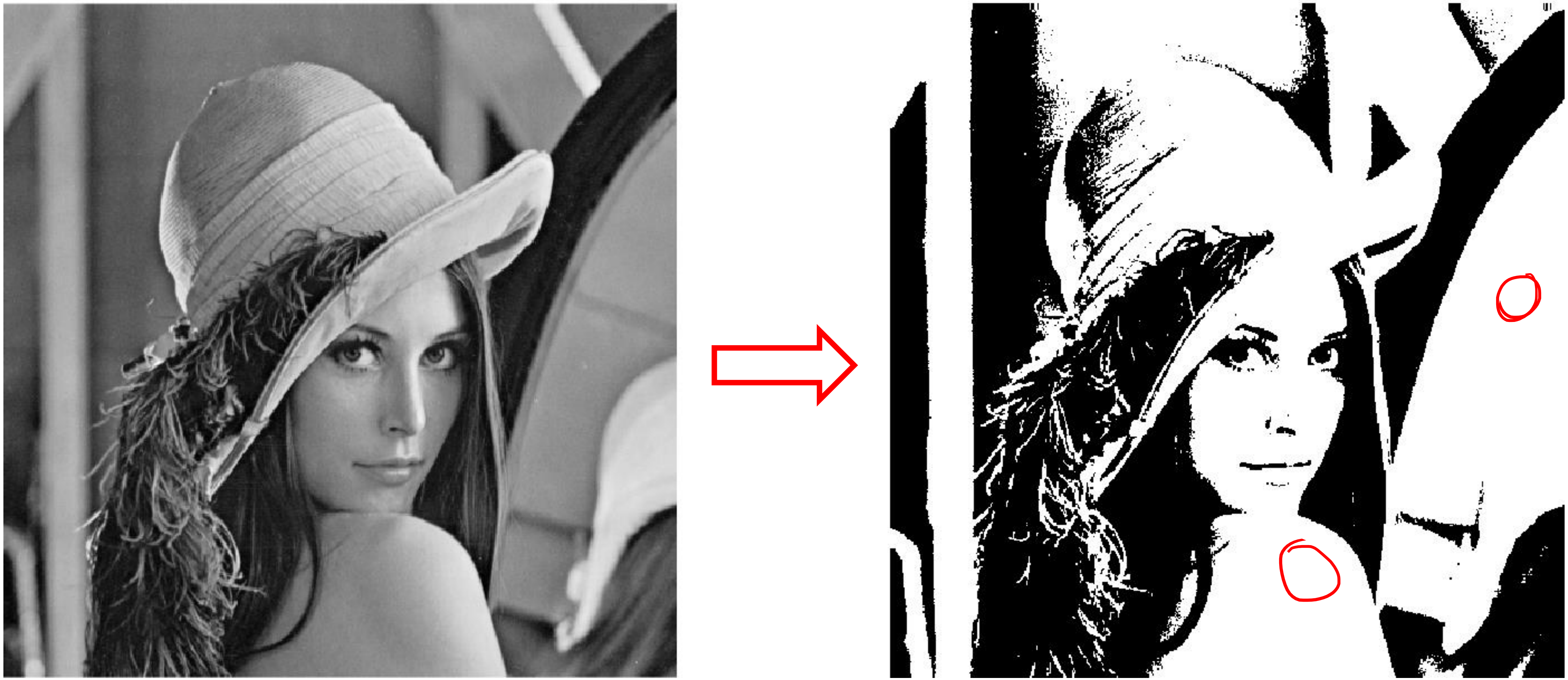
Giacomo Boracchi

Binarization using a local threshold

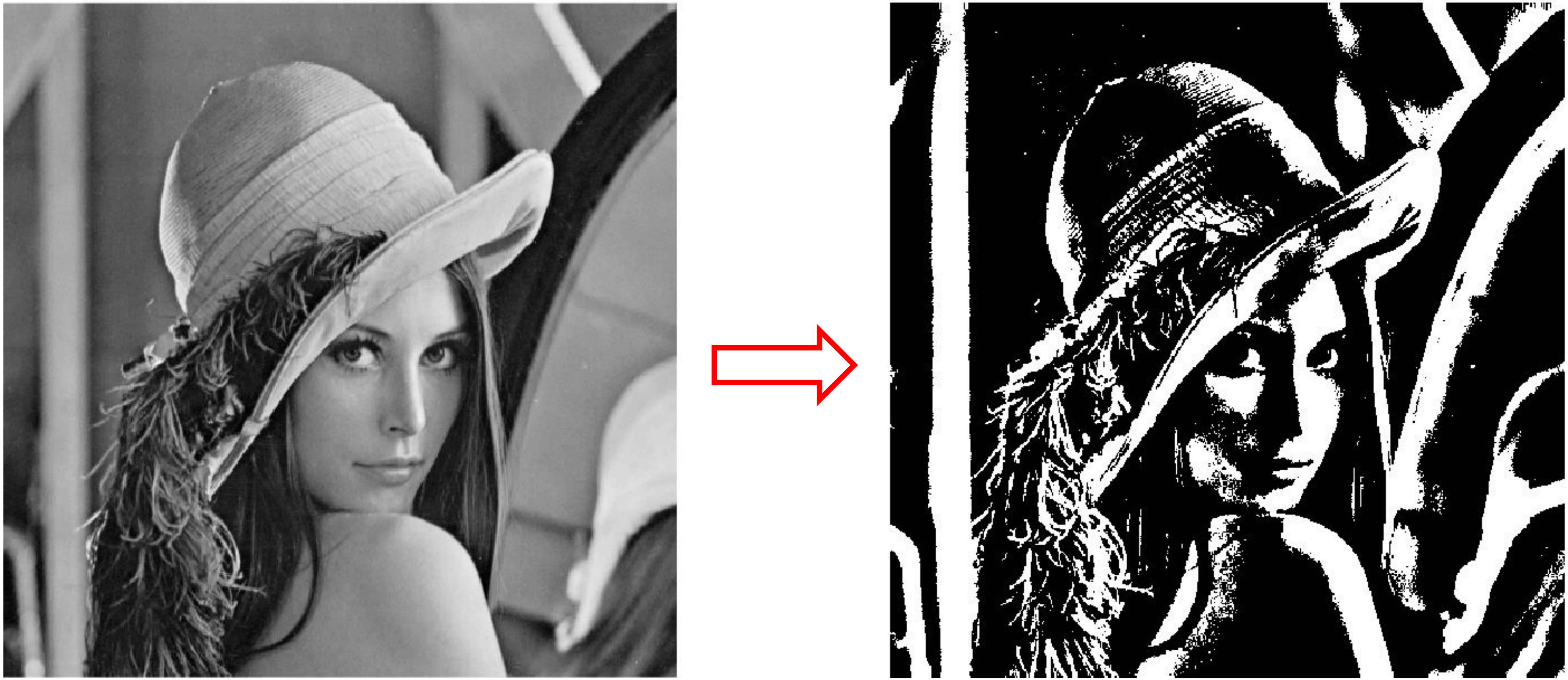




Binarization using a global threshold



Binarization using a local threshold



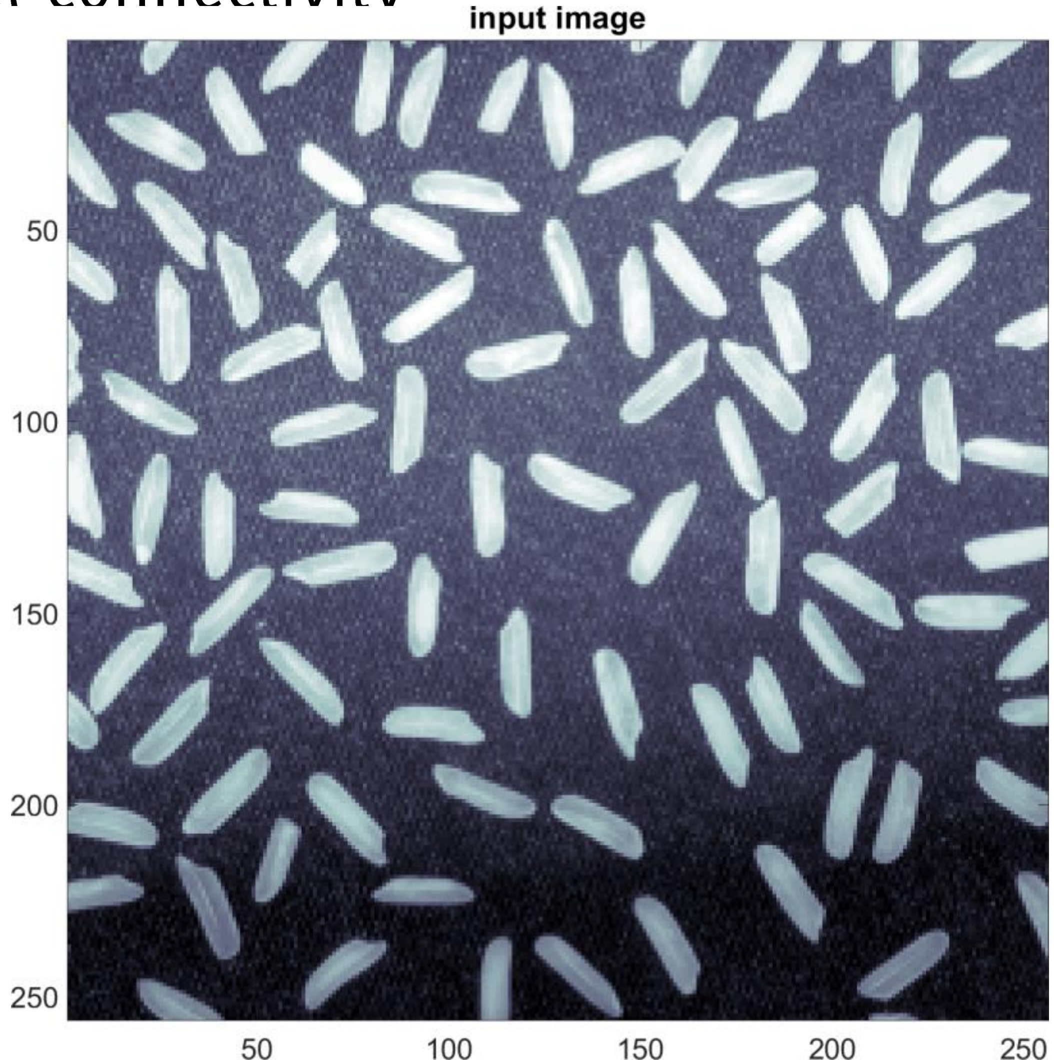
Other Morphological Operations

Blob Labeling, Connected Components

Giacomo Boracchi

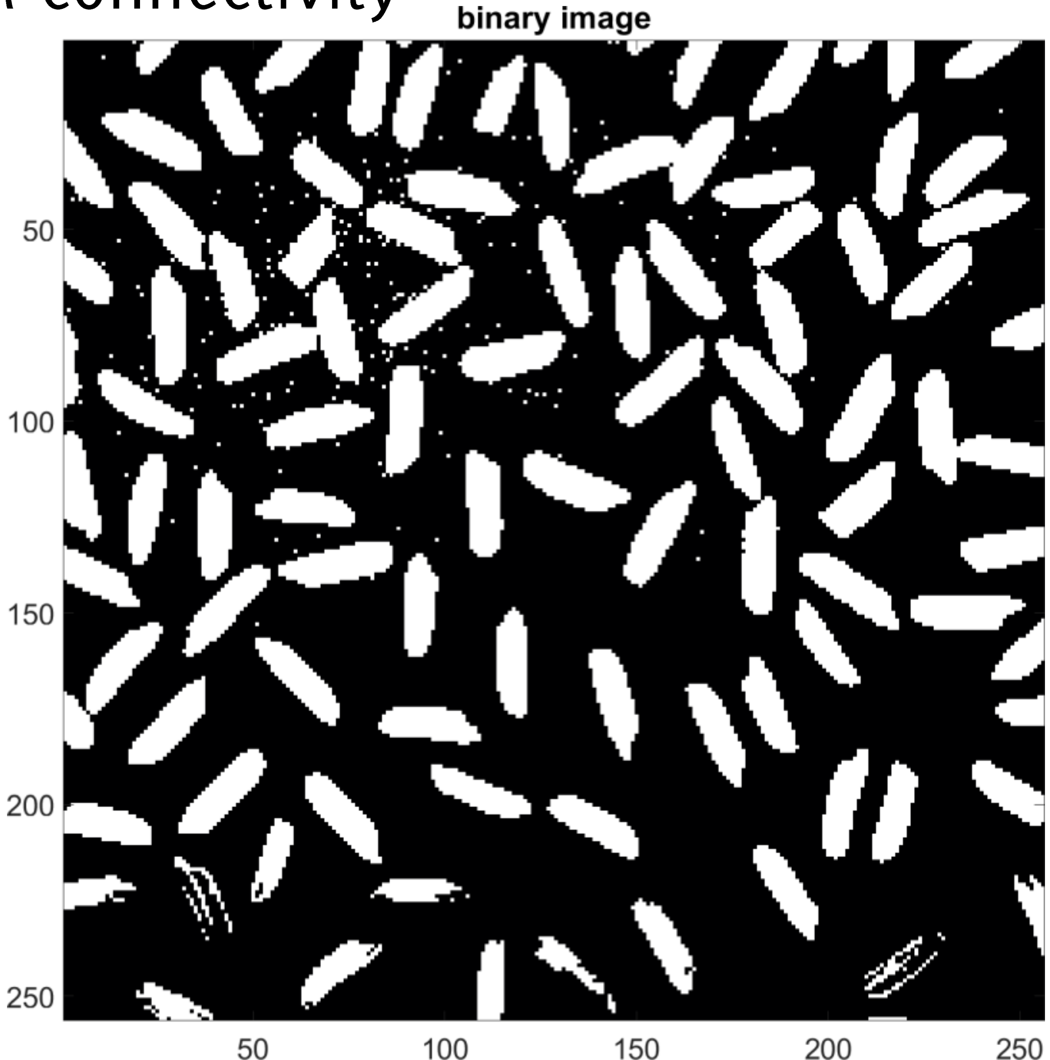
Extraction of connected components

Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity



Connected components

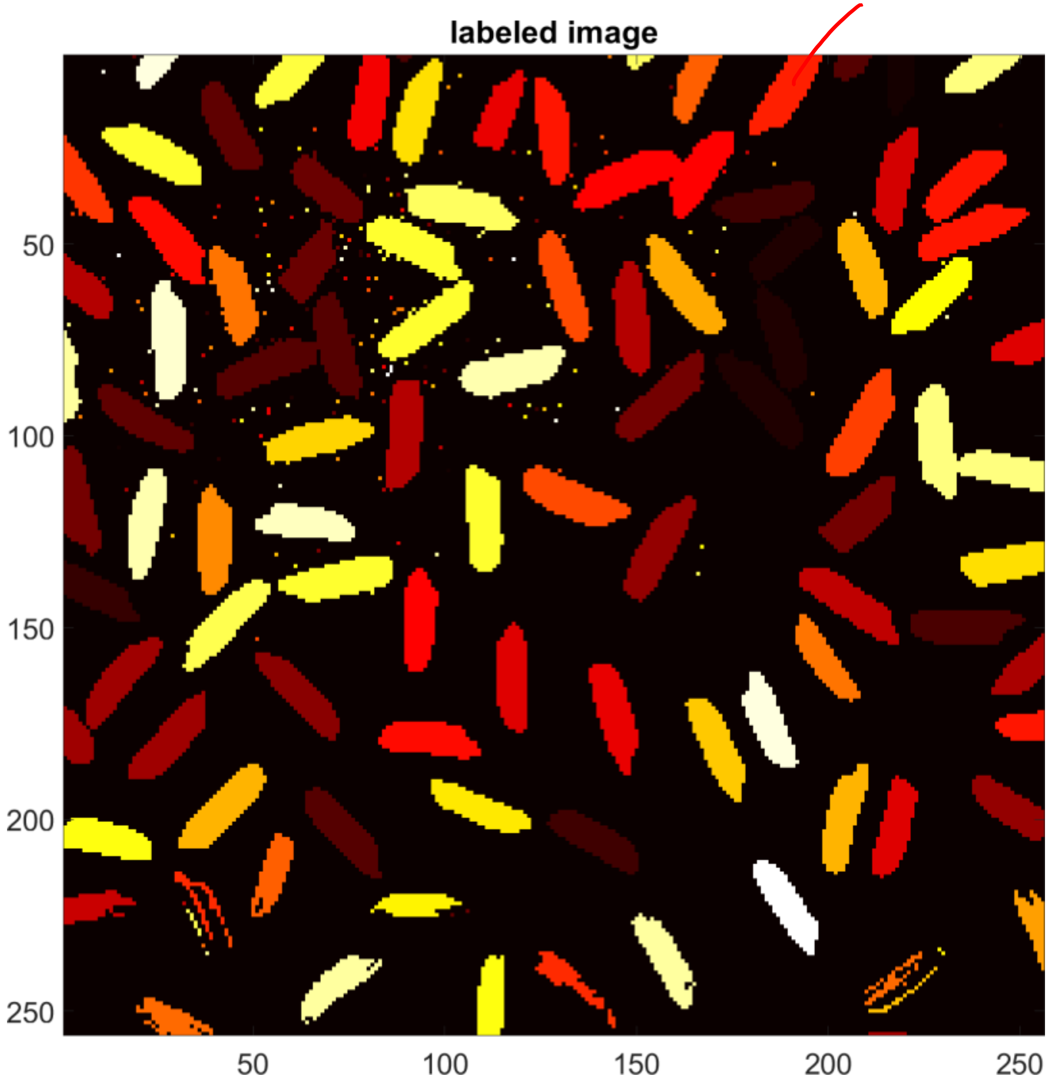
Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity



Connected components

This allows to identify different objects or target in the scene

Each color corresponds
to a different label



Here, each color denotes a different number, i.e. a label.

Connected components

