Non Linear Filters and Image Derivatives

Giacomo Boracchi

giacomo.boracchi@polimi.it

Politecnico di Milano

Image Analysis And Computer Vision

https://boracchi.faculty.polimi.it/

Nonlinear Filters

Nonlinear 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 or point t.

Examples of nonlinear filter are

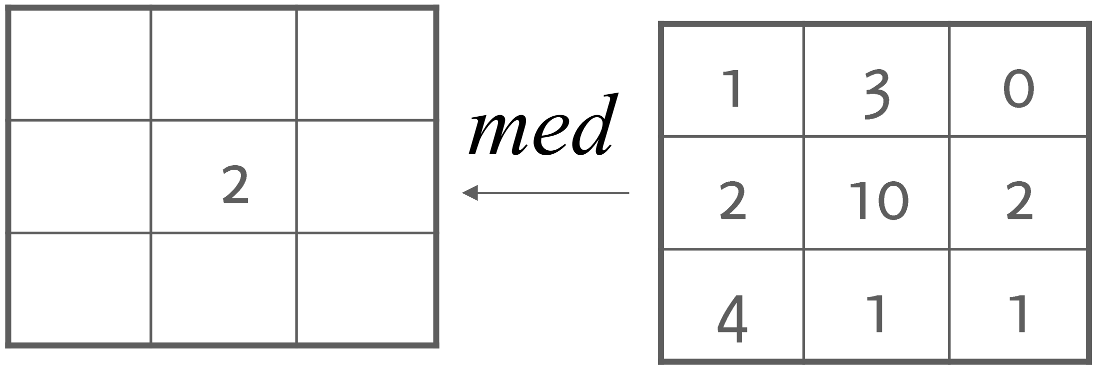
- Median Filter (Weighted Median)
- Ordered Statistics based Filters
- Threshold, Shrinkage

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

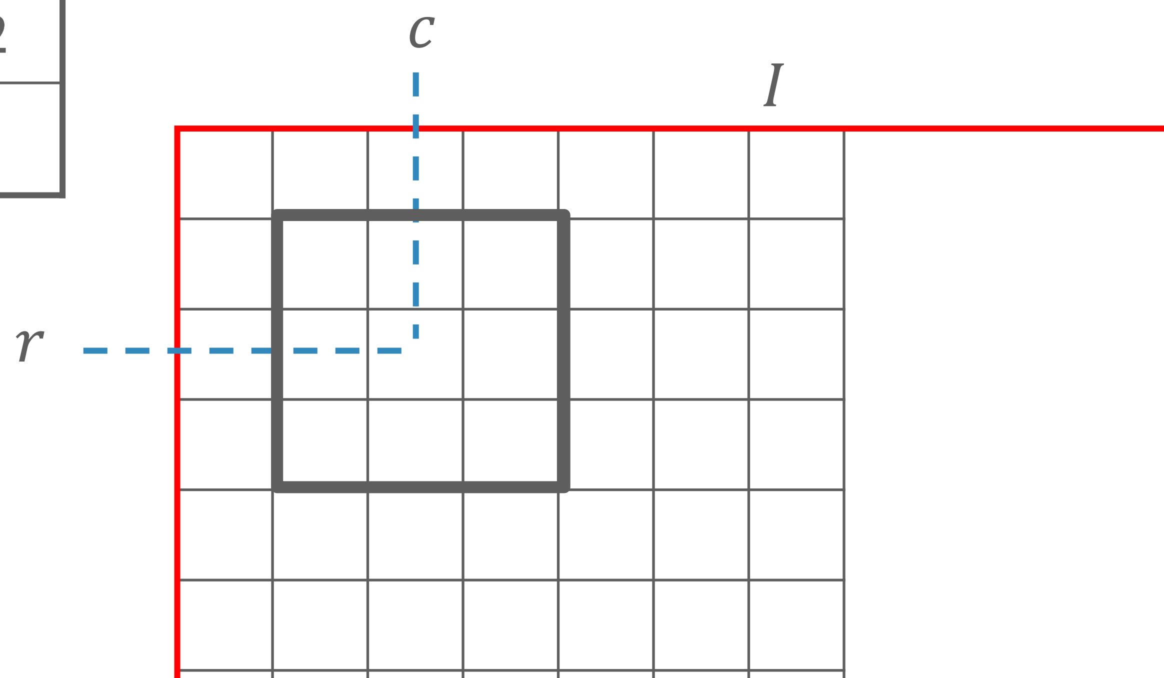
Blockwise Median

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

This is edge-preserving and robust to outliers!



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



Salt-and-pepper noise

Salt and Pepper (Impulsive) noise

Denoisng using local smoothing 3x3

Denoisng with median 3x3

Salt and Pepper (Impulsive) noise

Histogram matching and median filtering are useful! landslide monitoring

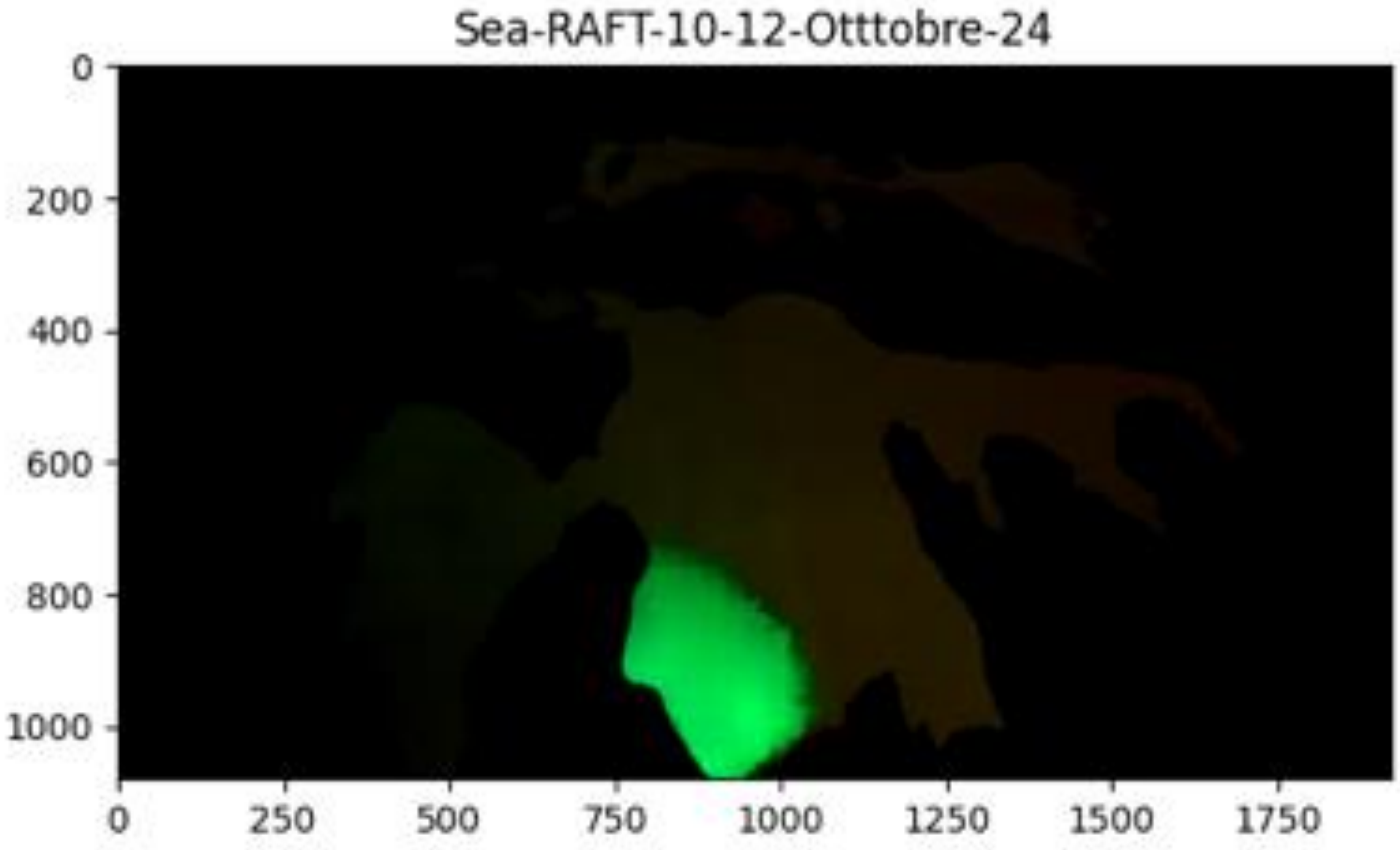
Input images

Input images

Median image + histogram equalization

Median images

Good to know, but optional



Pixels direction and magnitude

Morphological Operations

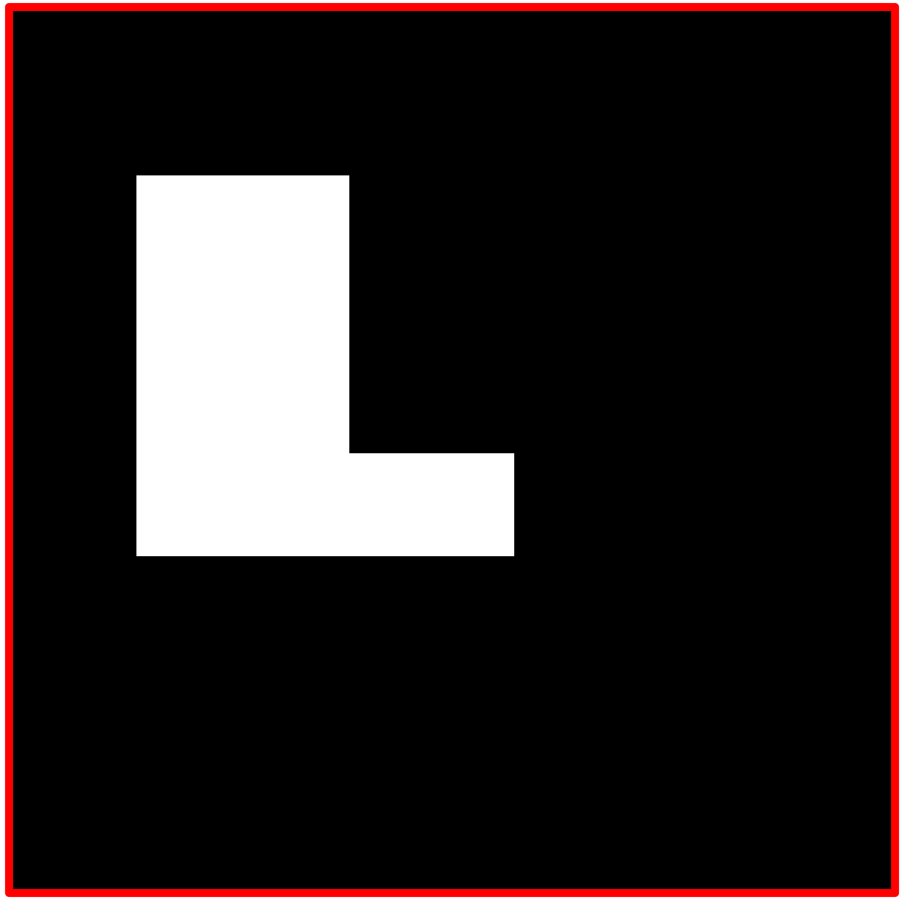
Ordered Statitiscs

Binary images

A binary image is defined as $I \in \{0,1\}^{R \times C}$

Each pixel can be either true (1) / false (0)

Typically binary images are the result of pre-processing operations including thresholding



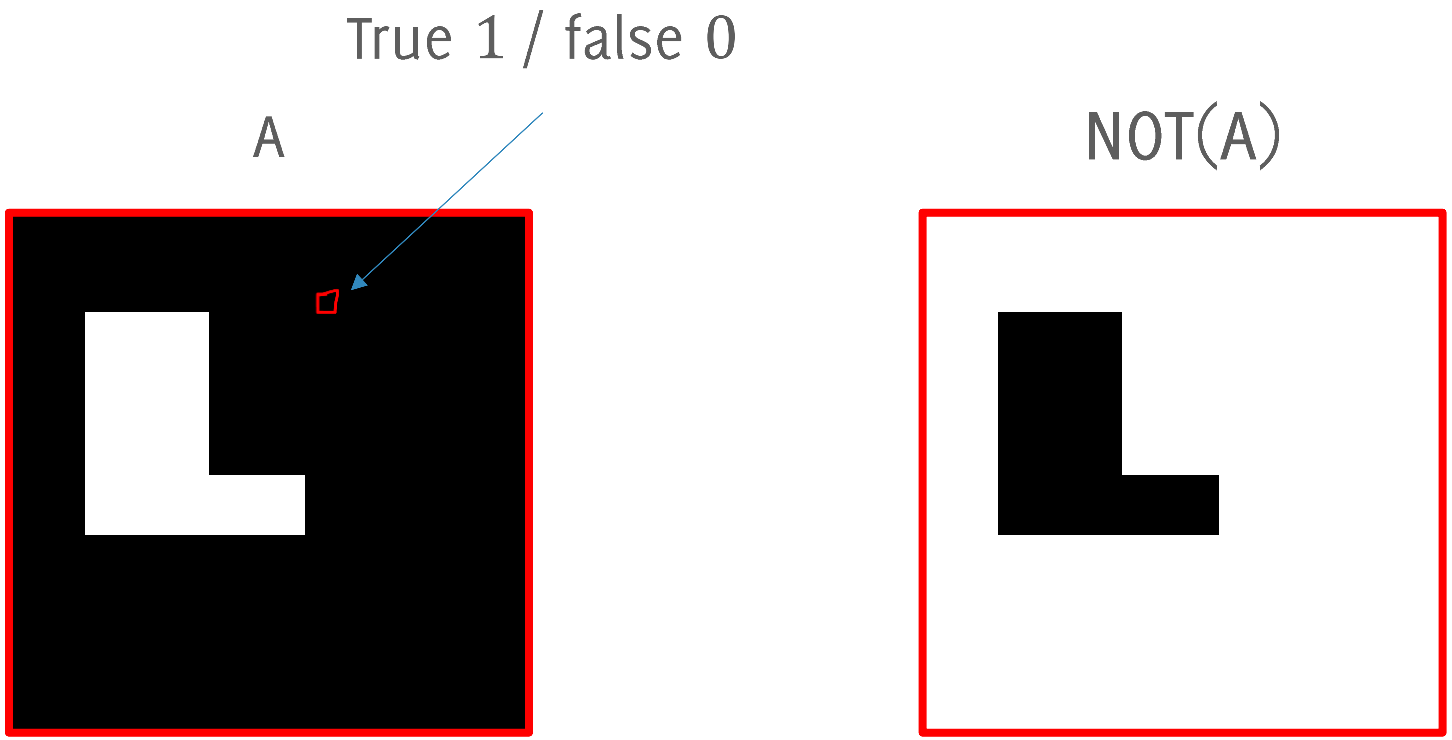
An overview on morphological operations

Erosion, Dilation

Open, Closure

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

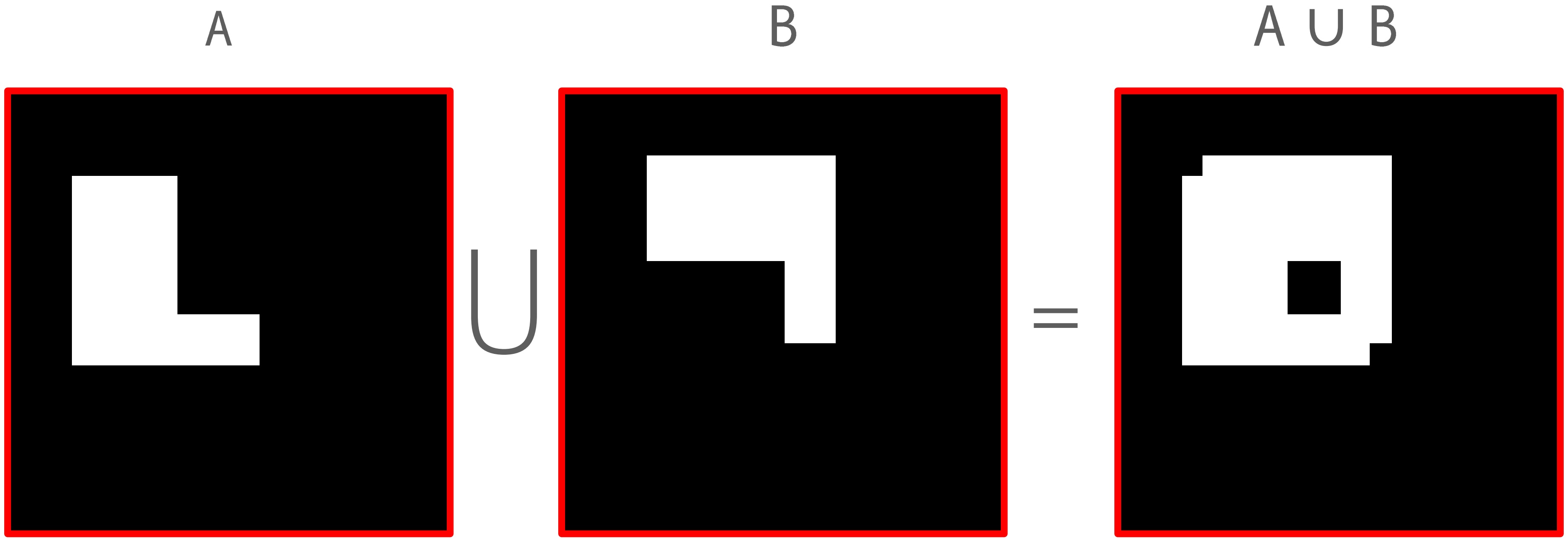
Boolean operations on binary images $I \in \{0,1\}^{R \times C}$



