

Nonlinear Filters

Giacomo Boracchi

Image Analysis and Computer Vision

19 November 2021

Politecnico di Milano

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

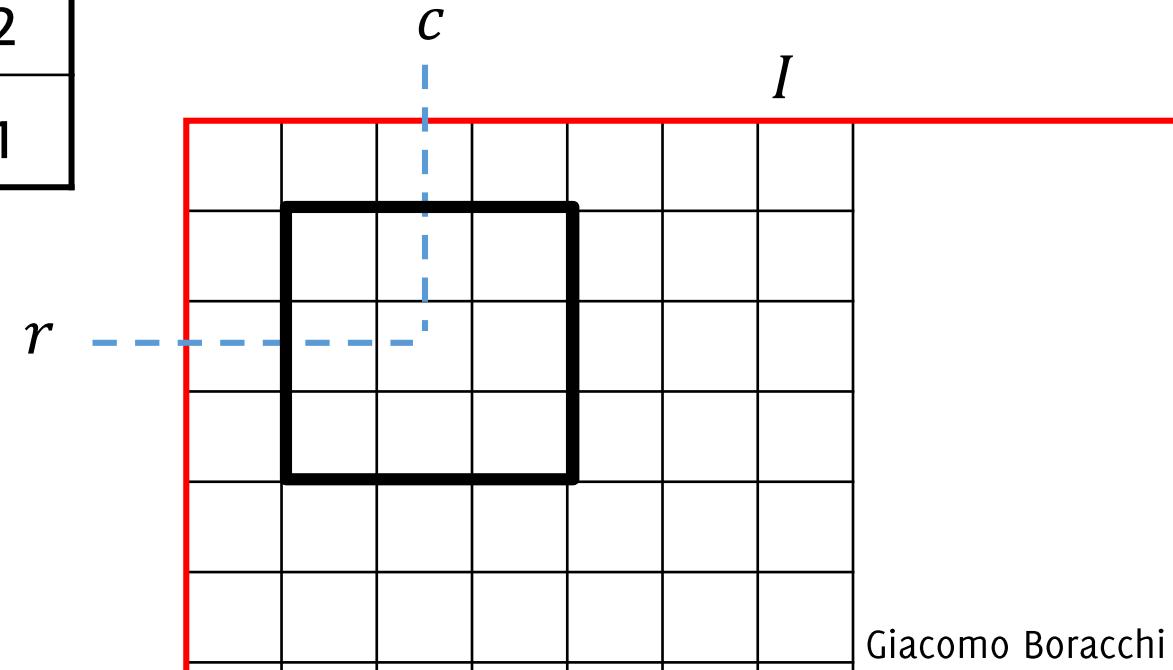
Block-wise median: replaces each pixel with the median of its neighborhood. It is still a **local spatial transformation!**

	2	

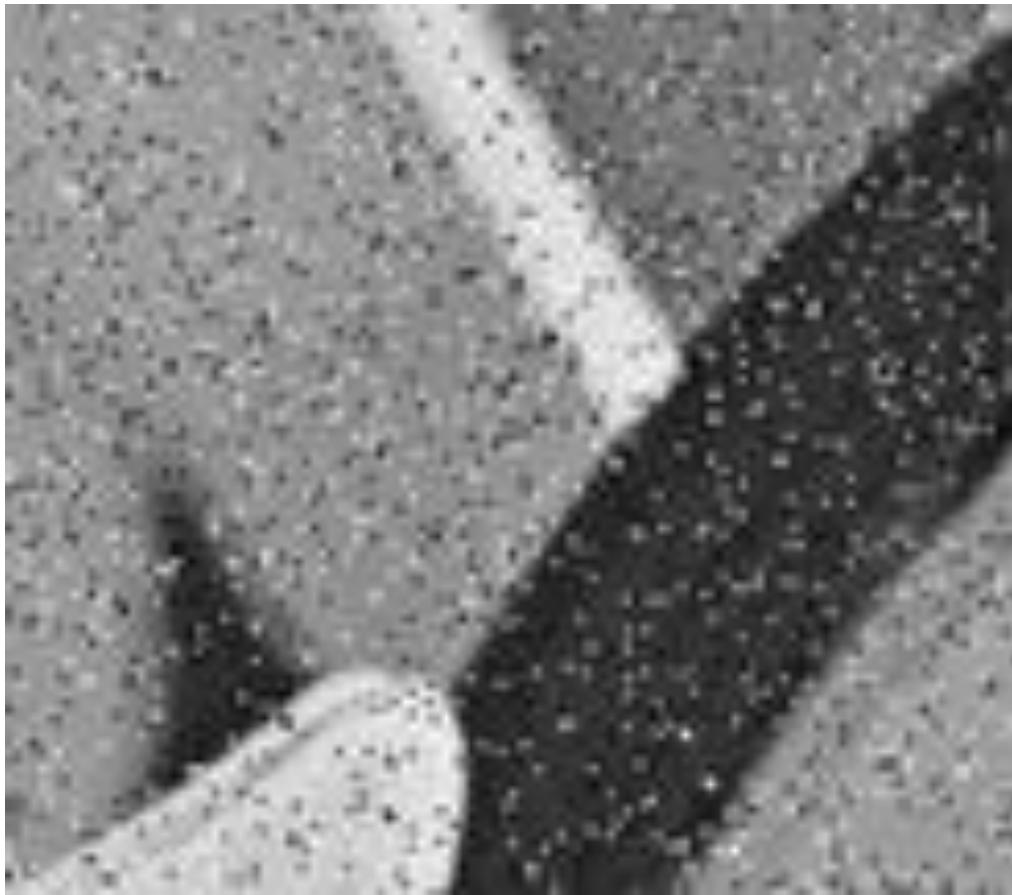
med

1	3	0
2	10	2
4	1	1

$$m = \text{median}(1, 3, 0, 2, 10, 2, 4, 1, 1) = 2$$



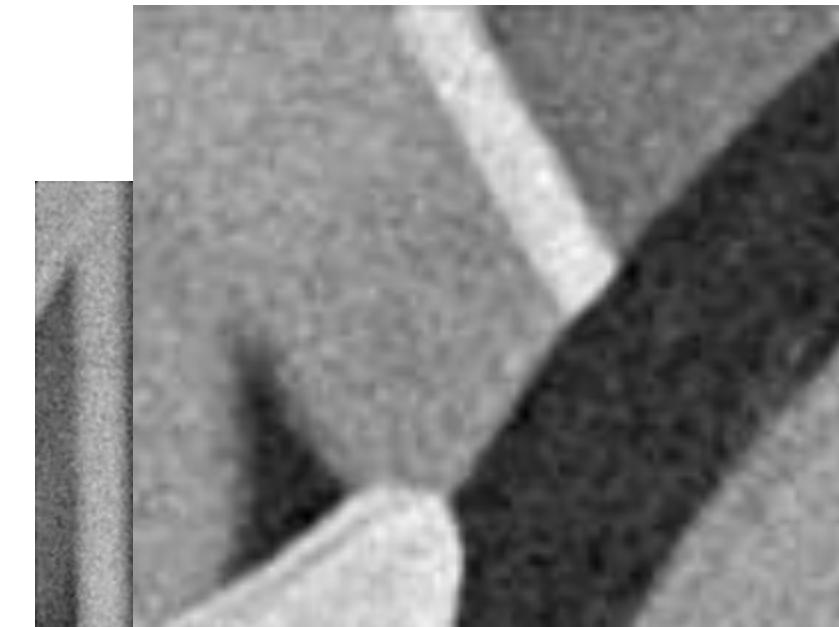
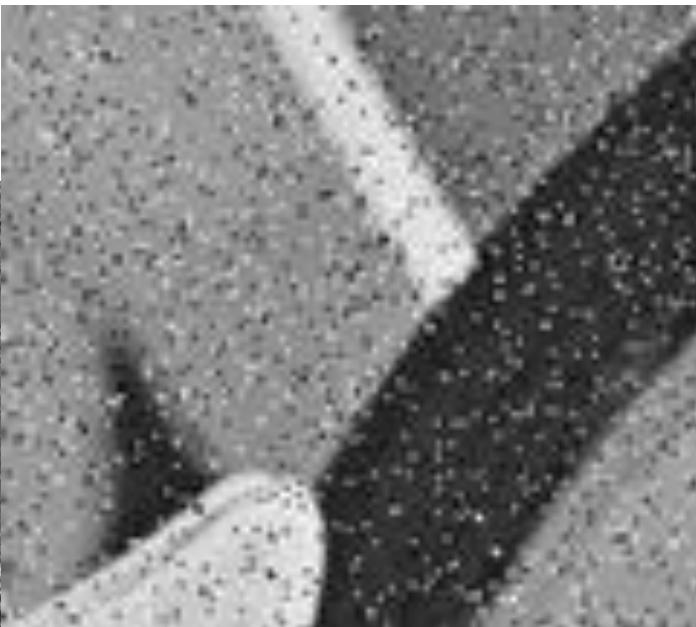
Salt-and-pepper noise



Salt and Pepper (Impulsive) noise

Denoising using local smoothing 3x3

Notice that he used
dirac impulse to denoise it
So, A black image with
only one white pixel
in the middle can be
seen a s dirac impulse



Denoising with median 3x3



Salt and Pepper (Impulsive) noise

Morphological Operations

Ordered Statistics and Blob Labeling

An overview on morphological operations

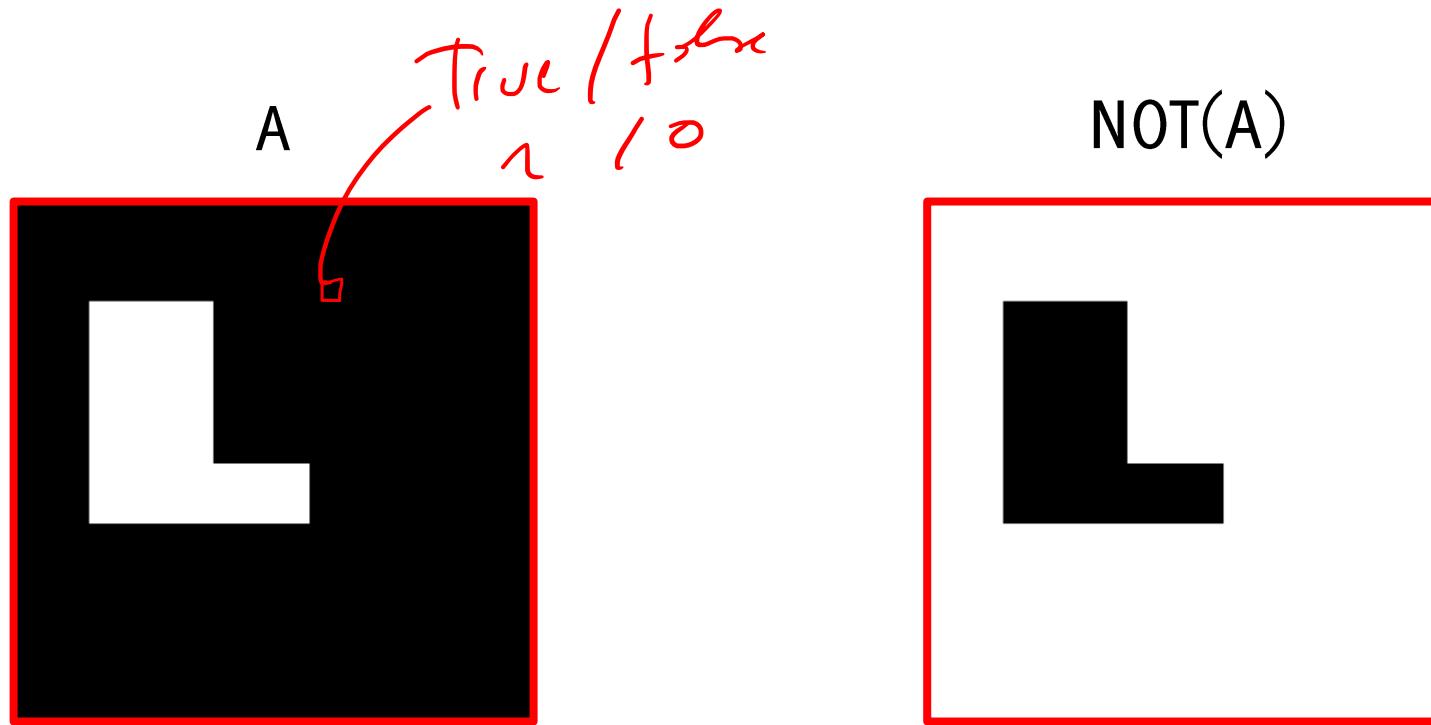
Erosion, Dilation

Open, Closure

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

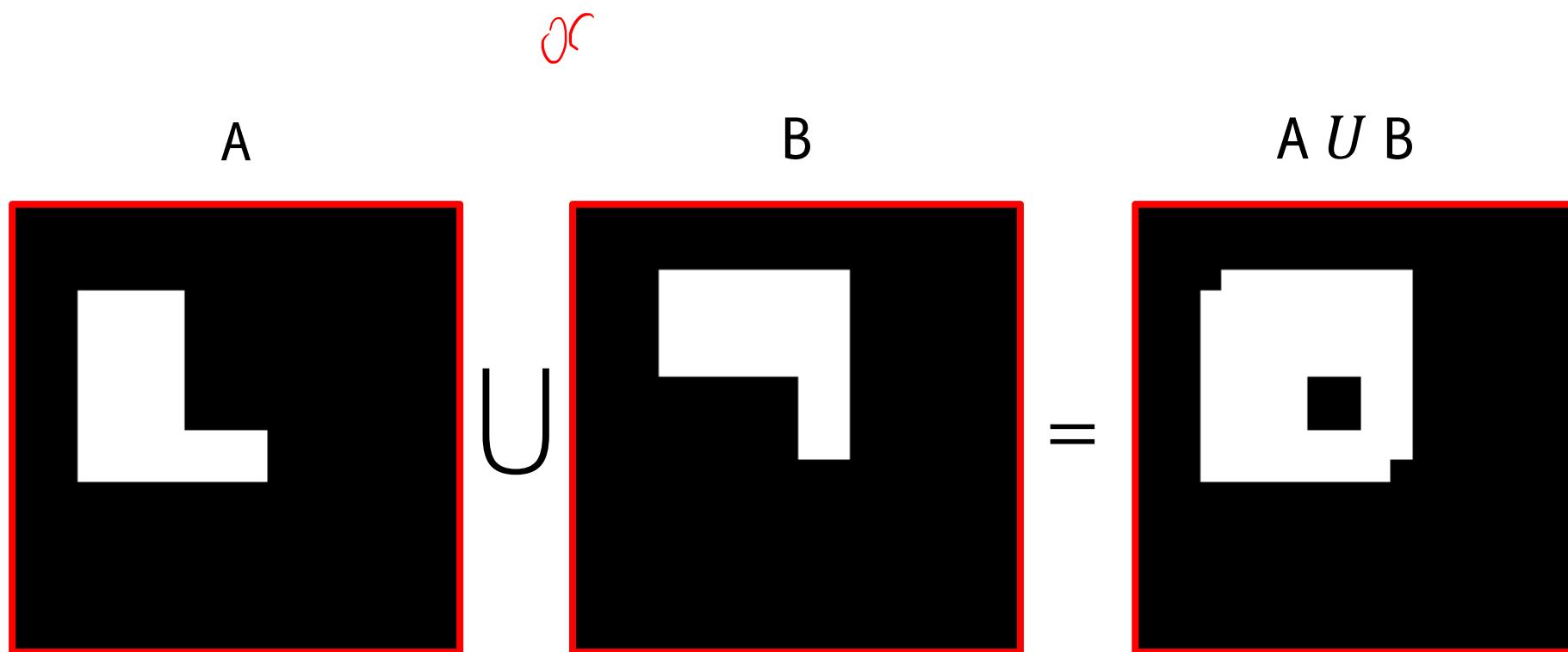
Boolean operations on binary images

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



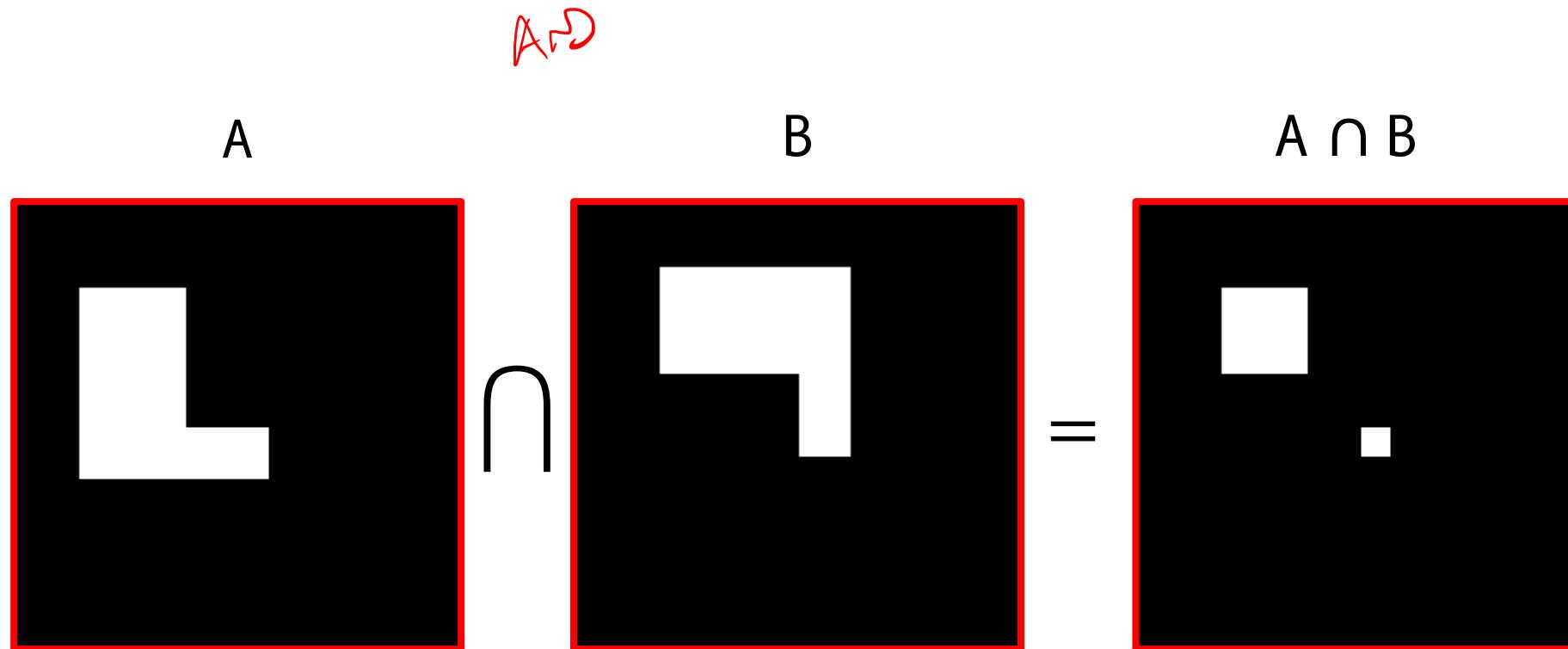
$$\text{NOT}_A = A == 0$$

UNION of binary images



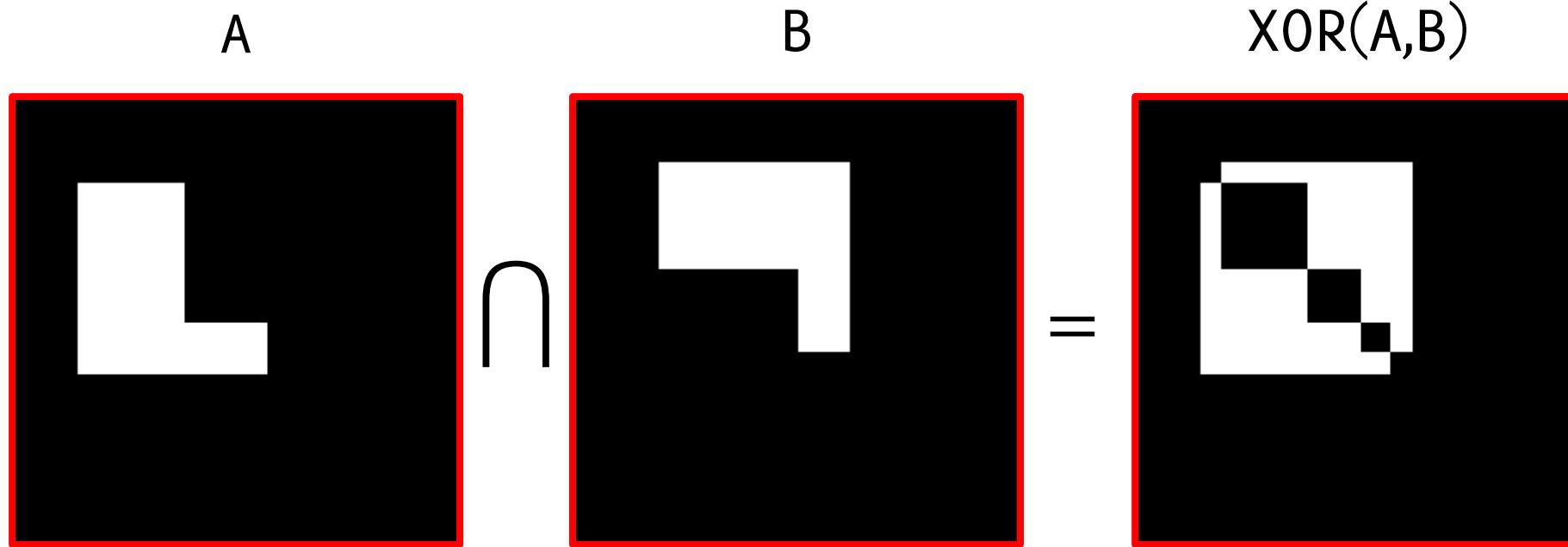
$$A \cup B = A + B > 0$$

INTERSECTION of binary images



$$A_AND_B = A + B > 0$$

On binary images it is possible to define XOR

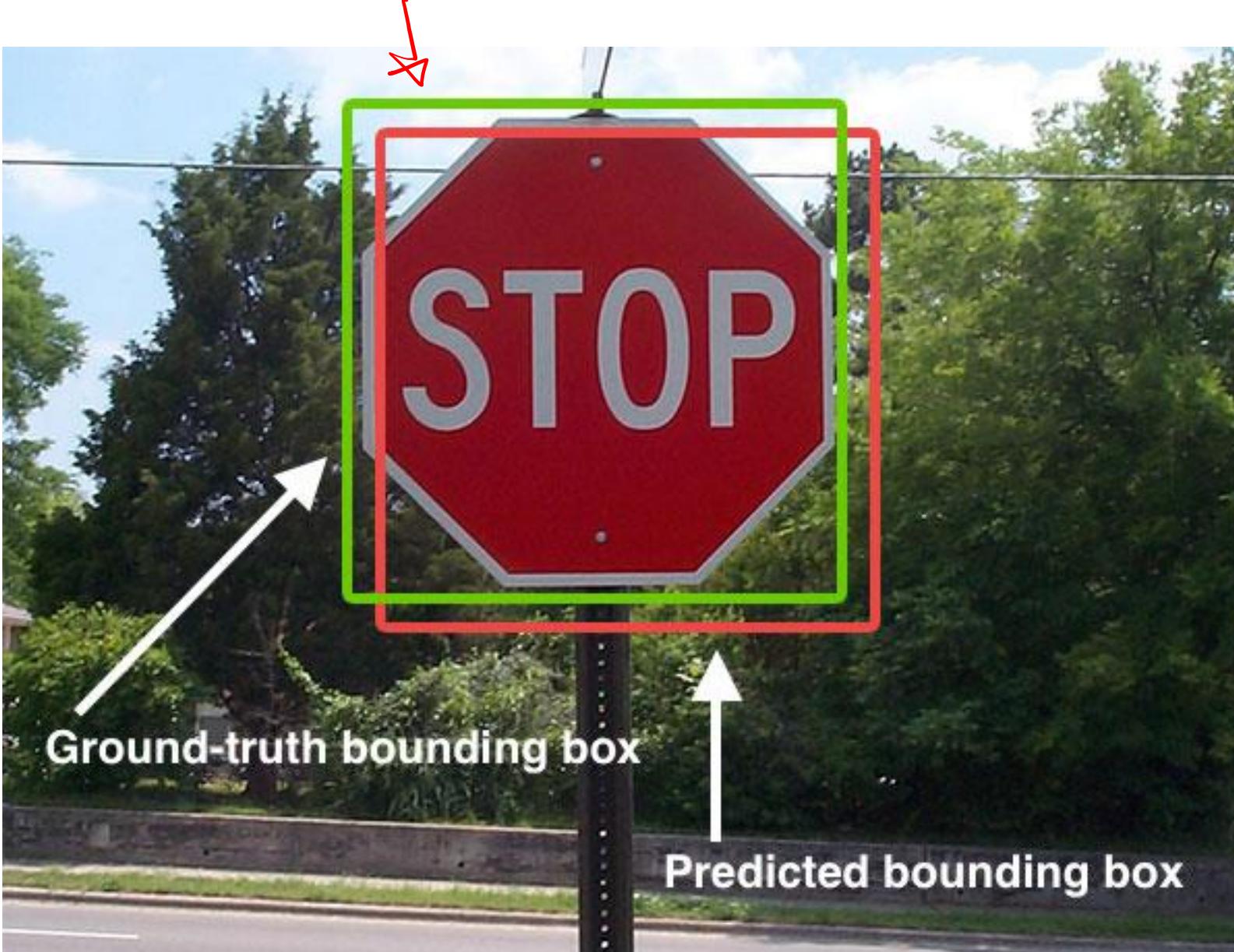


$$\text{XOR}(A,B) =$$

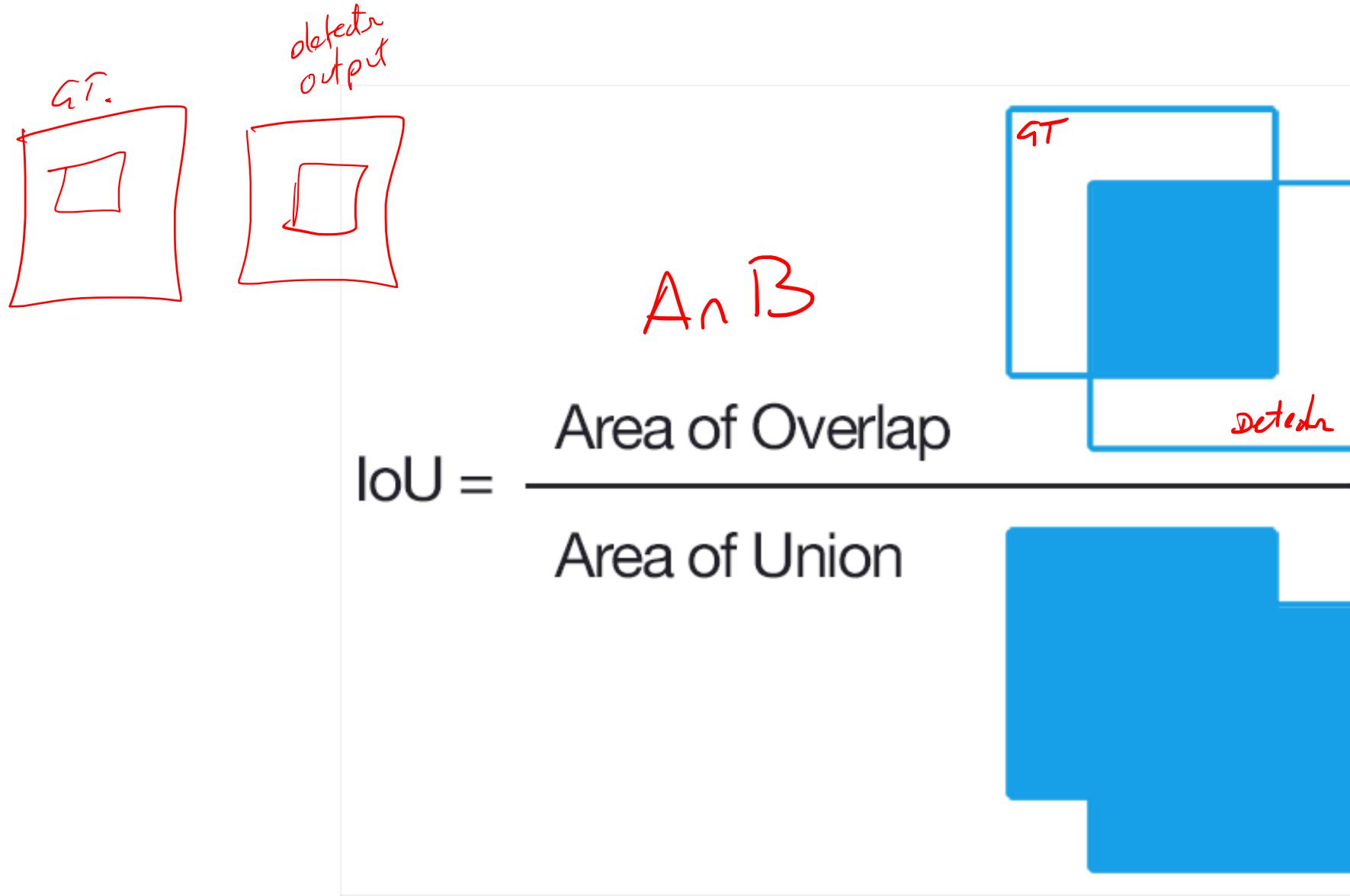
$$A \cup B - A \text{ AND } B$$

$$A \cap B$$

Intersection over the Union (IoU, Jaccard Index)