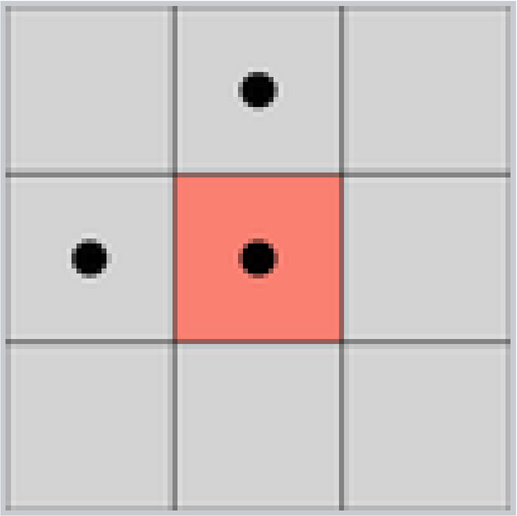
Here, each color denotes a different number, i.e. a label.

It is enough to scan the image twice.

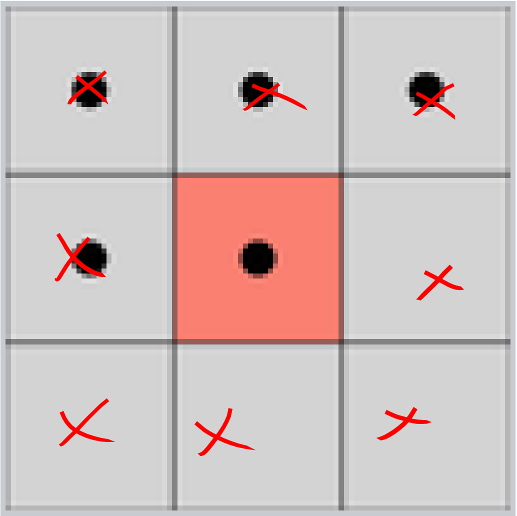
Here, each color denotes a different number, i.e. a label.

Connected components

Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity



4-connectivity



8-connectivity

Two Pass Algorithm: First Pass

Iterate through each pixel (r, c)

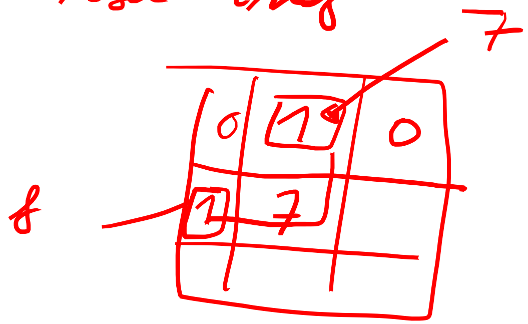
If $I(r, c) == 1$

- Get a neighbor $U_{(r,c)}$ of (r, c)
- If $I(u, v) == 0 \;\; \forall (u, v) \in U_{(r,c)}$
  - Assign a new label $\boxed{L(r, c)}$
- Else $L(r, c) = \min\big(L(u, v)\big)$ over $U_{(r,c)}$
  - If there are different labels in $U_{(r,c)}$
    - Record they are equivalent in a table

In Python **`skimage.measure.label`**

I "binarized image"