NOT_A = A == 0

Union of binary images

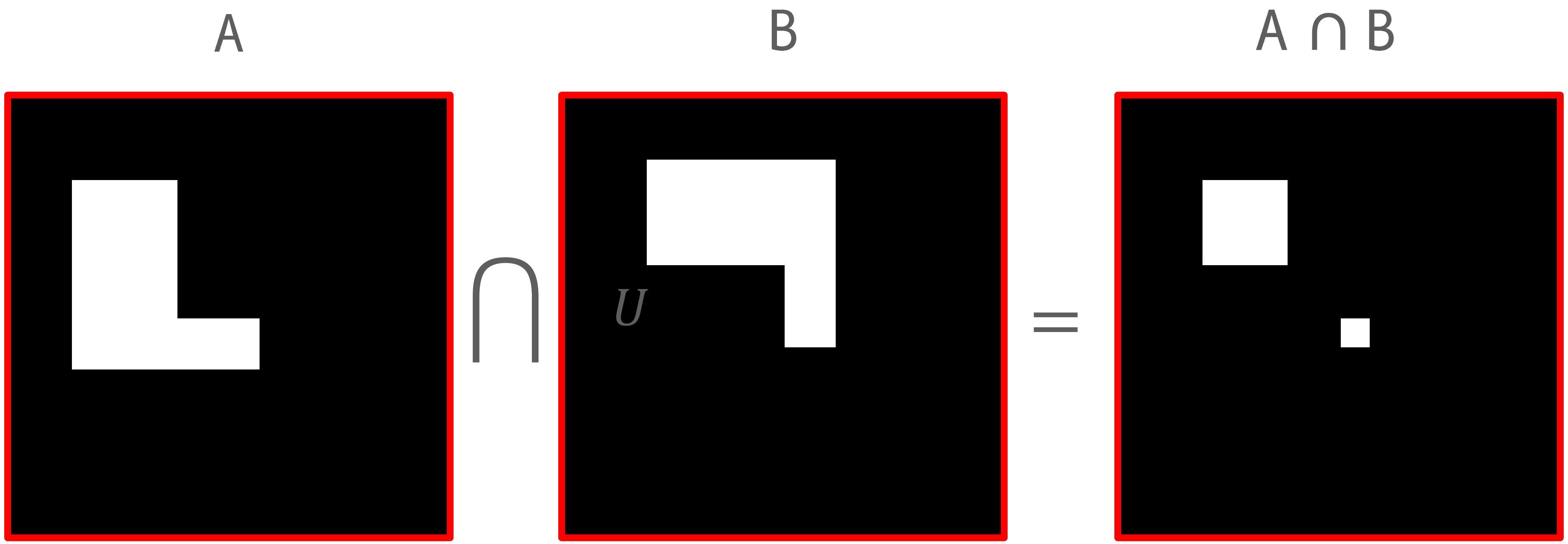
Equivalent to the OR operation



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

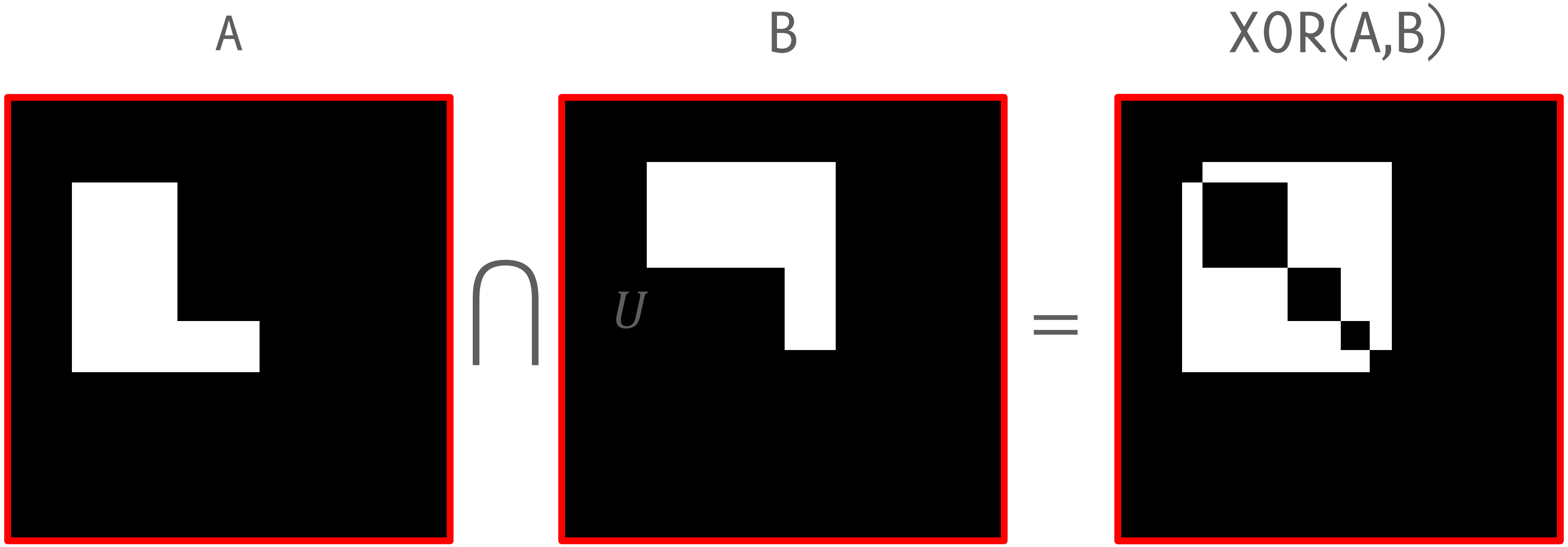
Intersection of Binary Images

Equivalent to the AND operation



$$A \cap B = A + B > 1$$

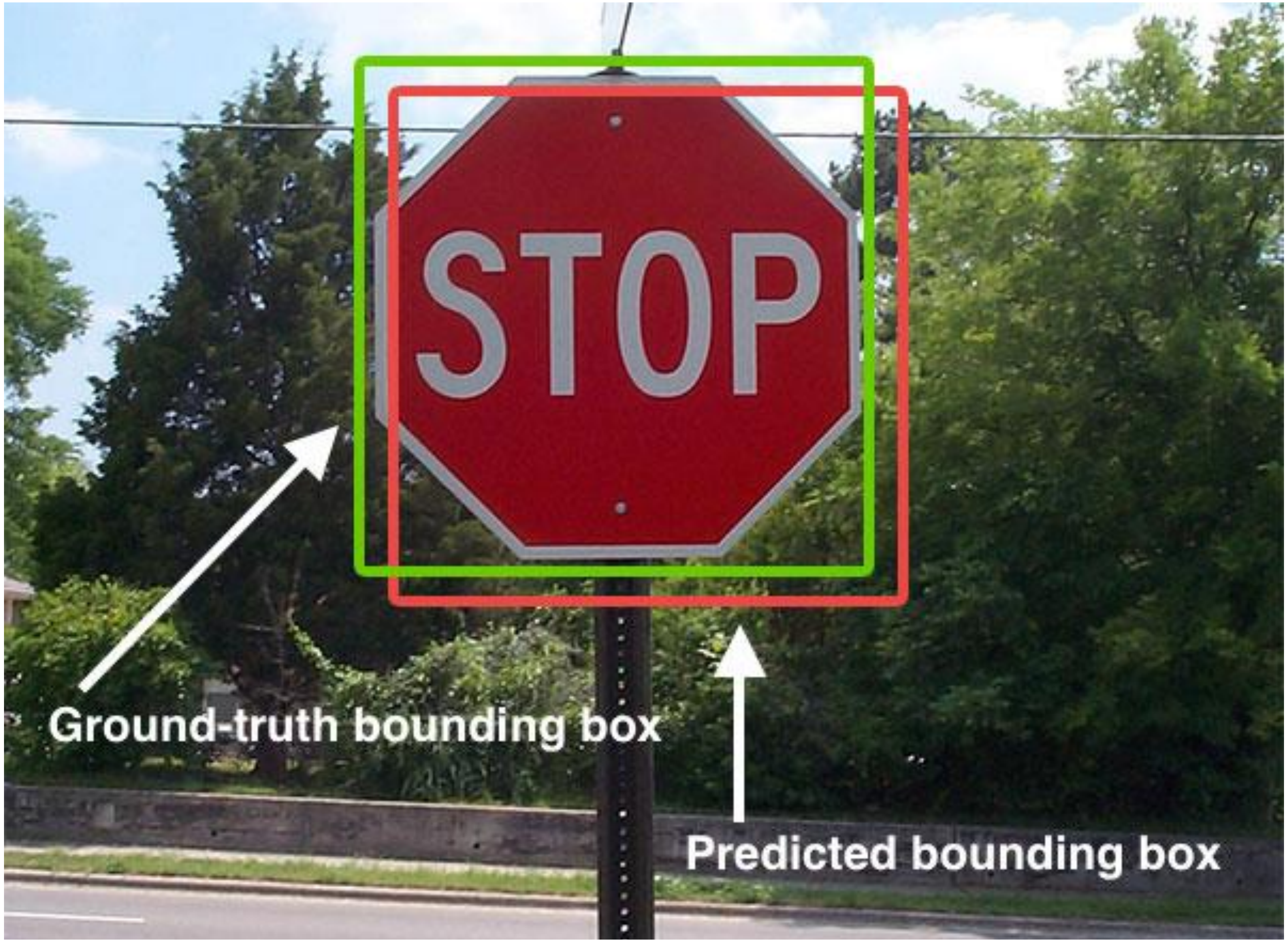
On binary images it is possible to define XOR



XOR(A,B) = $A \cup B \; - \; A \cap B$

What do we use this for?

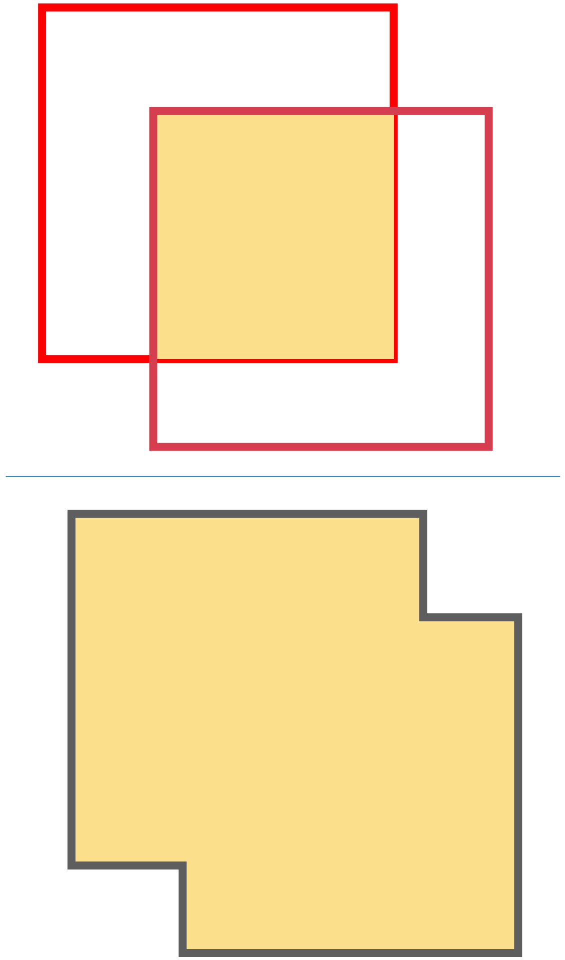
Intersection over the Union (IoU, Jaccard Index)



By Adrian Rosebrock http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=57718561

Intersection over the Union (IoU, Jaccard Index)

IOU = Area of intersection / Area of overlap



Intersection over the Union (IoU, Jaccard Index)

By Adrian Rosebrock http://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=57718559

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.

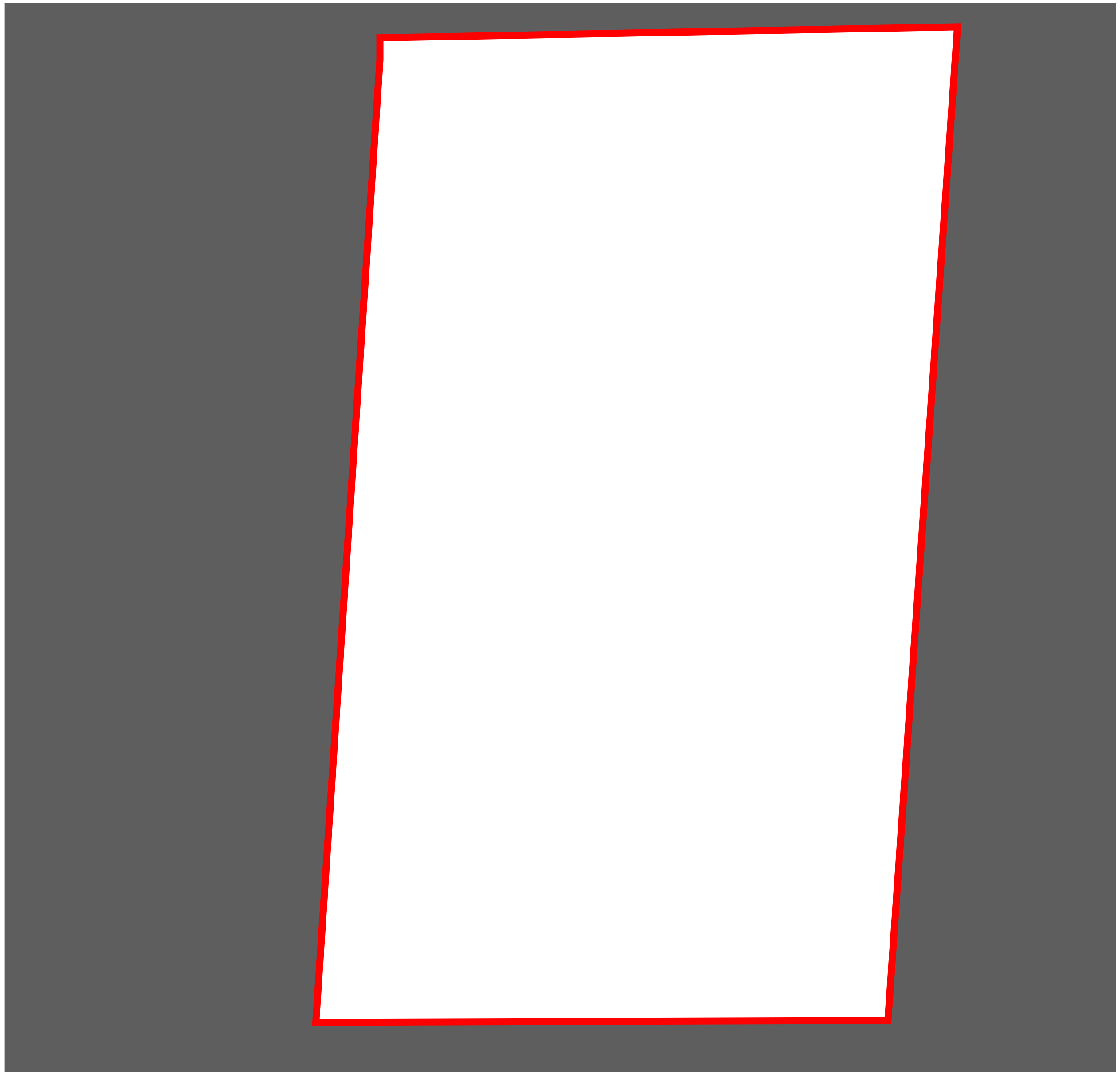
It is a standard reference measure for detection performance