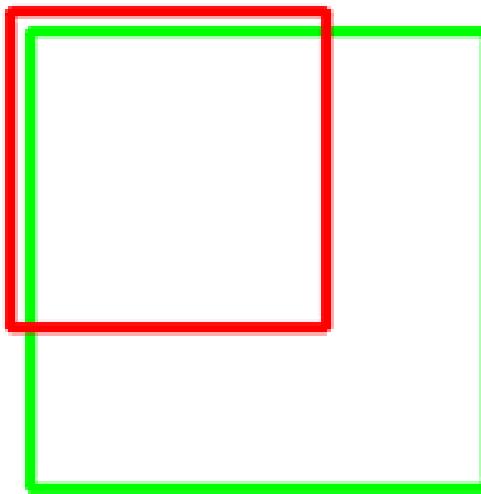


Intersection over the Union (IoU, Jaccard Index)



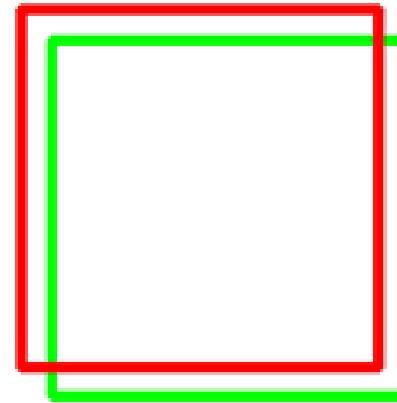
Intersection over the Union (IoU, Jaccard Index)

IoU: 0.4034



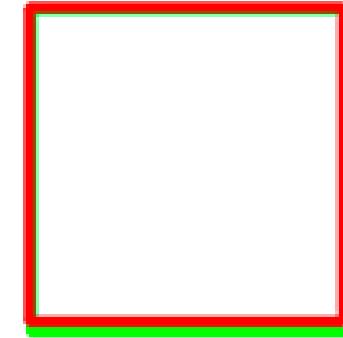
Poor

IoU: 0.7330



Good

IoU: 0.9264



Excellent

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

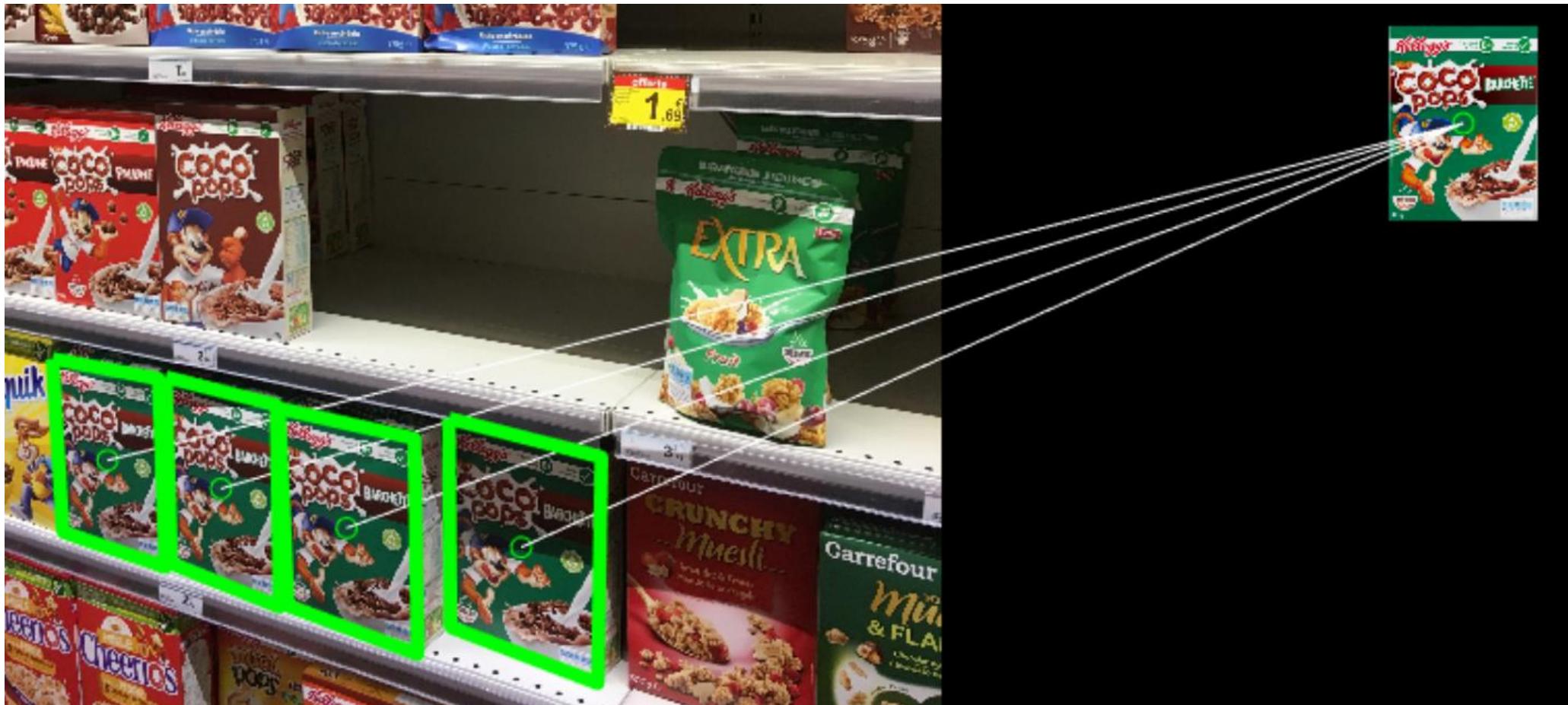
$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

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)

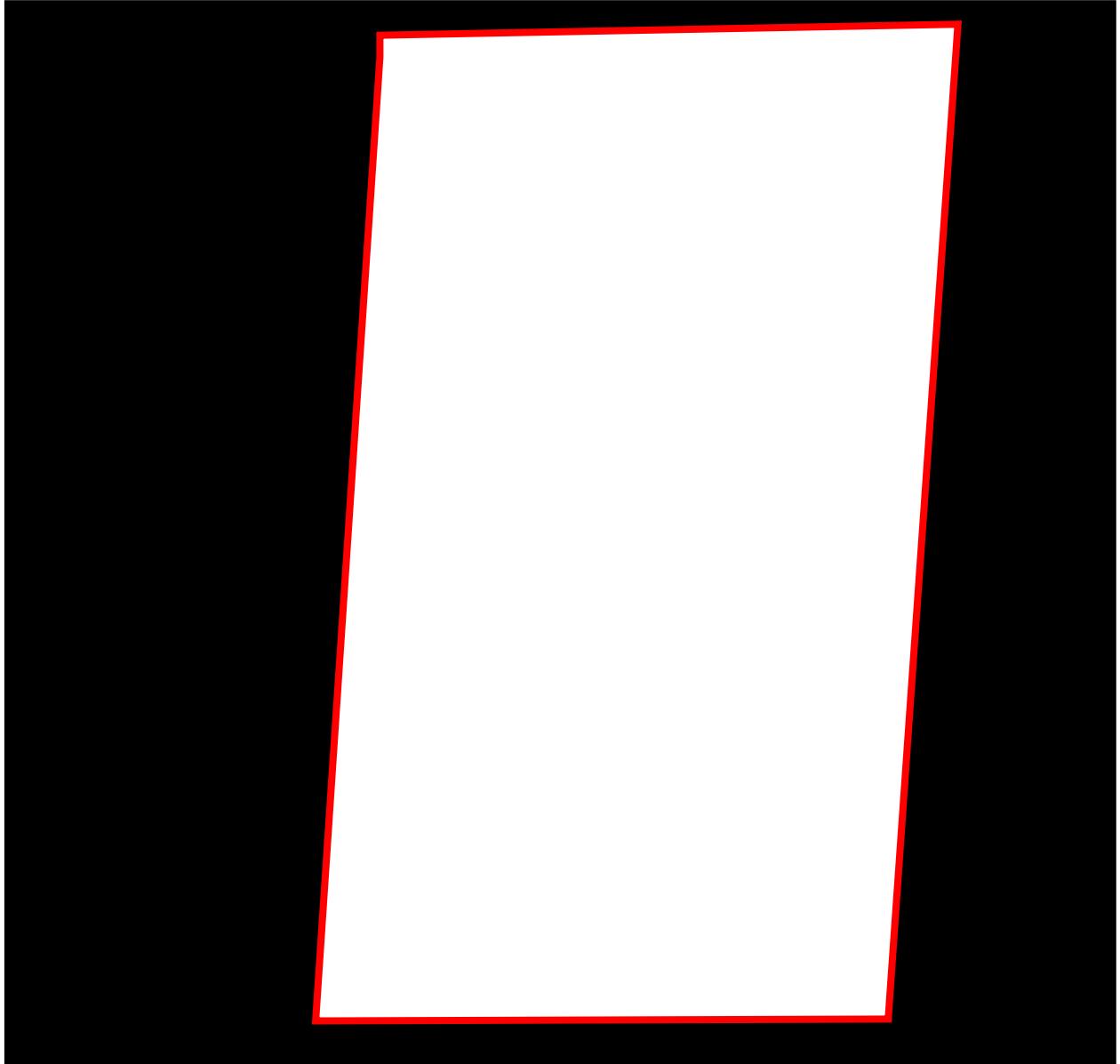
It is not necessarily defined for bounding boxes (even though most of deep learning networks for detections provide bb as outputs)



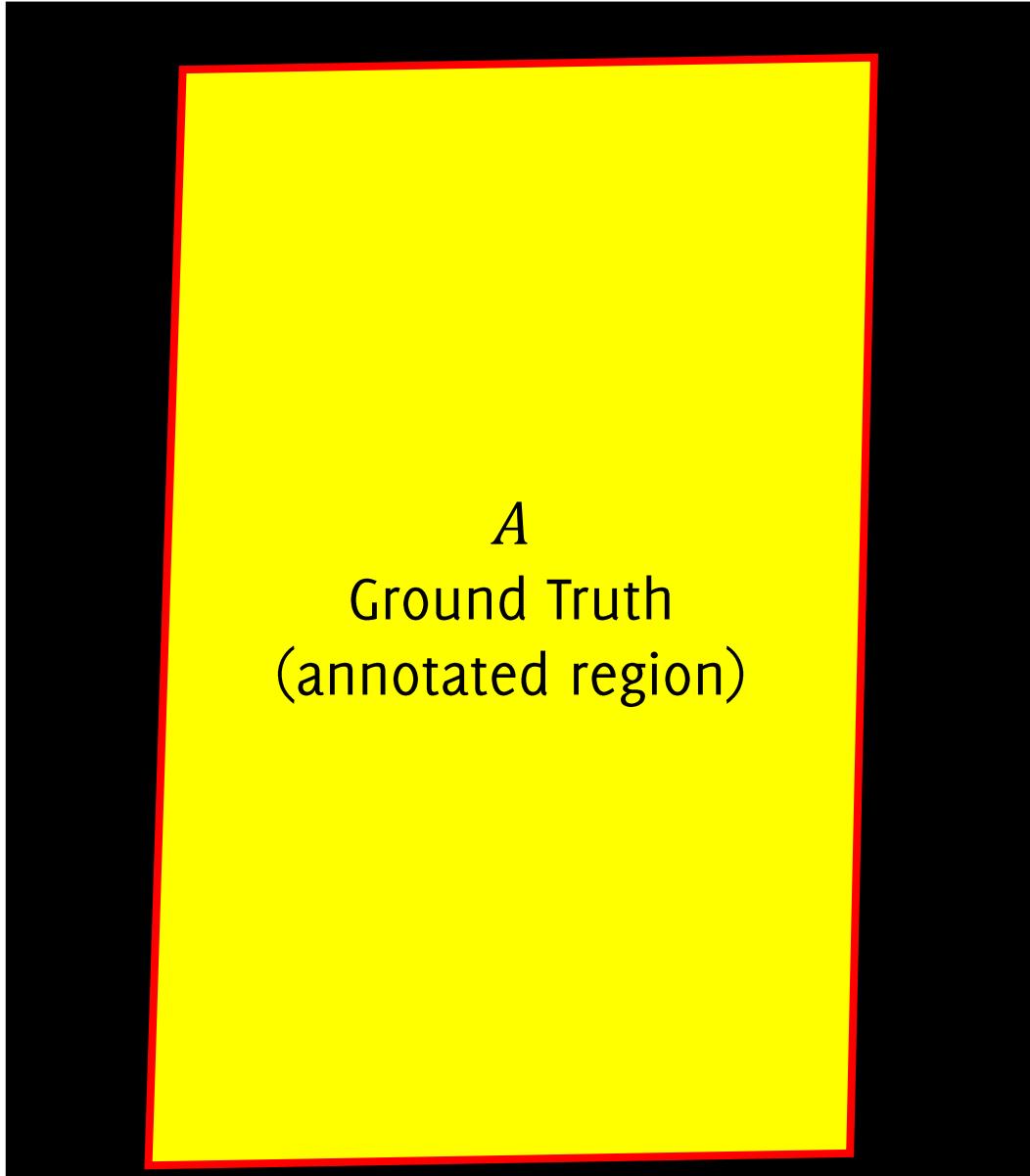
Jaccard Index (IoU)



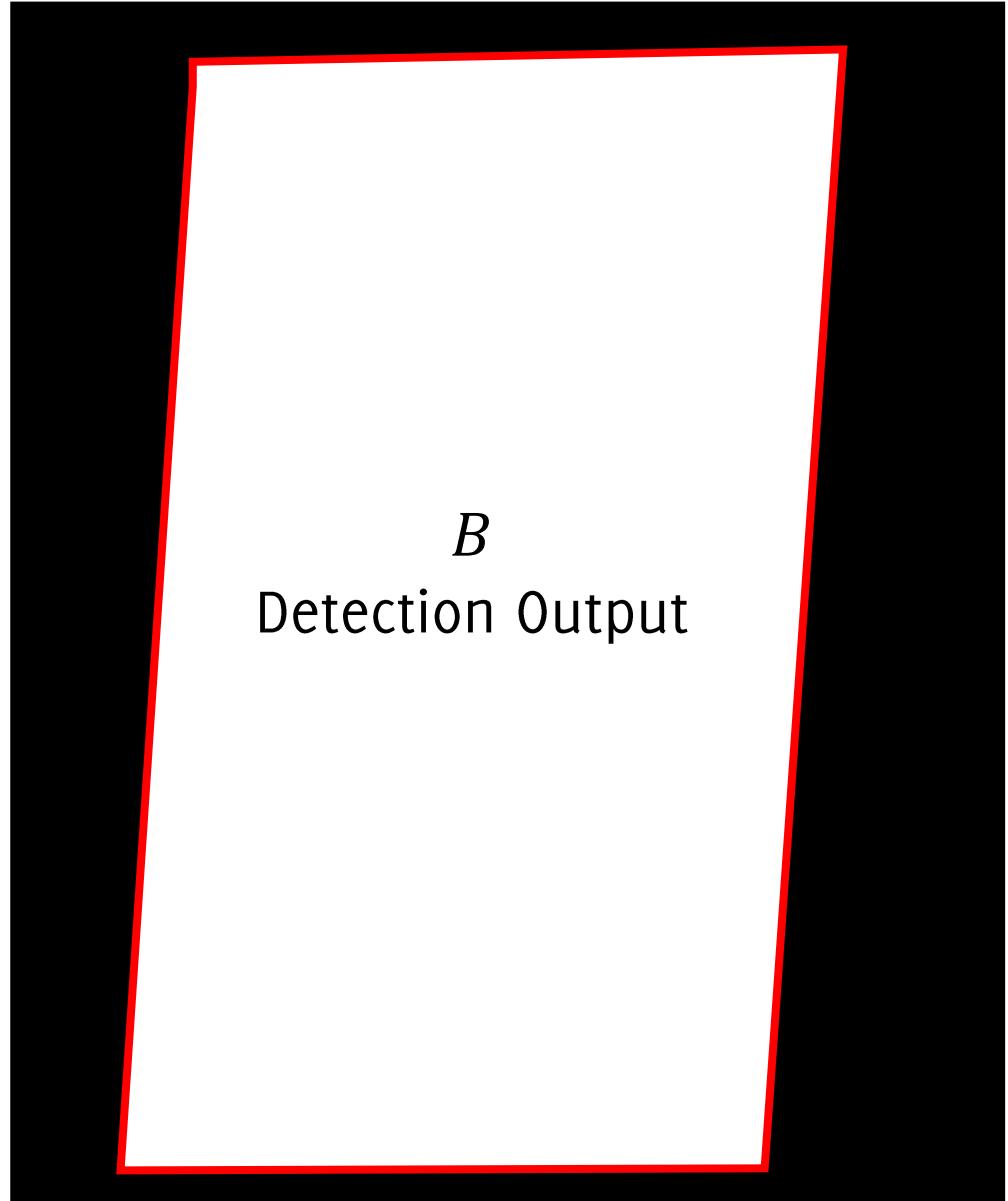
Jaccard Index (IoU)



Jaccard Index (IoU)



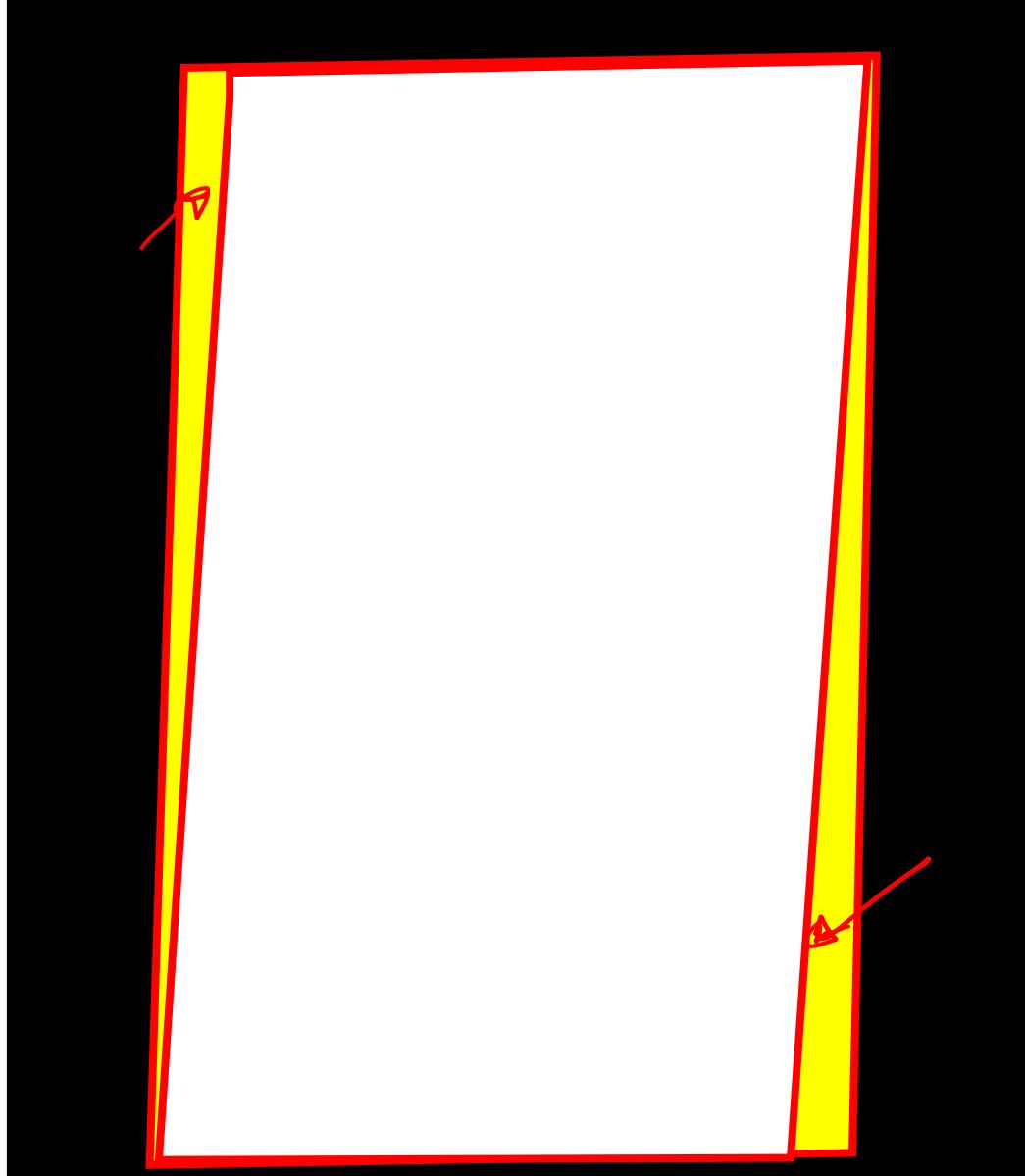
A
Ground Truth
(annotated region)



B
Detection Output

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

General definition:

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

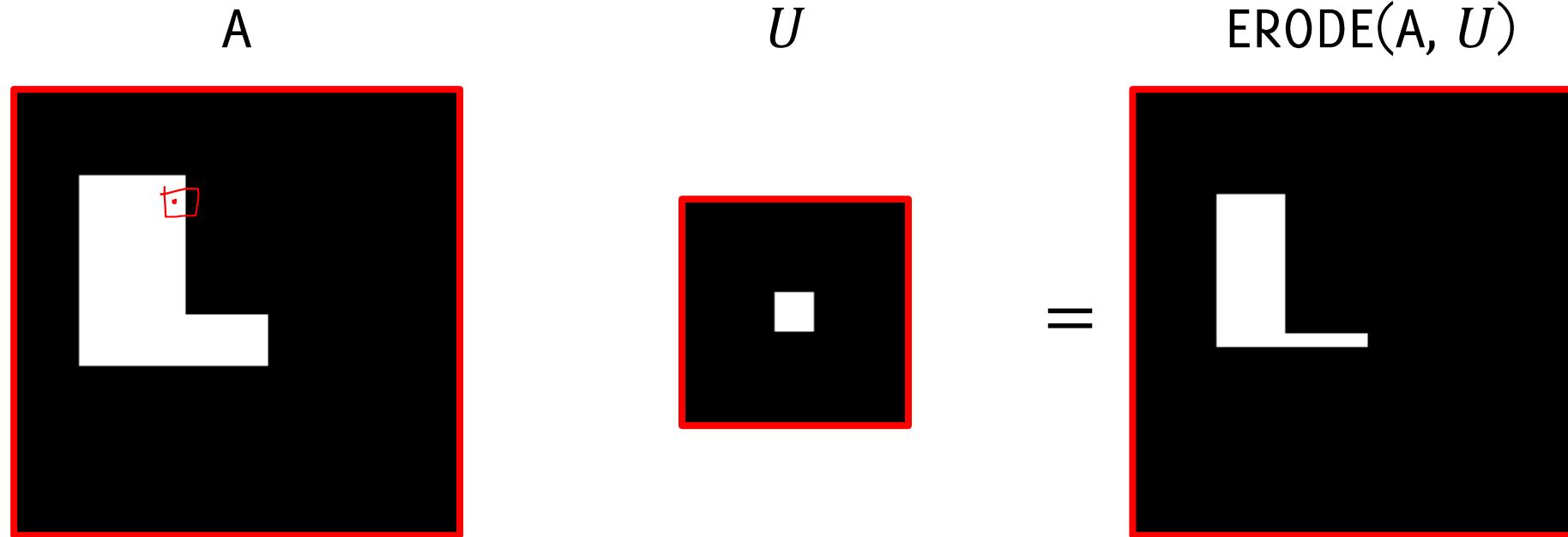
As a consequence on binary images, it is equivalent to the following rule:

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

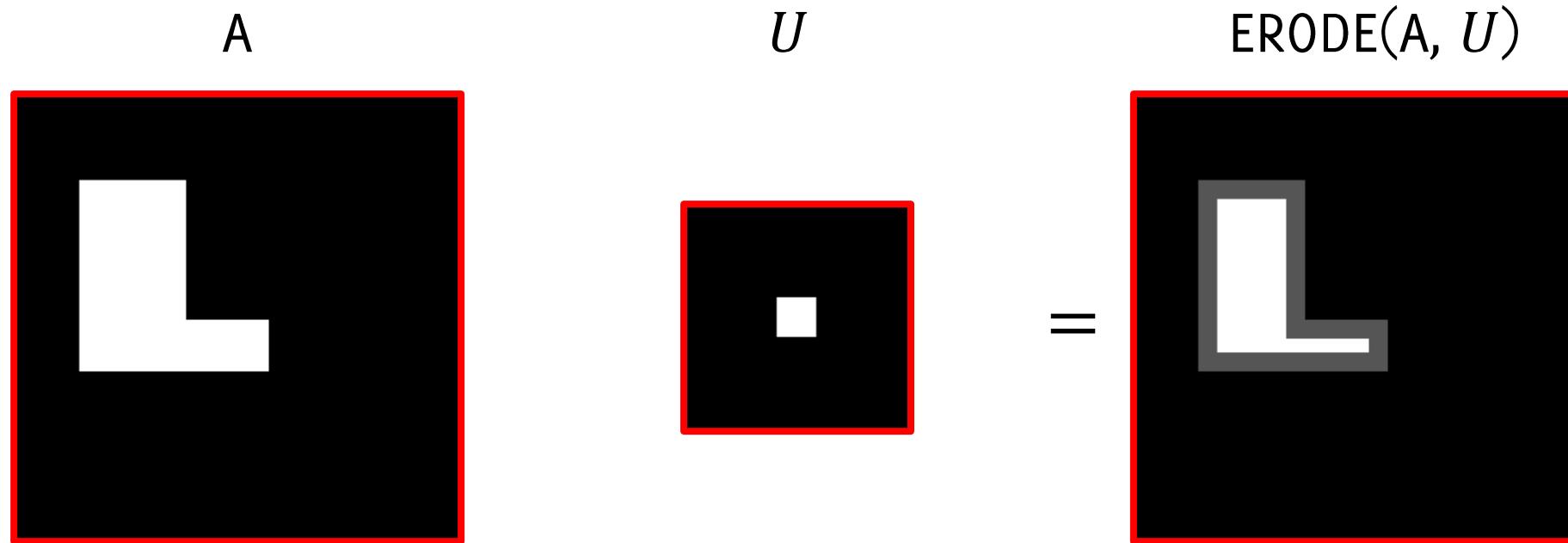
This operation reduces thus the boundaries of binary images