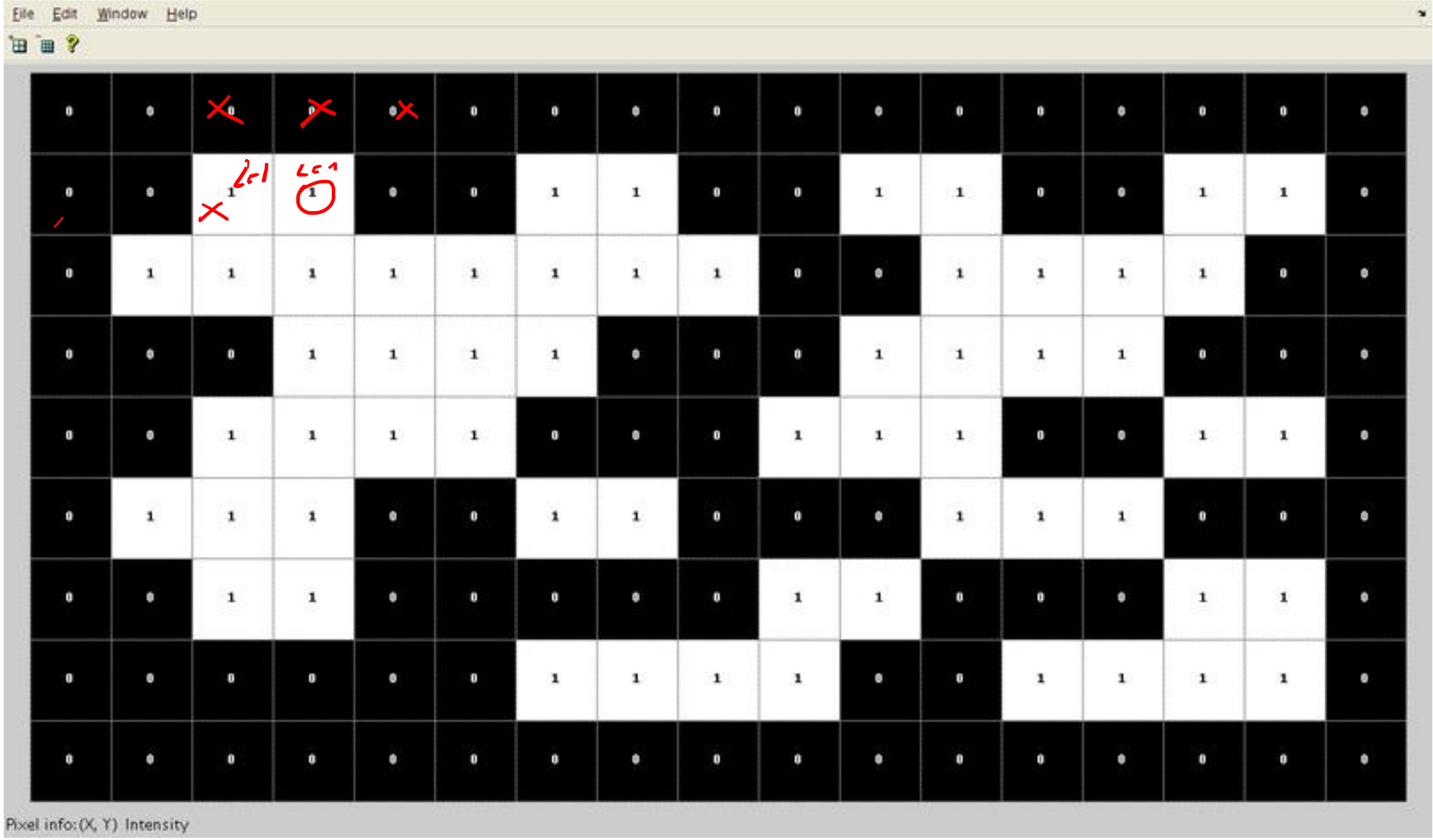
L "label image"



$7 = 8$

Binary input image

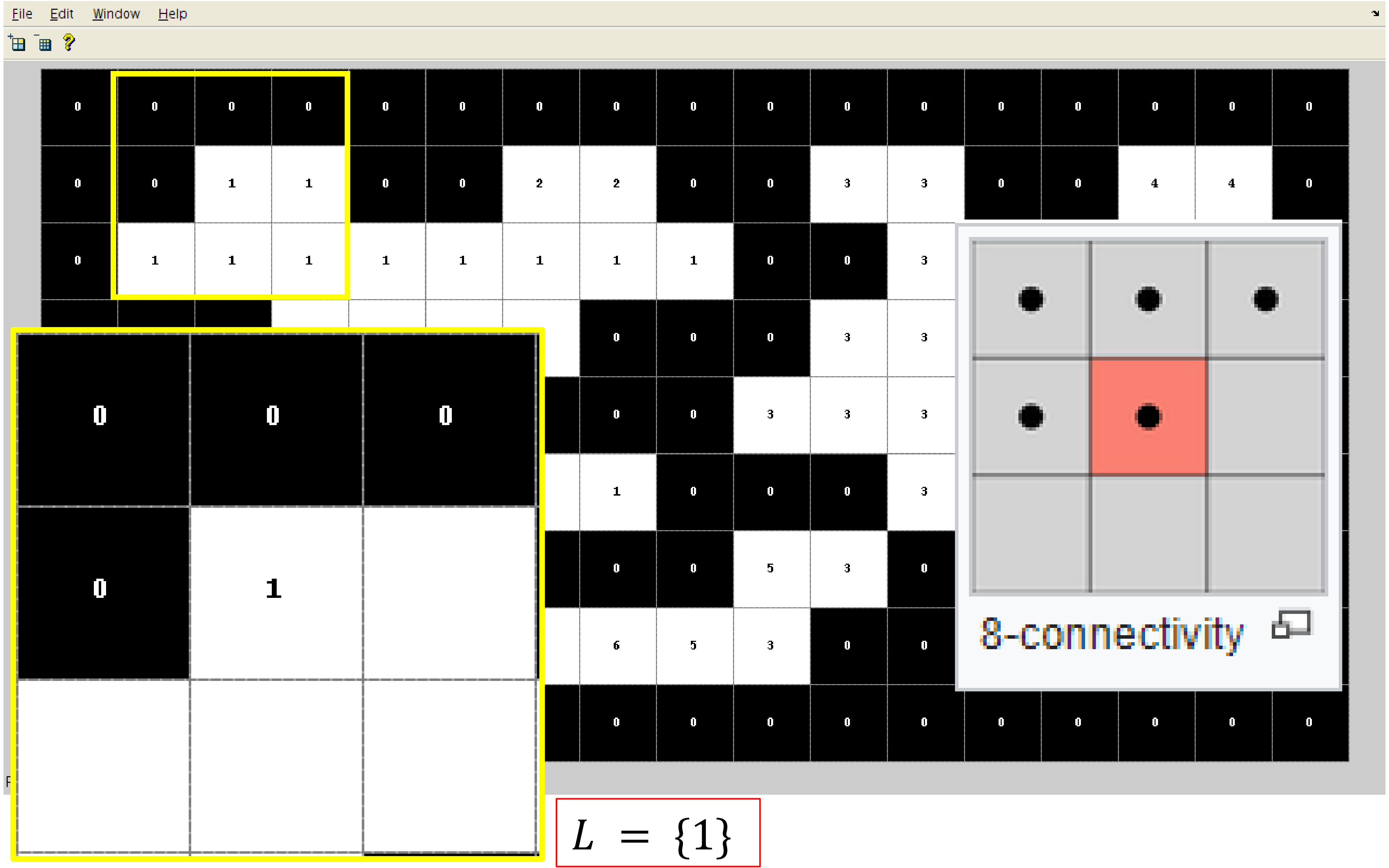
I L



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0
0	1	1	1	1	1	1	1	1	0	0	1	1	1	1	0	0
0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	0	1	1	1	0	0	1	1	0
0	1	1	1	0	0	1	1	0	0	0	1	1	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	0	0	0	1	1	0
0	0	0	0	0	0	1	1	1	1	0	0	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

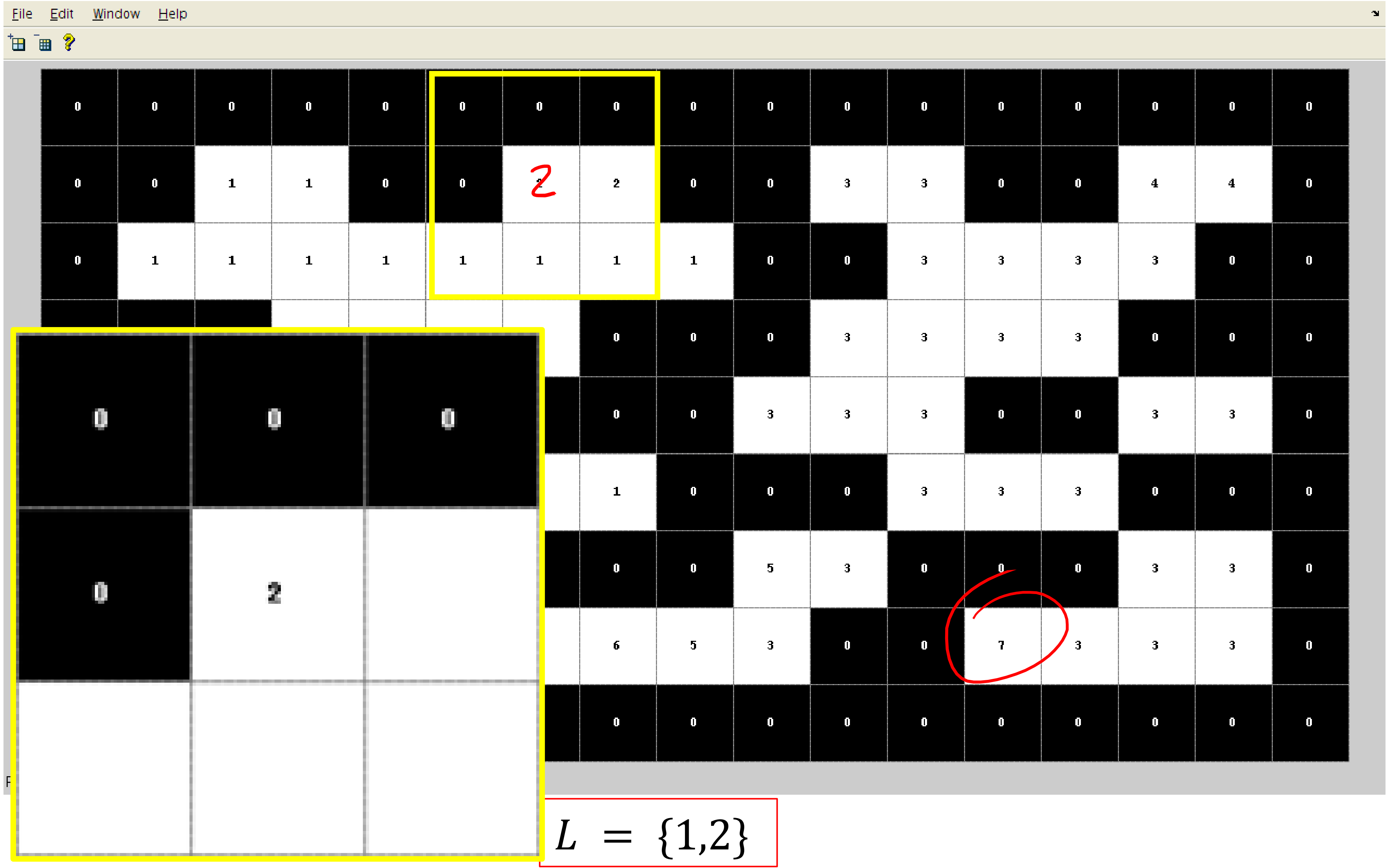
By Dhull003 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166860