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)

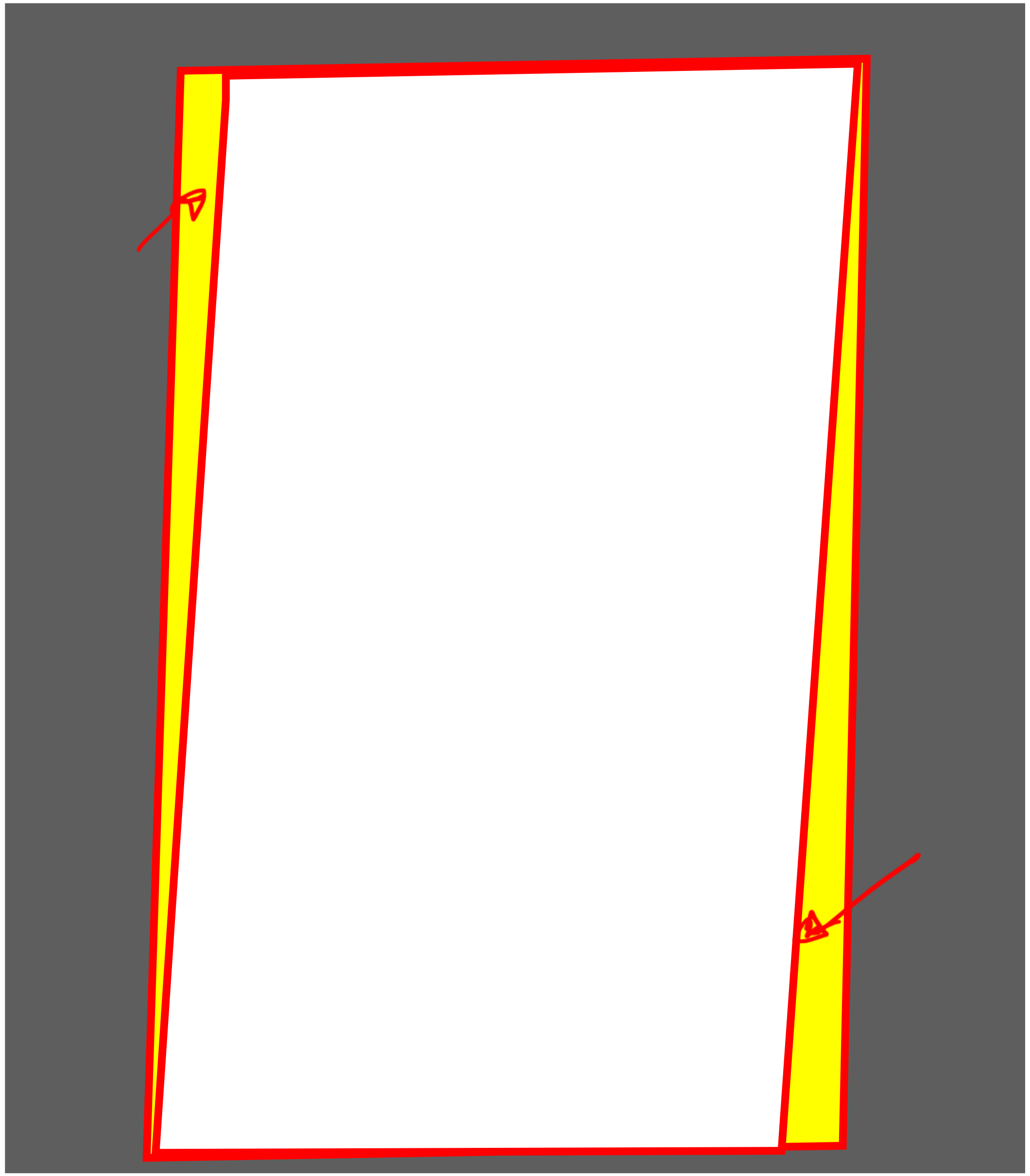
Jaccard Index (IoU)



Jaccard Index (IoU)

Jaccard Index (IoU)

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



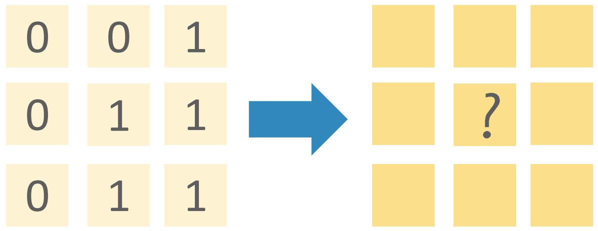
Filters on binary images

It is possible to define filtering operations between binary images

Consider also binary filters, i.e. spatial filters having binary weights.

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

Erosion



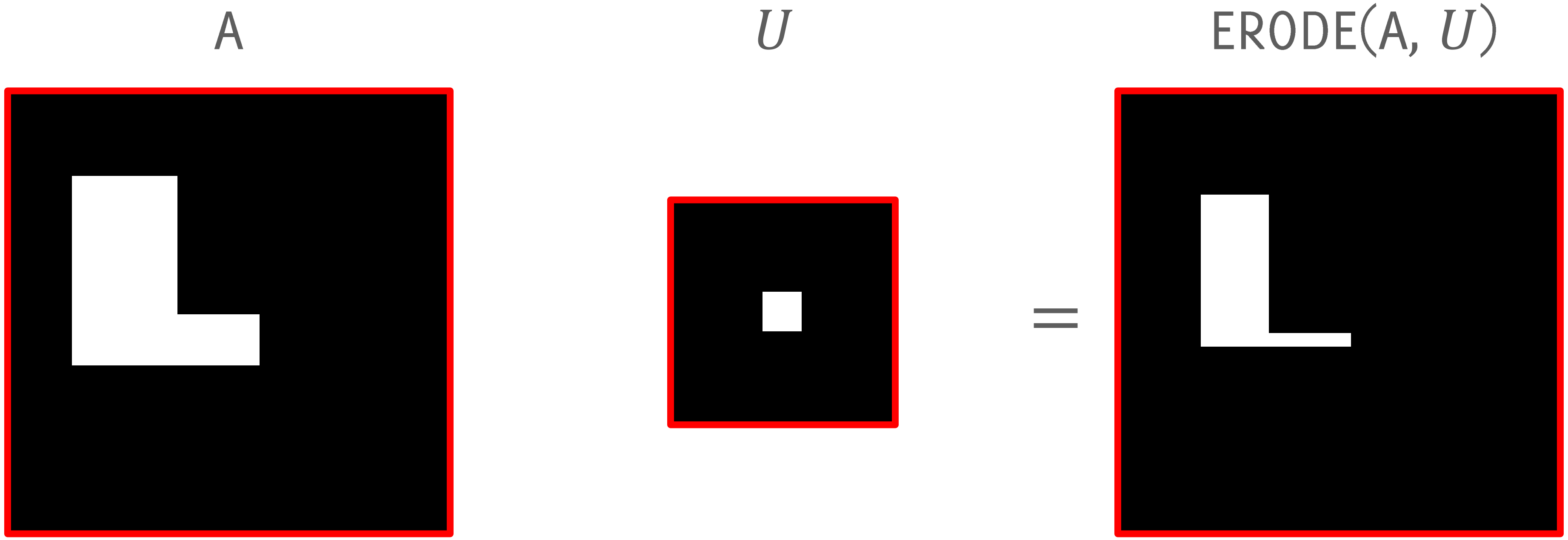
General definition:
Nonlinear Filtering procedure that replaces each pixel value, ***with the minimum on a given neighbor***

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

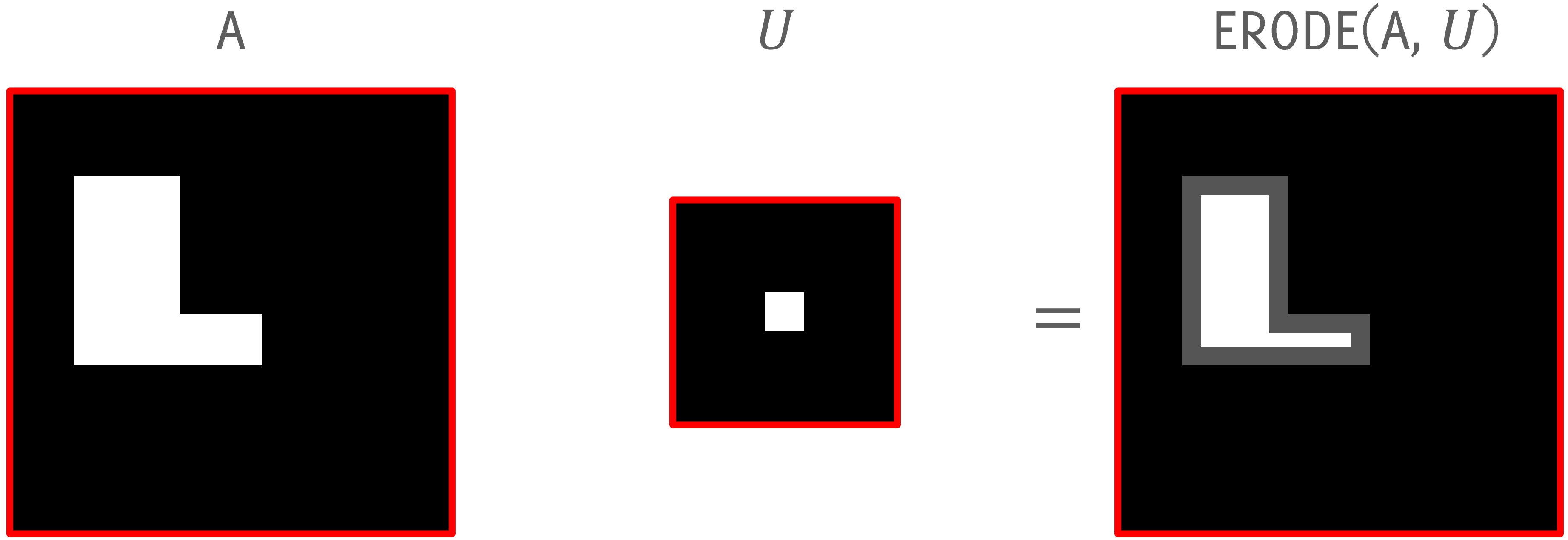
This operation reduces thus the boundaries of binary images