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

Erosion



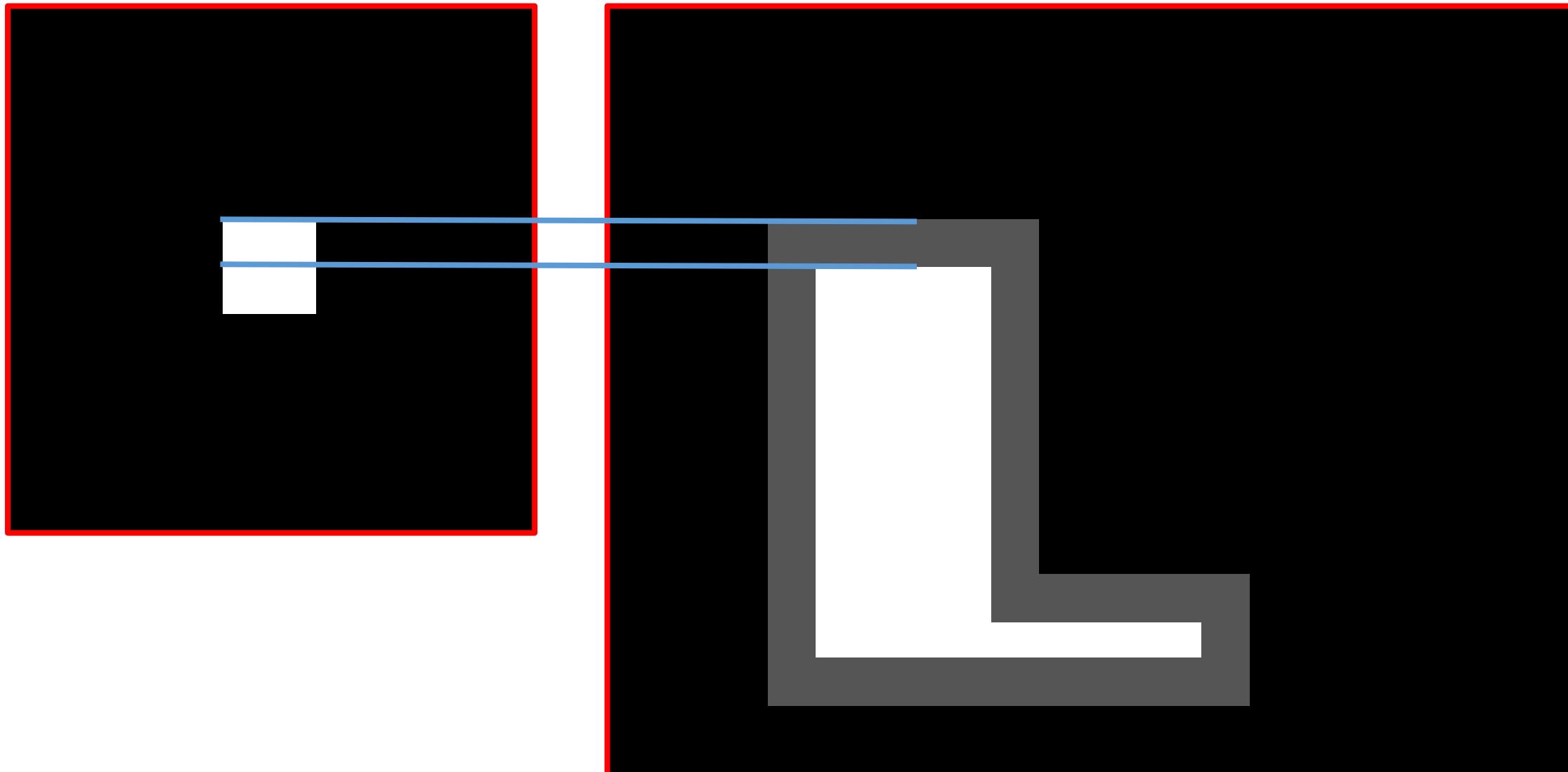
Erosion



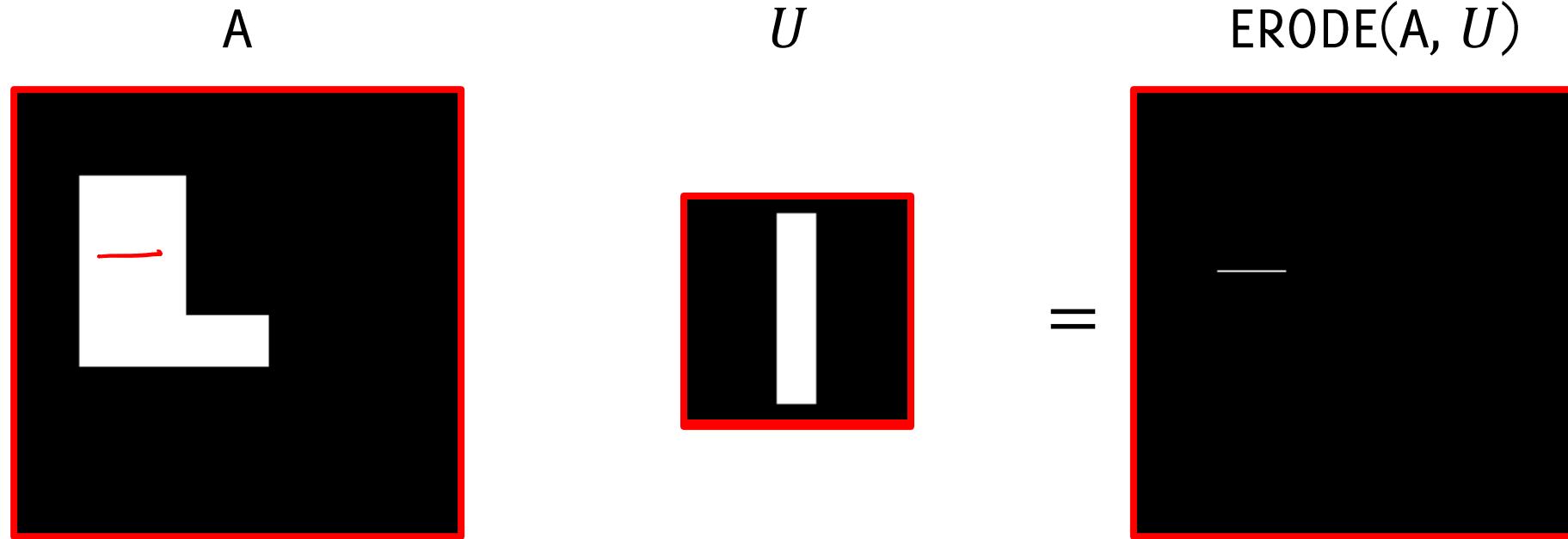
The gray area corresponds
to the input

Erosion

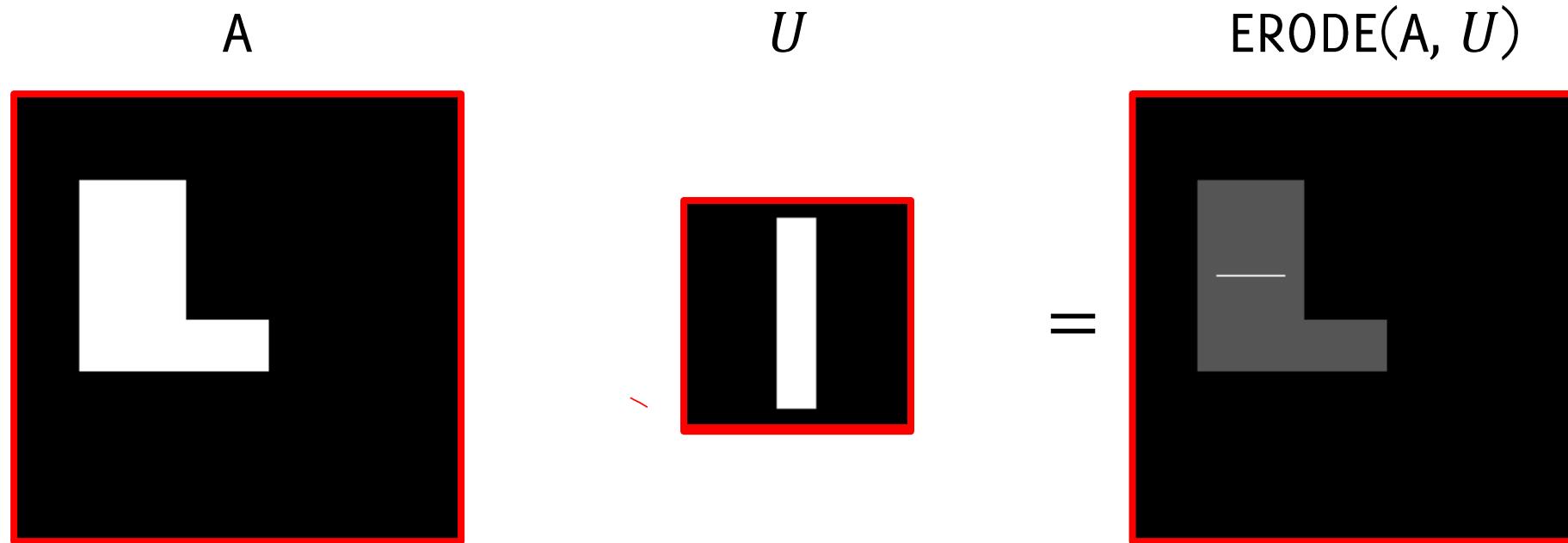
Erosion removes half size of the structuring element used as filter



Erosion



Erosion



Dilation

General definition:

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

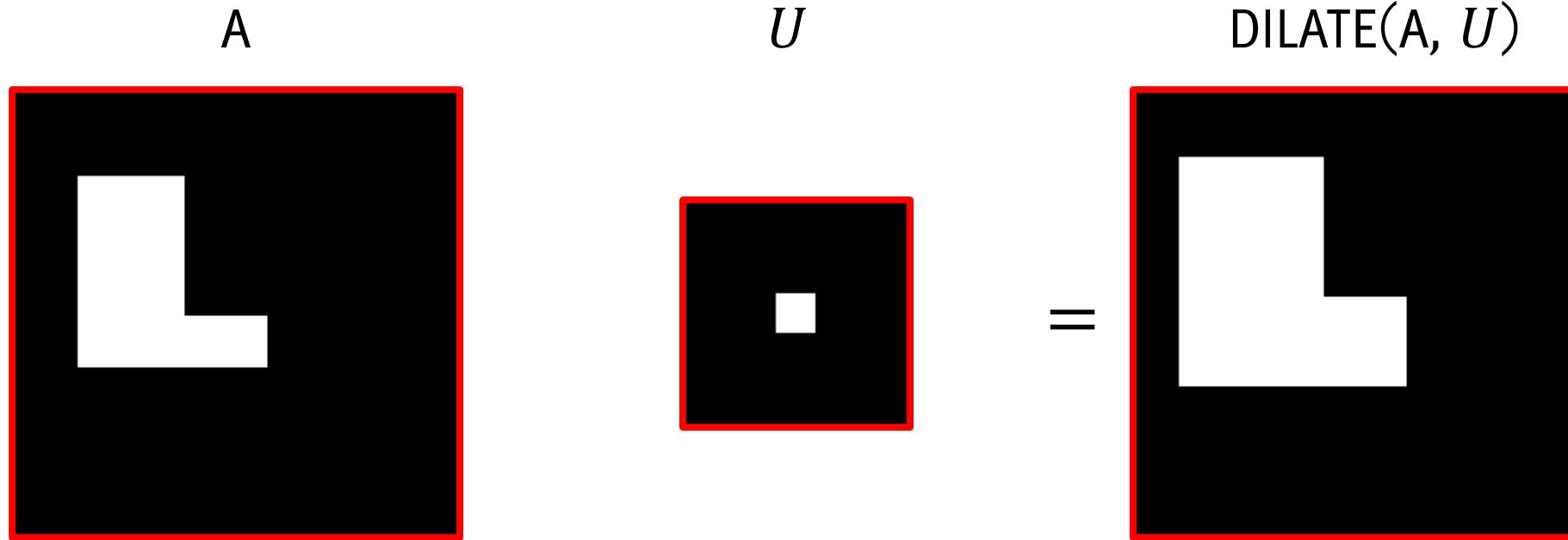
As a consequence on binary images, it is equivalent to the following rule:

$$E(x)=1 \text{ iff at least a pixel in the neighbor is } 1$$

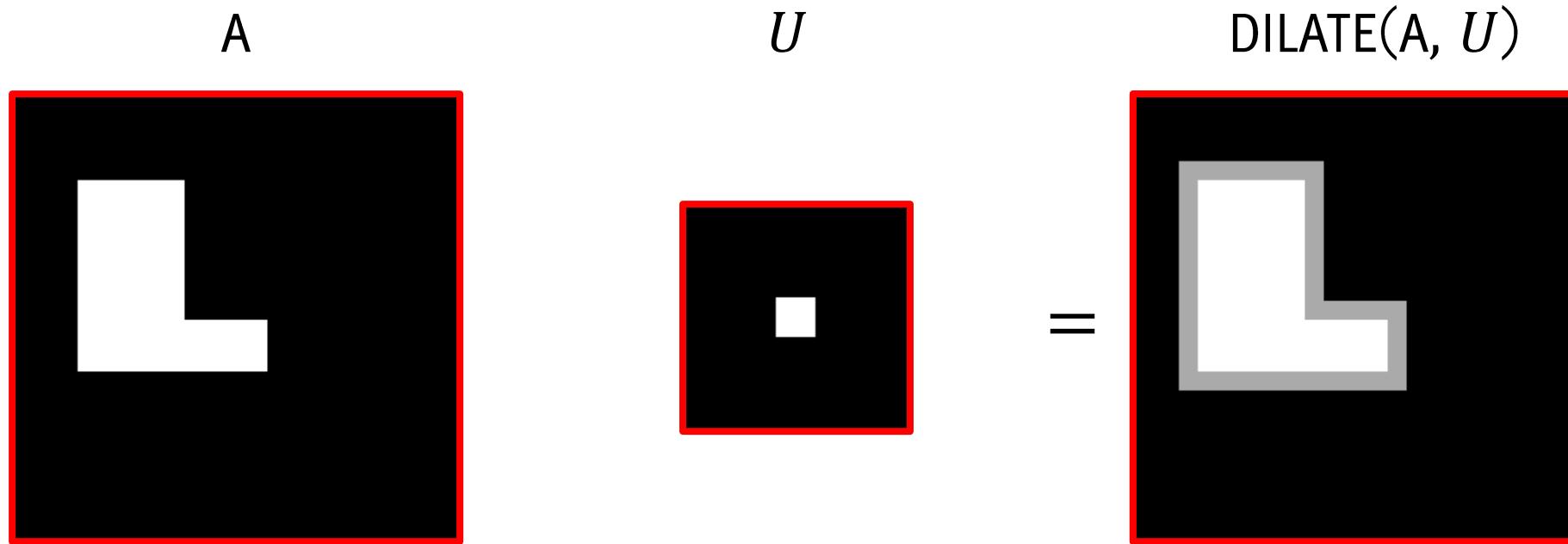
This operation grows fat the boundaries of binary images

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

Dilation



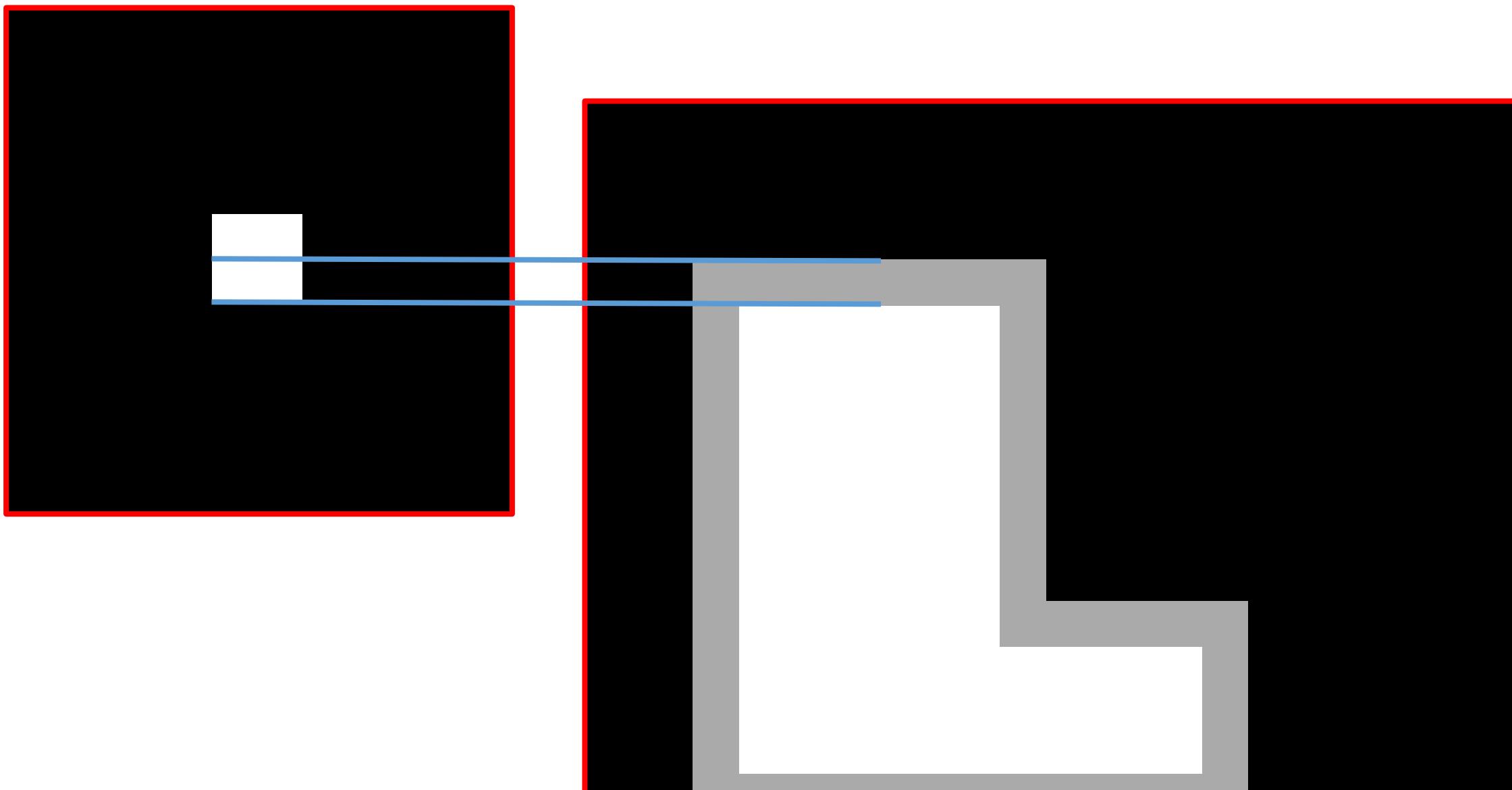
Dilation



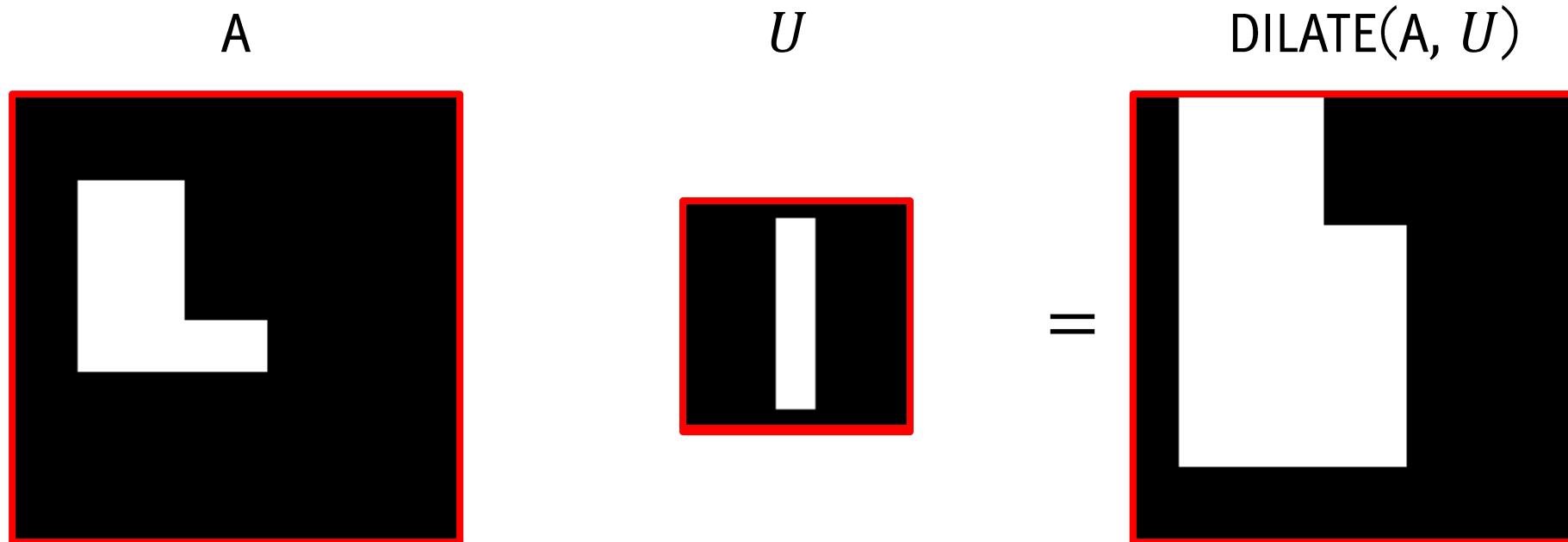
The brighter area now
corresponds to the input

Dilation

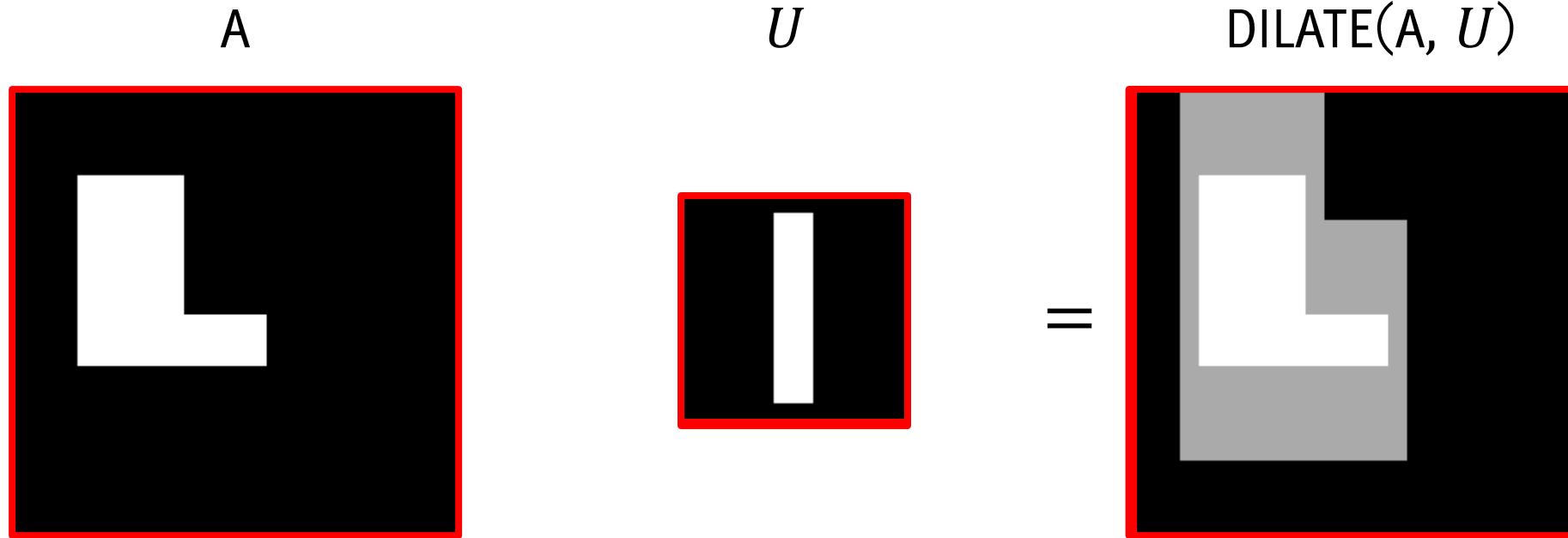
Dilation expands half size of the structuring element used as filter



Dilation



Dilation



Open and Closure

Open Erosion followed by a Dilation

Closure Dilation followed by an Erosion

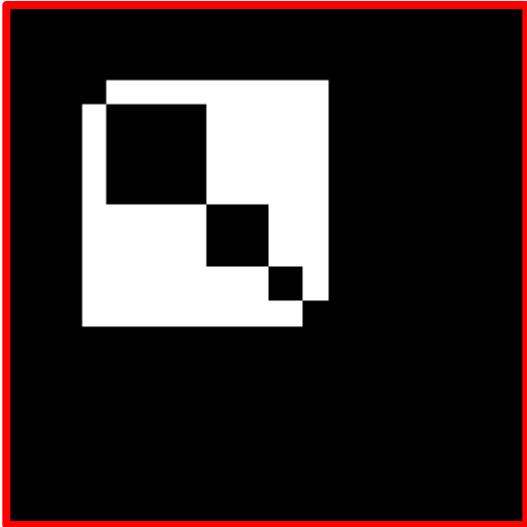
Open

Open Erosion followed by a Dilation

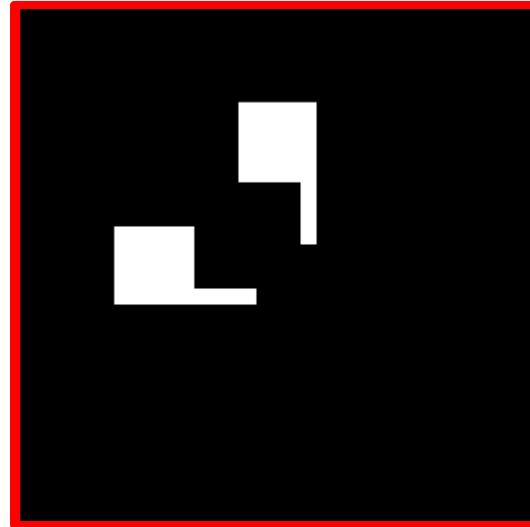
- Smooths the contours of an object
- Typically eliminates thin protrusions