Iterations of the first pass



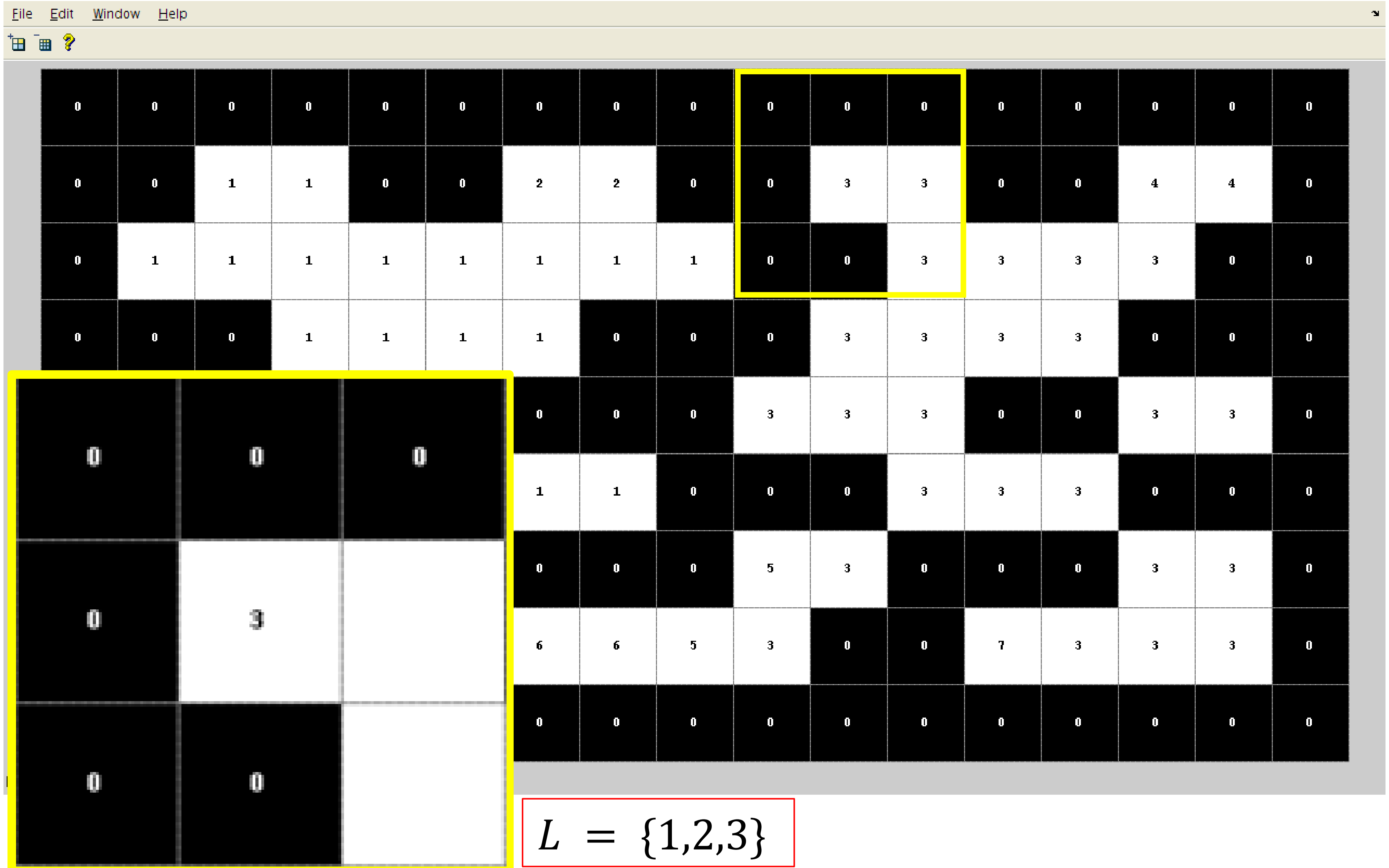
$L = \{1\}$

Iterations of the first pass



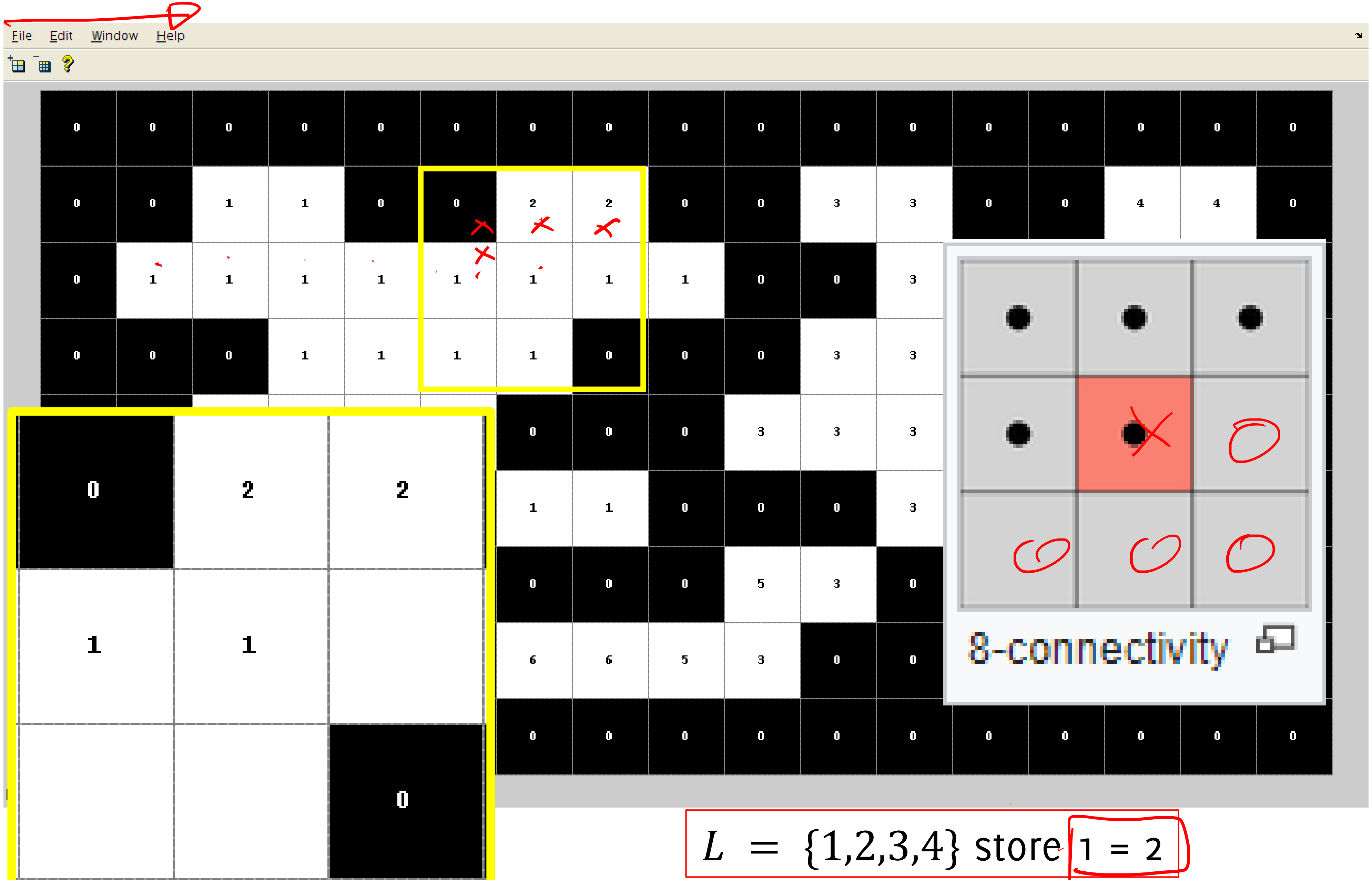
$L = \{1,2\}$

Iterations of the first pass



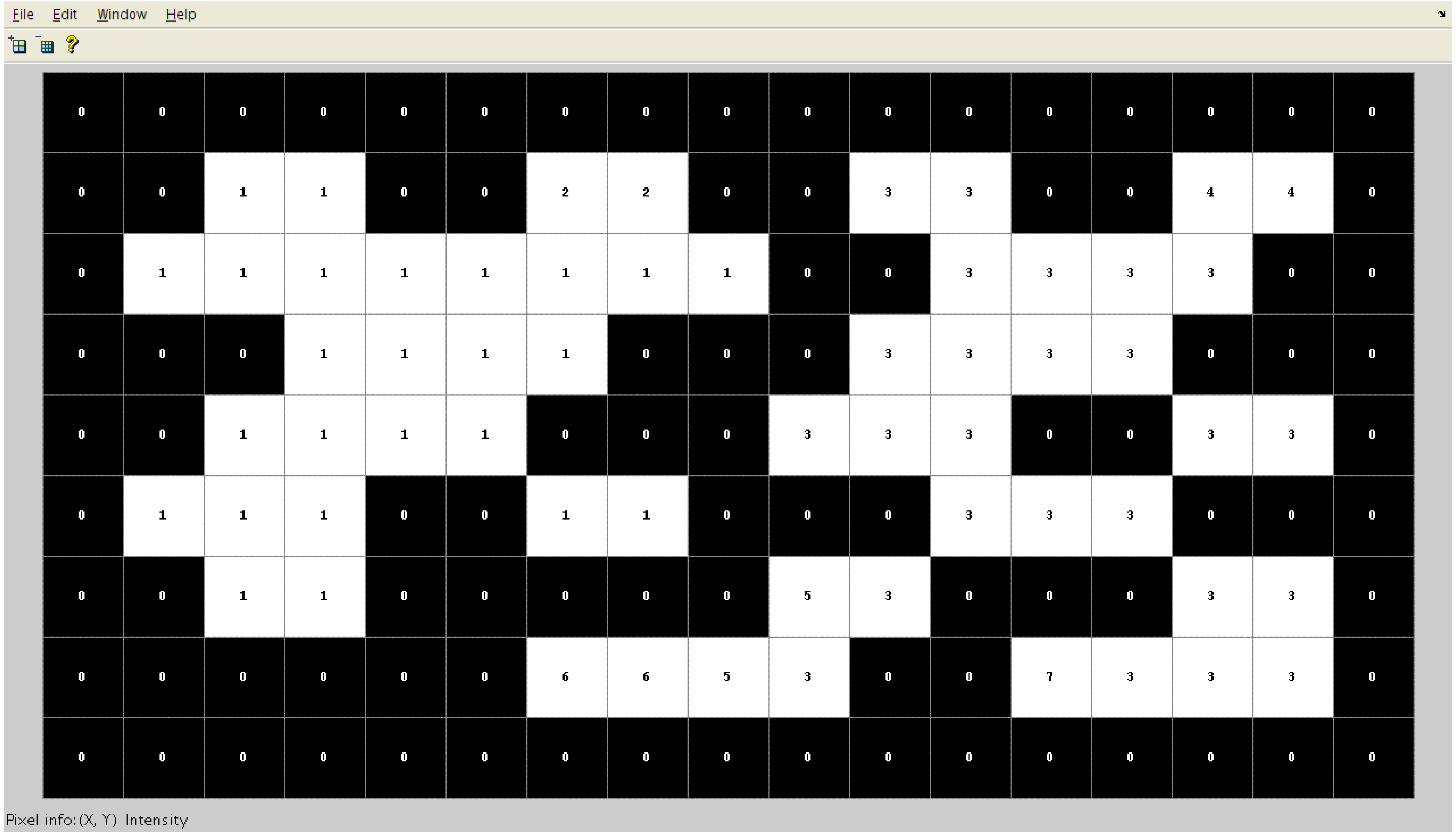
$L = \{1,2,3\}$

Iterations of the first pass



$L = \{1,2,3,4\}$ store $1 = 2$

Output of the first pass



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	2	2	0	0	3	3	0	0	4	4	0
0	1	1	1	1	1	1	1	1	0	0	3	3	3	3	0	0
0	0	0	1	1	1	1	0	0	0	3	3	3	3	0	0	0
0	0	1	1	1	1	0	0	0	3	3	3	0	0	3	3	0
0	1	1	1	0	0	1	1	0	0	0	3	3	3	0	0	0
0	0	1	1	0	0	0	0	0	5	3	0	0	0	3	3	0
0	0	0	0	0	0	6	6	5	3	0	0	7	3	3	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