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

Erosion



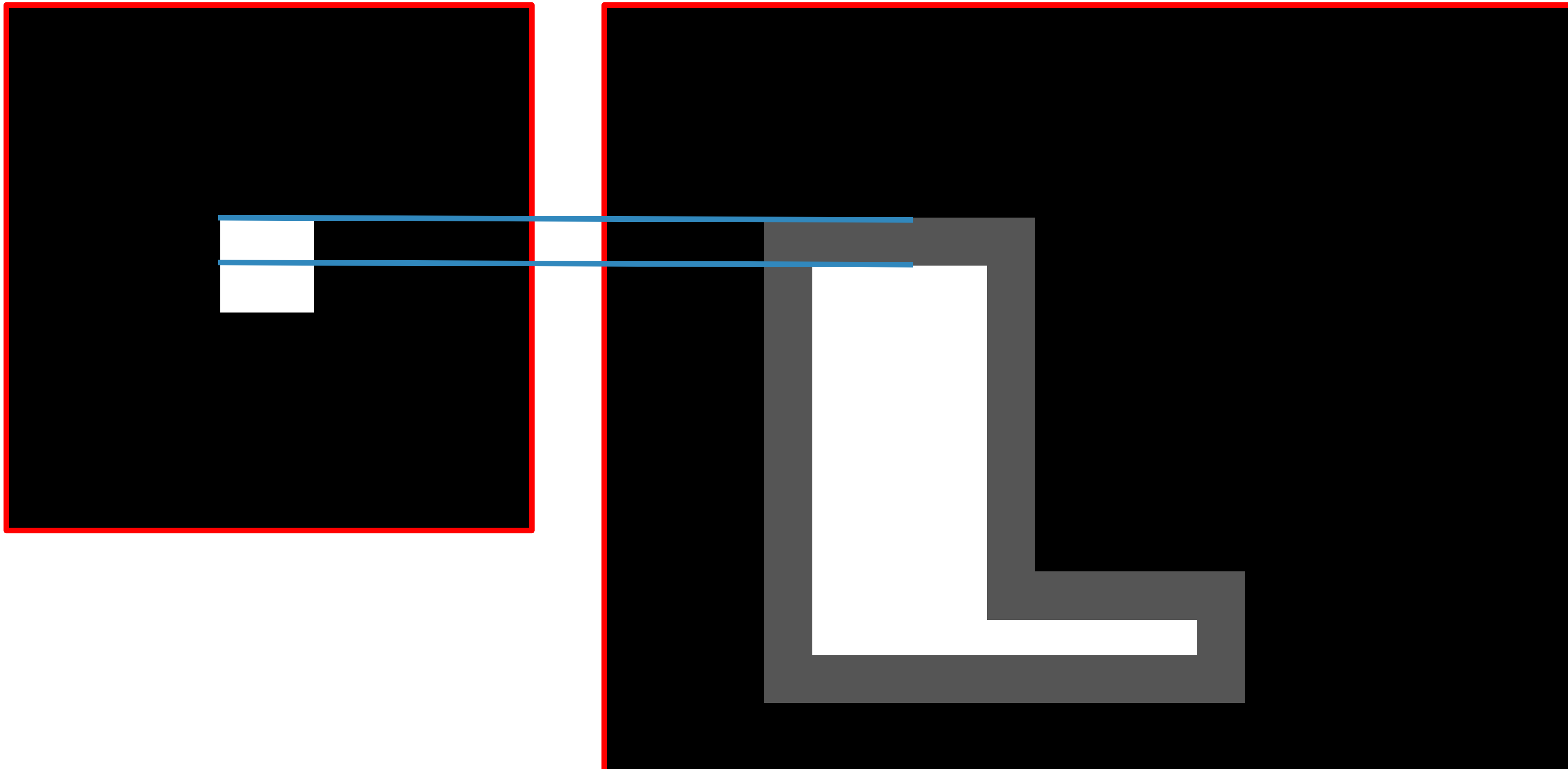
Erosion



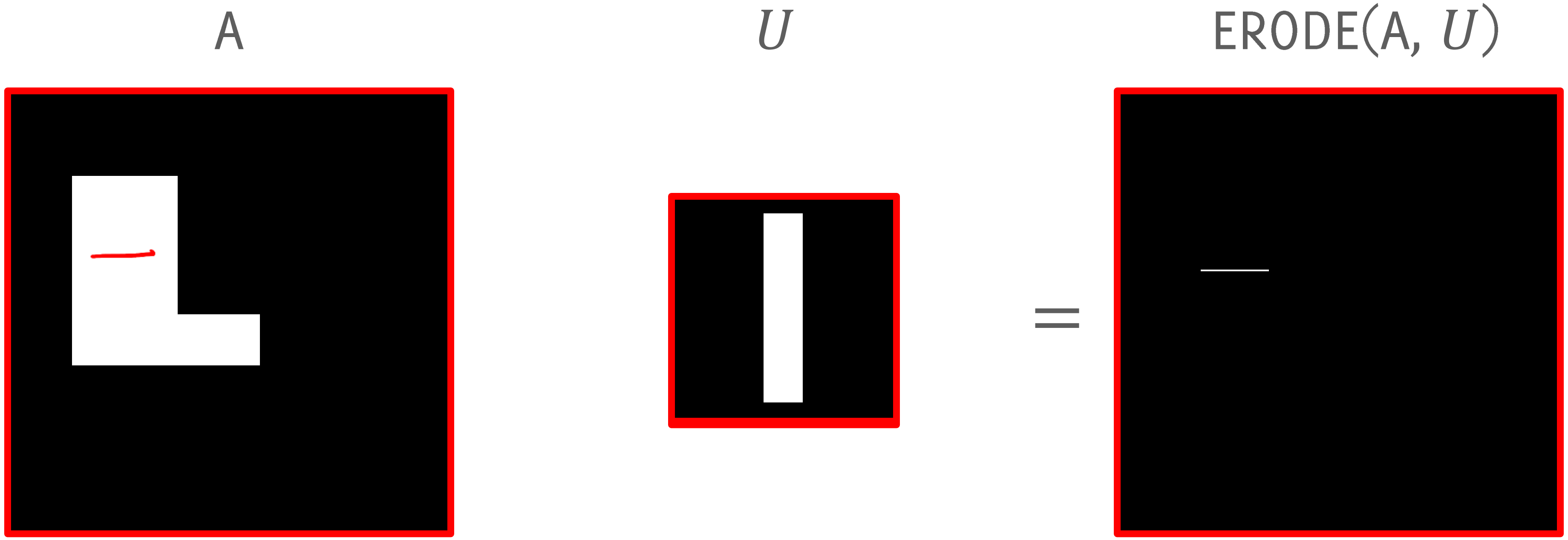
The gray area corresponds
to the input

Erosion

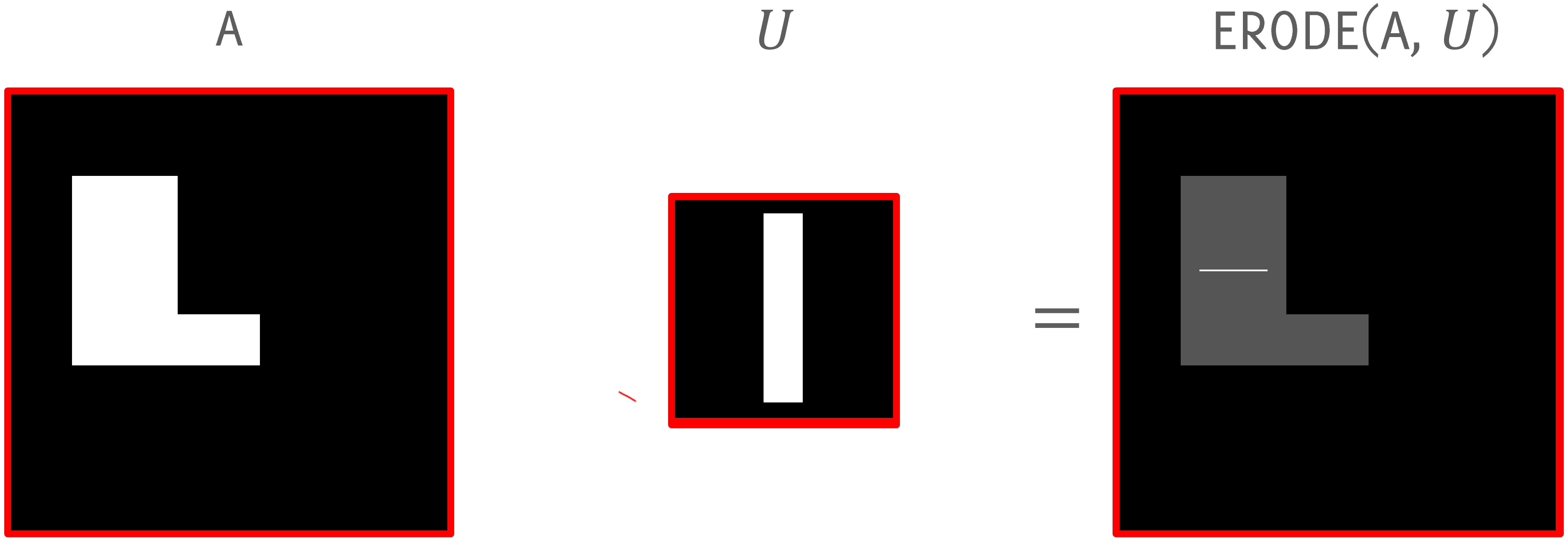
Erosion removes half size of the structuring element used as filter



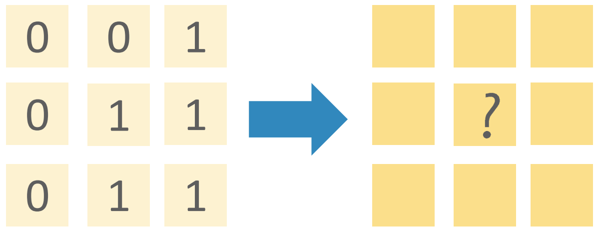
Erosion



Erosion



Dilation



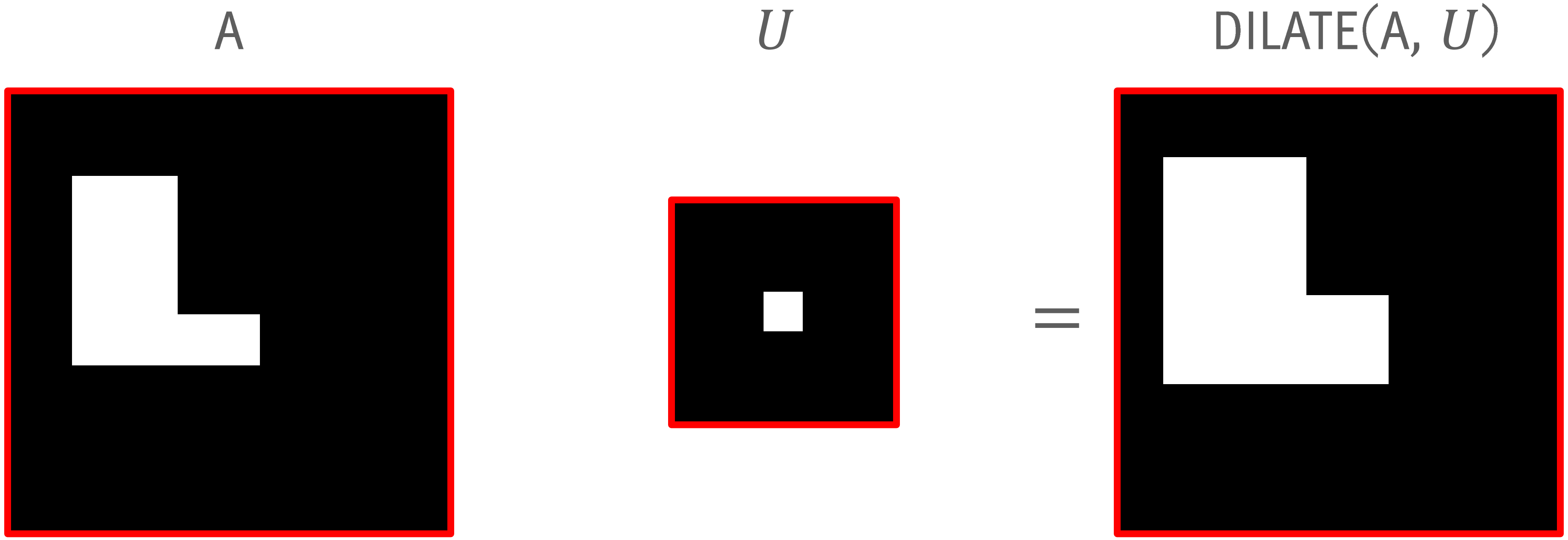
General definition:
Nonlinear Filtering procedure that replaces to each pixel value, ***with the maximum on a given neighbor***

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

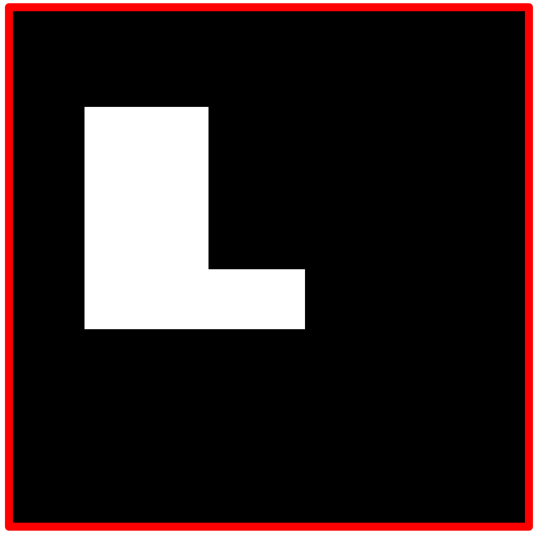
It can be interpreted as an OR operation of the image and the neighbour overlapped at each pixel

Dilation

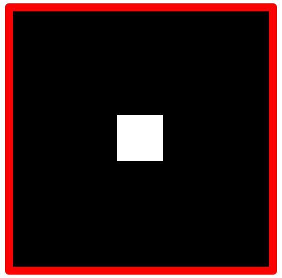


Dilation

A

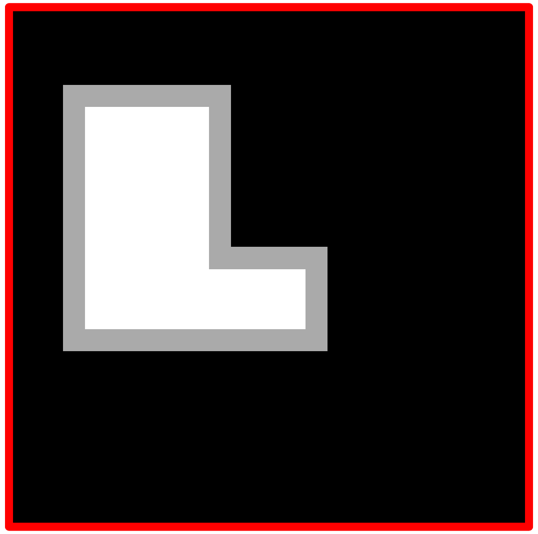


U



=

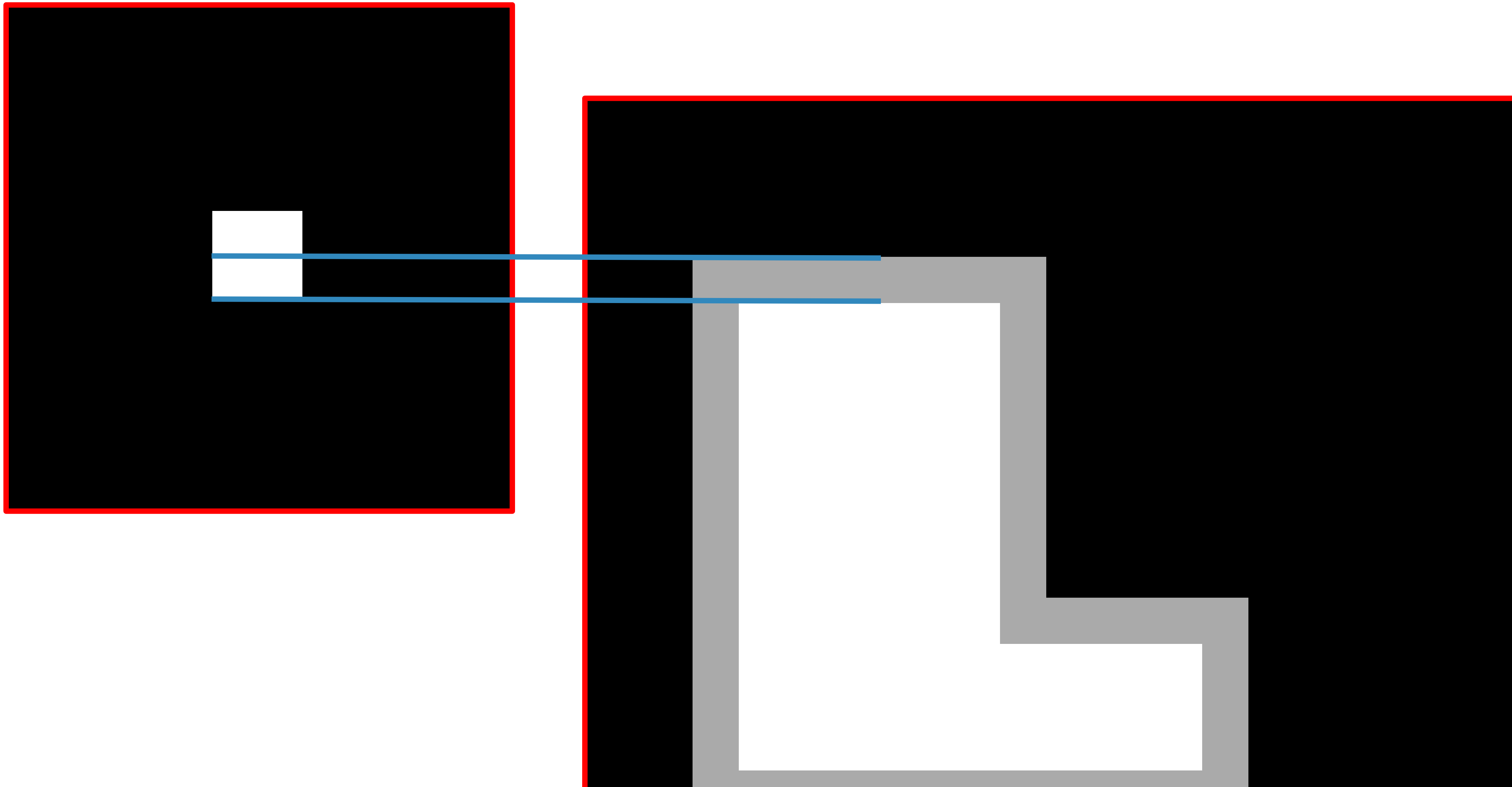
DILATE(A, U)



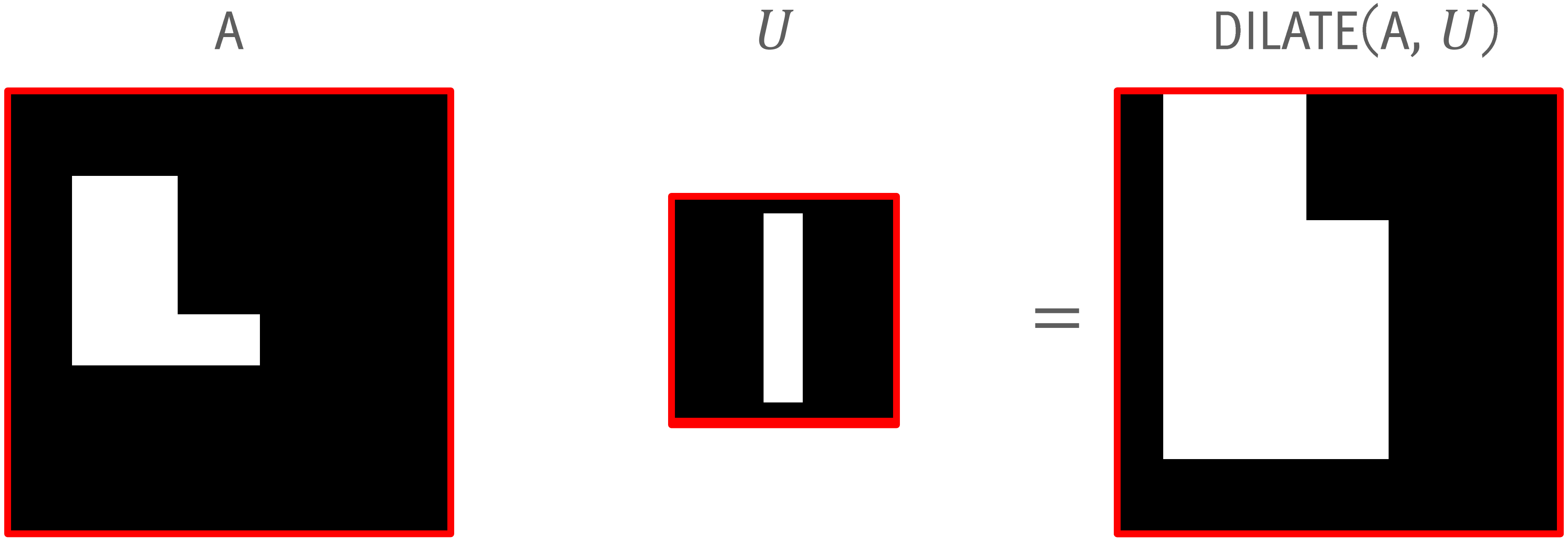
The brighter area now corresponds to the input