Open

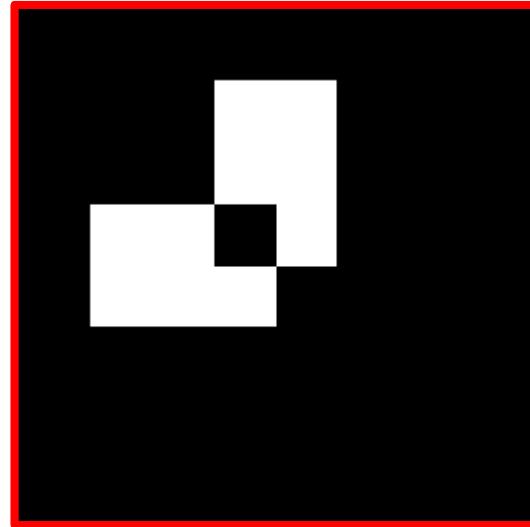
A



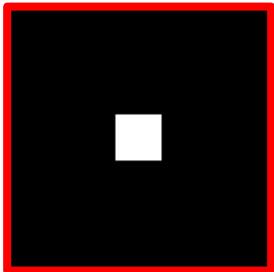
$O = \text{ERODE}(A, U)$



$O = \text{DILATE}(O, U)$

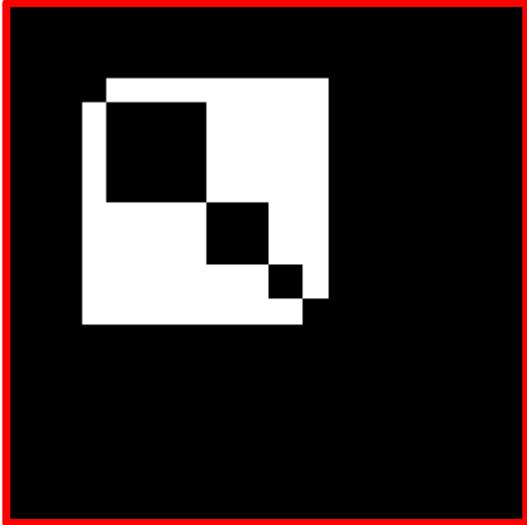


U

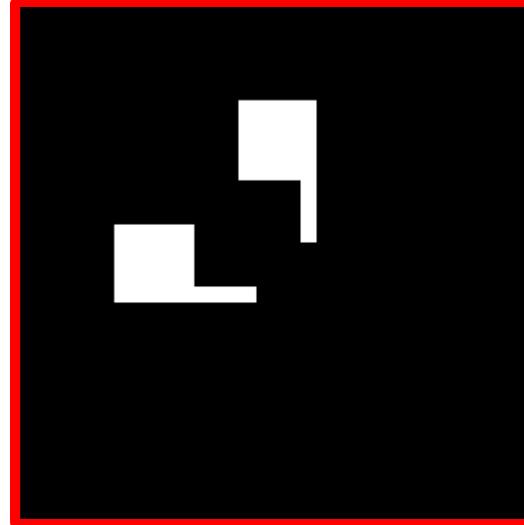


Open

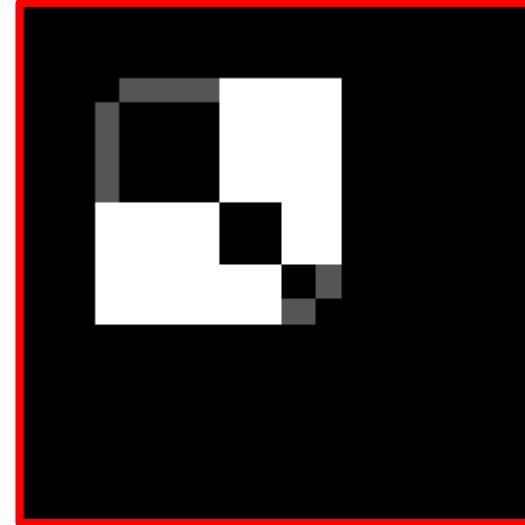
A



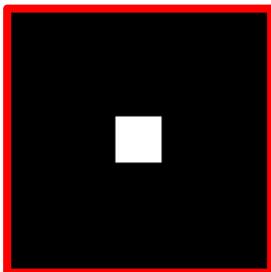
$O = \text{ERODE}(A, U)$



$O = \text{DILATE}(O, U)$



U



The gray area corresponds
to the input

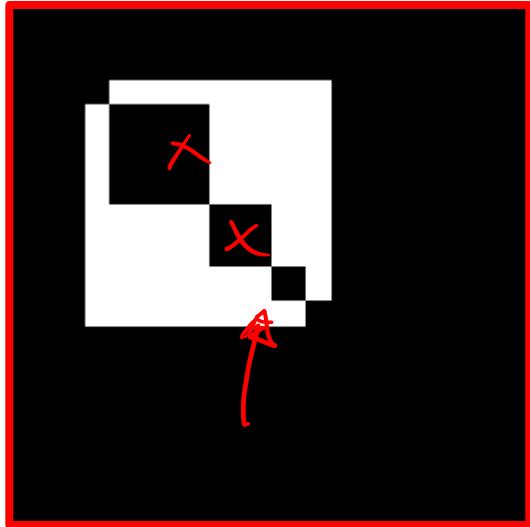
Closure

Closure Dilation followed by an Erosion

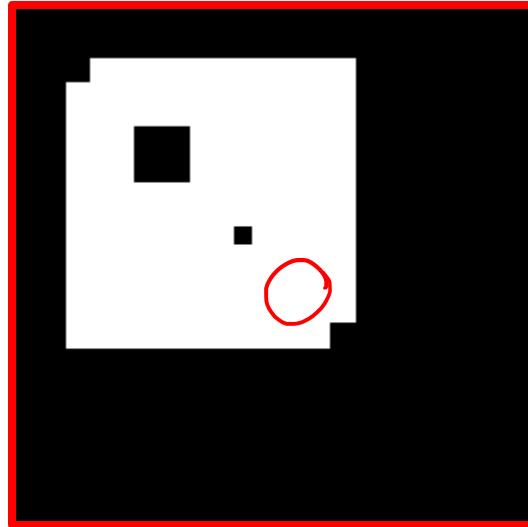
- Smooths the contours of an object, typically creates bridges
- Generally fuses narrow breaks

Close

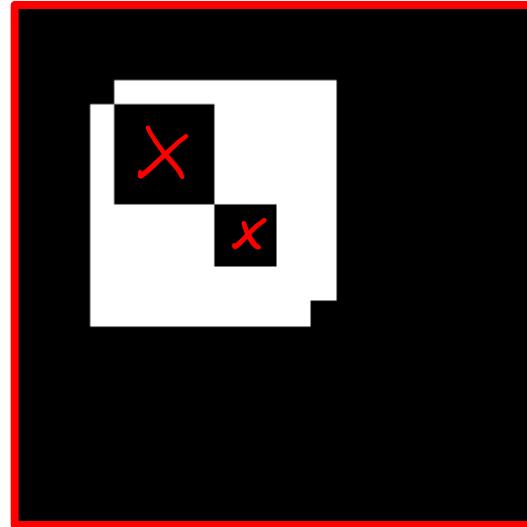
A



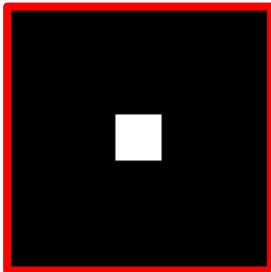
$O = \text{DILATE}(A, U)$



$O = \text{ERODE}(O, U)$

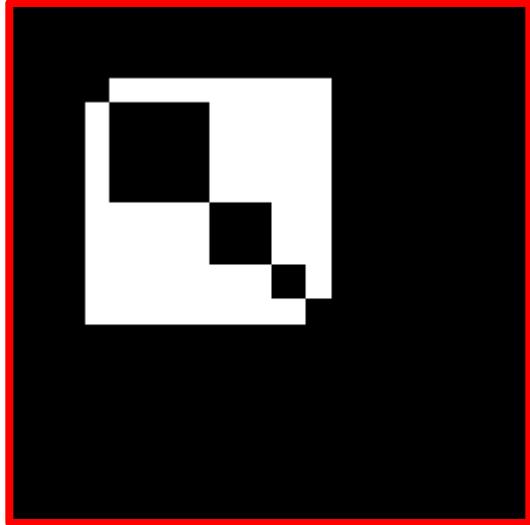


U

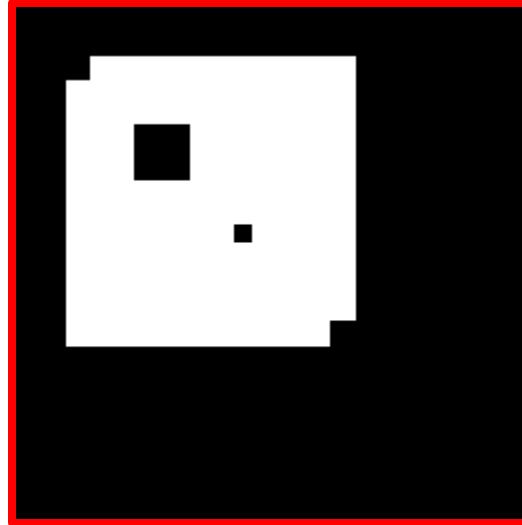


Close

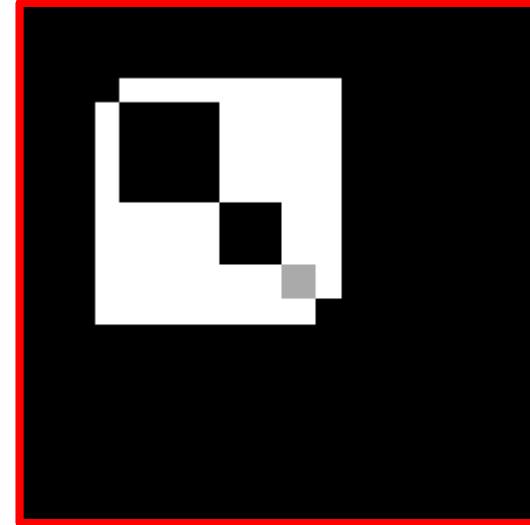
A



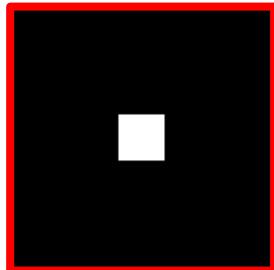
$O = \text{DILATE}(A, U)$



$O = \text{ERODE}(O, U)$



U



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

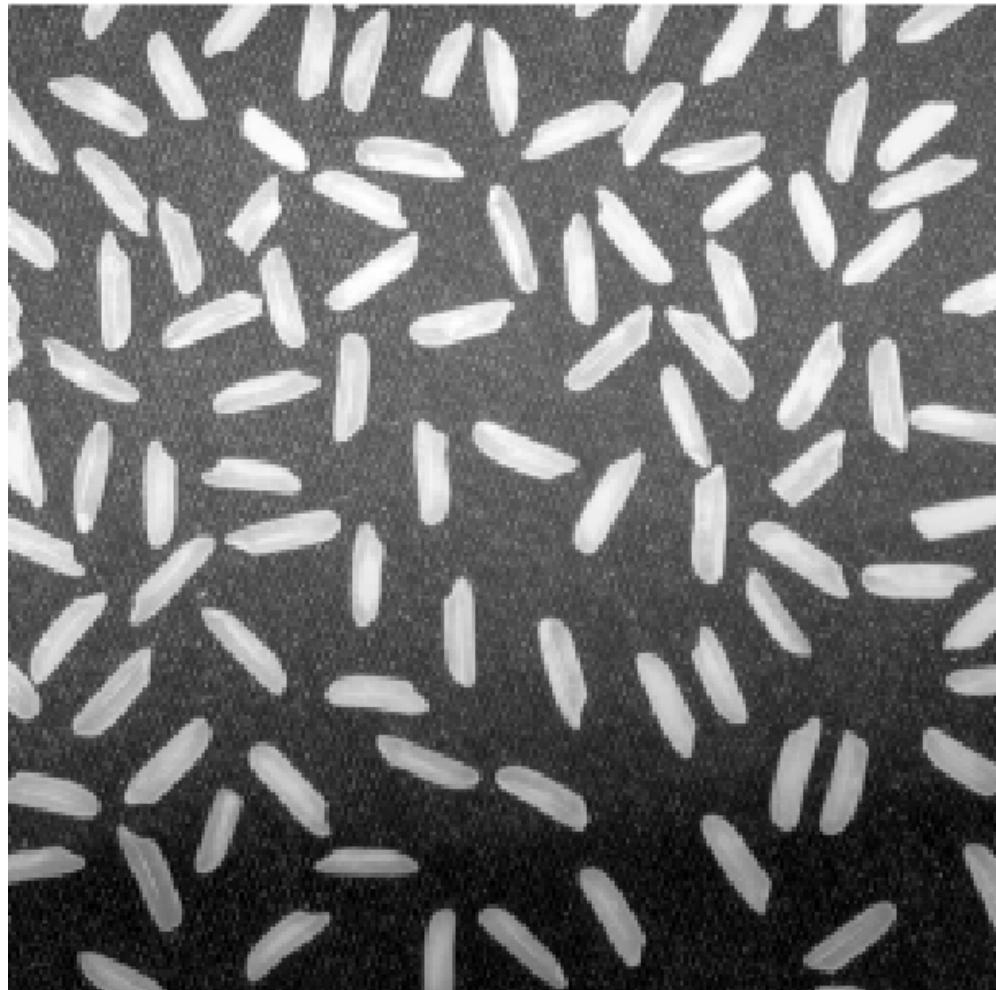
A bit more on Nonlinear Filters

Edge Detection (Canny Edge Detector)

Going to Binary Images

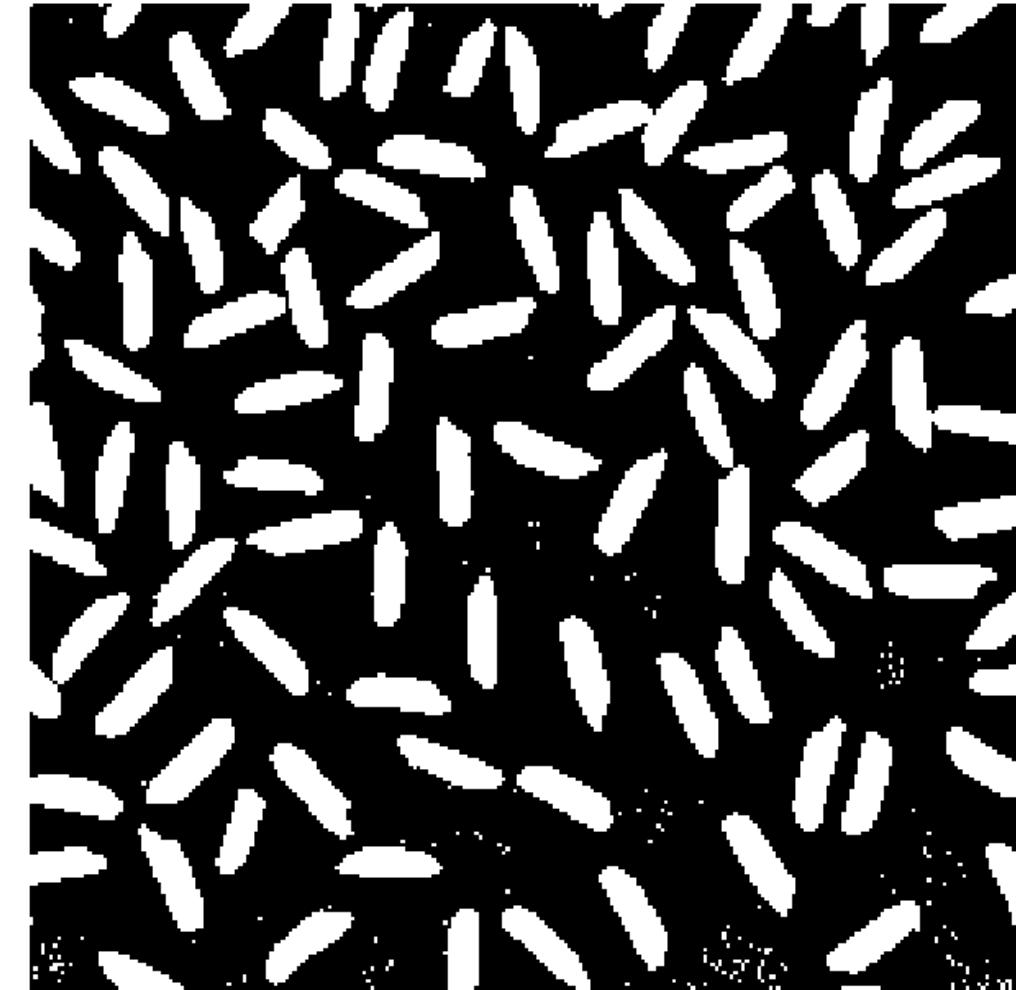
Plenty morphological information can be extracted
from binary images

Binarization



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

$I > r$



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

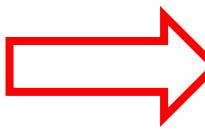
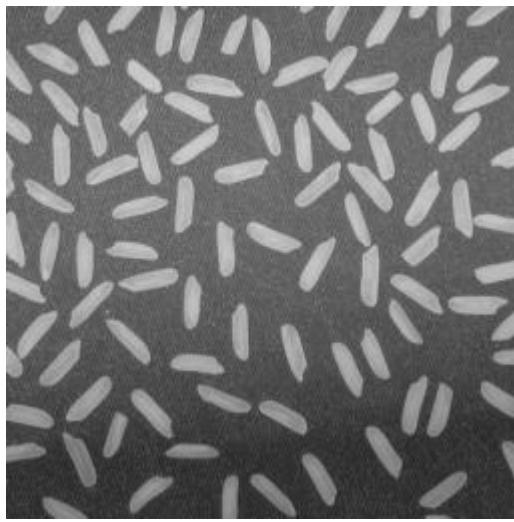


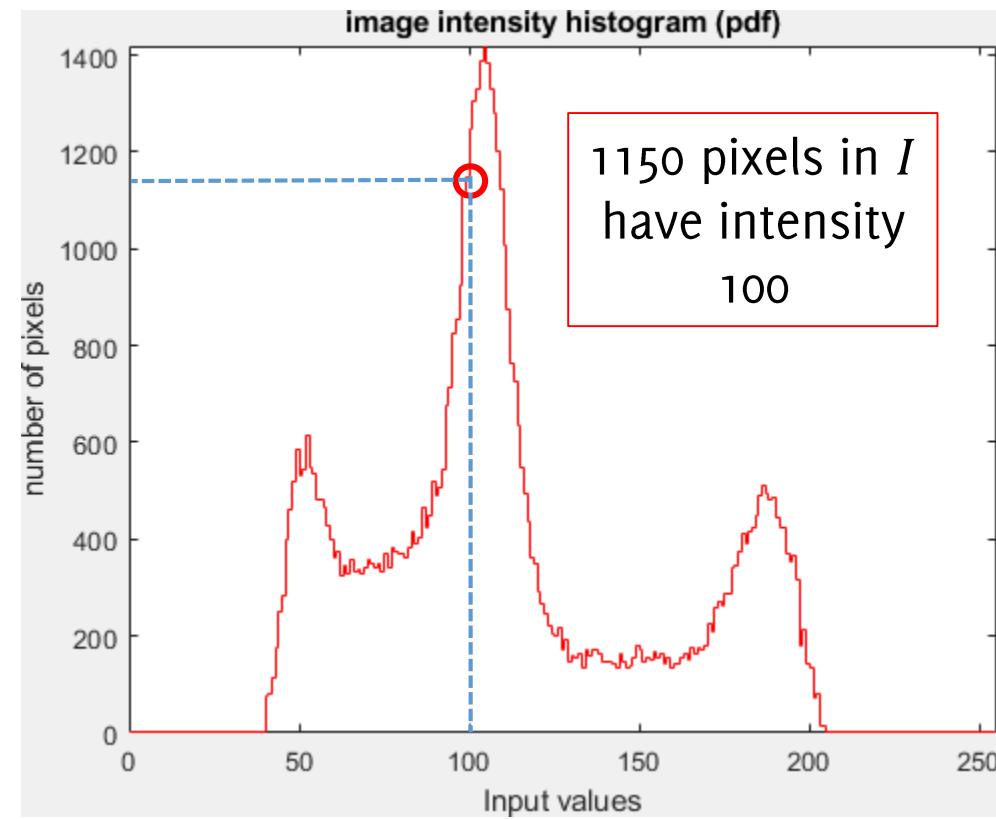
Image histograms

Histogram of pixel intensities can be used to define intensity transformation



img

$$h_{100} = 1150$$

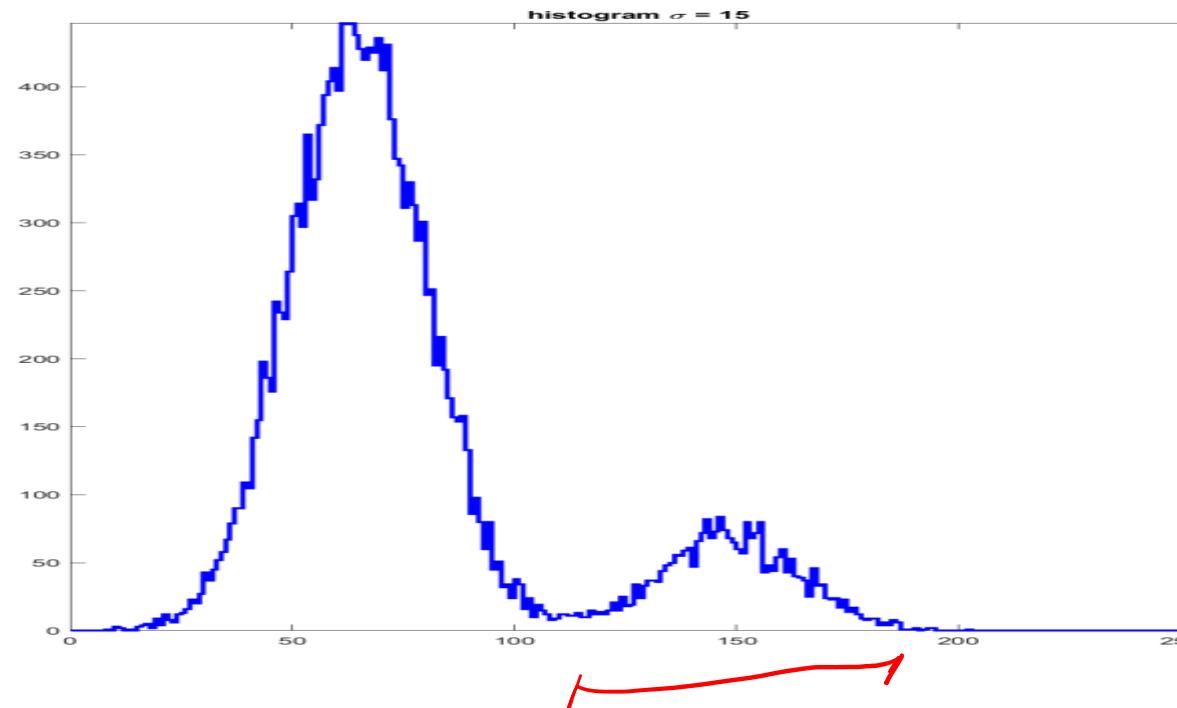


$i = 100$

Histogram

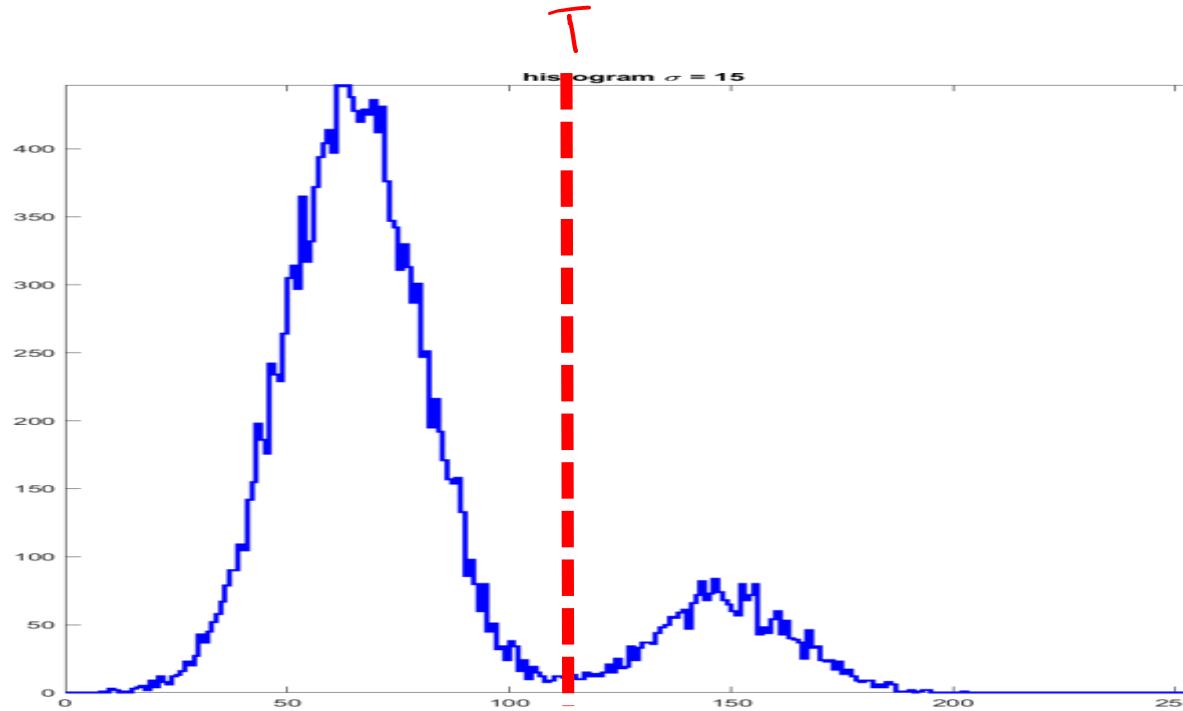
Threshold definition by Otsu Method

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



Threshold definition by Otsu Method

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



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

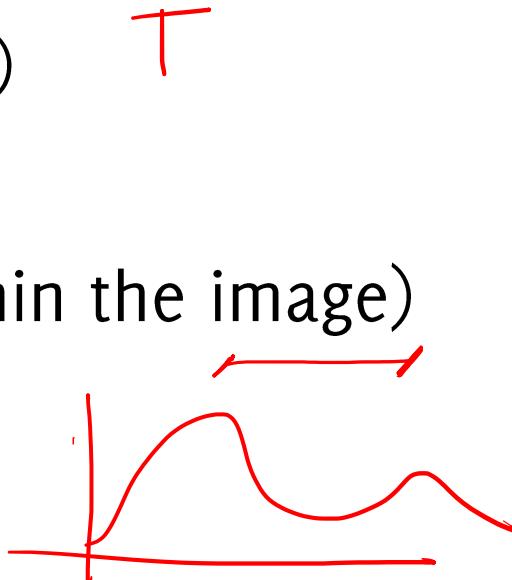
Thresholding

Different options:

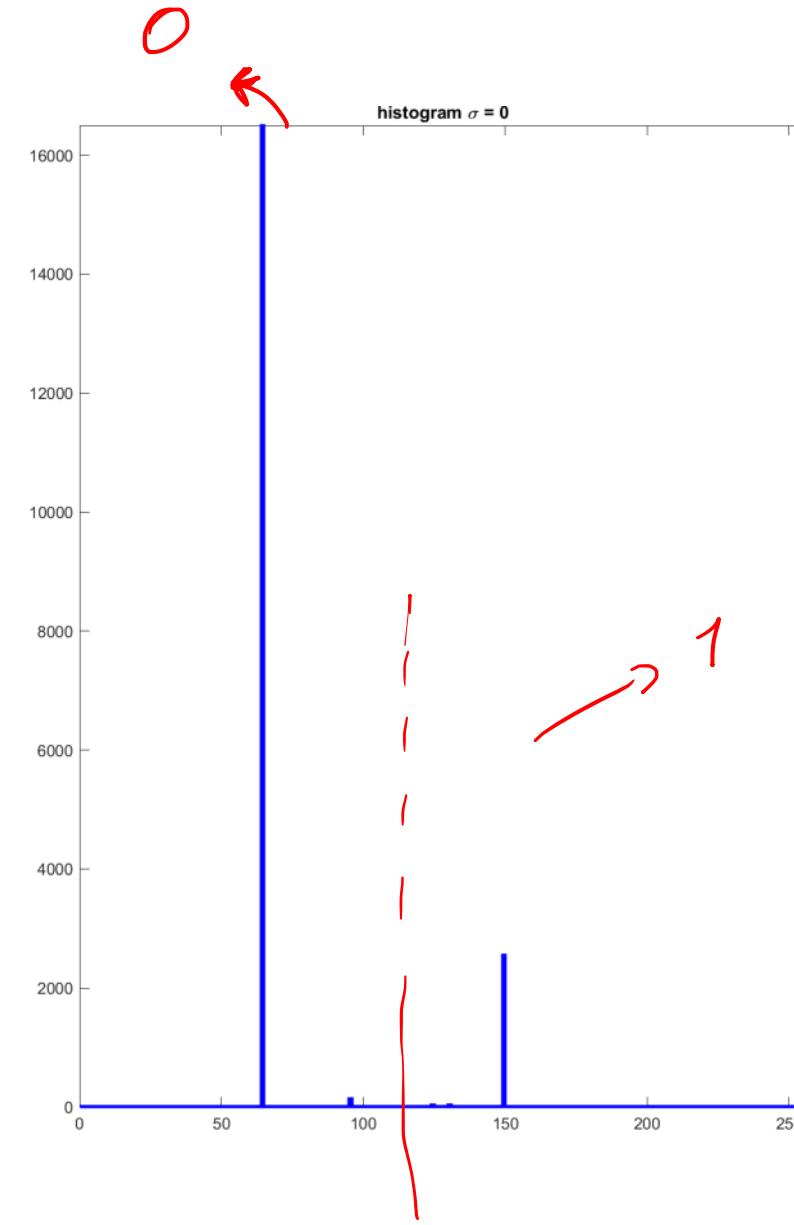
- **Global** (a fixed threshold for the whole image)
- Variable (threshold defined locally)
- Adaptive (the way threshold is set varies within the image)

The success of thresholding depends on:

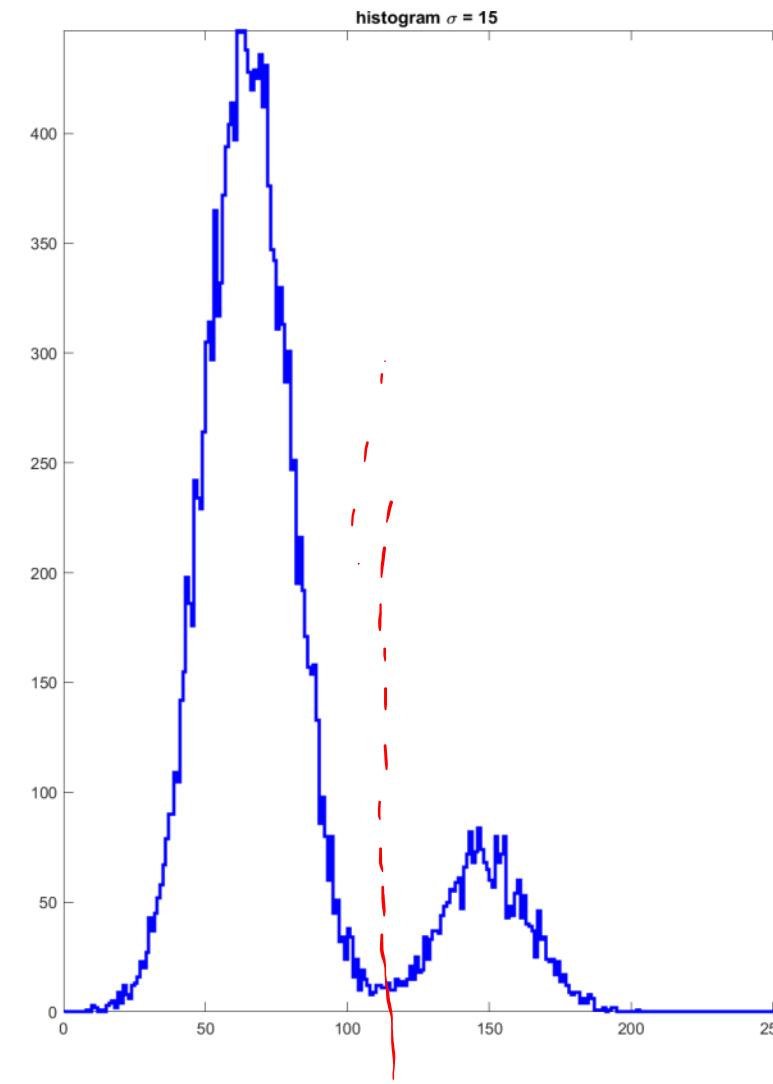
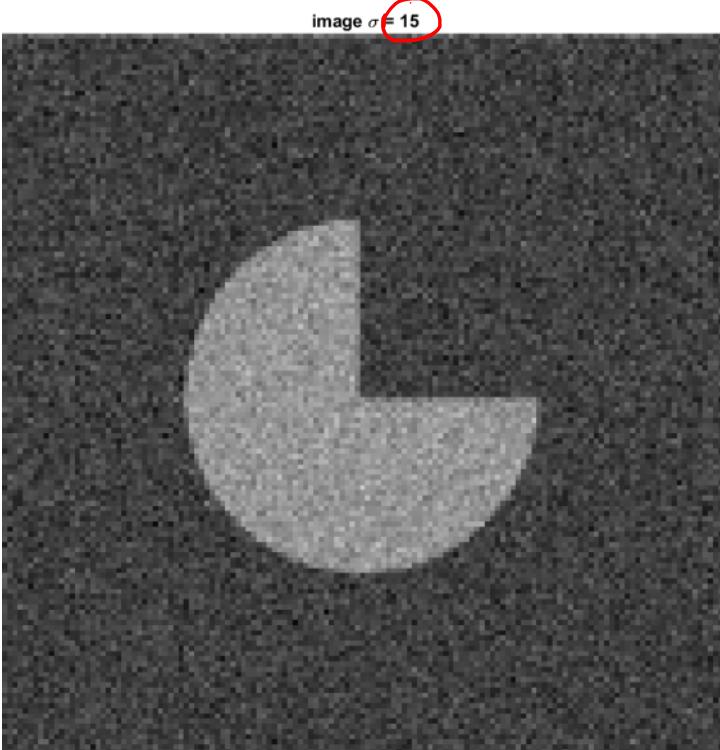
- Peaks separation in the grayscale image
- Noise
- Relative size of the two peaks
- Uniformity of scene illumination / reflectance



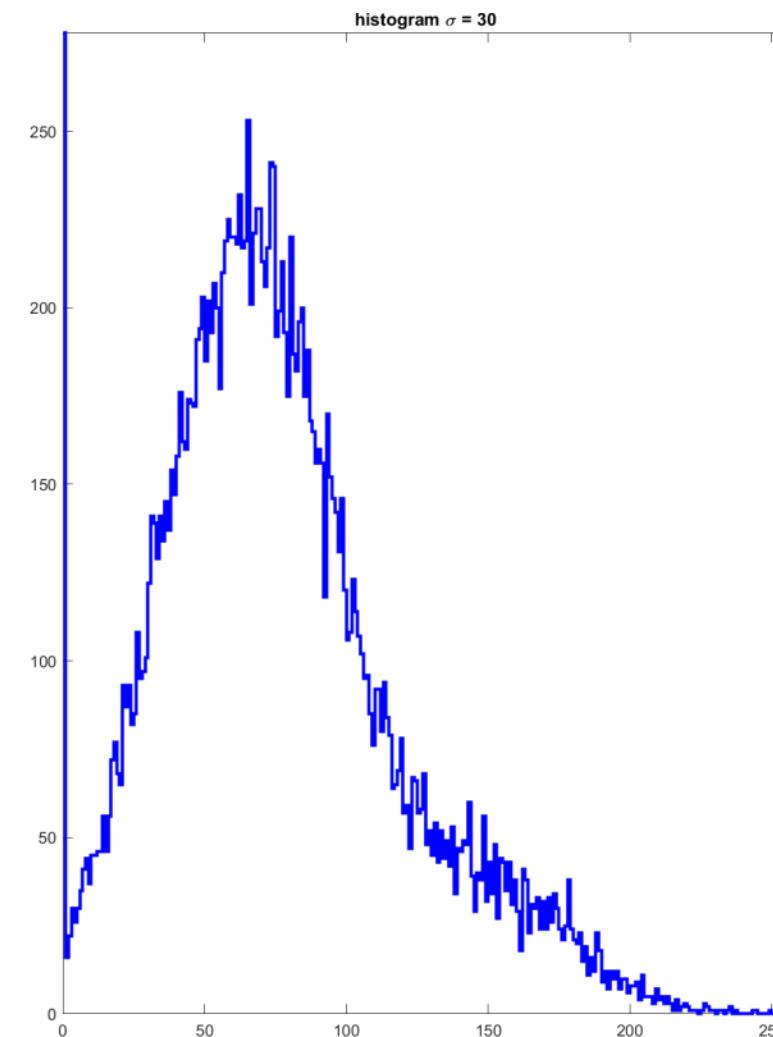
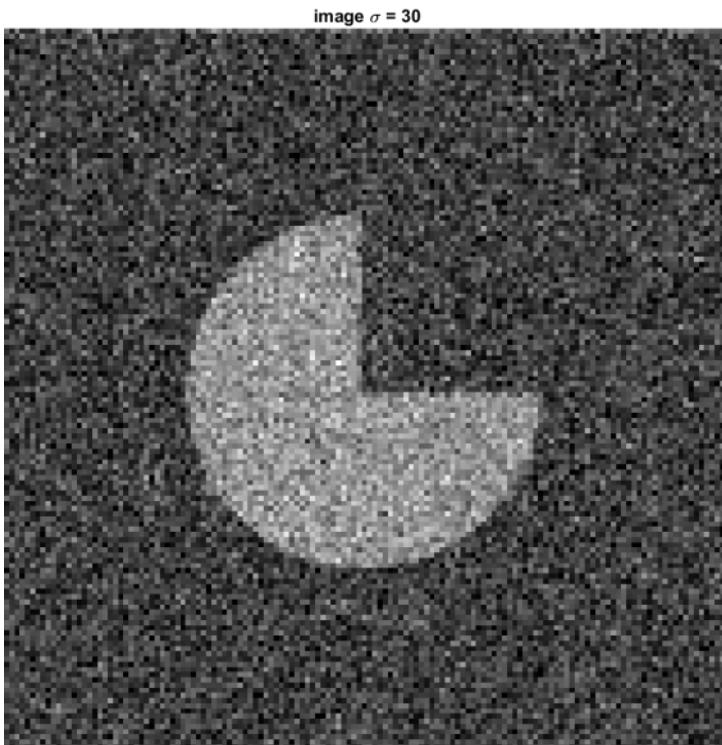
Binary Image



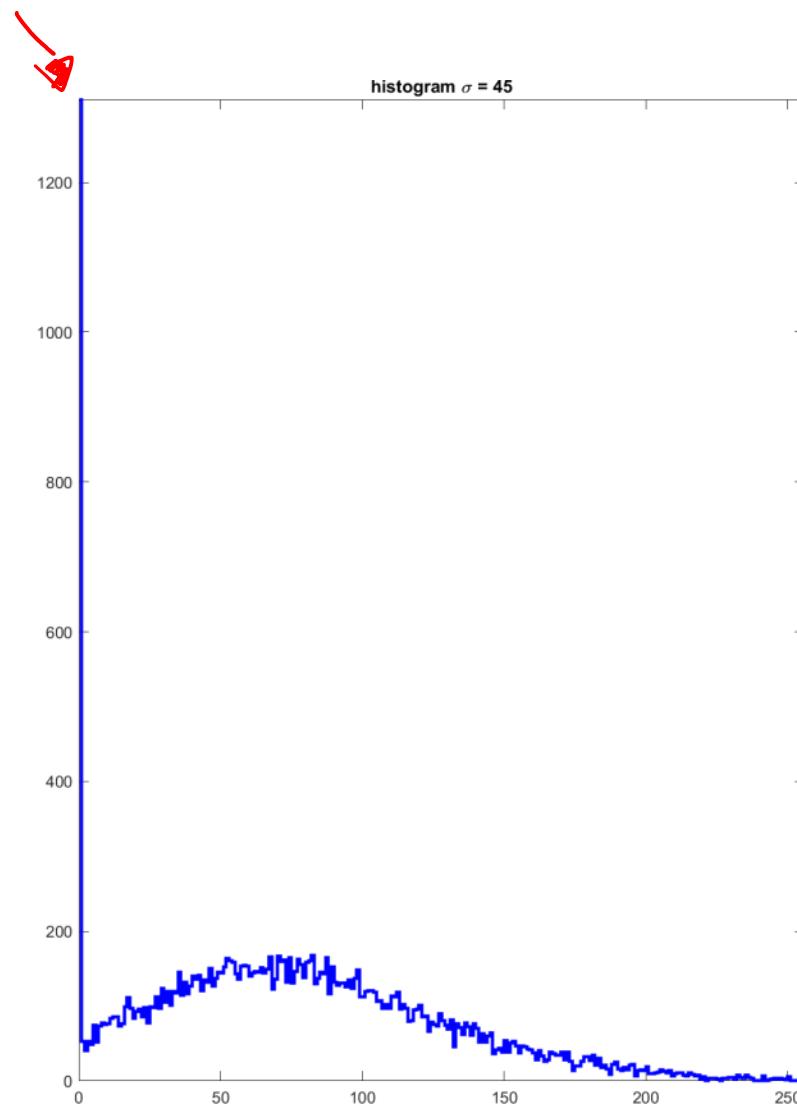
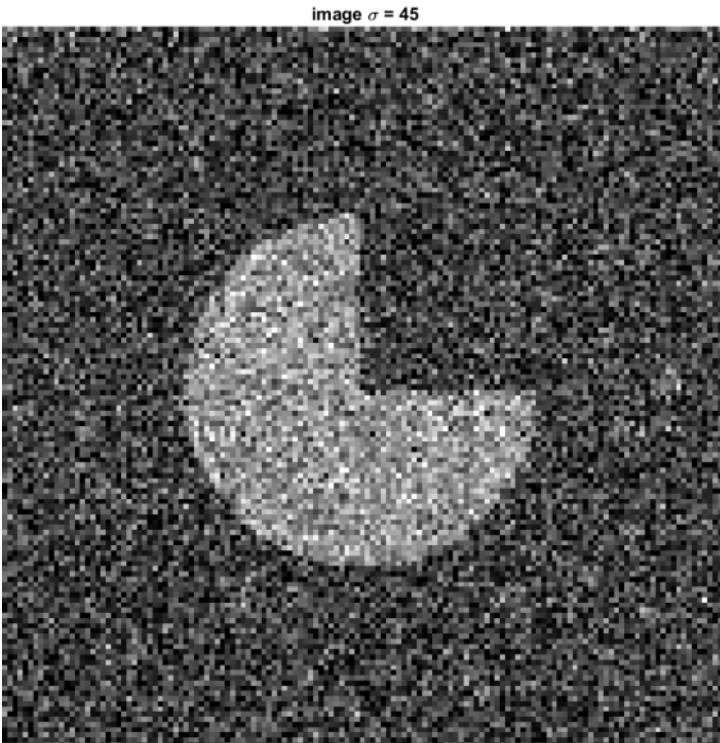
Noisy Binary Image



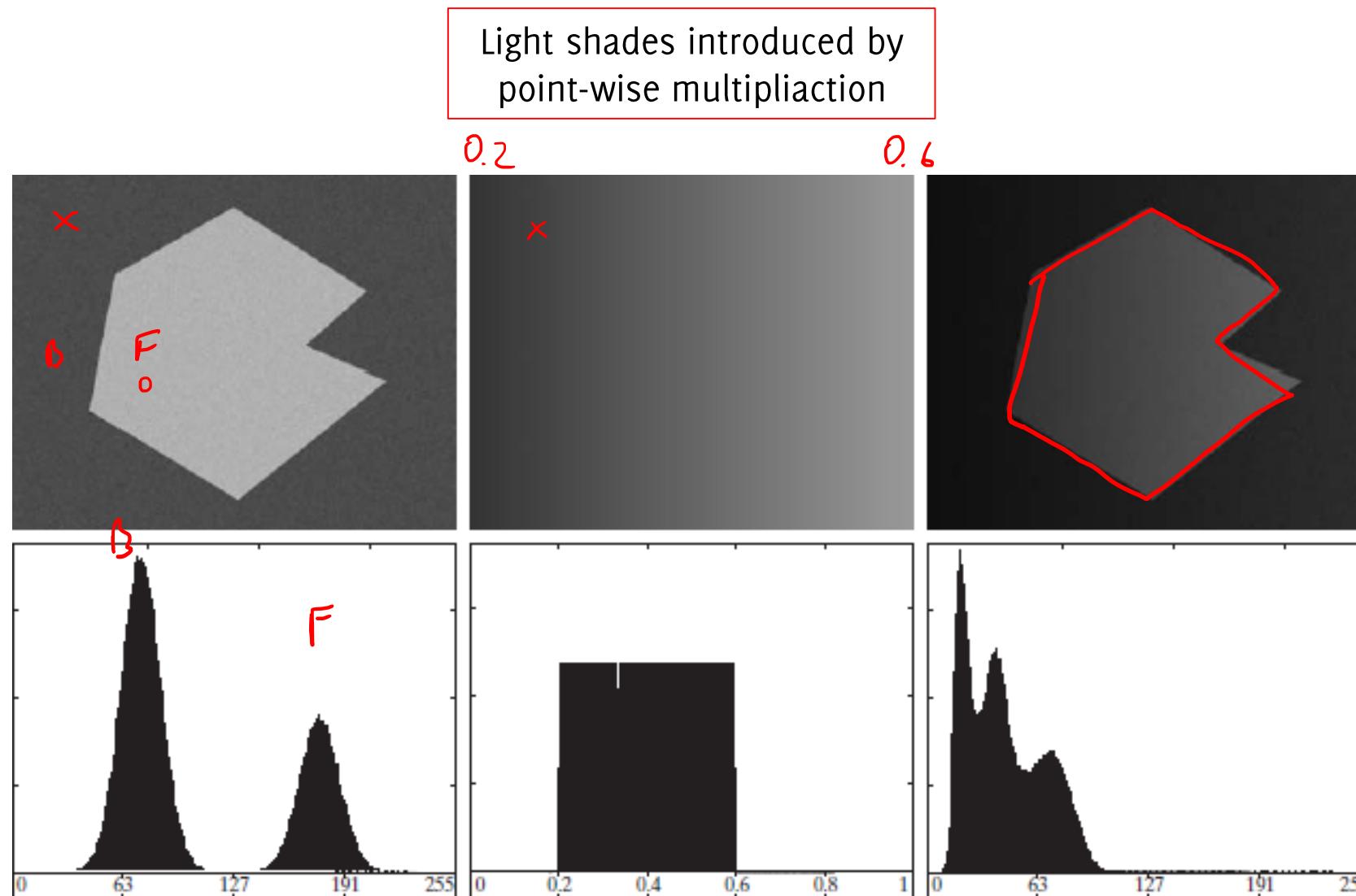
Noisy Binary Image



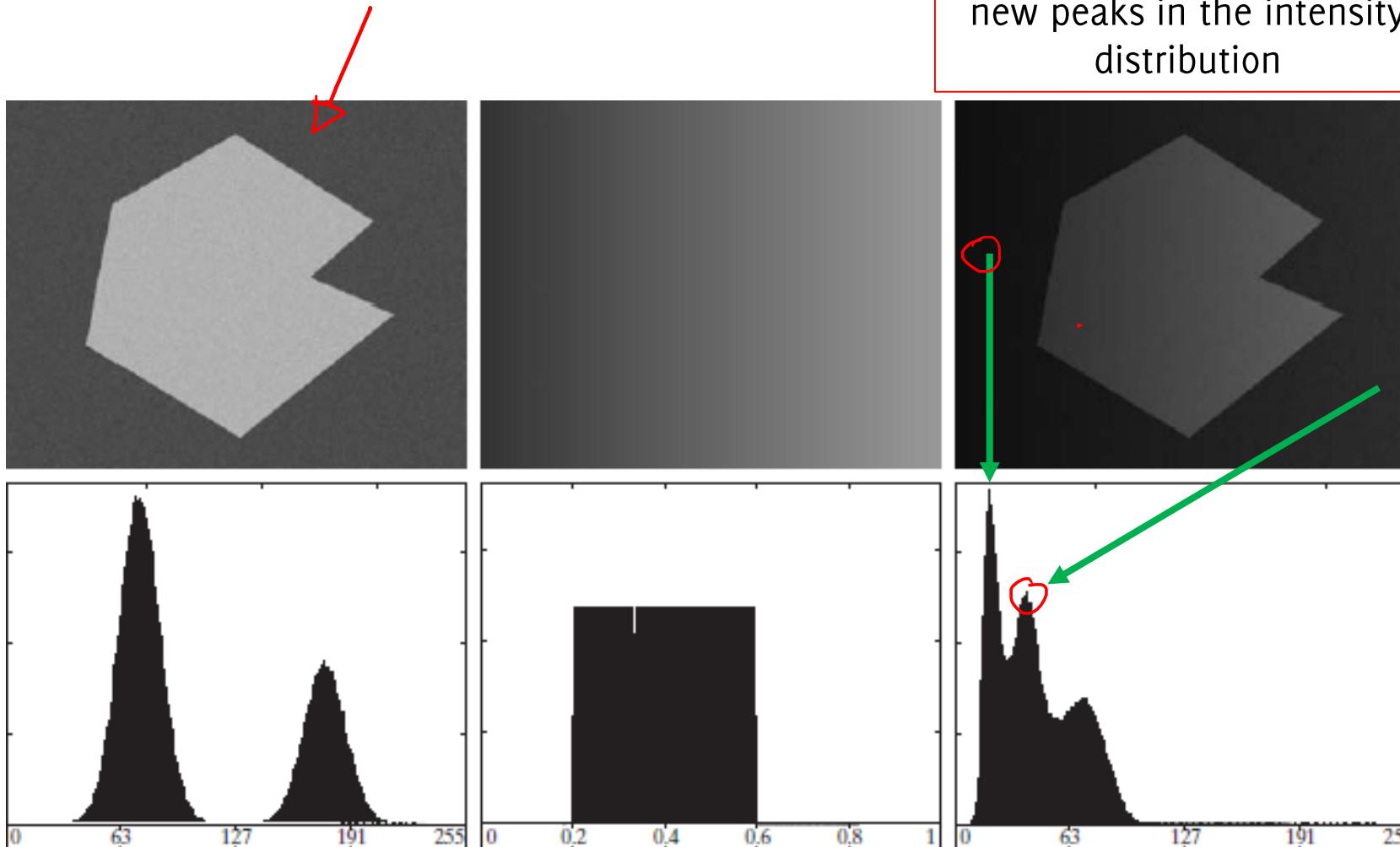
Noisy Binary Image



The impact of light shades on image binarization



The impact of light shades on image binarization



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**

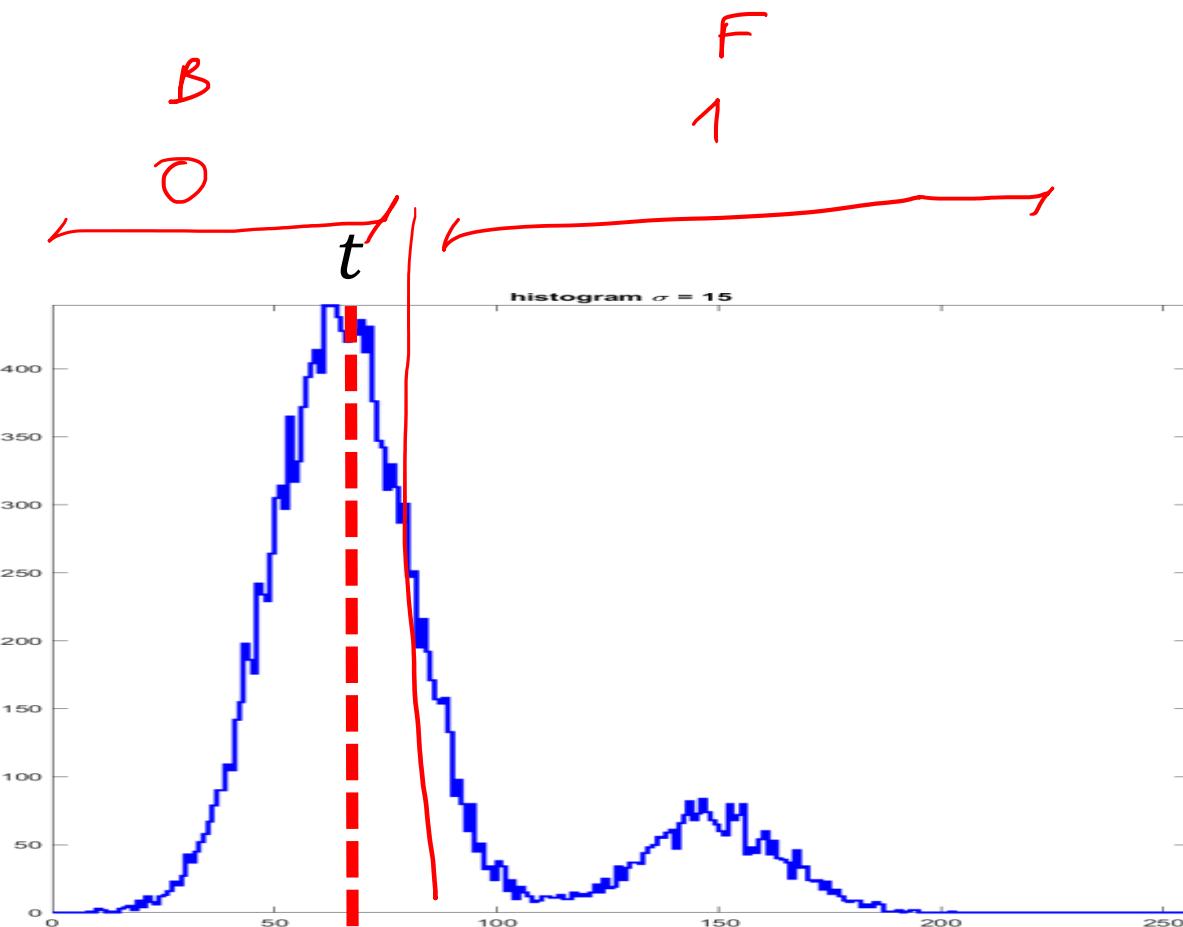
This is equivalent to maximizing the inter-class variance, which can be done more efficiently using iterative expressions and histogram representation of the image.