$L = \{1,2,3,4,5,6,7\}$ equivalence sets $\{1,2\}$, $\{3,4,5,6,7\}$

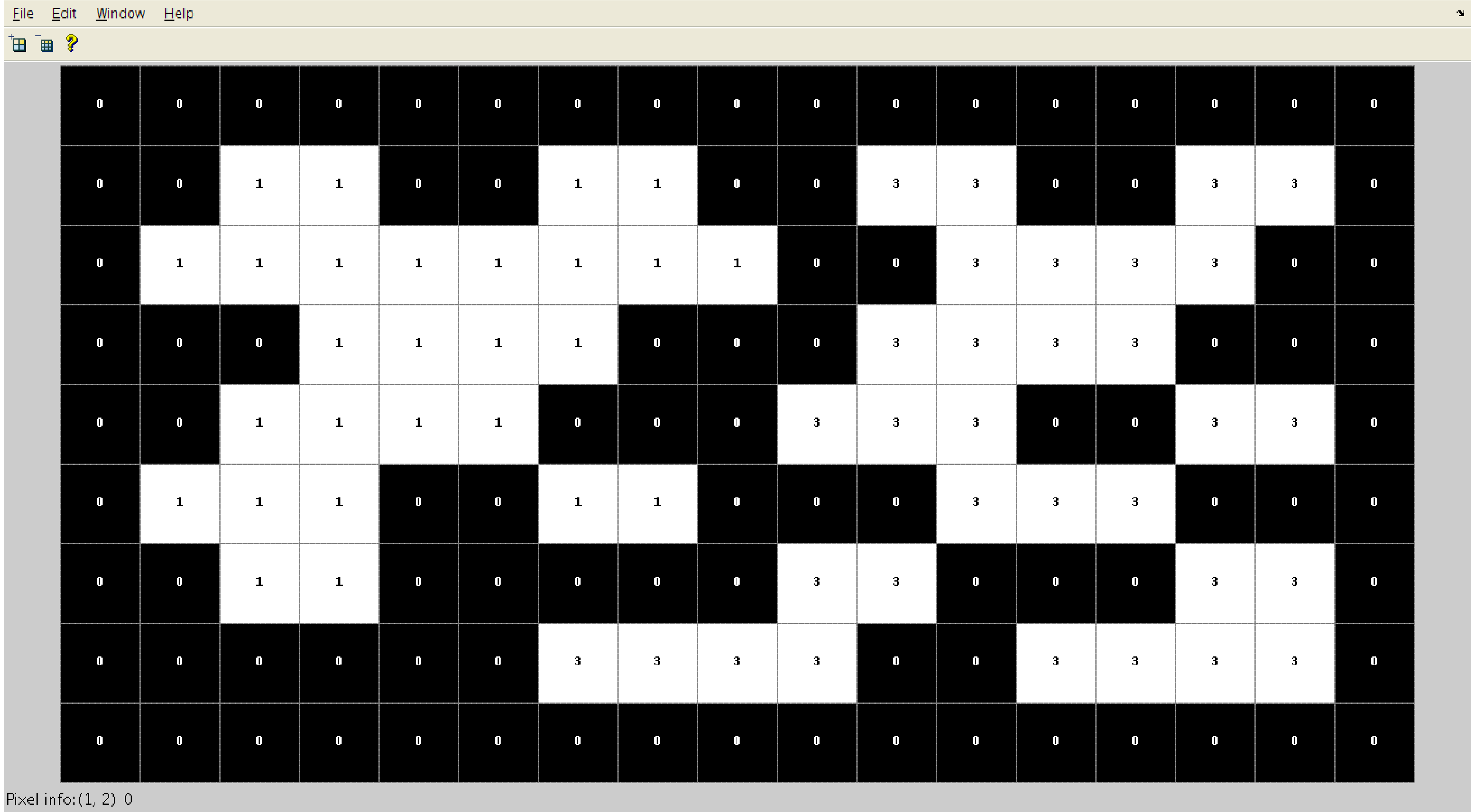
Two Pass Algorithm: Second Pass

Iterate through each pixel (r, c)

If $I(r, c) == 1$

Relabel the element with the lowest equivalent label

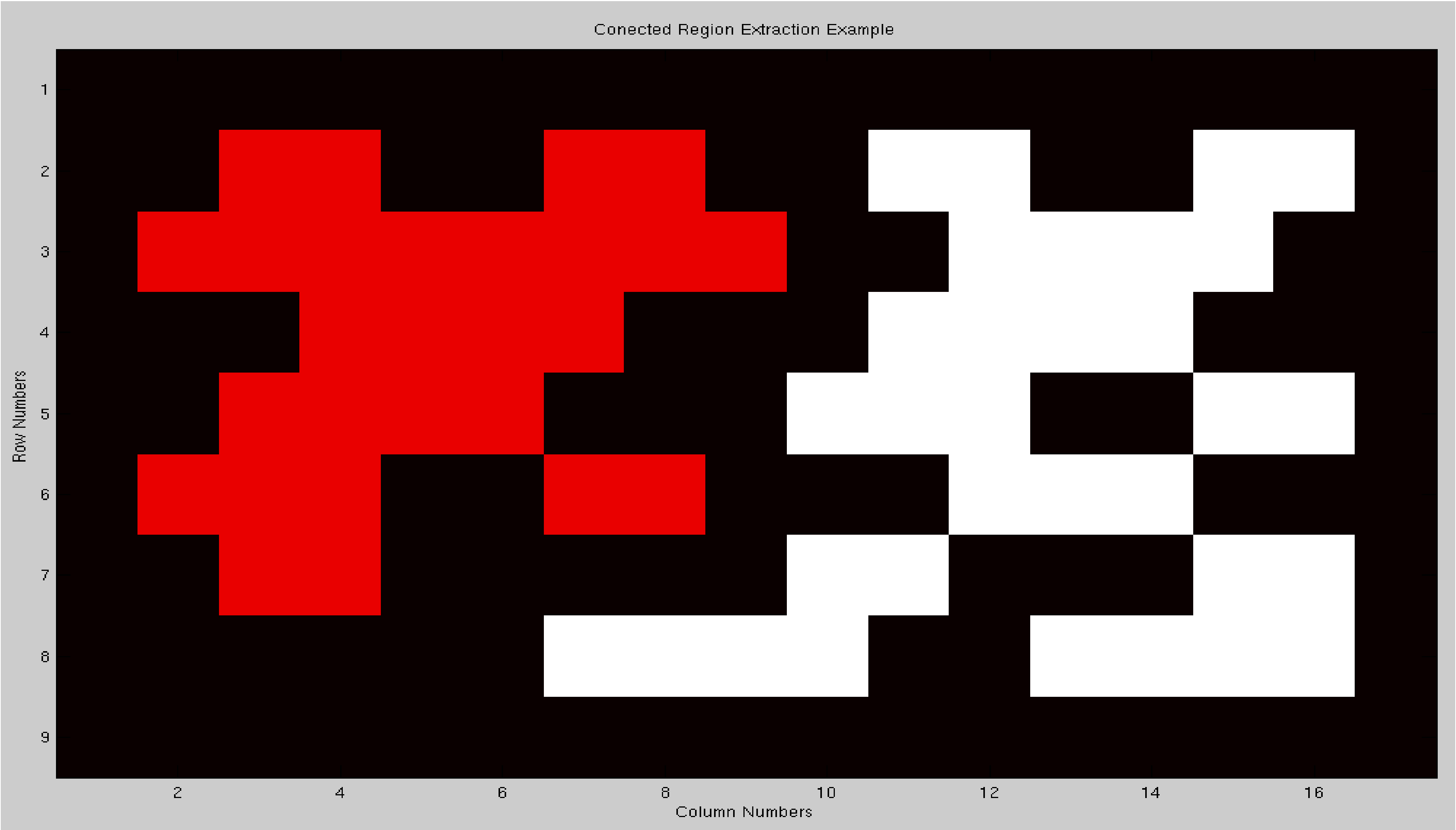
Output of the Second Pass



0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	3	3	0	0	3	3	0
0	1	1	1	1	1	1	1	1	0	0	3	3	3	3	0	0
0	0	0	1	1	1	1	0	0	0	3	3	3	3	0	0	0
0	0	1	1	1	1	0	0	0	3	3	3	0	0	3	3	0
0	1	1	1	0	0	1	1	0	0	0	3	3	3	0	0	0
0	0	1	1	0	0	0	0	0	3	3	0	0	0	3	3	0
0	0	0	0	0	0	3	3	3	3	0	0	3	3	3	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