Dilation

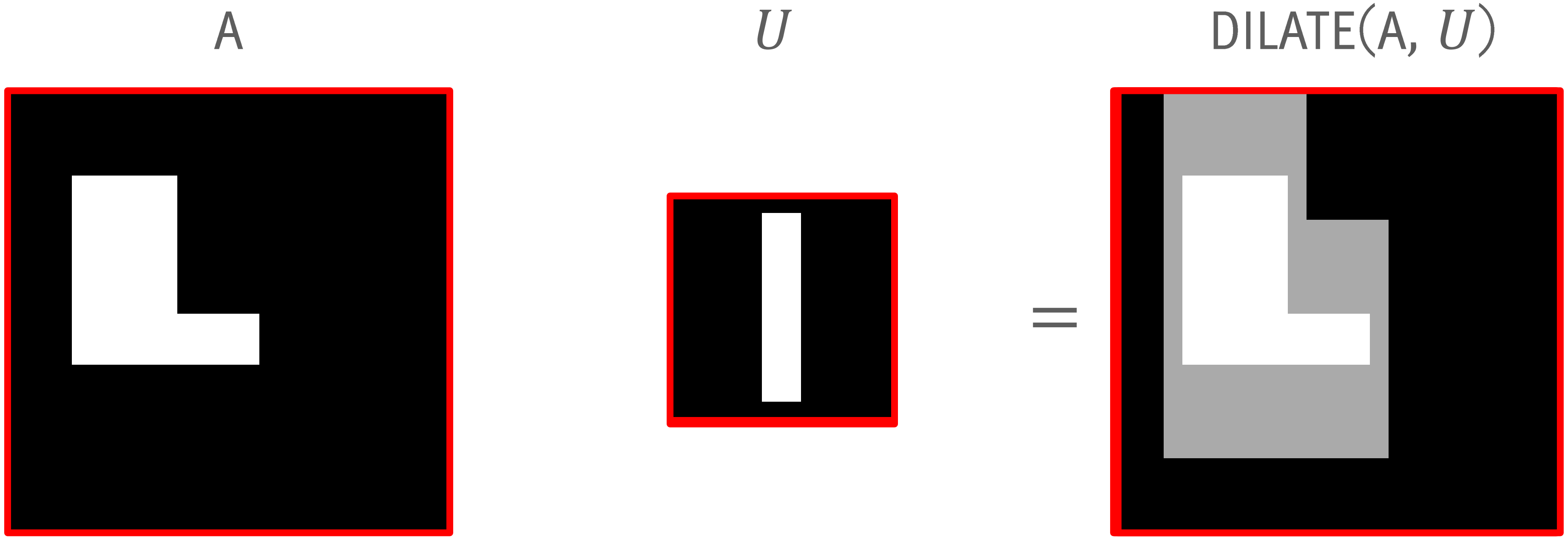
Dilation expands half size of the structuring element used as filter



Dilation



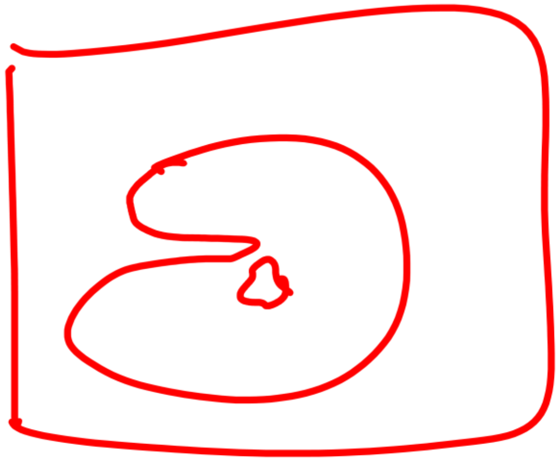
Dilation



Open and Closure

Open Erosion followed by a Dilation

Closure Dilation followed by an Erosion

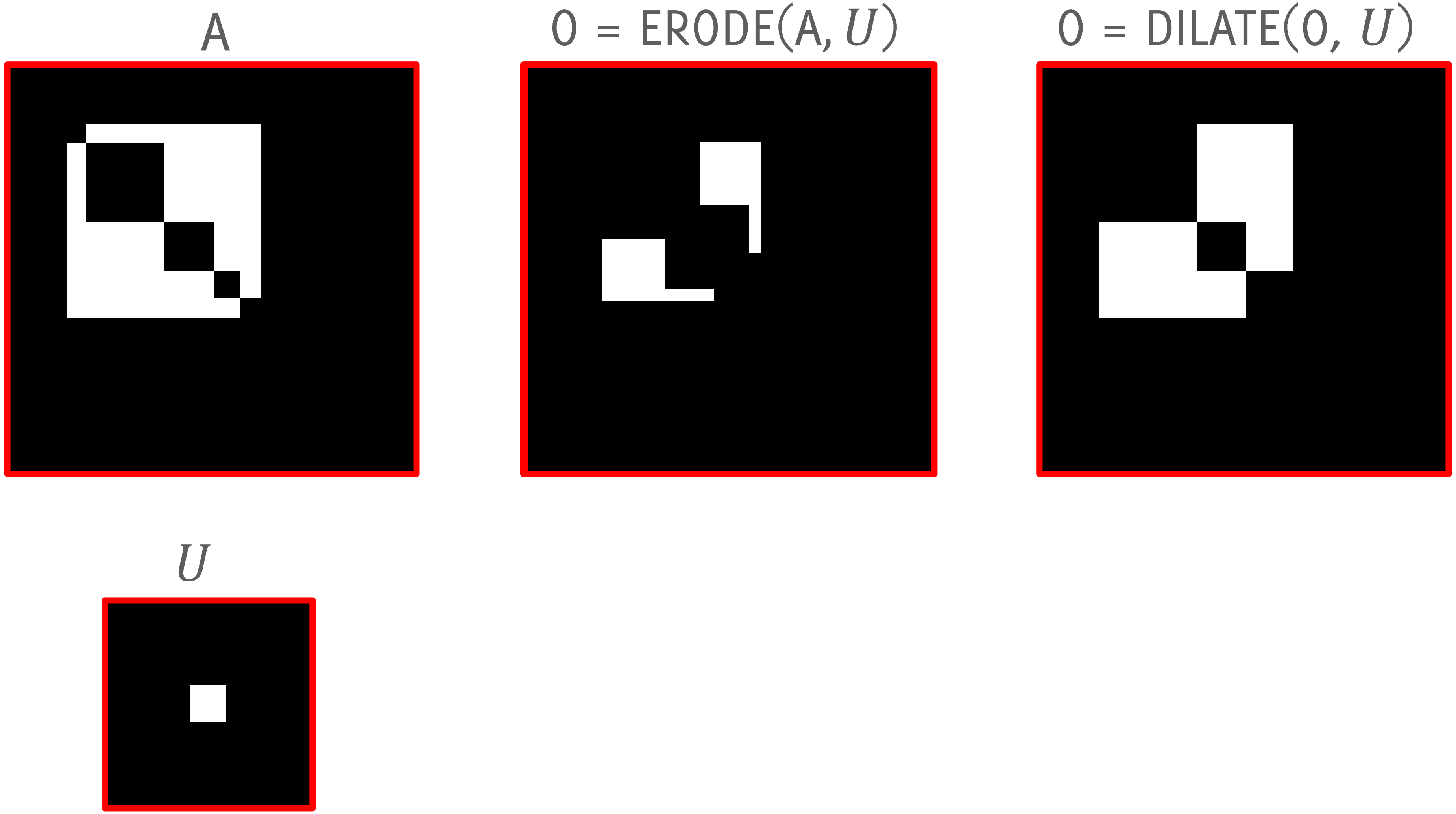


Open

Open Erosion followed by a Dilation

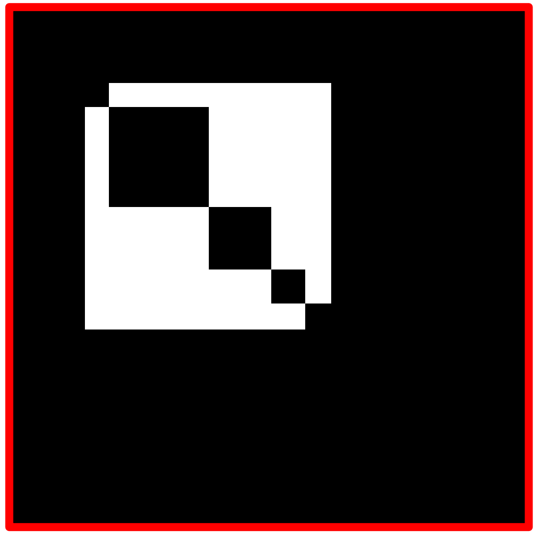
- Smooths the contours of an object
- Typically eliminates thin protrusions

Open

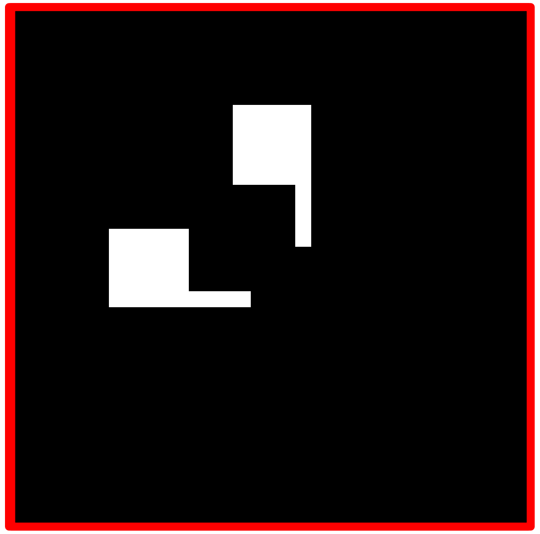


Open

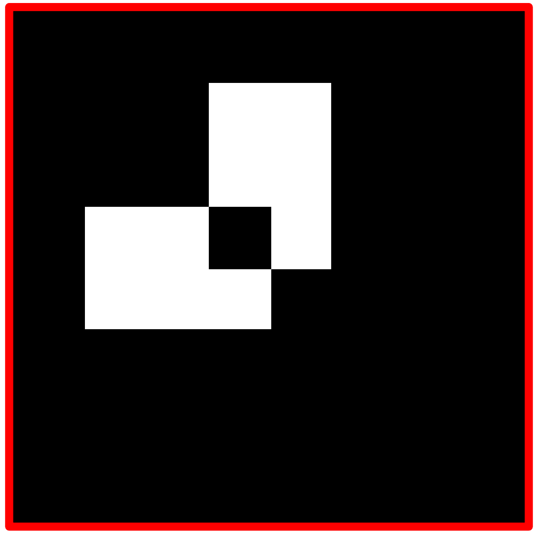
A



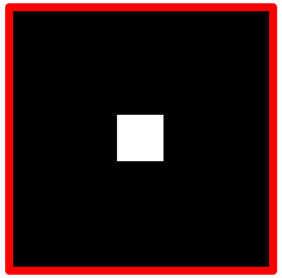
O = ERODE(A, U)



O = DILATE(O, U)

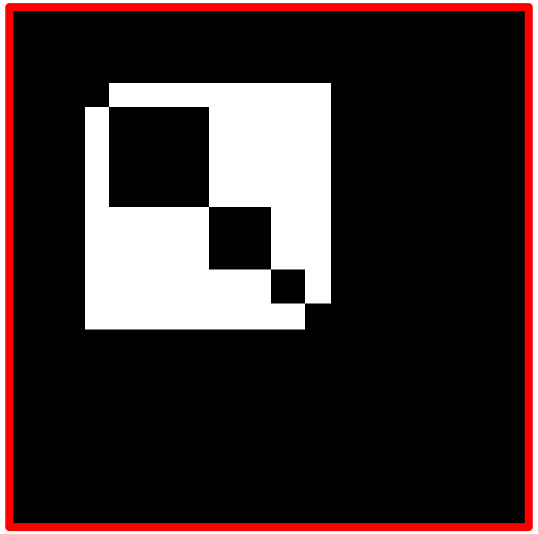


U

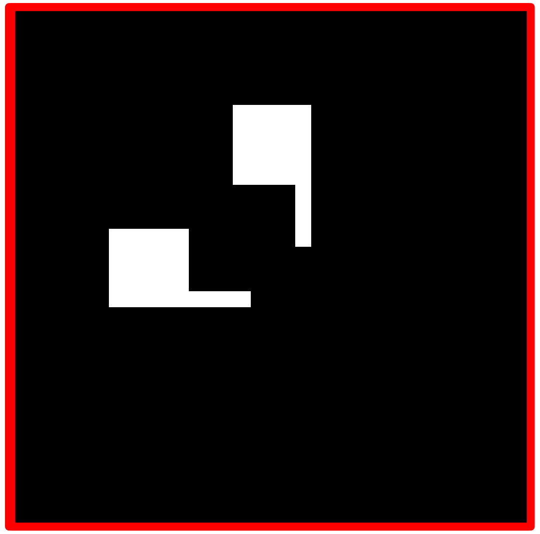


Open

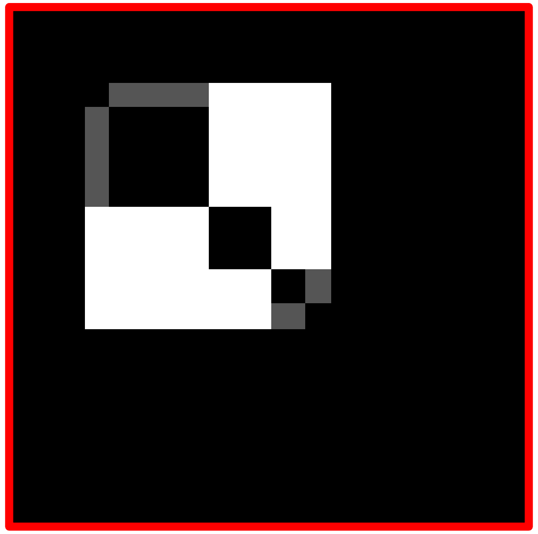
A



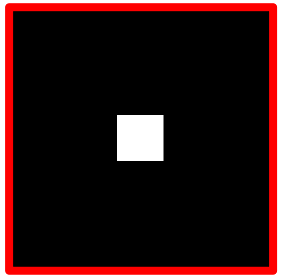
O = ERODE(A, U)



O = DILATE(O, U)



U



The gray area corresponds to the input

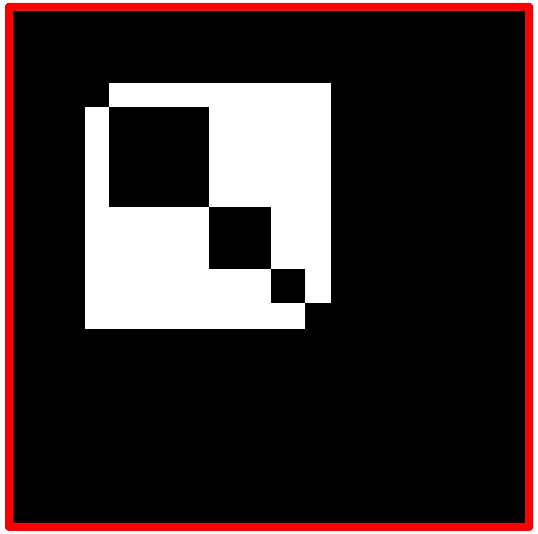
Closure

Closure Dilation followed by an Erosion

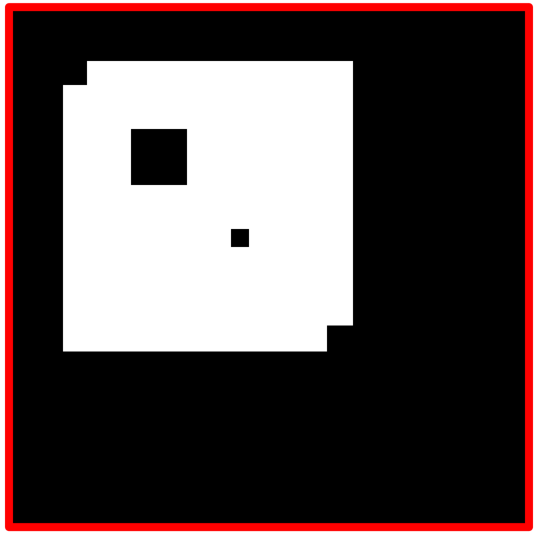
- Smooths the contours of an object, typically creates bridges
- Generally fuses narrow breaks