In Python **`skimage.filters.threshold_otsu`**

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

$$t = \operatorname{argmin}_{\tau} \sigma_w^2(\tau)$$



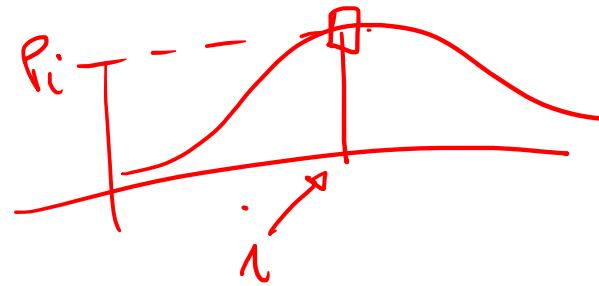
where the intra-class variance is

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

$[0, 255]$

And w_0, w_1 are the class proportions

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



Otsu Method

This is shown to be equivalent to

$$t = \operatorname{argmax}_{\tau} \sigma_b^2(\tau)$$

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

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

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

$$\begin{aligned}w_0(t-1), w_1(t-1) \\w_0(t) = w_0(t-1) + P_{c-1}\end{aligned}$$

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

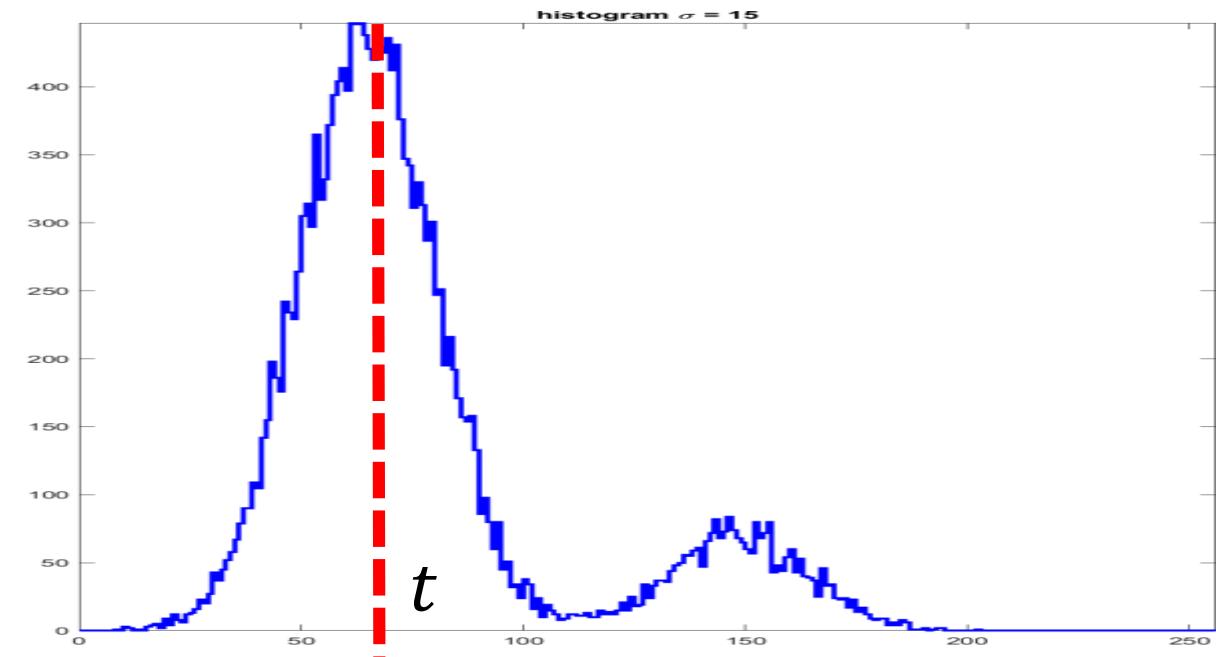
$$w_1(t) = w_1(t-1) - P_{c-1}$$

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

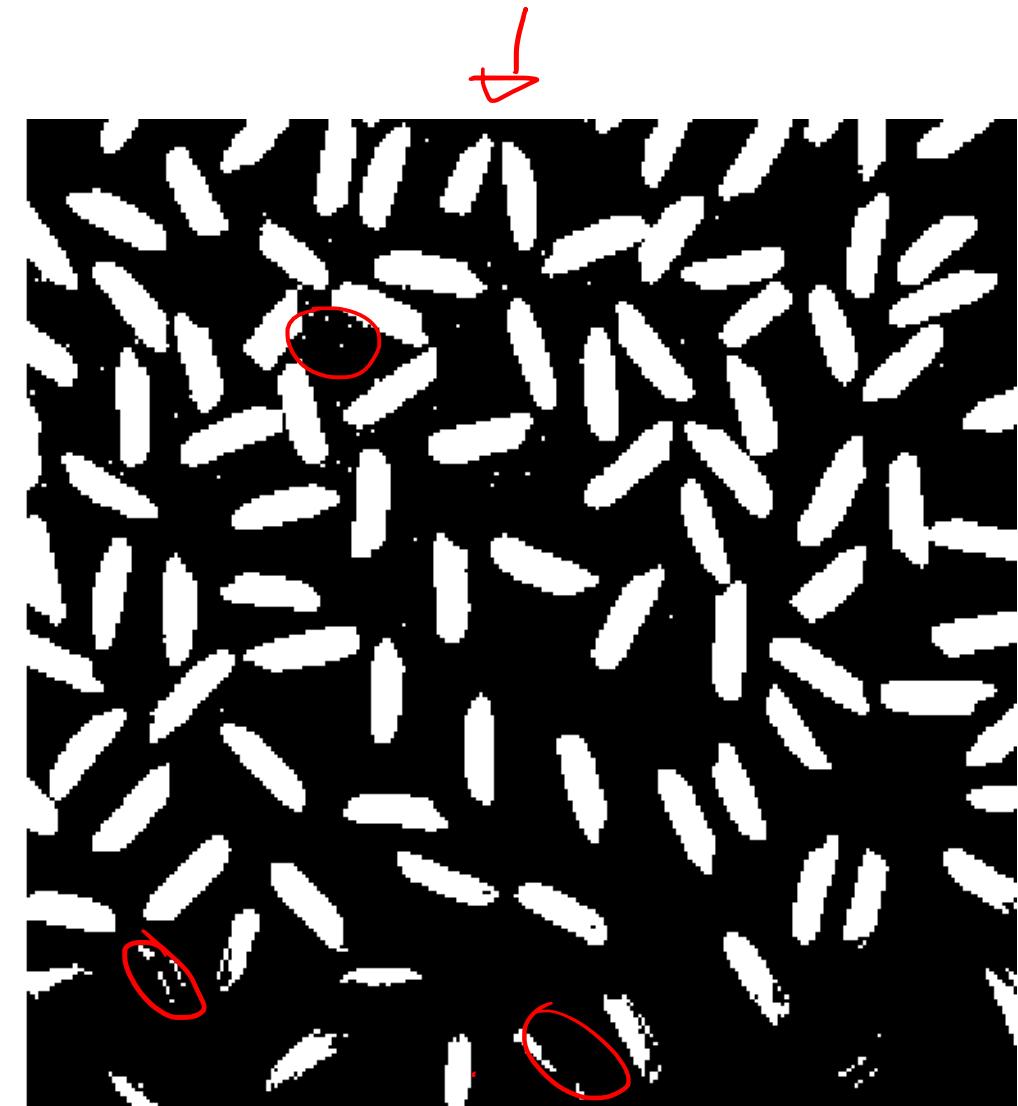
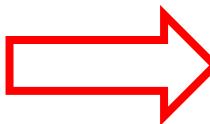
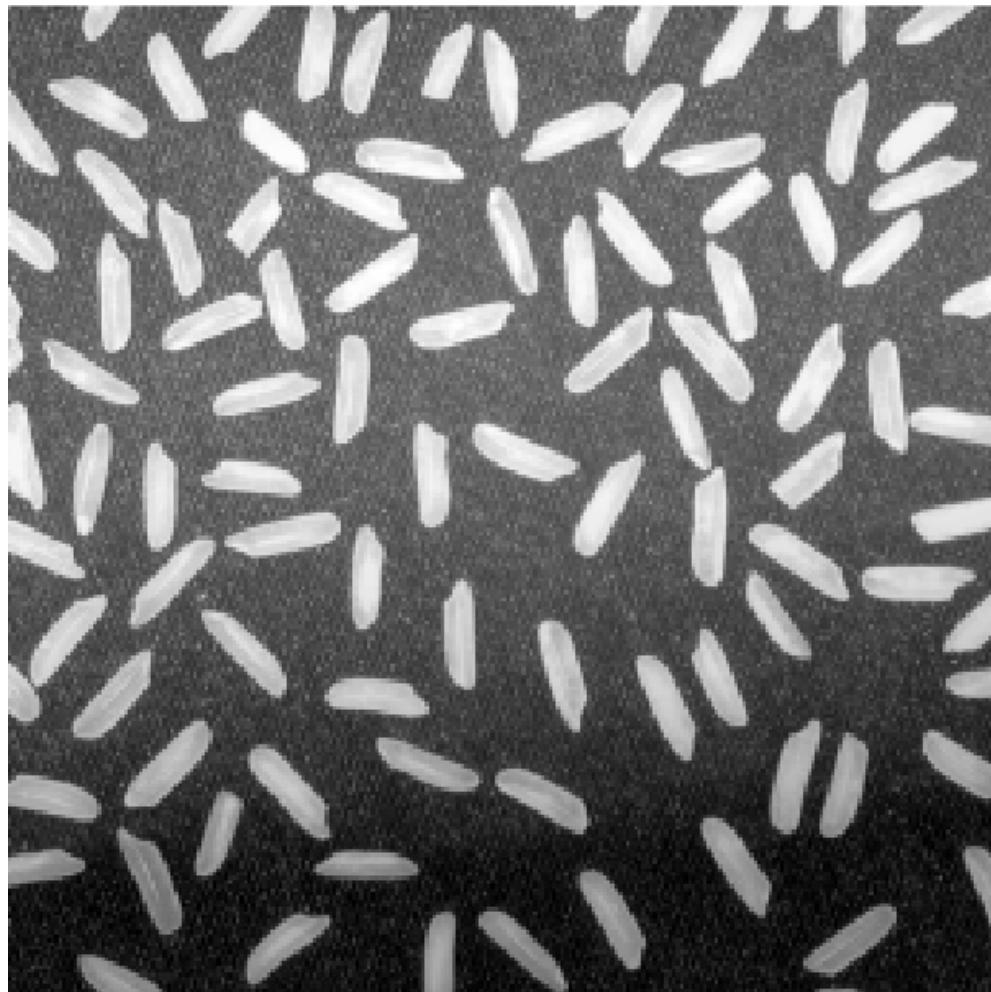
$$\mu_0(t) = \frac{\sum_{i=0}^{t-1} i p_i}{w_0(t)} \text{ and } \mu_1(t) = \frac{\sum_{i=t}^{L-1} i p_i}{w_1(t)}$$

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

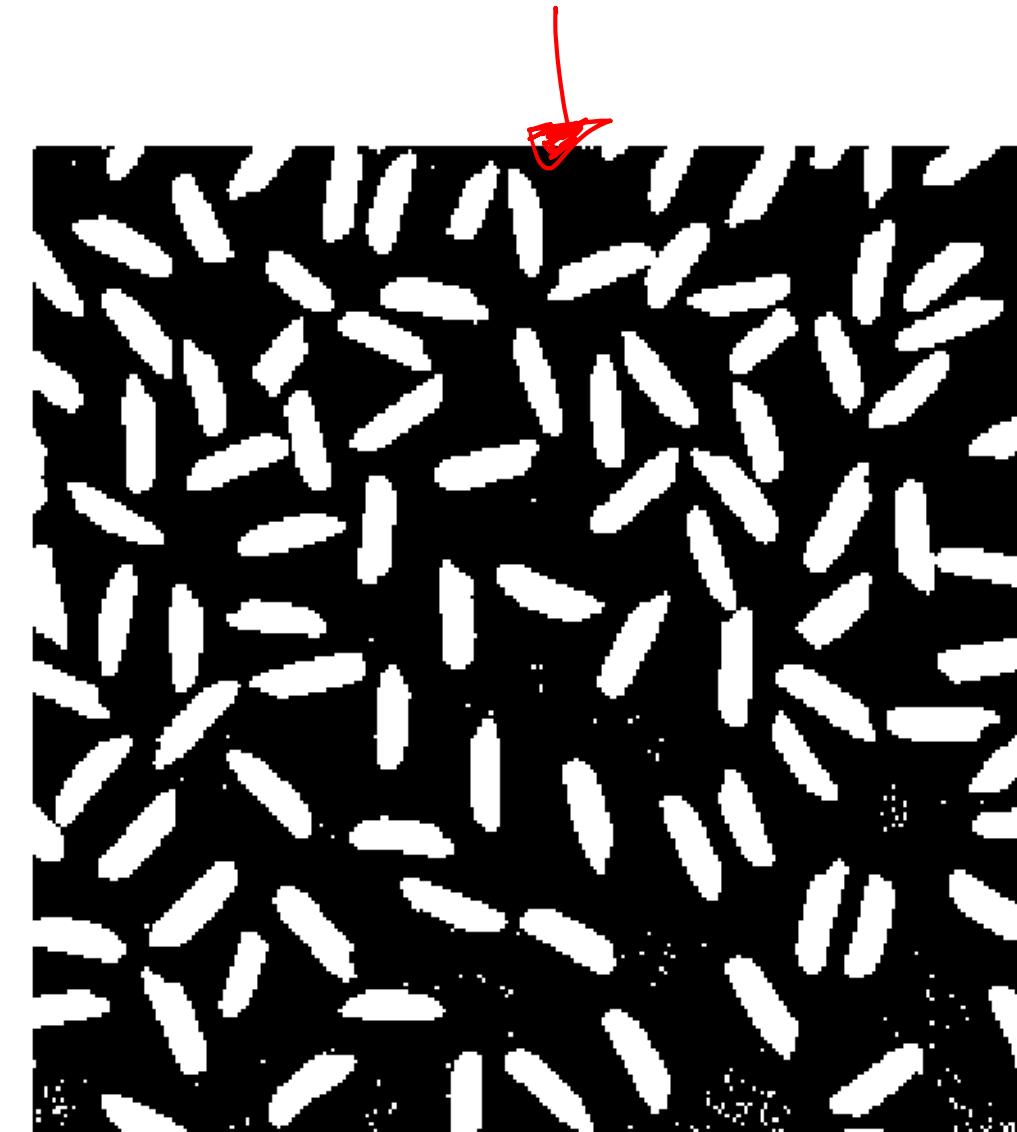
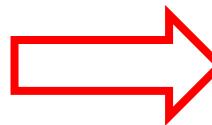
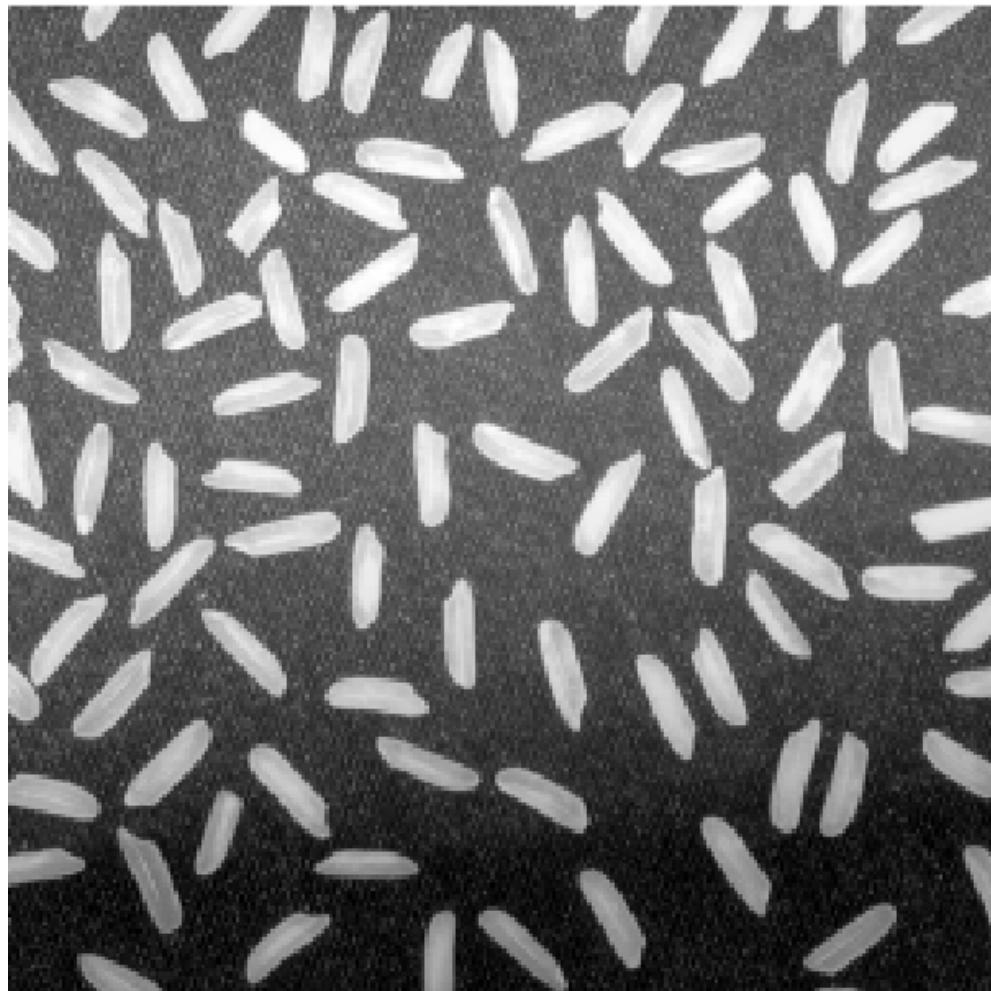
$$T = \arg \max_t [\sigma_b^2(t)]^2$$



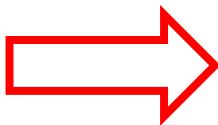
Binarization using a global threshold



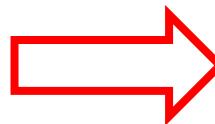
Binarization using a local threshold



Binarization using a global threshold



Binarization using a local threshold

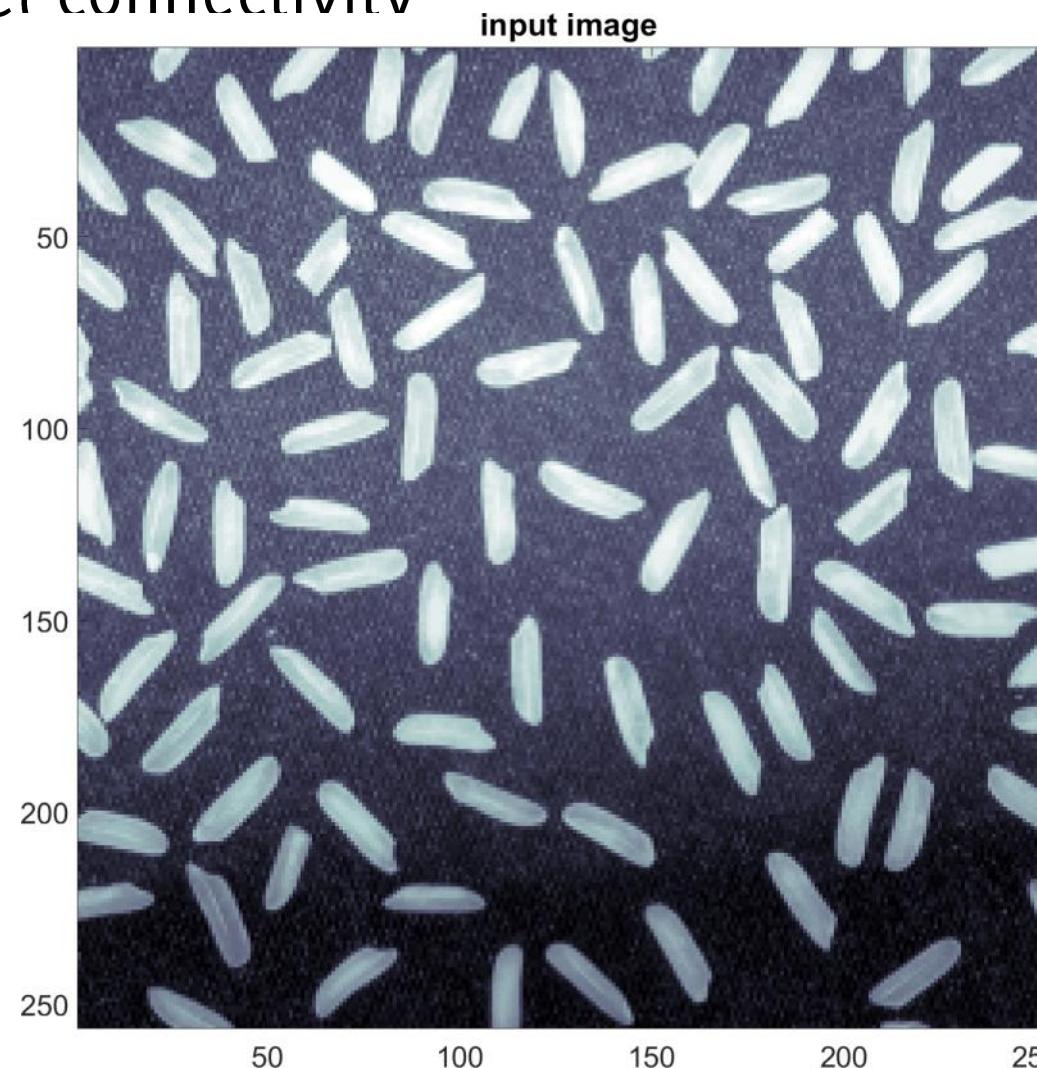


Other Morphological Operations

Blob Labeling, Connected Components

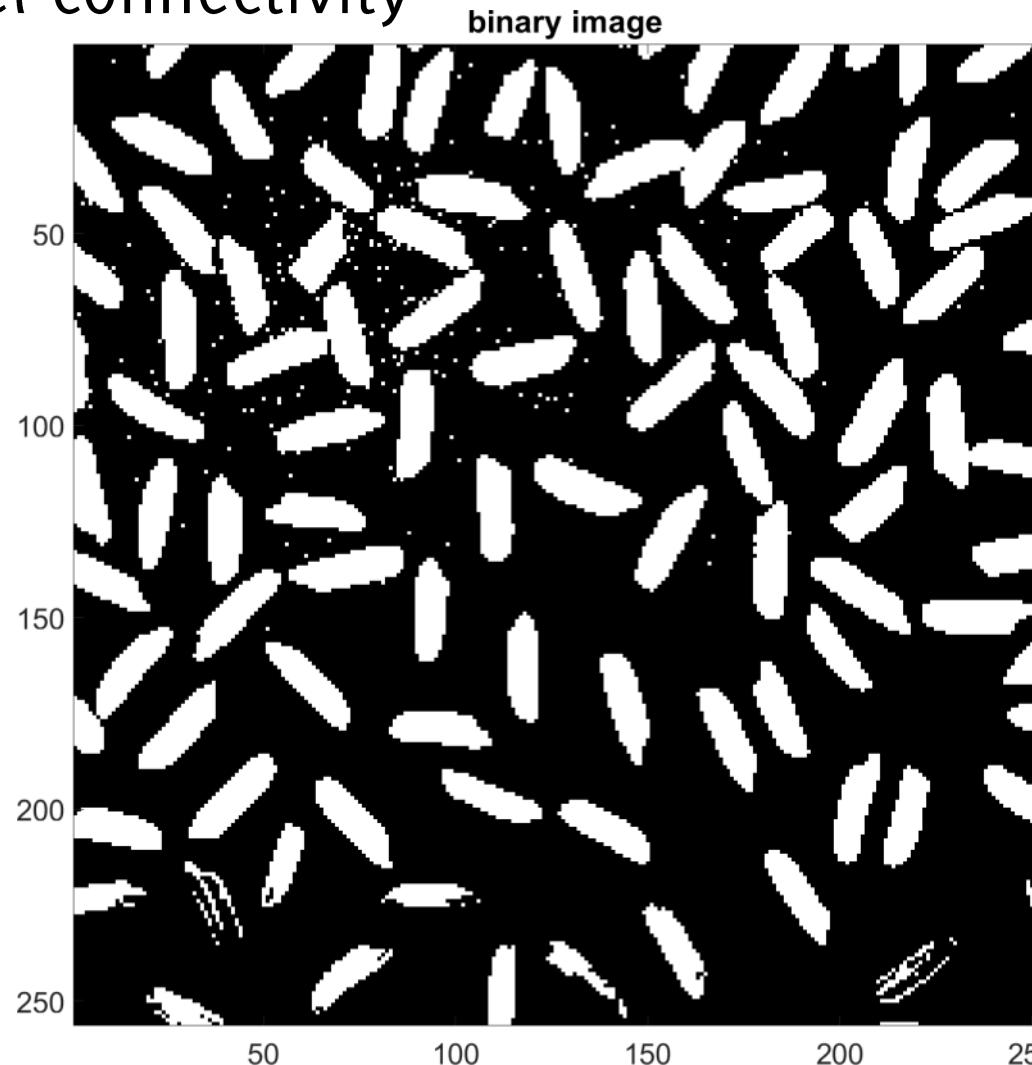
Extraction of connected components

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



Connected components

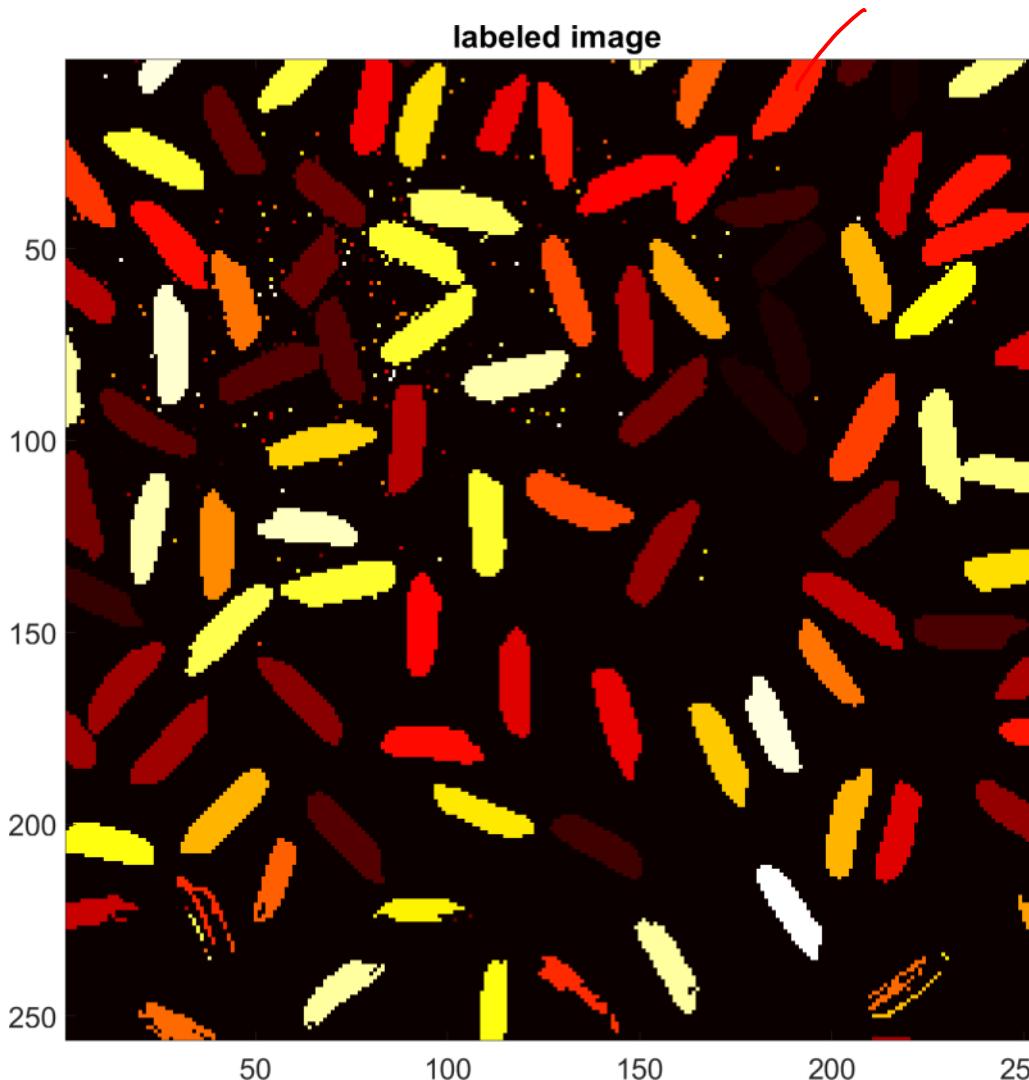
Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity



Connected components

This allows to identify different objects or target in the scene

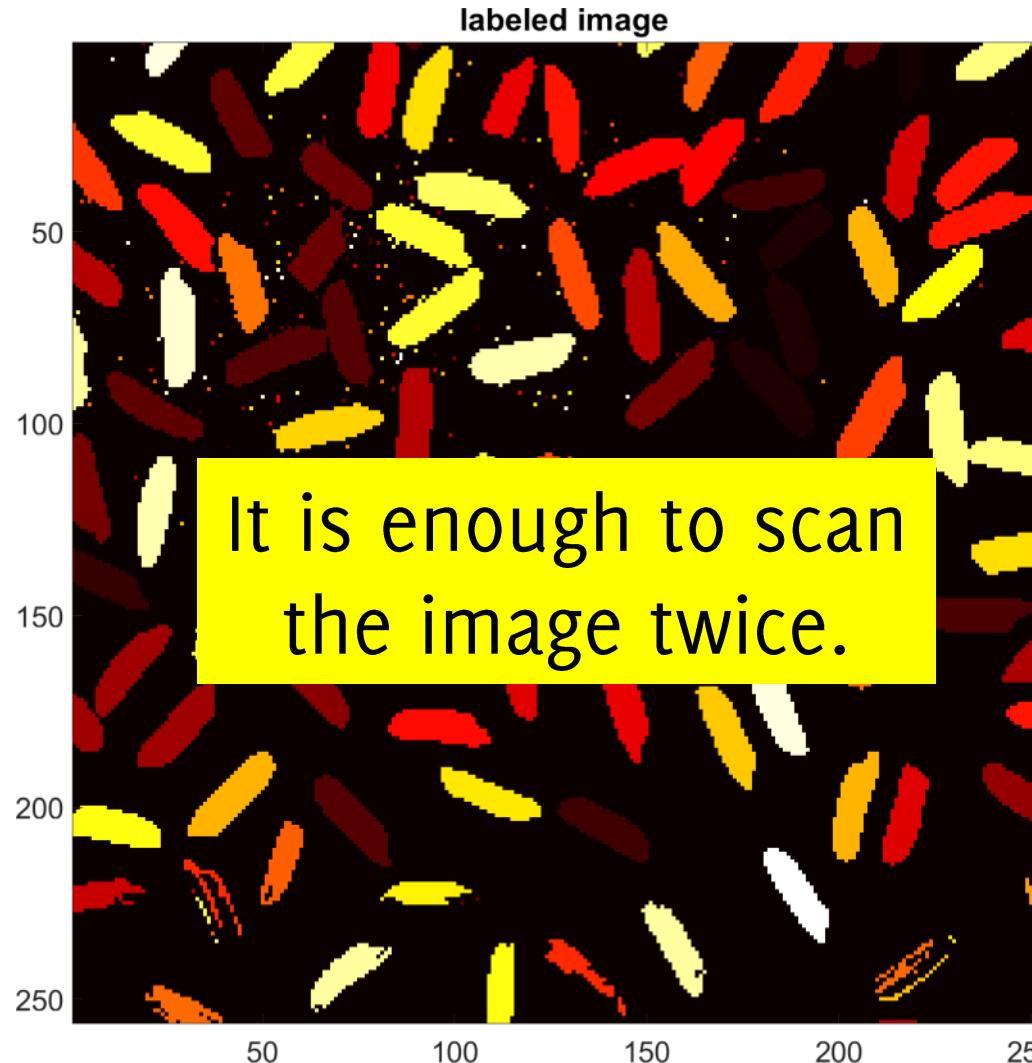
Each color corresponds
to a different label



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

Connected components

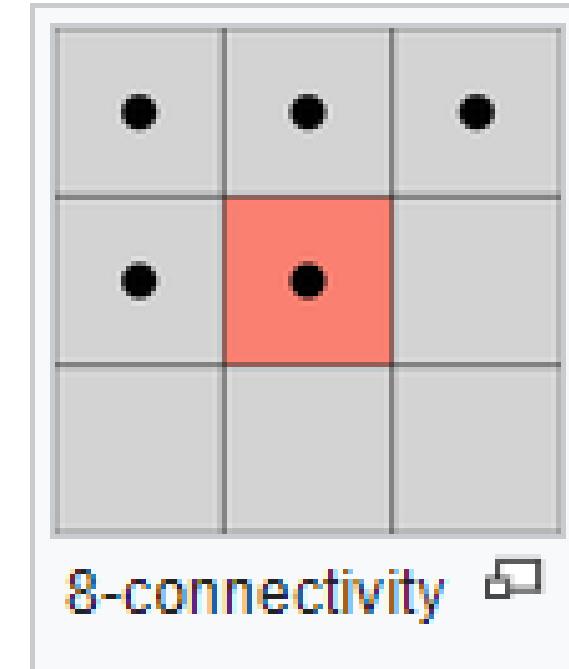
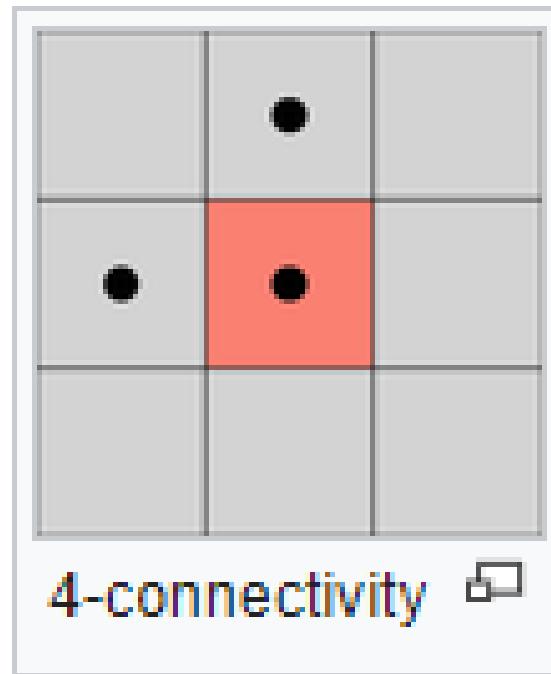
Here, each color denotes a different number, i.e. a label.



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



Two Pass Algorithm: First Pass

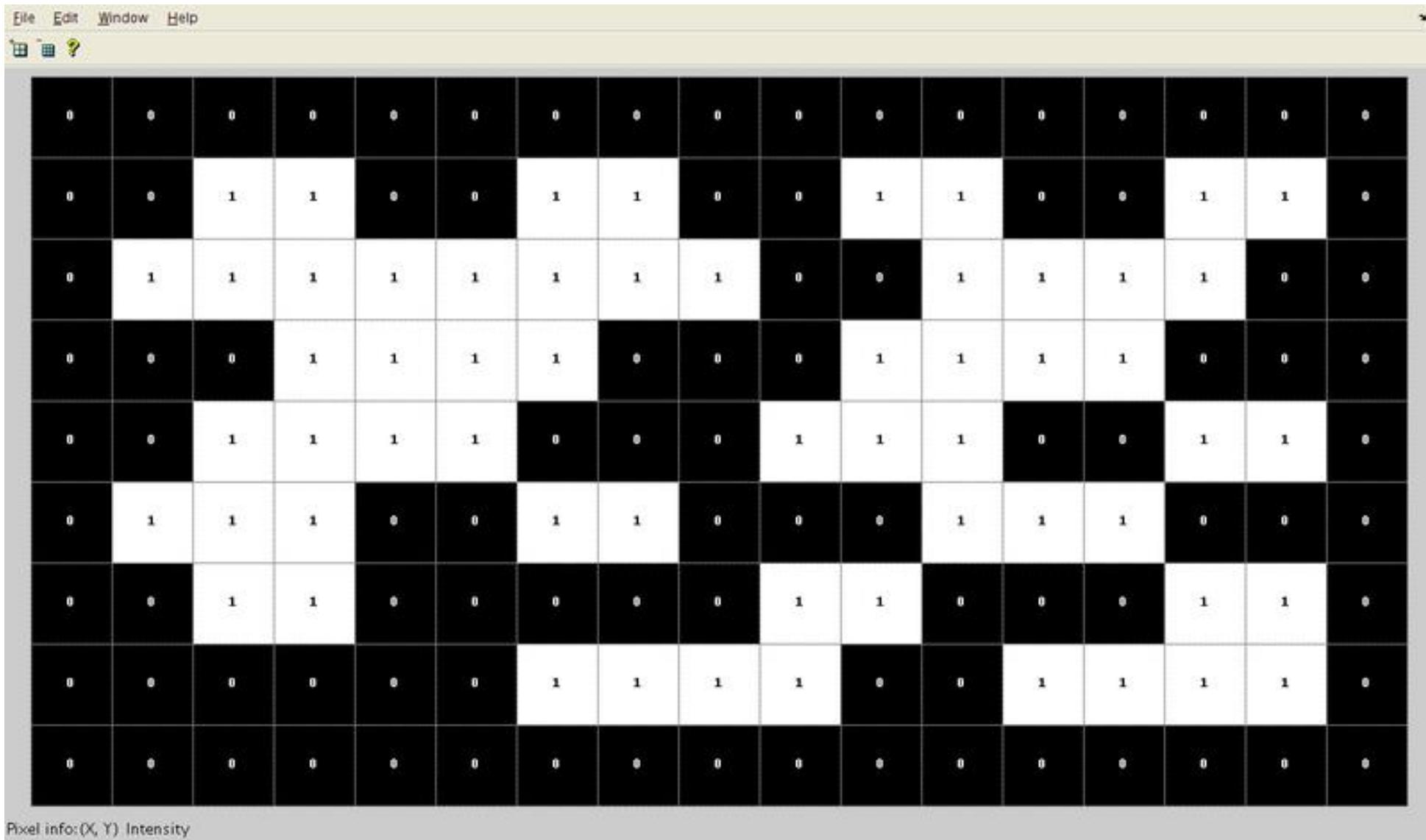
Iterate through each pixel (r, c)

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

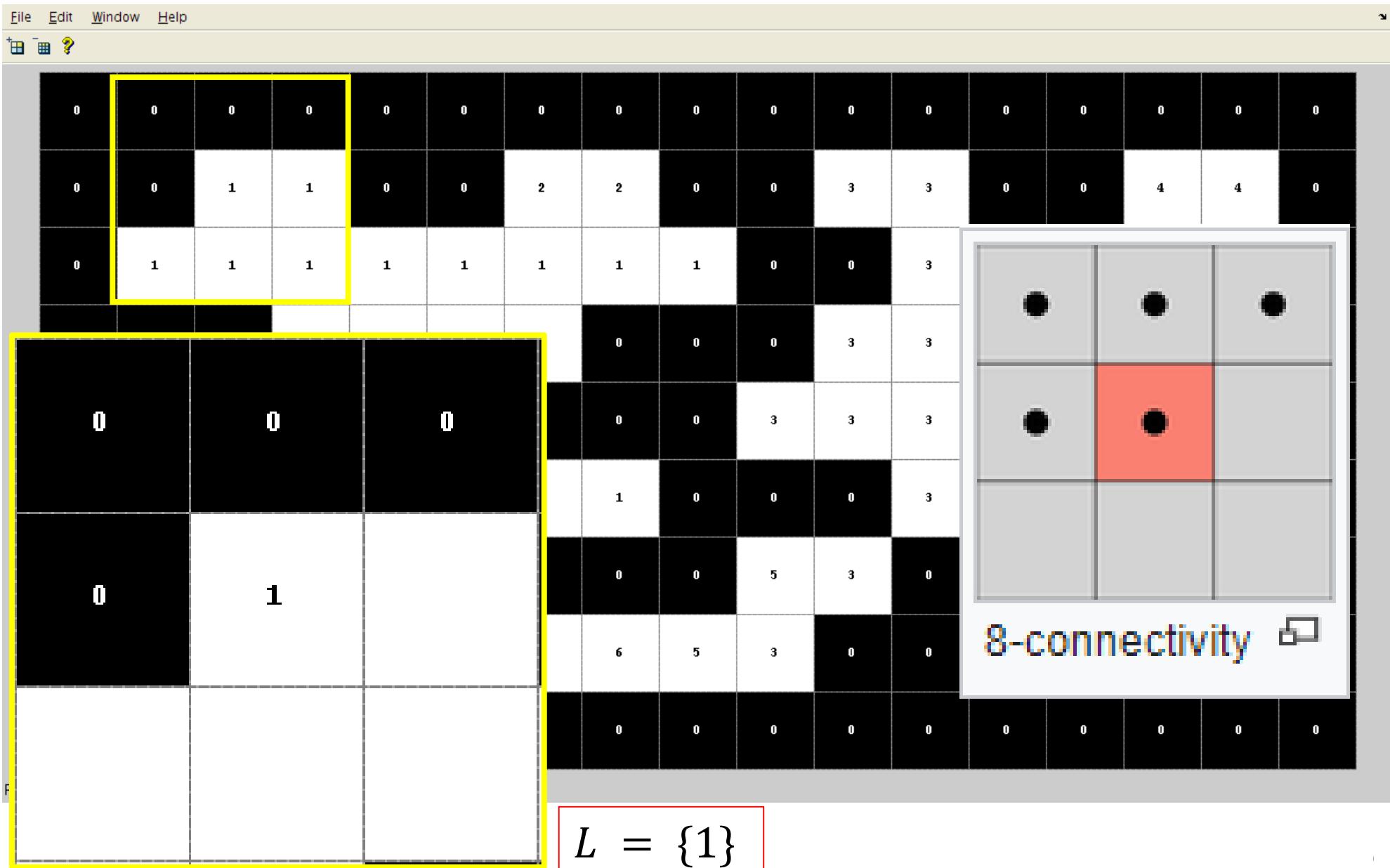
- Get a neighbor $U_{(r,c)}$ of (r, c)
- If $I(u, v) == 0 \ \forall (u, v) \in U_{(r,c)}$
 - Assign a new label $L(r, c)$
- Else $L(r, c) = \min(L(u, v))$ over $U_{(r,c)}$
 - If there are different labels in $U_{(r,c)}$
 - Record they are equivalent in a table