By Dhull003 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166888

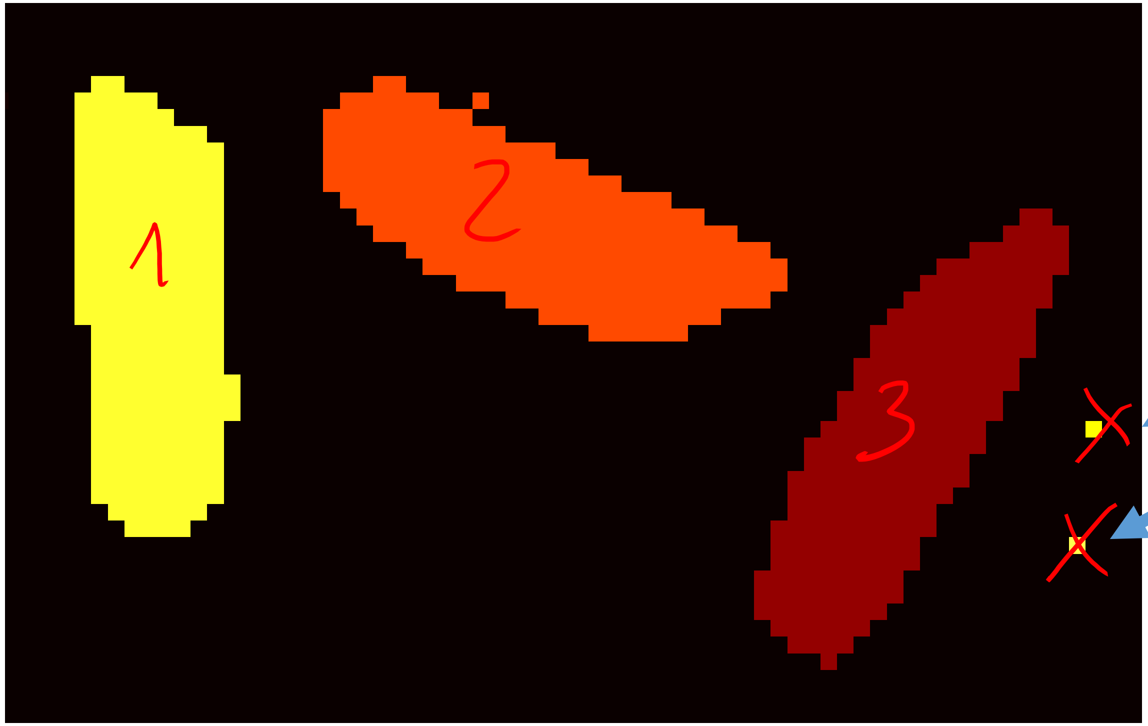
Output of the Second Pass



By Dhull003 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166888

Bounding Box vs Axis

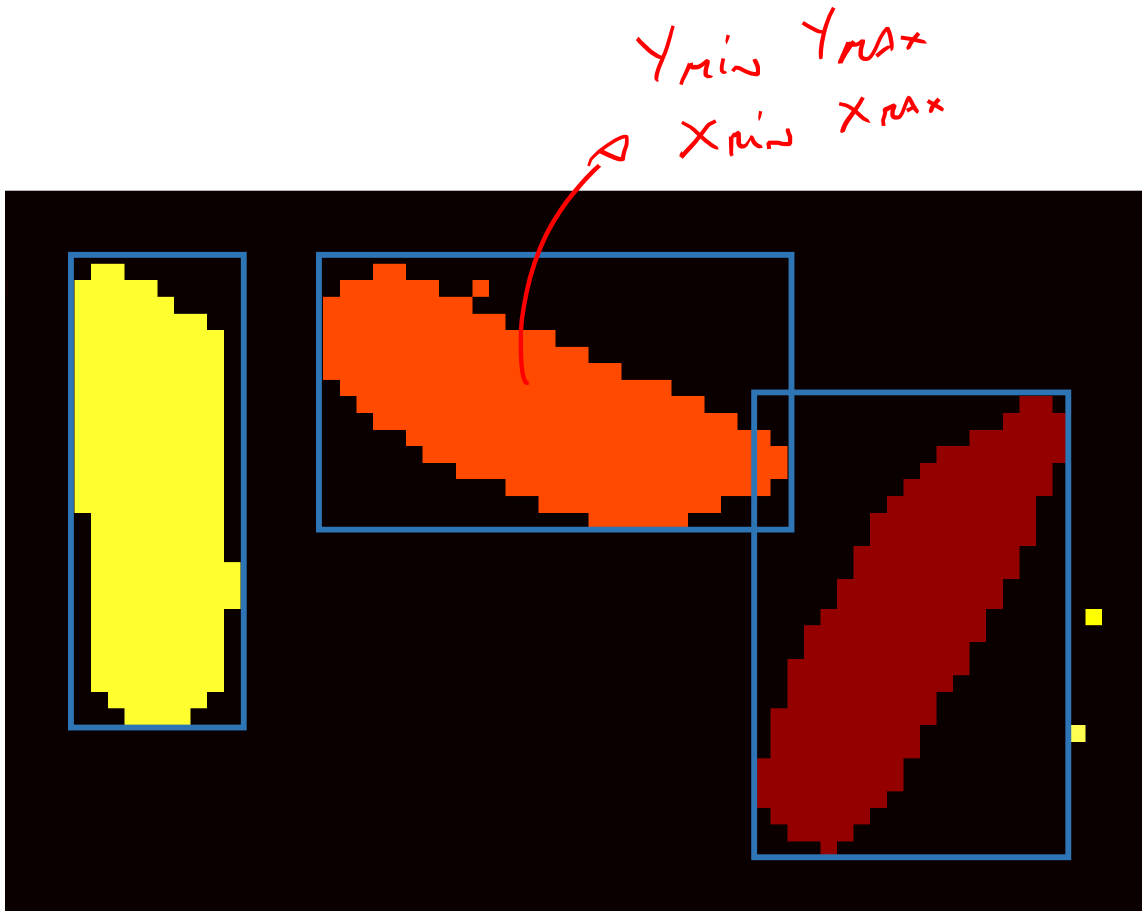
These provide information about size and orientation of the object



Too small, can be removed analyzing the area

Bounding Box vs Axis

These provide information about size and orientation of the object

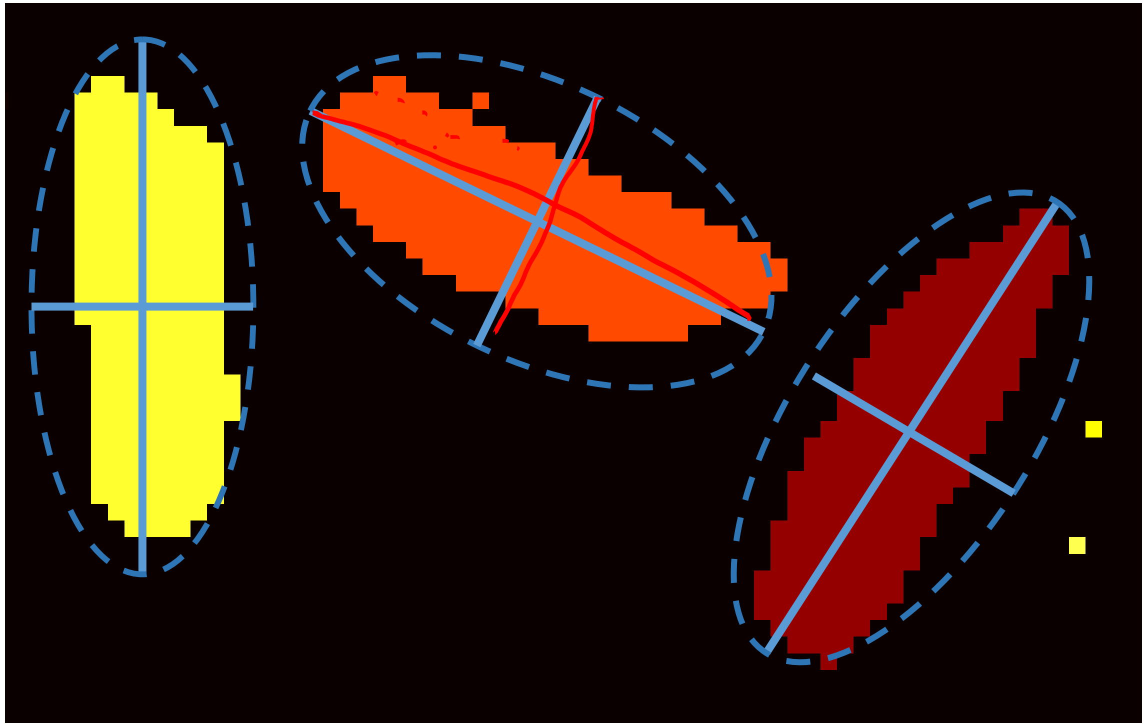


Bounding box allow to crop separate images for each component

(these are defined as the range of values for each coordinate)

Bounding Box vs Axis

These provide information about size and orientation of the object



Blob axis are computer as axis the of the ellipse that has the same second-moments as the region.

In Python: **`skimage.measure.regionprops`**

Features from Morphological Image Processing

Now operations can be performed on each blob individually

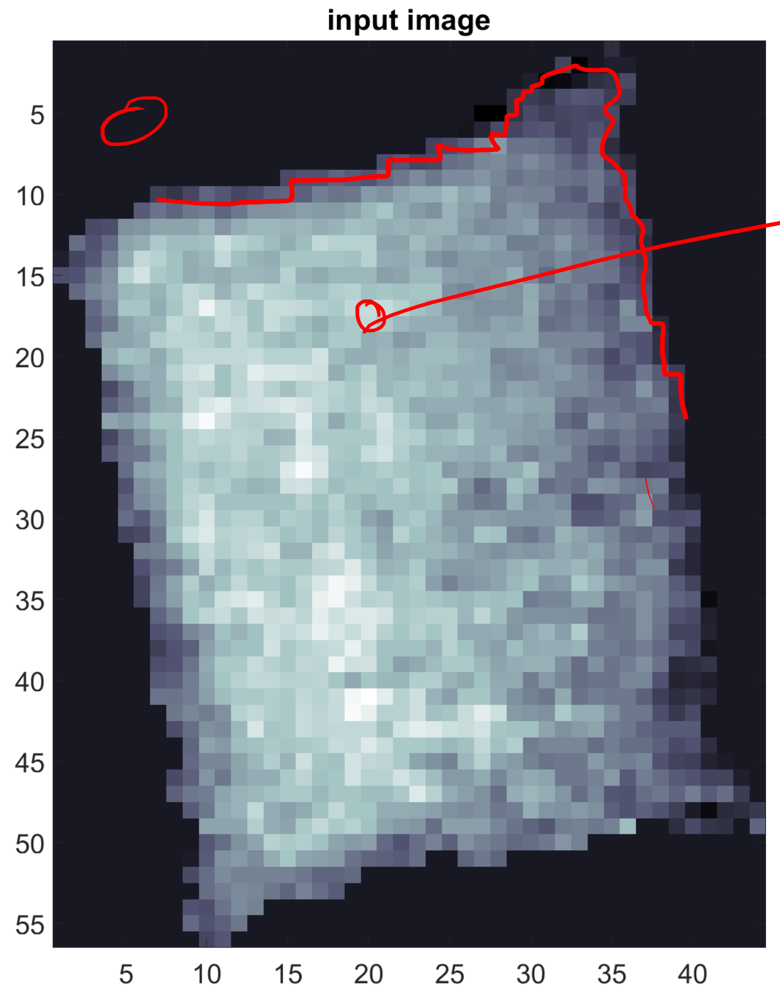
- Remove blobs that are too small
- Ehnance boundaries by morphological operation

For instance, you can for each blob

- Compute the area
- Compute different image statistics over each blob (average intensity, standard deviation, squared error w.r.t. regression model) and over central / perimetral area
- Compute Bounding Box and the aspect ratio
- Compute the Axis
- Analyze the location in the image
- …. compute the convex hull…