Close

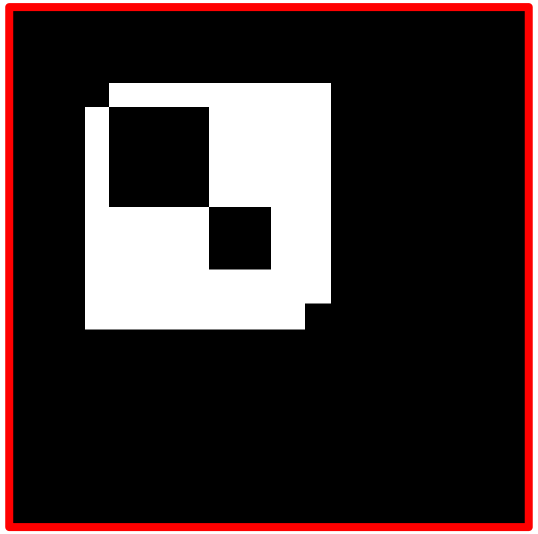
A



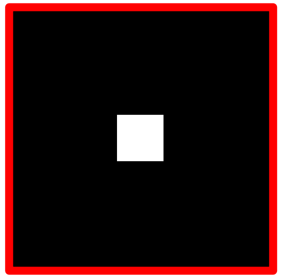
$O = \text{DILATE}(A, U)$



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

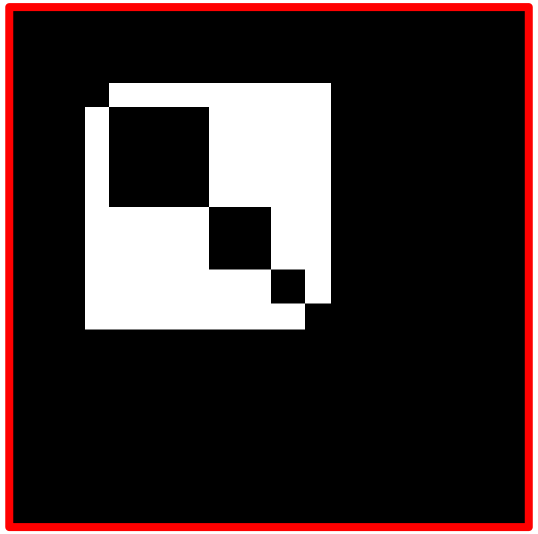


U

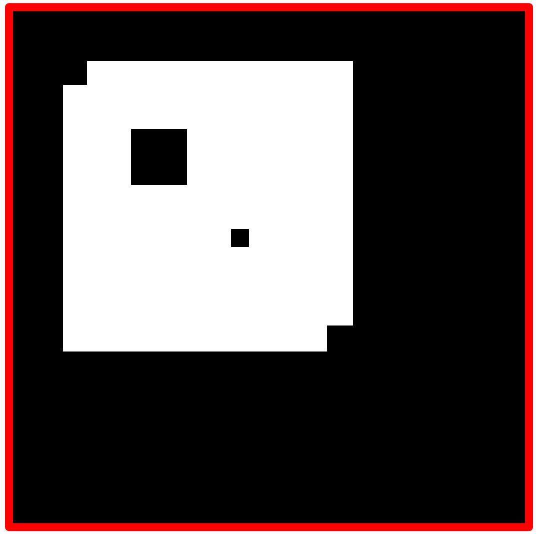


Close

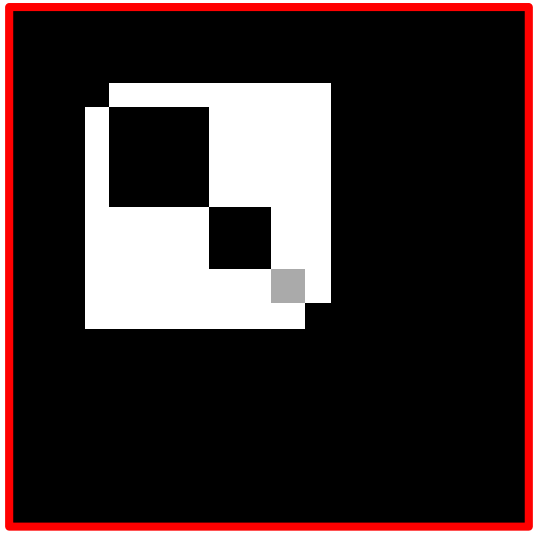
A



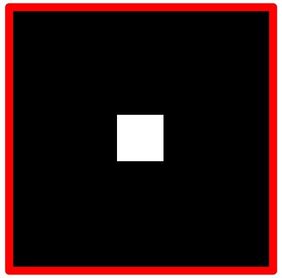
O = DILATE(A, U)



O = ERODE(O, U)



U



The gray spot was «false»
in the input

There are several other Non Linear Filters

Ordered Statistic based

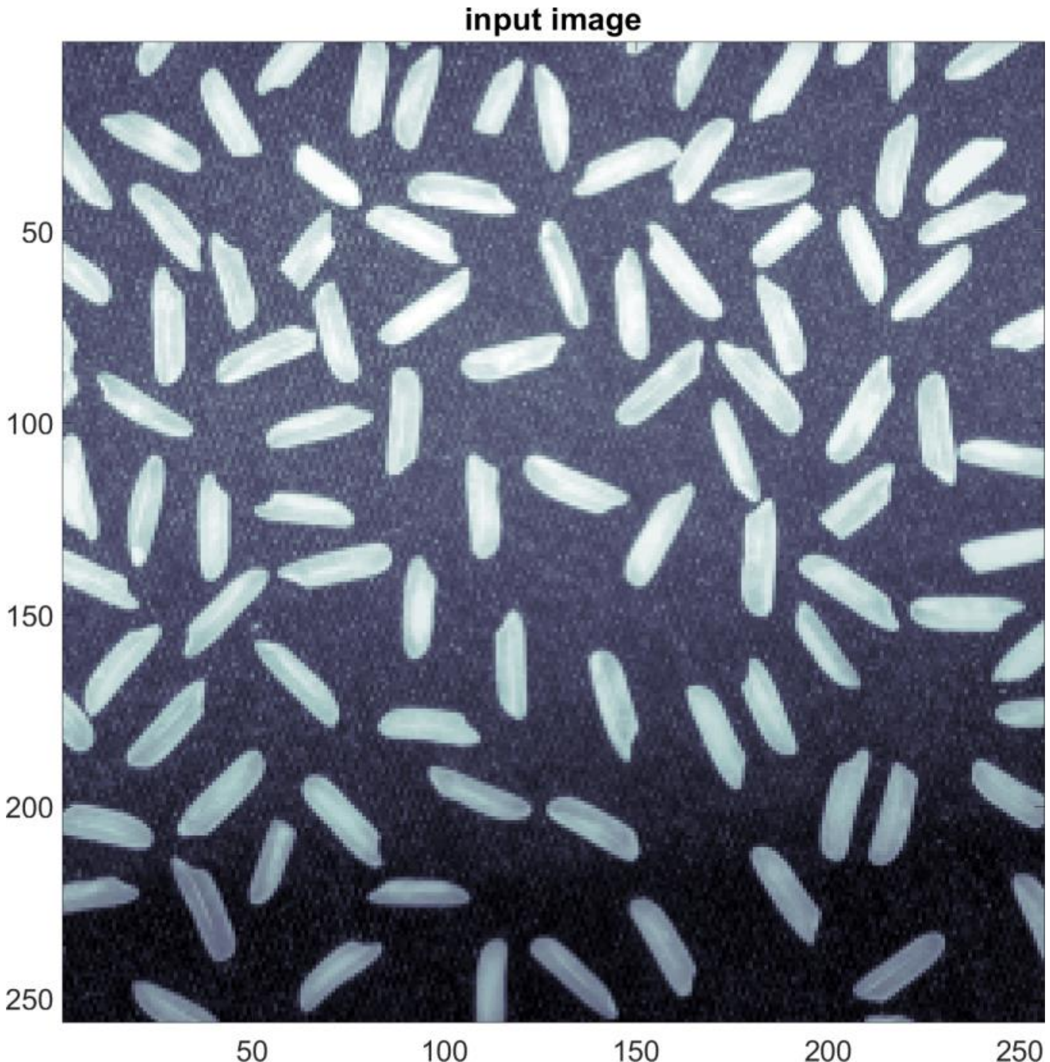
- Median Filter
- Weight Ordered Statistic Filter (being erosion and dilation special cases)
- Trimmed Mean
- Hybrid Median

Ordered statistics filters (including erosion and dilation) can be applied to grayscale images as well, as their definition is general

In Python: `skimage.morphology`

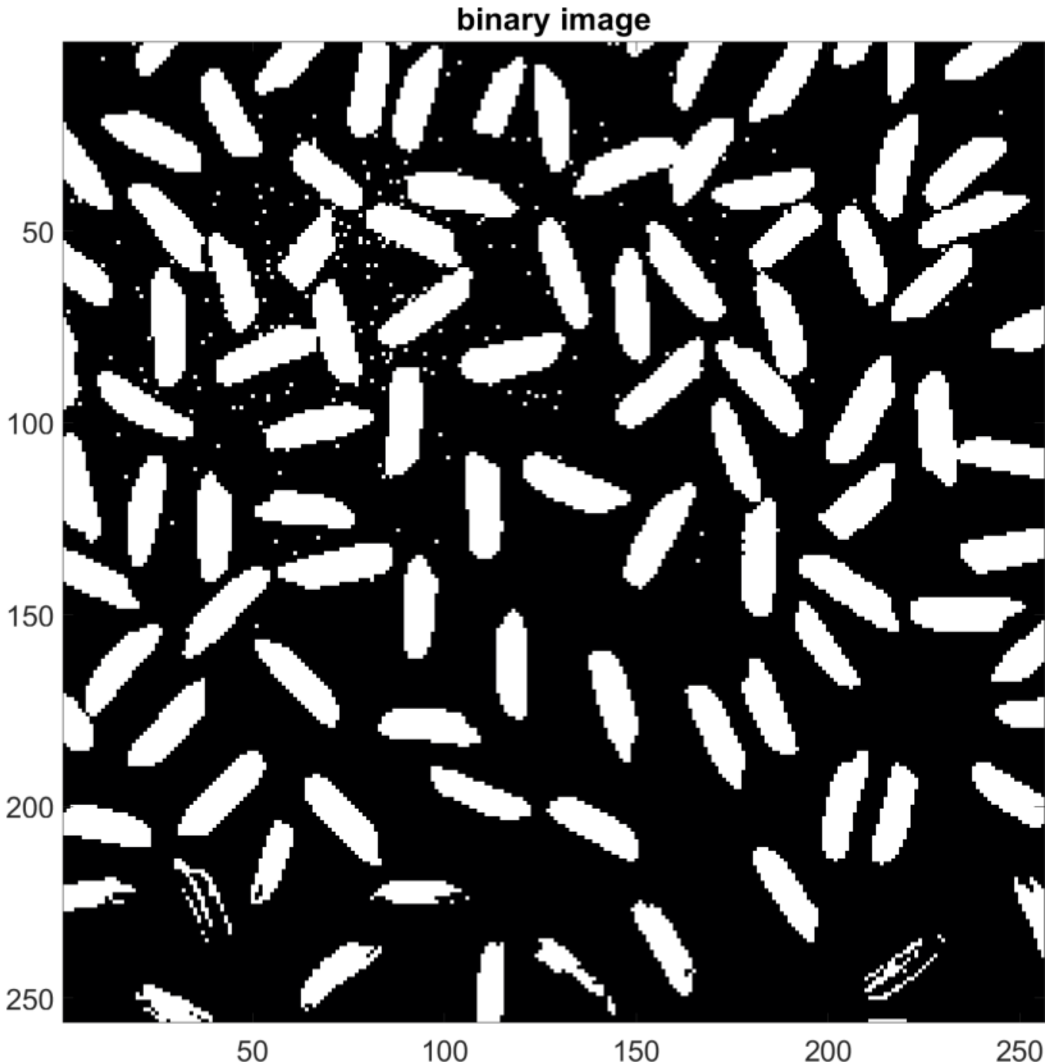
Extraction of connected components

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



Connected components

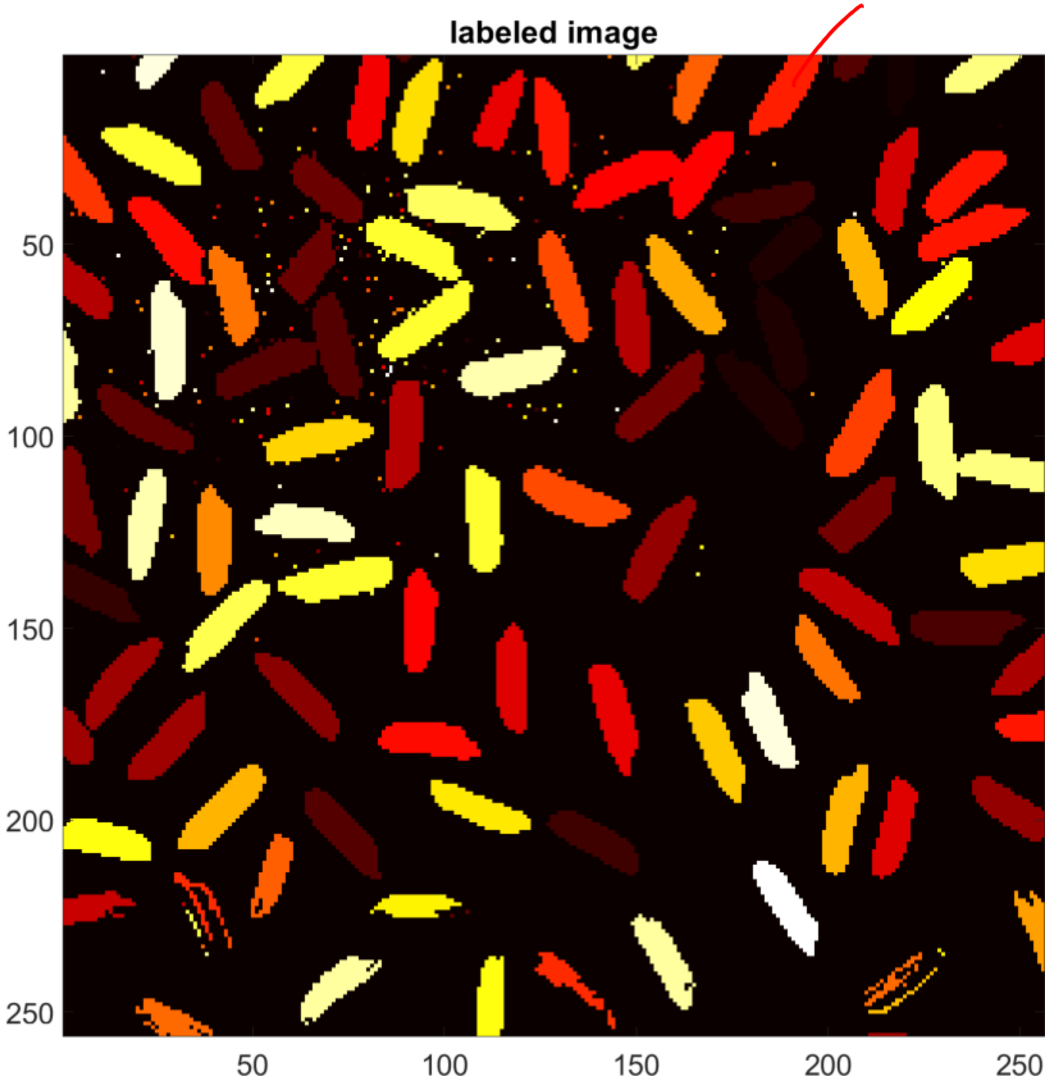
Extract subsets of pixels that are connected according to 4-pixel connectivity or 8-pixel connectivity



Connected components

This allows to identify different objects or target in the scene

Each color correspond to a different label



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

Connected components

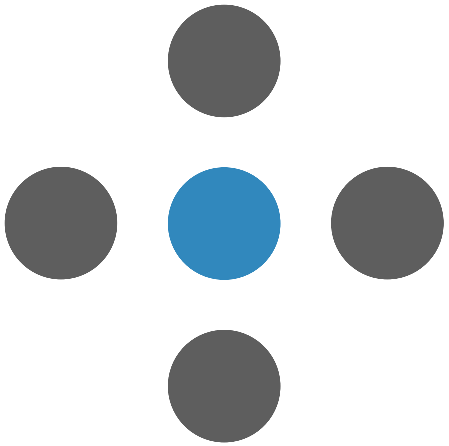
Here, each color denotes a different number, i.e. a label.