In Python `skimage.measure.label`

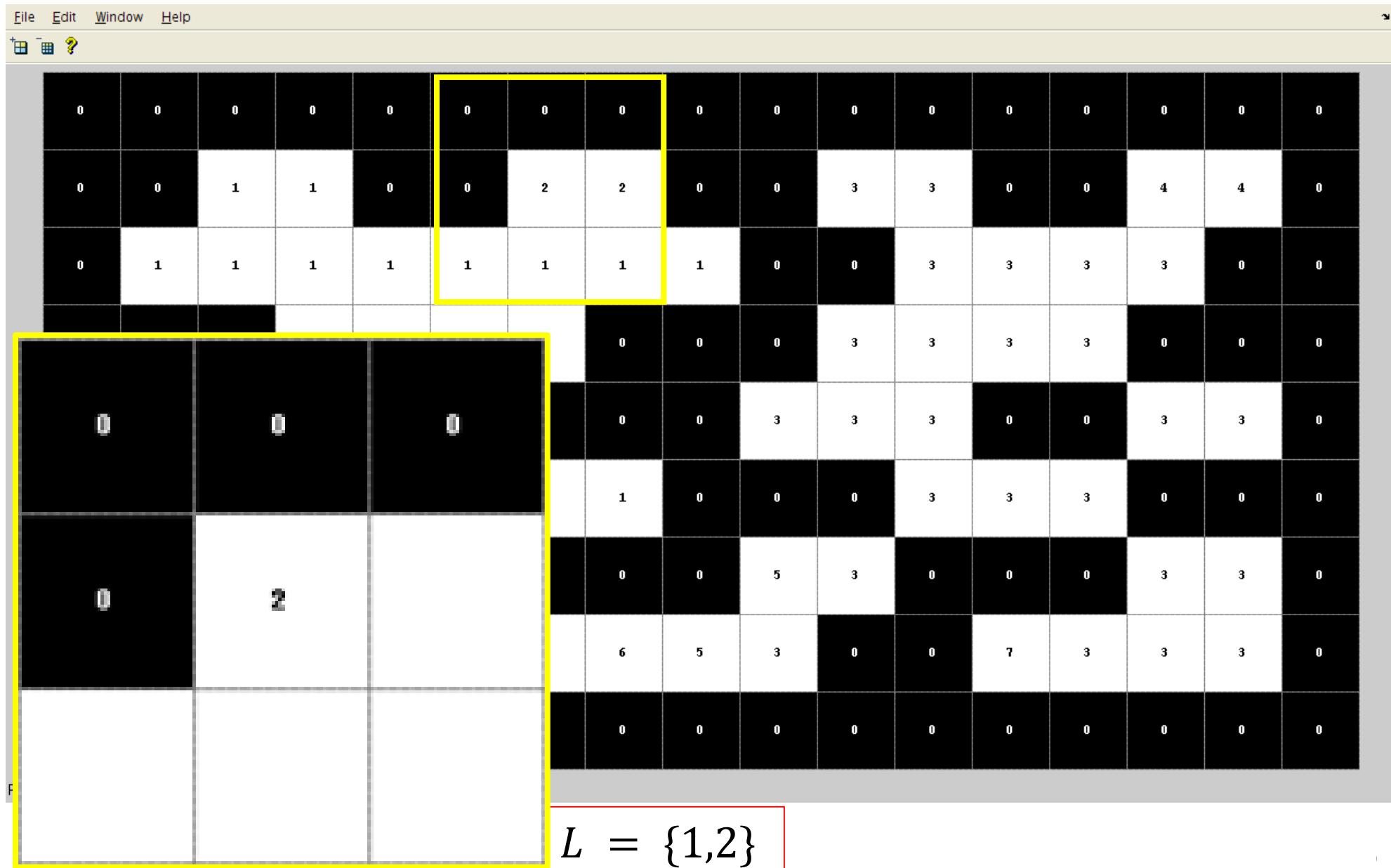
Binary input image



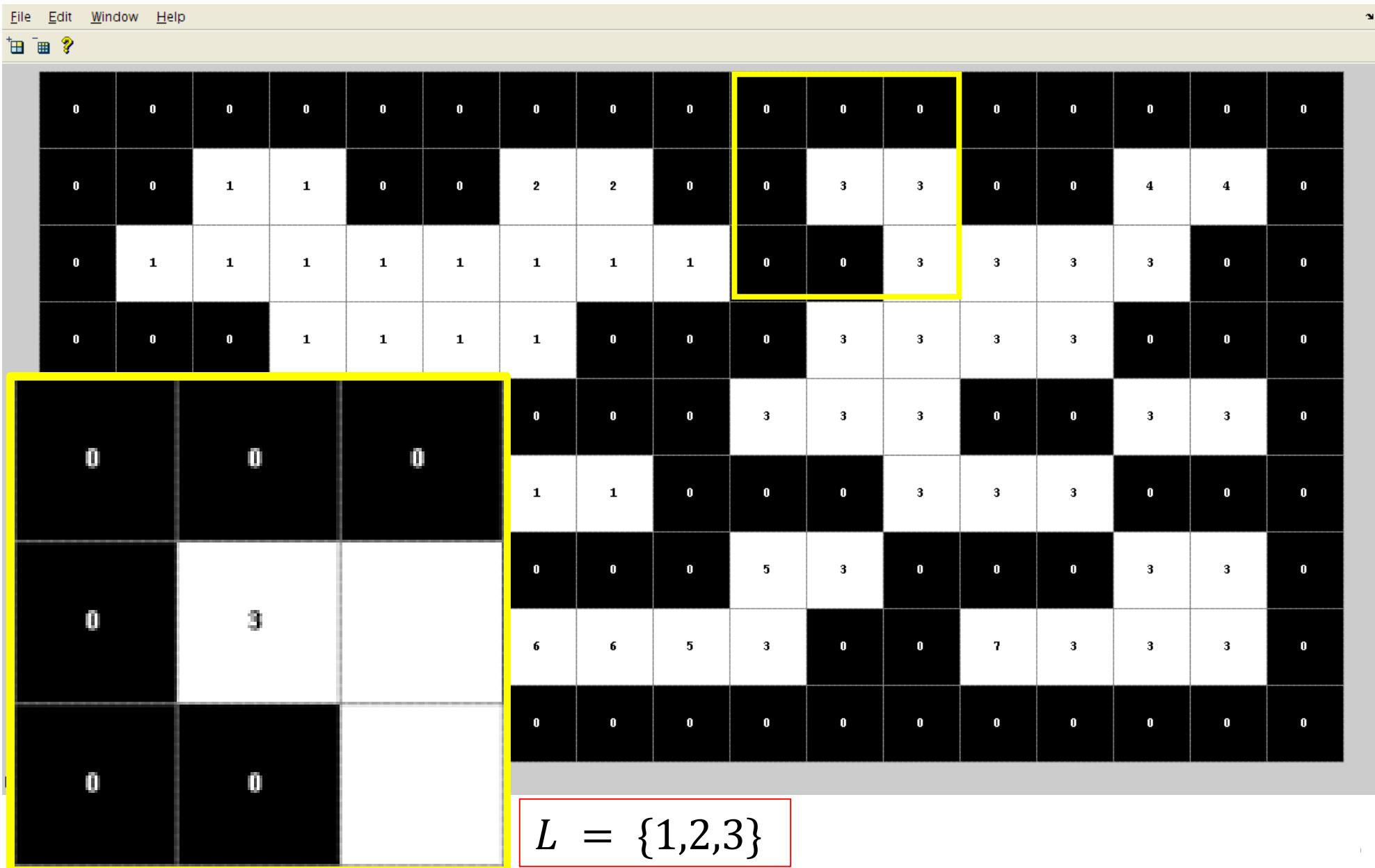
Iterations of the first pass



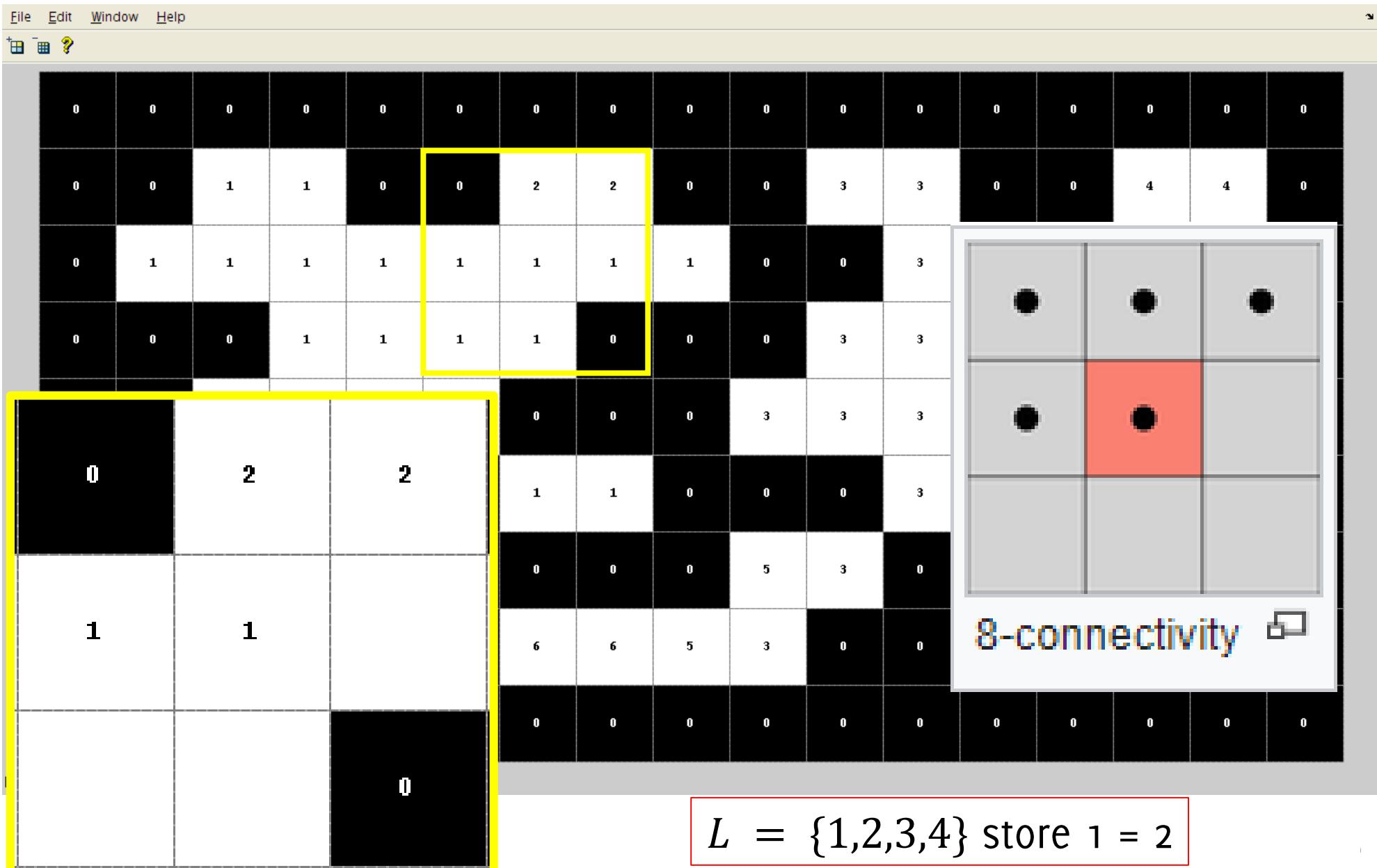
Iterations of the first pass



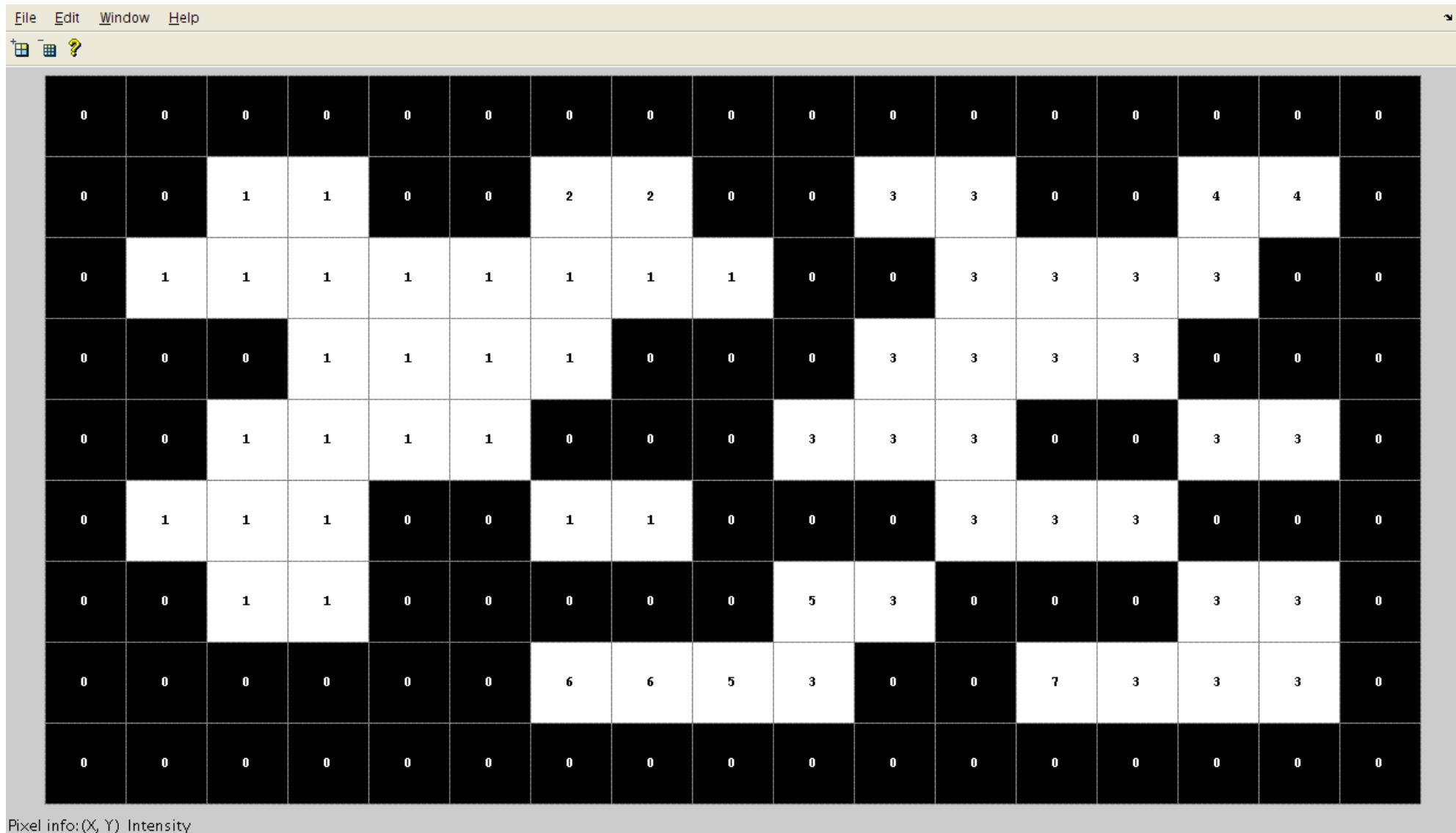
Iterations of the first pass



Iterations of the first pass



Output of the first pass



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

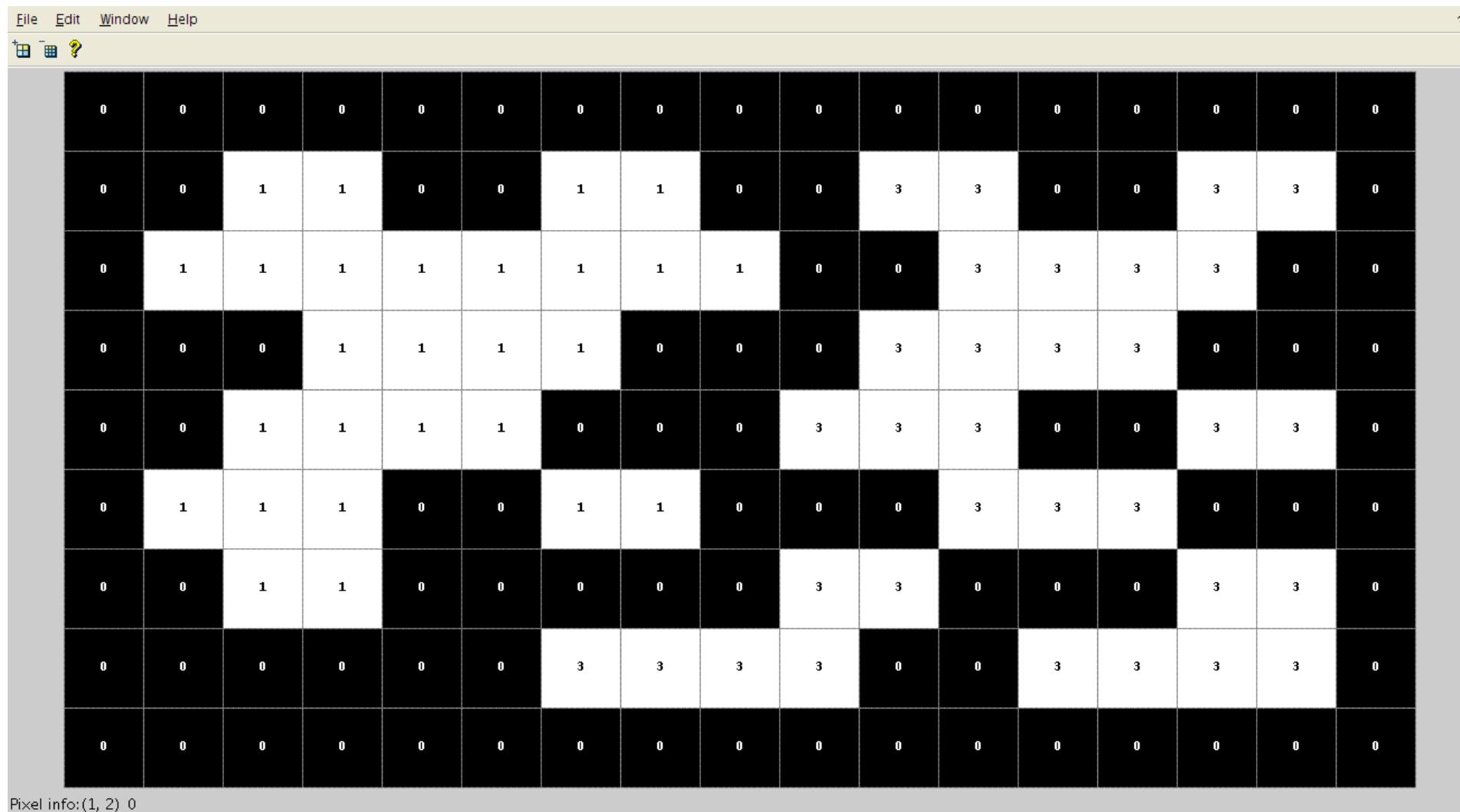
Two Pass Algorithm: Second Pass

Iterate through each pixel (r, c)

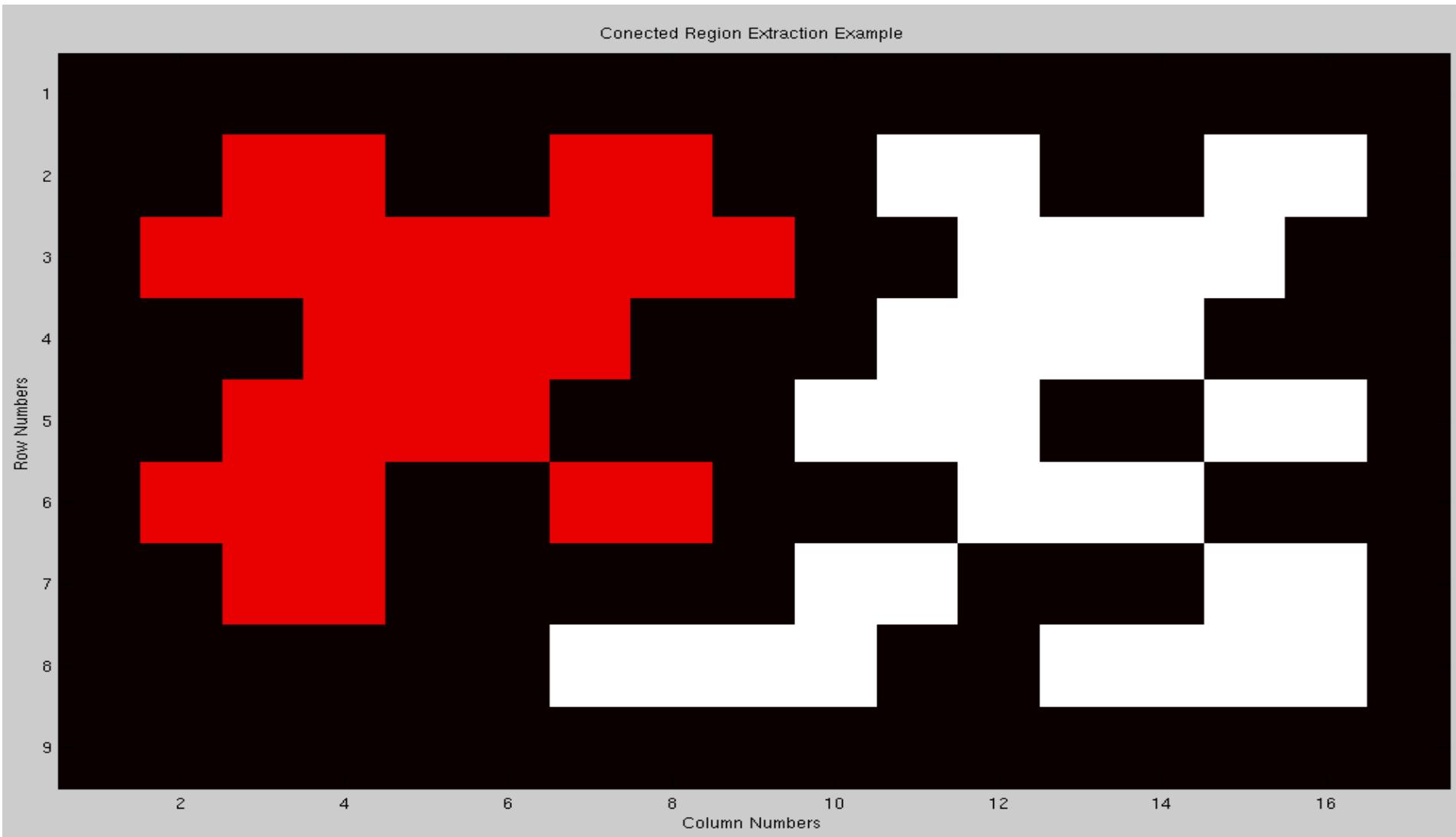
If $I(r, c) == 1$

Relabel the element with the lowest equivalent label

Output of the Second Pass



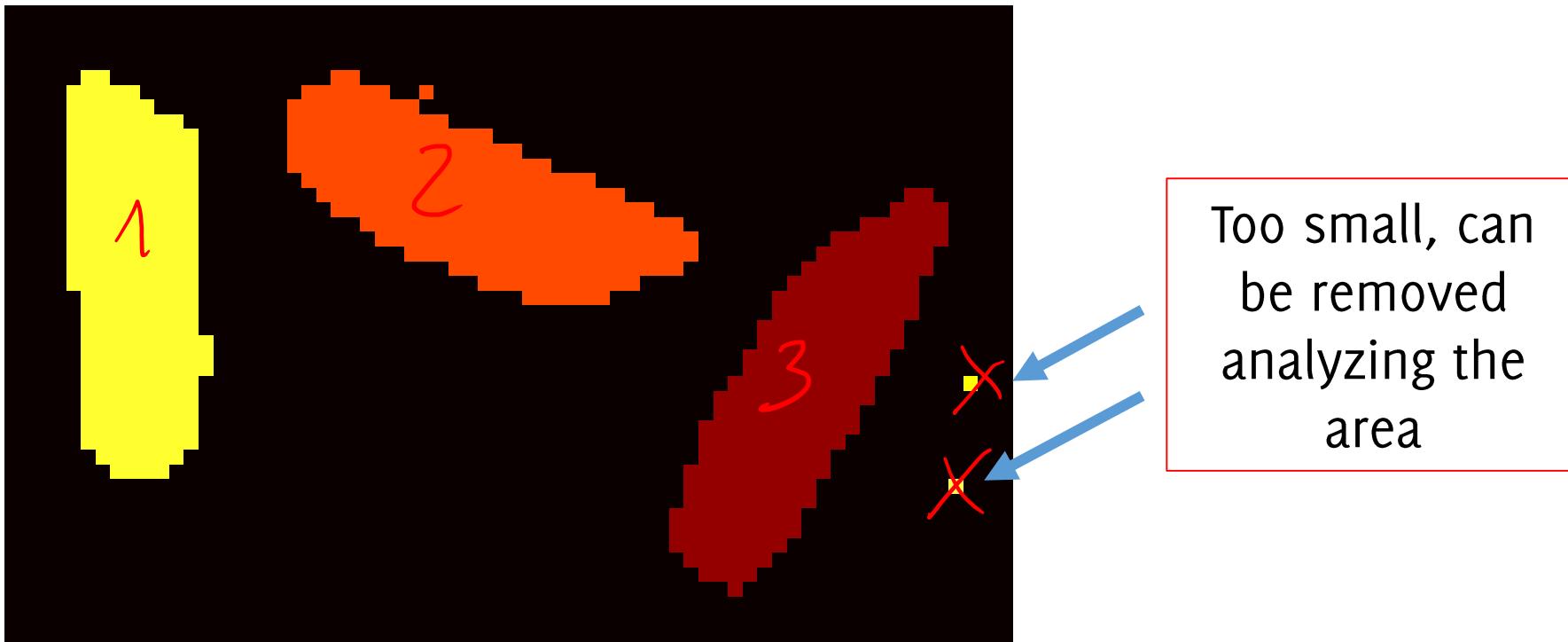
Output of the Second Pass



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

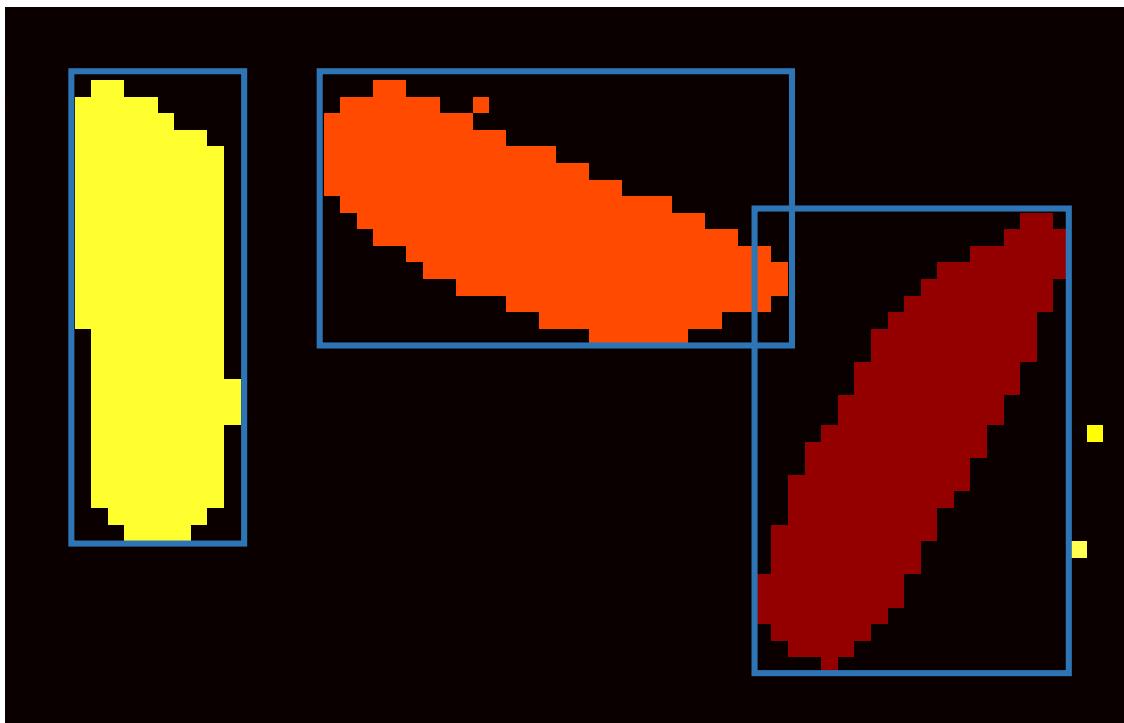
Bounding Box vs Axis

These provide information about size and orientation of the object



Bounding Box vs Axis

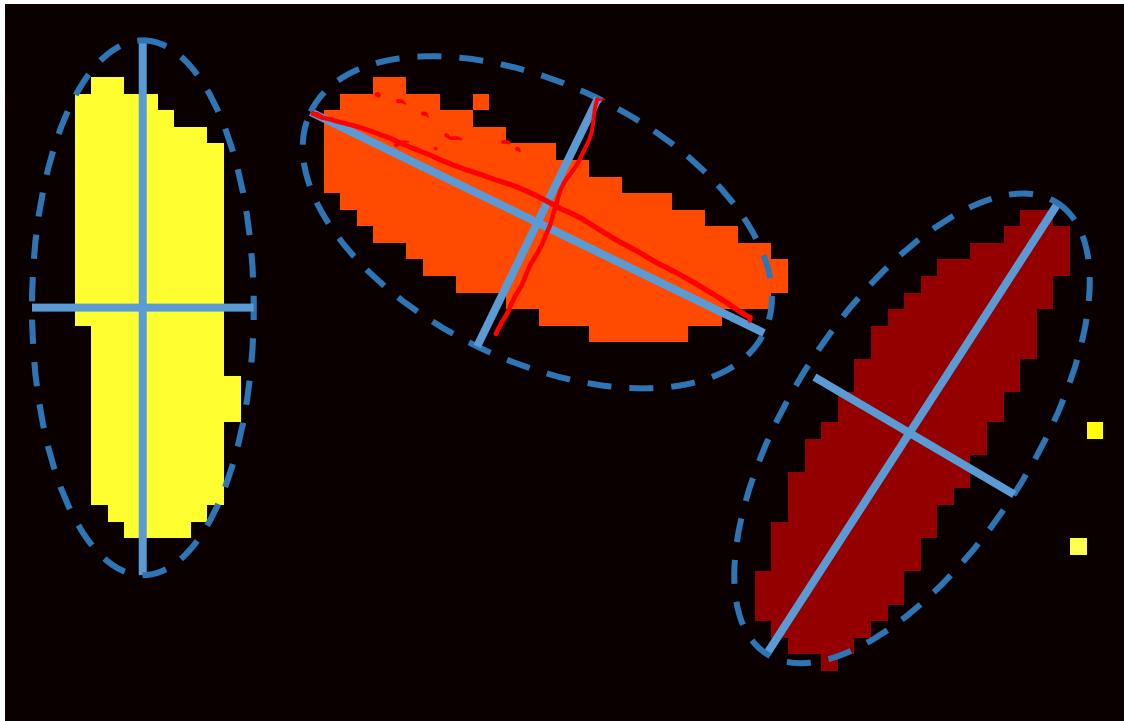
These provide information about size and orientation of the object



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

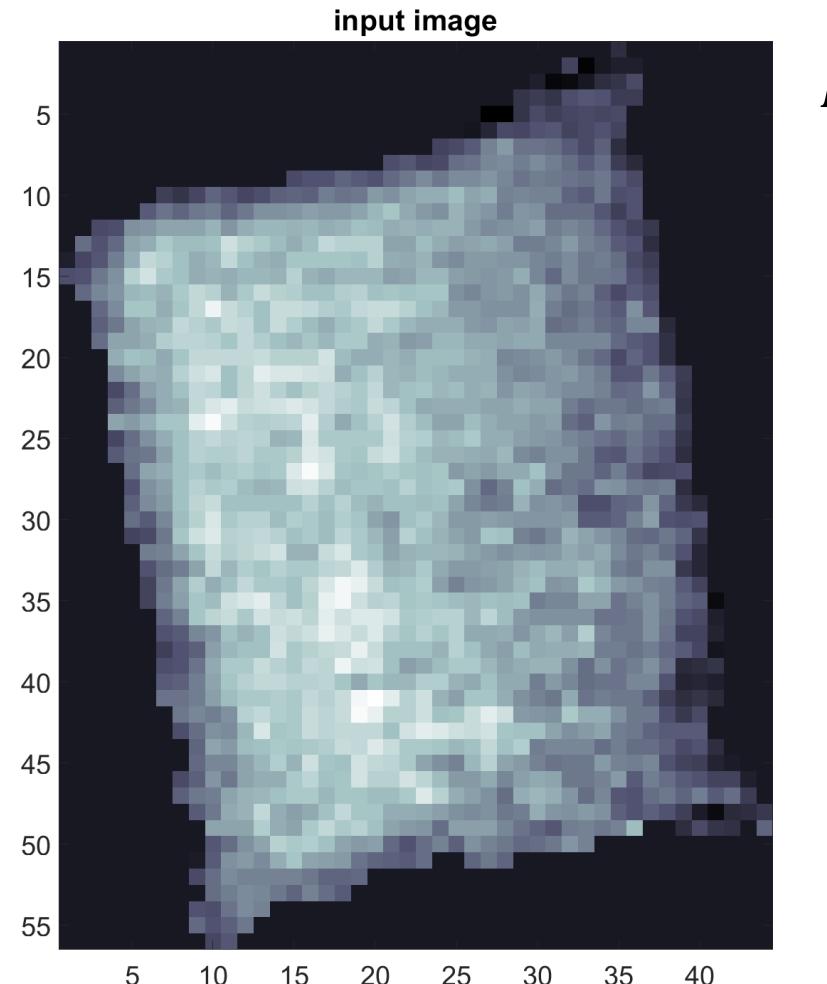
- Remove blobs that are too small
- Enhance boundaries by morphological operation

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...

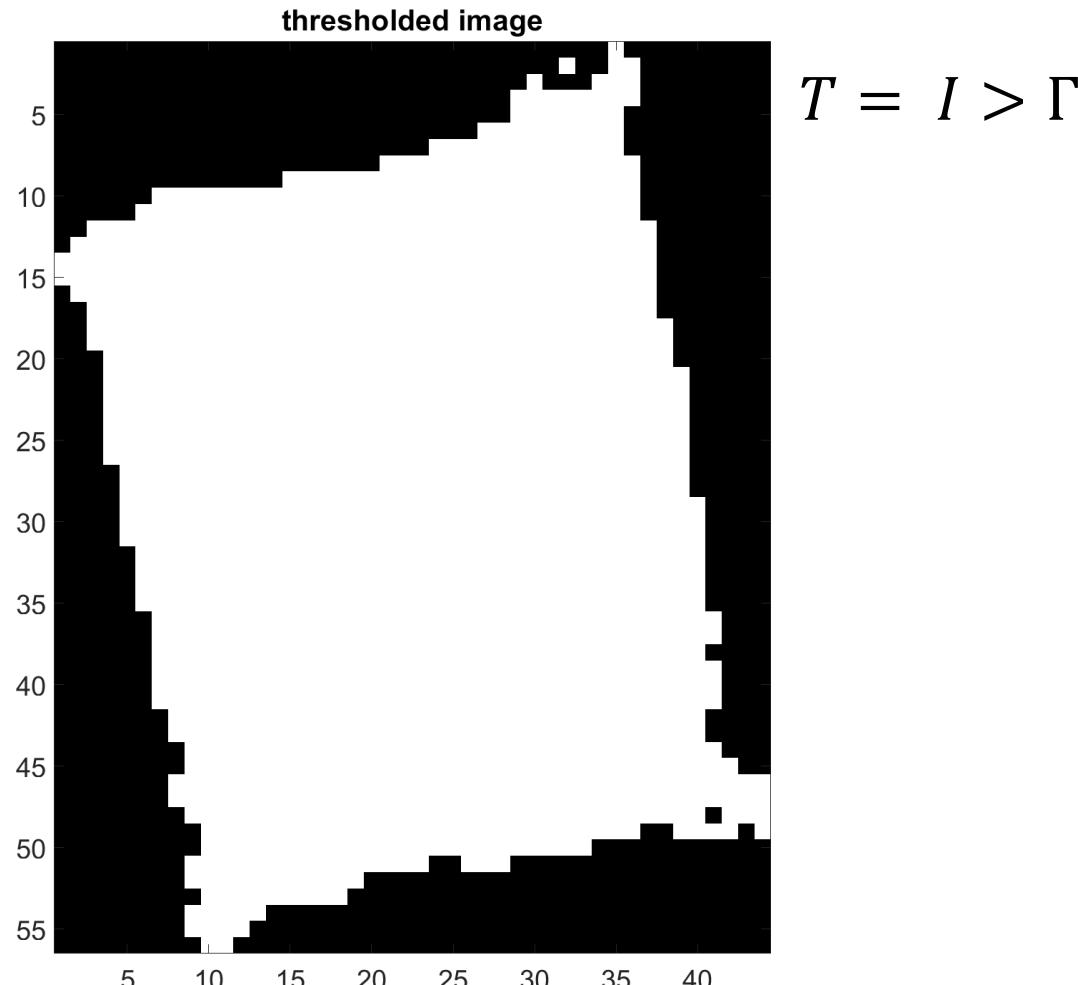
Boundary Extraction

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



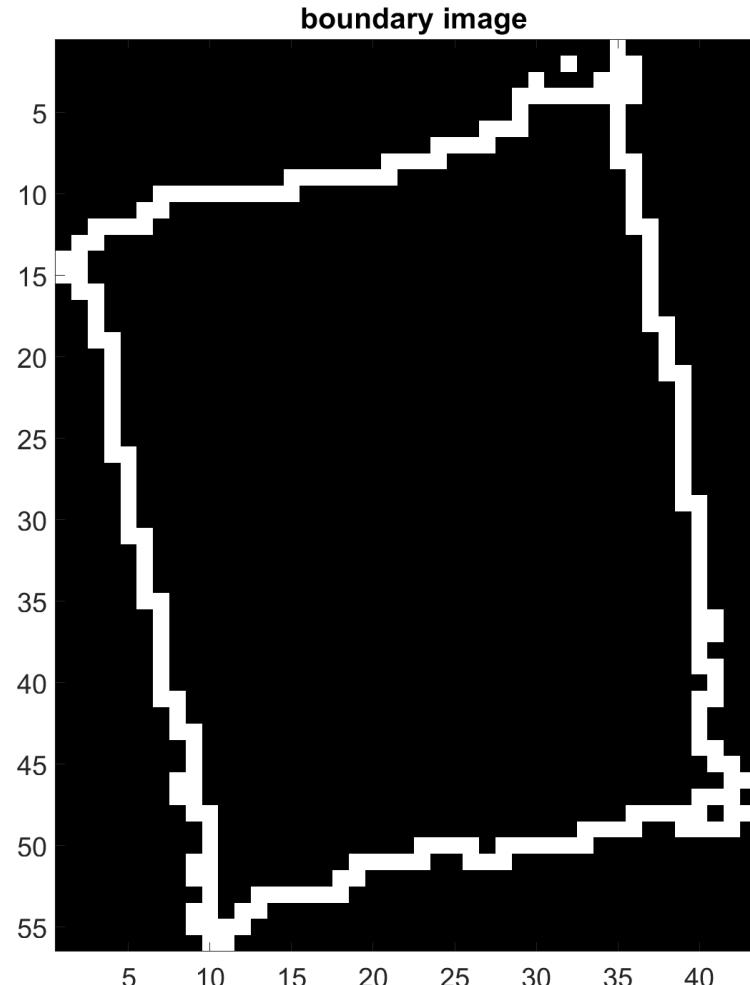
Boundary Extraction

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



Boundary Extraction

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



$$B = T \ominus h$$

$$\#[\bar{B} - \text{erode}(B)]$$

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

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)