Boundary Extraction

The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



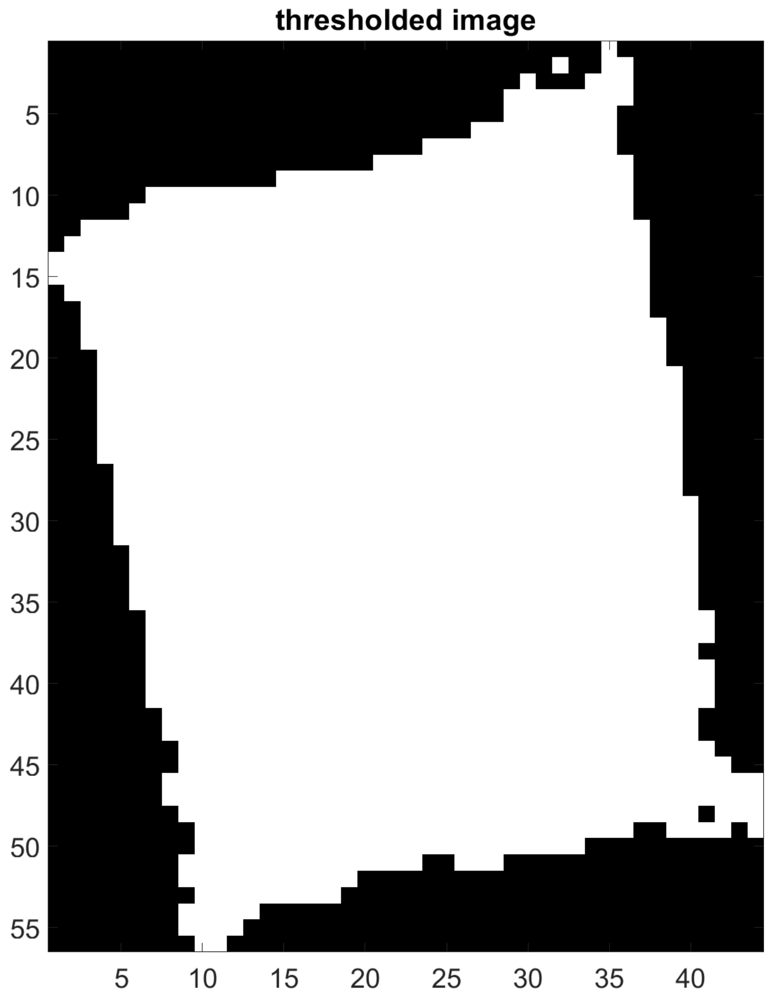
I

$[0-1]^m$

T

Boundary Extraction

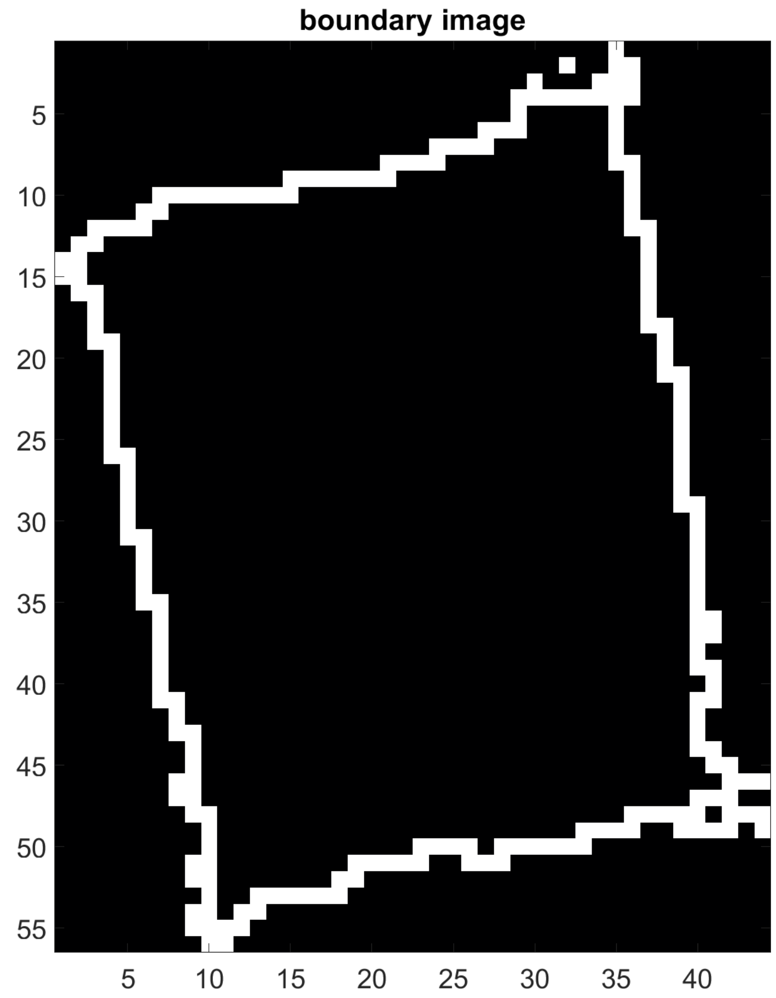
The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



$$T = I > \Gamma$$

Boundary Extraction

The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



$$B = T \ominus h$$

[B - erode(B)]

$$h = \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Feature Extraction From Boundaries

For each blob, extract

- Perimeter
- Area
- Perimeter-area ratio
- Edges orientation (it would be good to suitably rotate the image before)

Image Classification By Hand-Crafted Features

The Feature Extraction Perspective

Images can not be directly fed to a classifier

We need some intermediate step to:

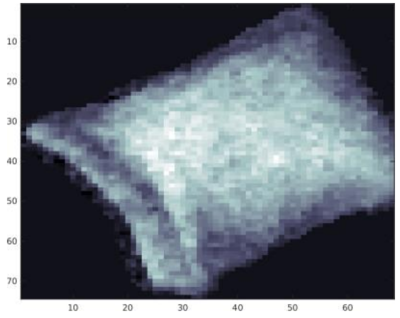
- Extract meaningful information (to our understanding)
- Reduce data-dimension

We need to extract features:

- The better our features, the better the classifier

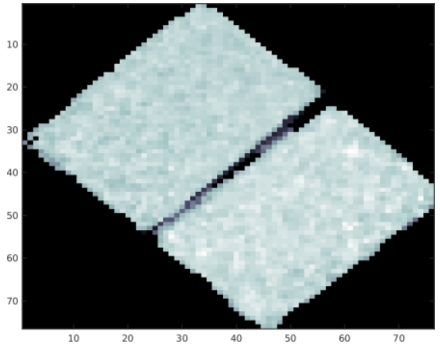
The Feature Extraction Perspective

Input image



$I_1 \in \mathbb{R}^{r_1 \times c_1}$

Input image



$I_1 \in \mathbb{R}^{r_2 \times c_2}$

Feature Extraction Algorithms

perimetro

area

$\mathbf{x} \in \mathbb{R}^{d}$

$\mathbf{x} \in \mathbb{R}^{d}$

$(d \ll r \times c)$

Classifier

"parcel"

$y \in \Lambda$

"double"

$y \in \Lambda$

The Feature Extraction Perspective

Images can not be directly fed to a classifier

We need some intermediate step to:

- Extract meaningful information (to our understanding)
- Reduce data-dimension

We need to extract features:

- The better our features, the better the classifier

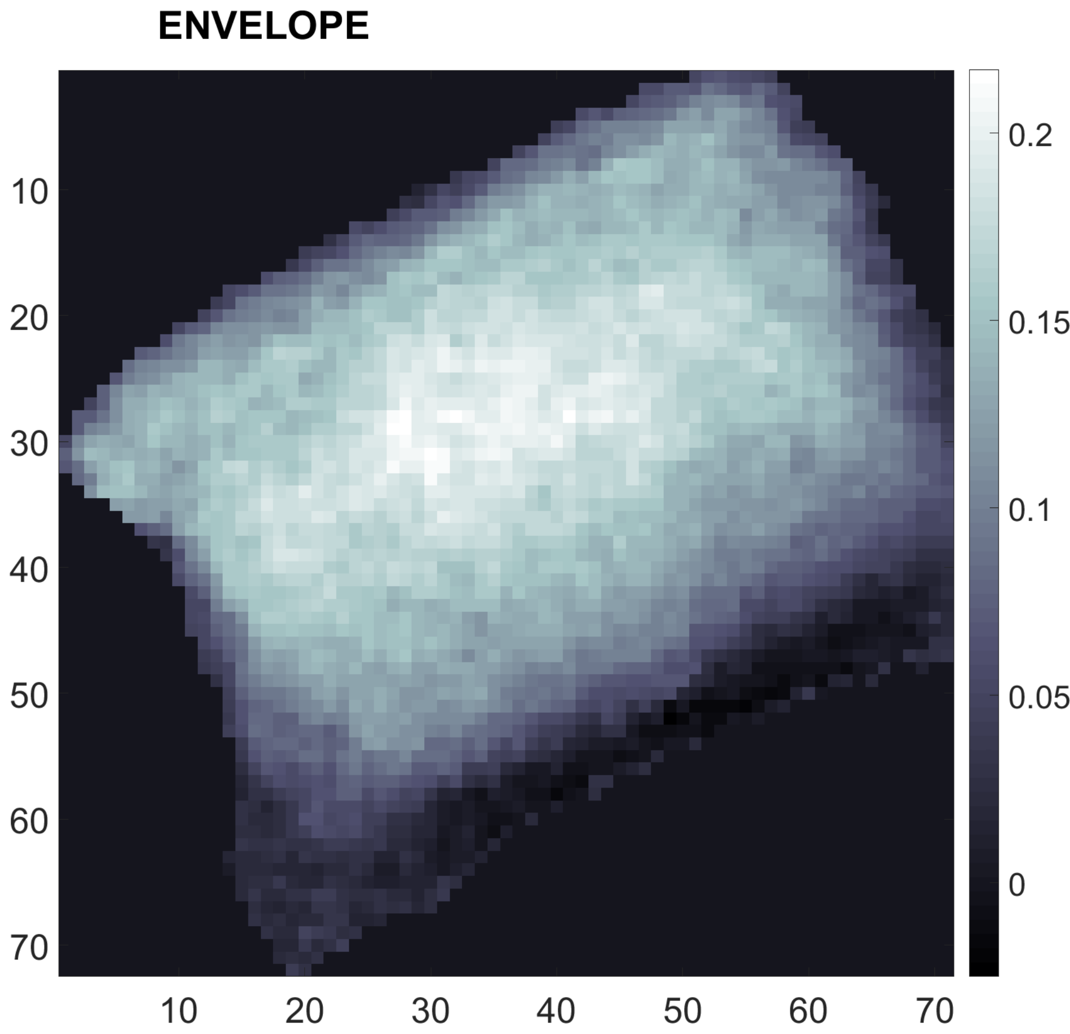
Different Options:

- **Hand Crafted Features**
- **Computer Vision Features**
- **Learned Features**