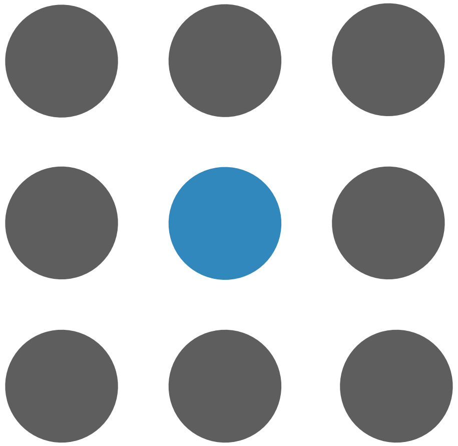
It is enough to scan the image twice.

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





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 $L(r, c)$

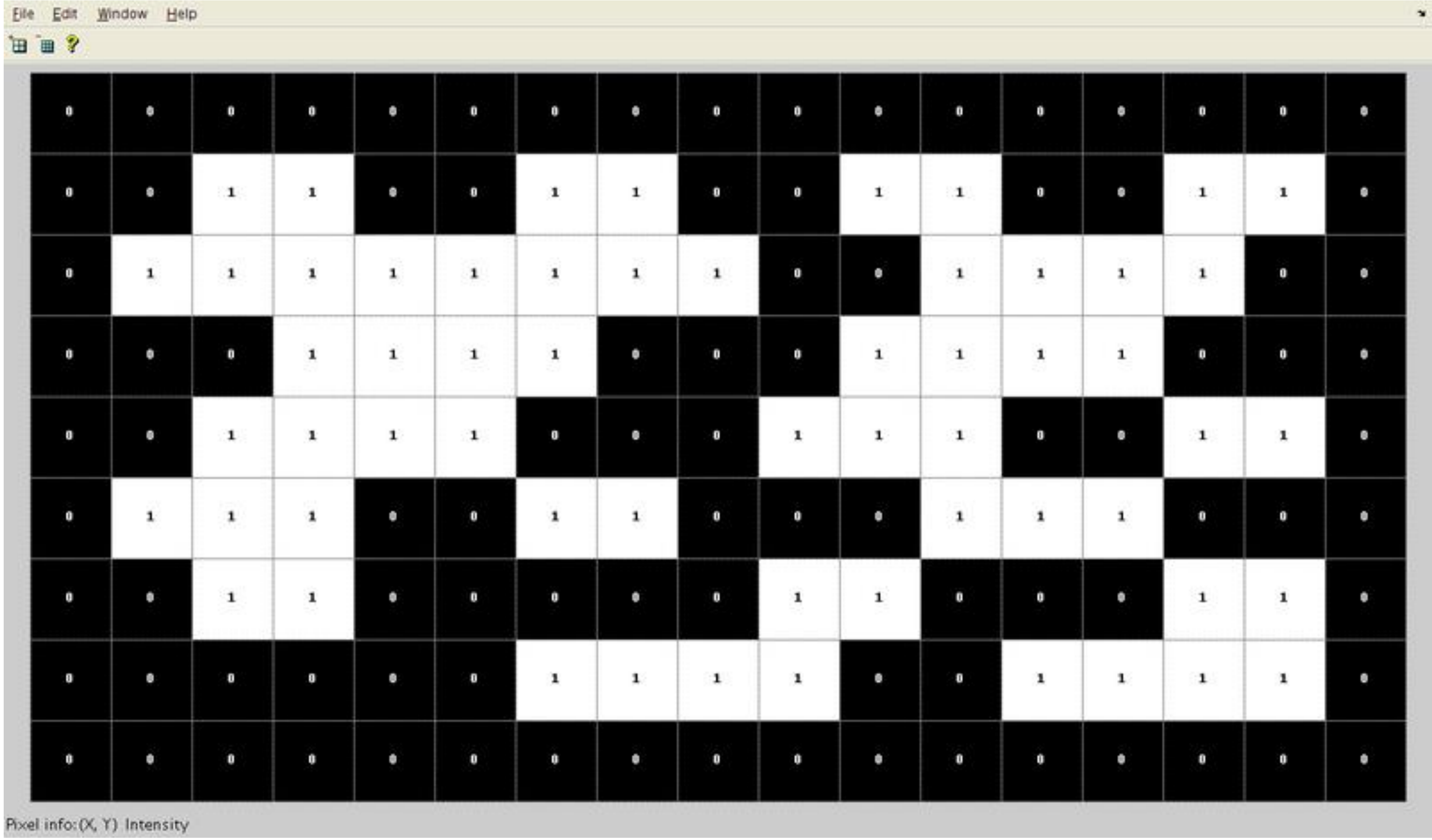
Else $L(r, c) =\ \min\big(L(u, v)\big)$ over $U_{(r,c)}$

If there are different labels in $U_{(r,c)}$

Record they are equivalent in a table

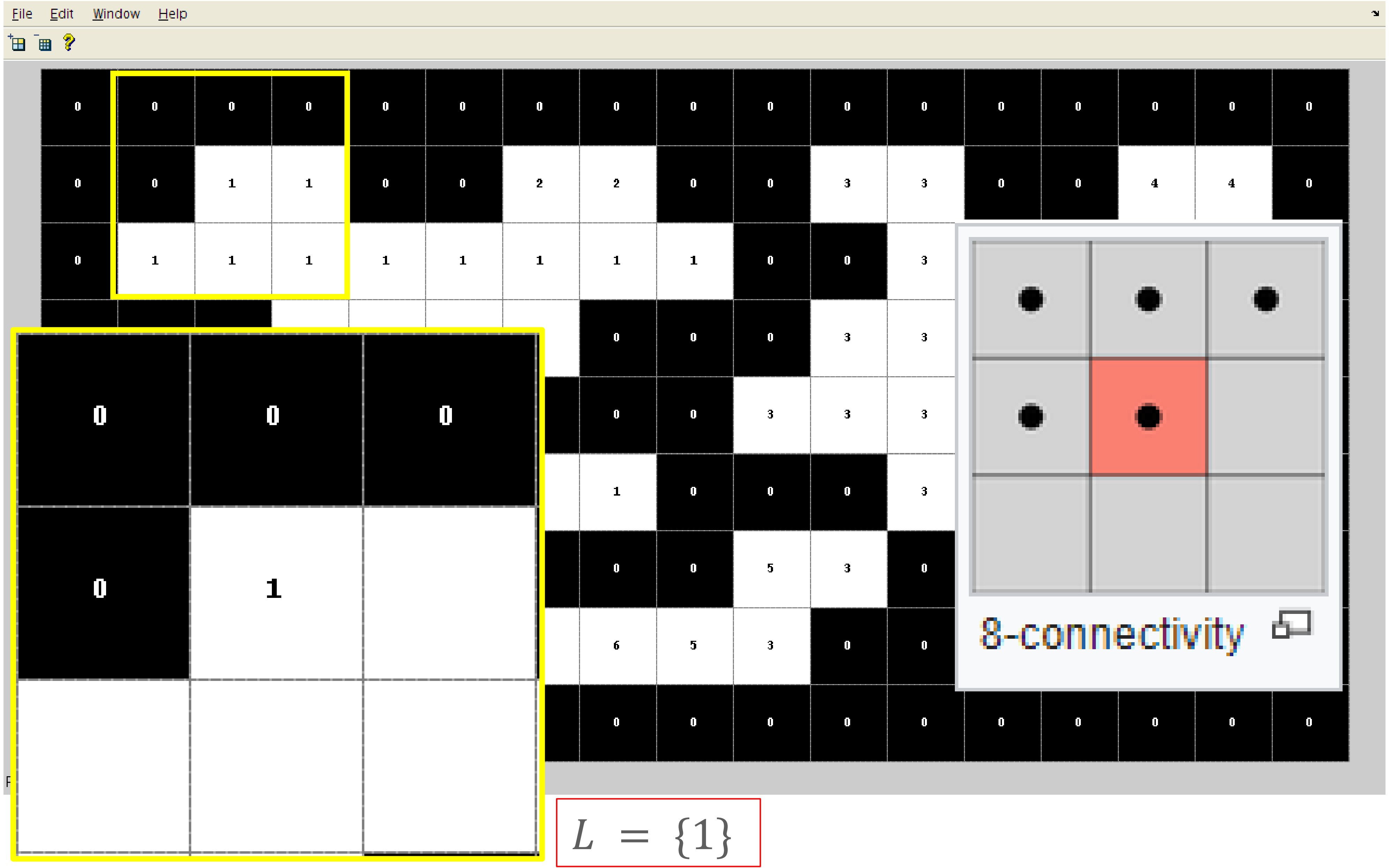
In Python **`skimage.measure.label`**

Binary input image



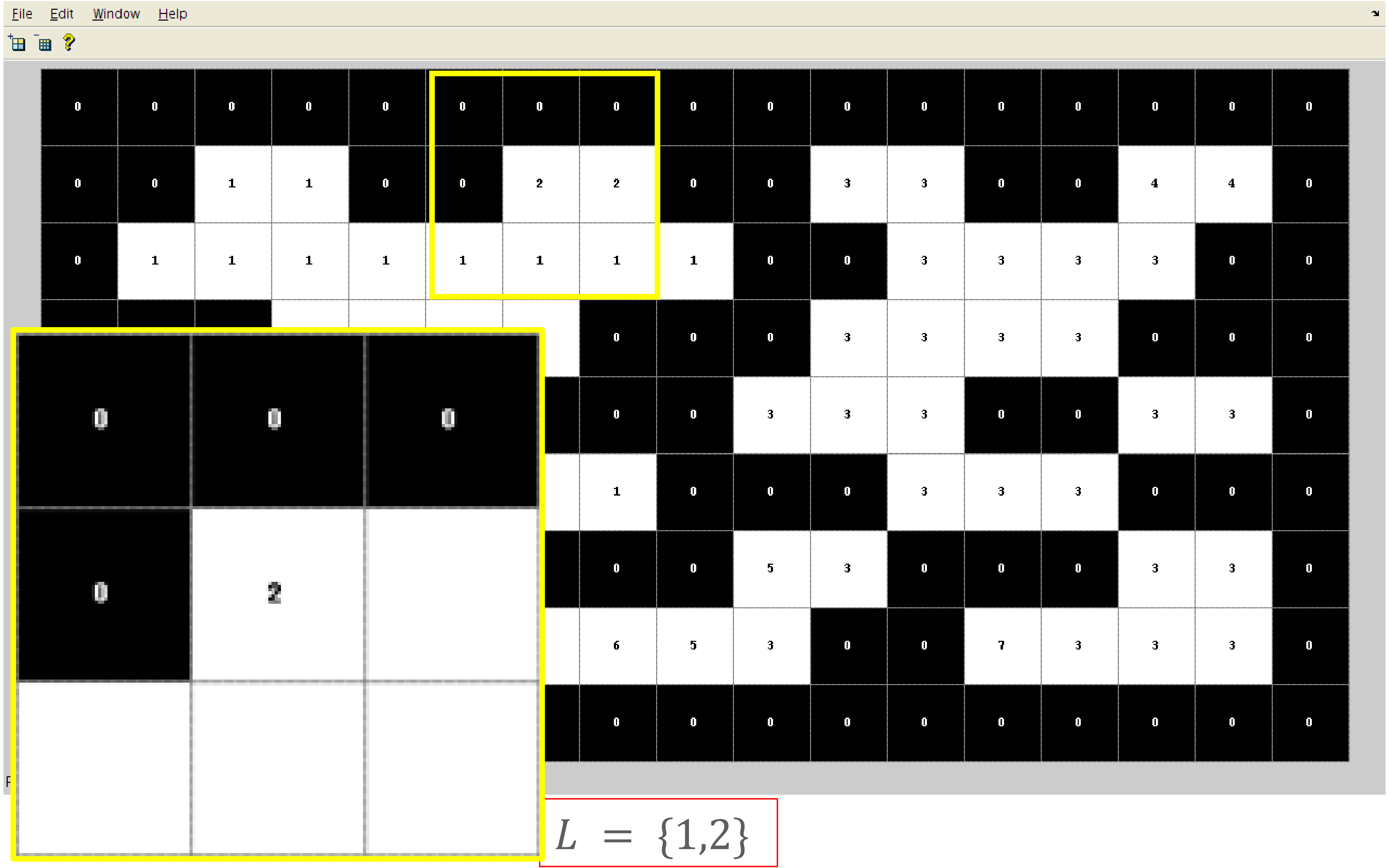
By Dhull003 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=10166860