The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

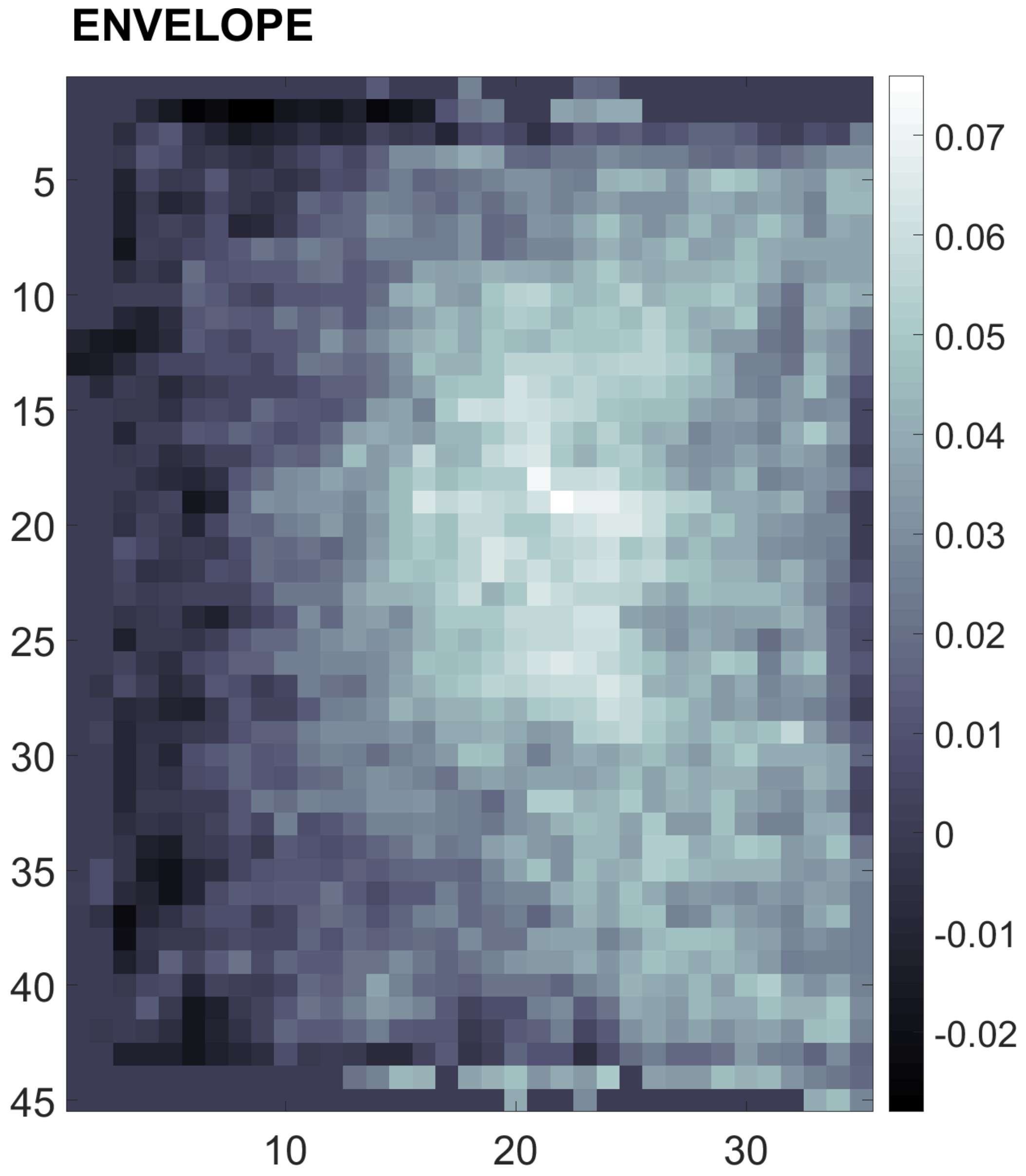
- No color information provided
- A few pixels report depth measures
- Images of 3 classes
    - ENVELOPE
    - PARCEL
    - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

- No color information provided
- A few pixels report depth measures
- Images of 3 classes
    - ENVELOPE
    - PARCEL
    - DOUBLE

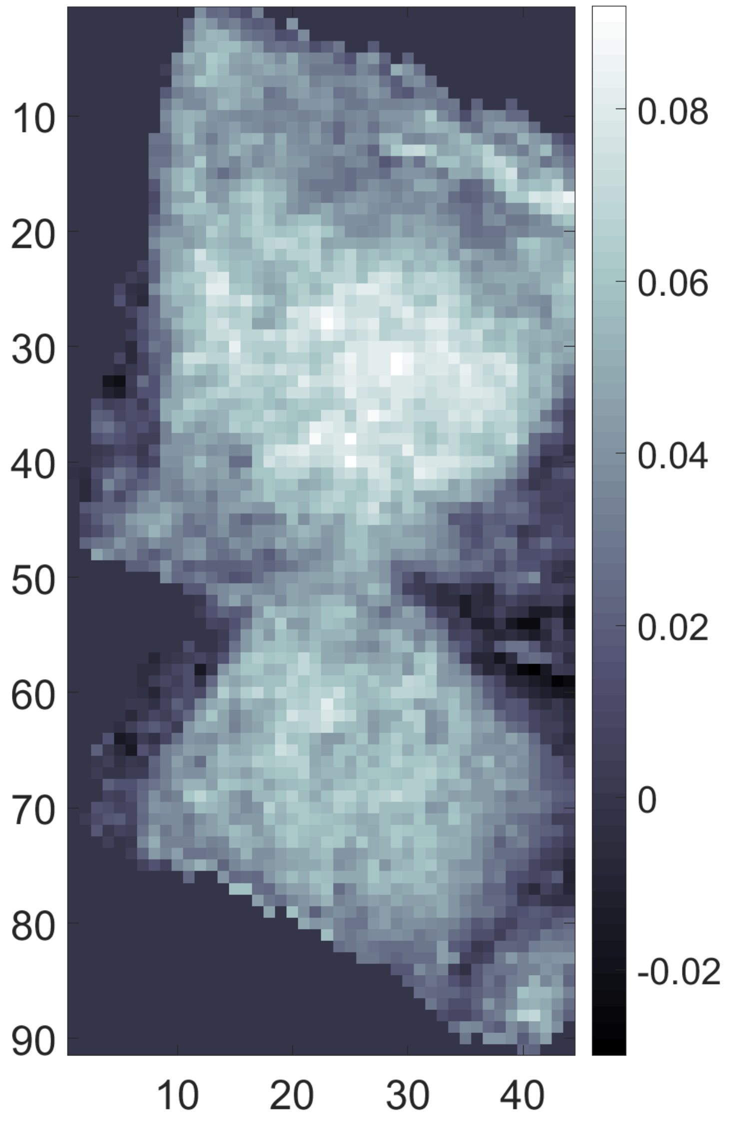


The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

- No color information provided
- A few pixels report depth measures
- Images of 3 classes
    - ENVELOPE
    - PARCEL
    - DOUBLE

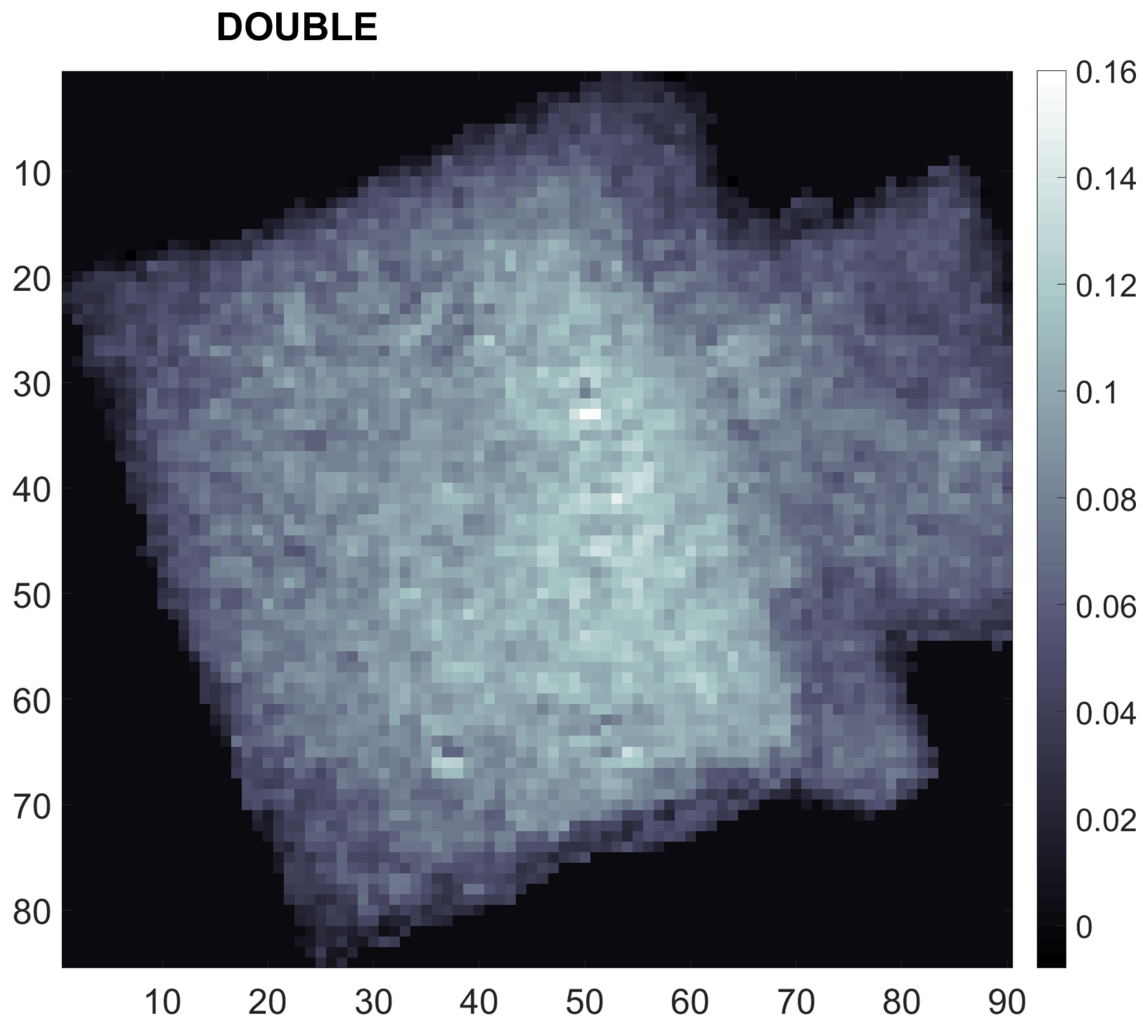
DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

- No color information provided
- A few pixels report depth measures
- Images of 3 classes
    - ENVELOPE
    - PARCEL
    - DOUBLE



The Application Scenario: Parcel Classification

Images acquired from a RGB-D sensor:

- No color information provided
- A few pixels report depth measures
- Images of 3 classes
    - ENVELOPE
    - PARCEL
    - DOUBLE