Iterations of the first pass



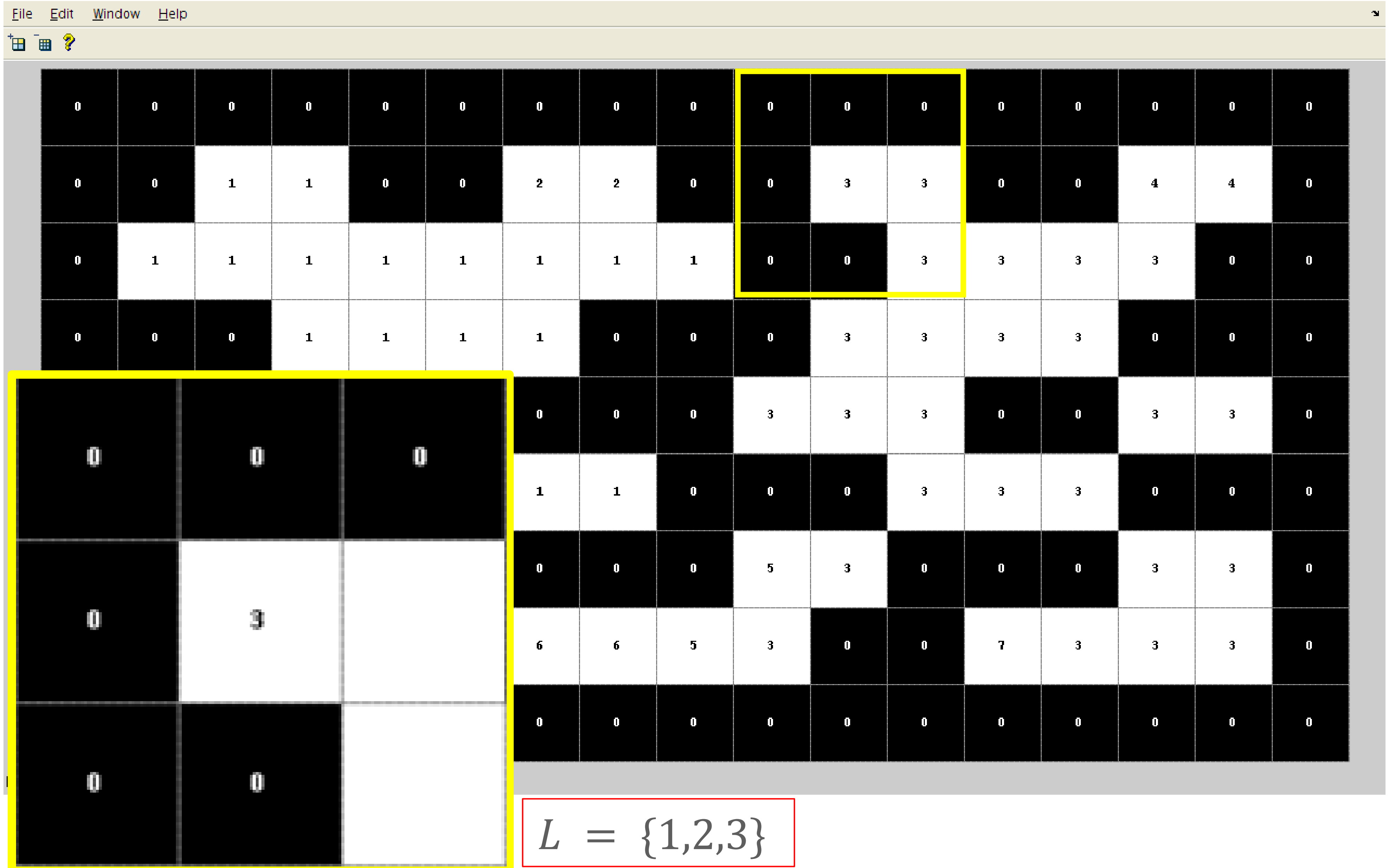
$L = \{1\}$

Iterations of the first pass



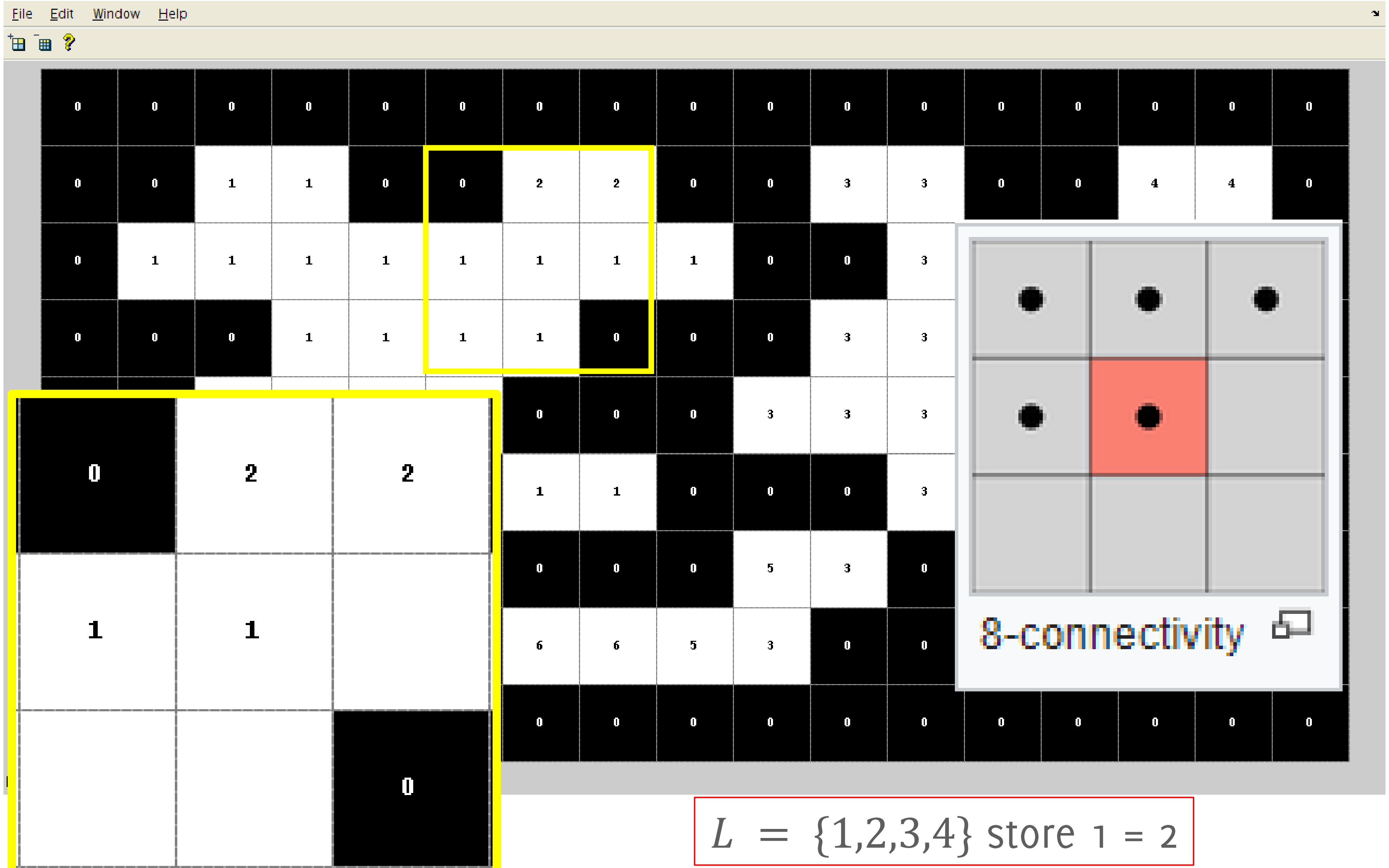
$$L = \{1,2\}$$

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

File Edit Window Help

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

Pixel info: (X, Y) Intensity

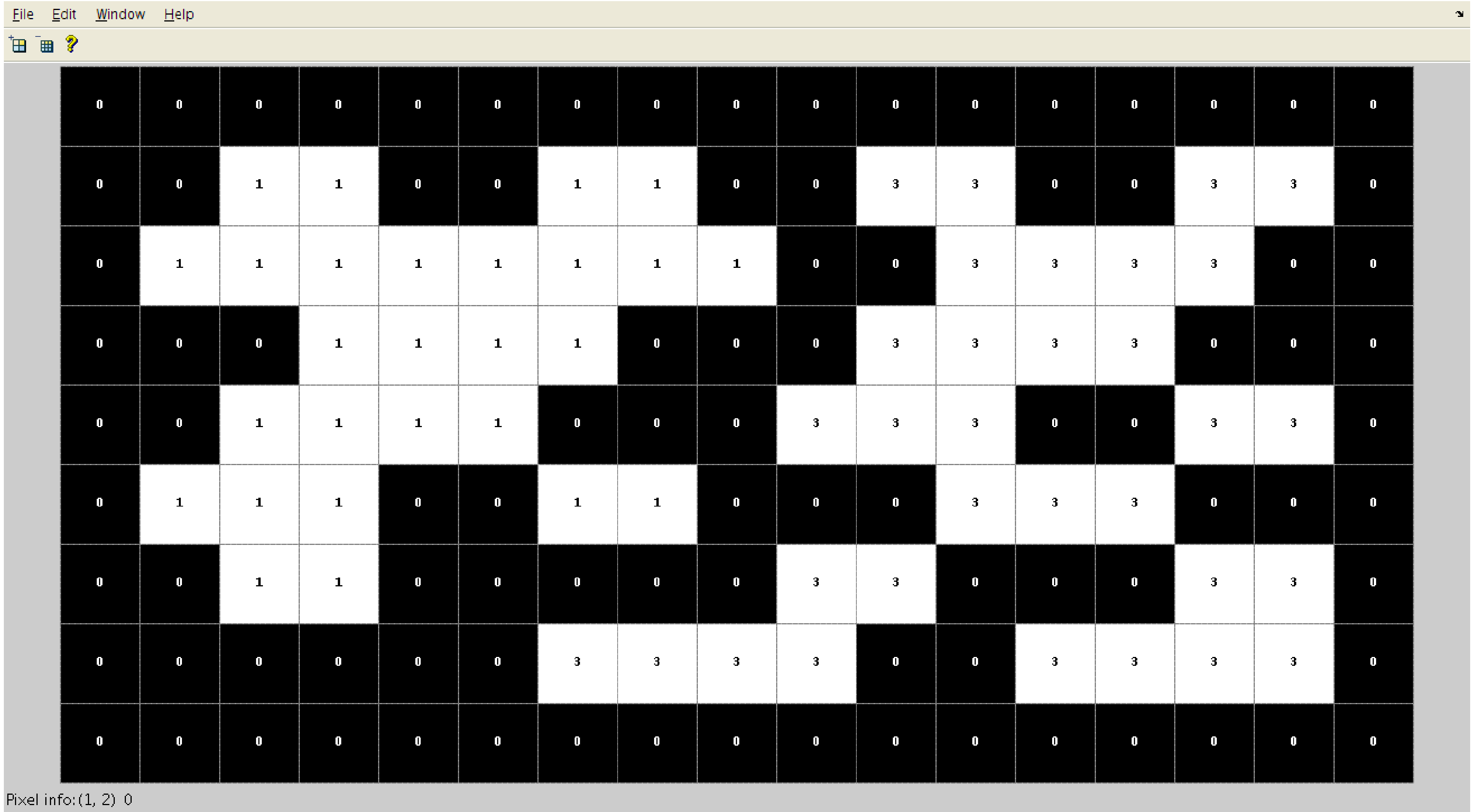
$L = \{1,2,3,4,5,6,7\}$ equivalence sets $\{1,2\}$, $\{3,4,5,6,7\}$

Two Pass Algorithm: Second Pass

Iterate through each pixel (r, c)

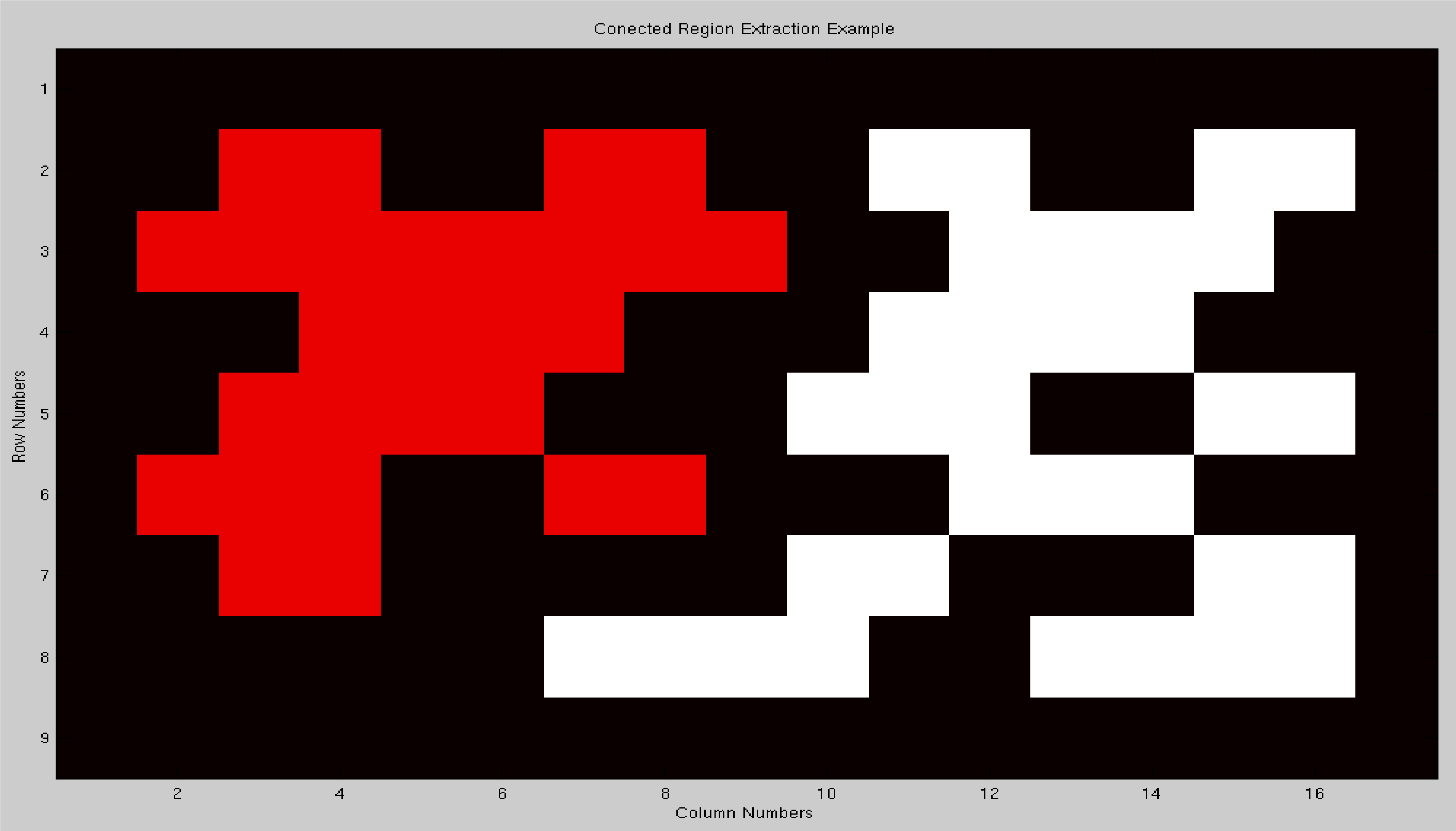
- If $I(r, c) == 1$
  - Relabel the element with the lowest equivalent label

Output of the Second Pass



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

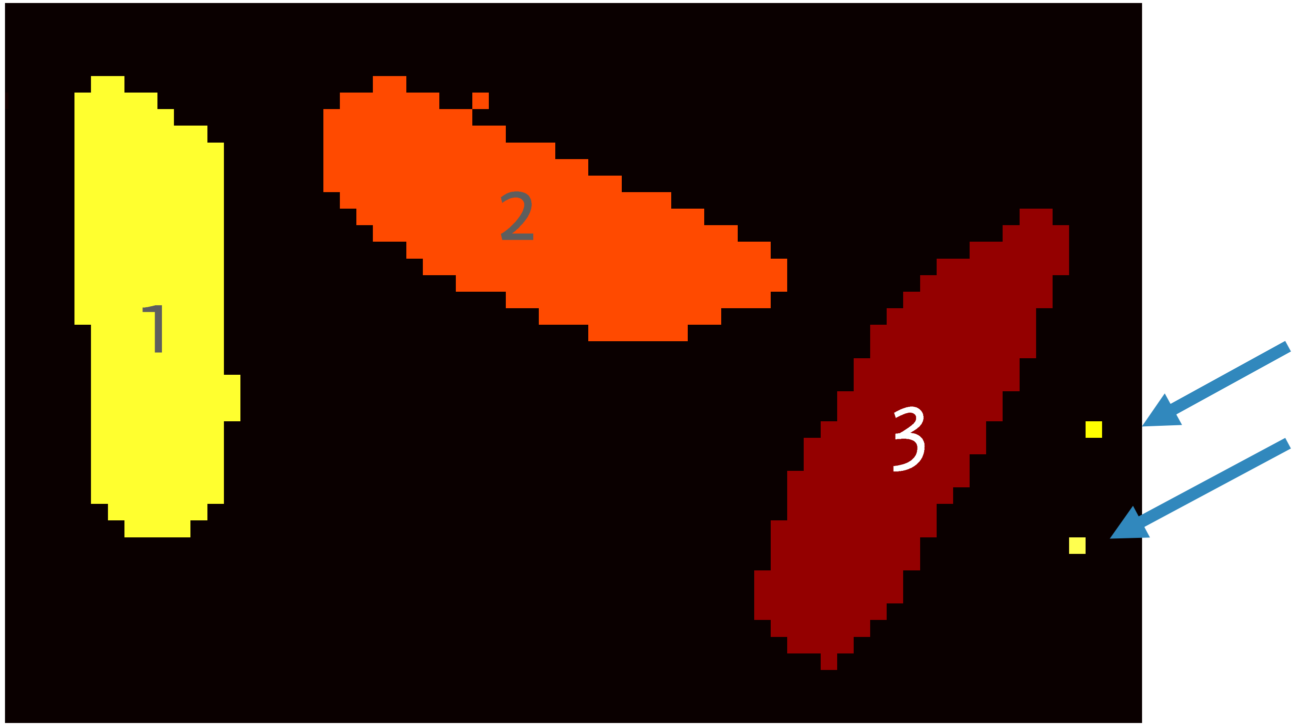
Output of the Second Pass



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

Bounding Box vs Axis

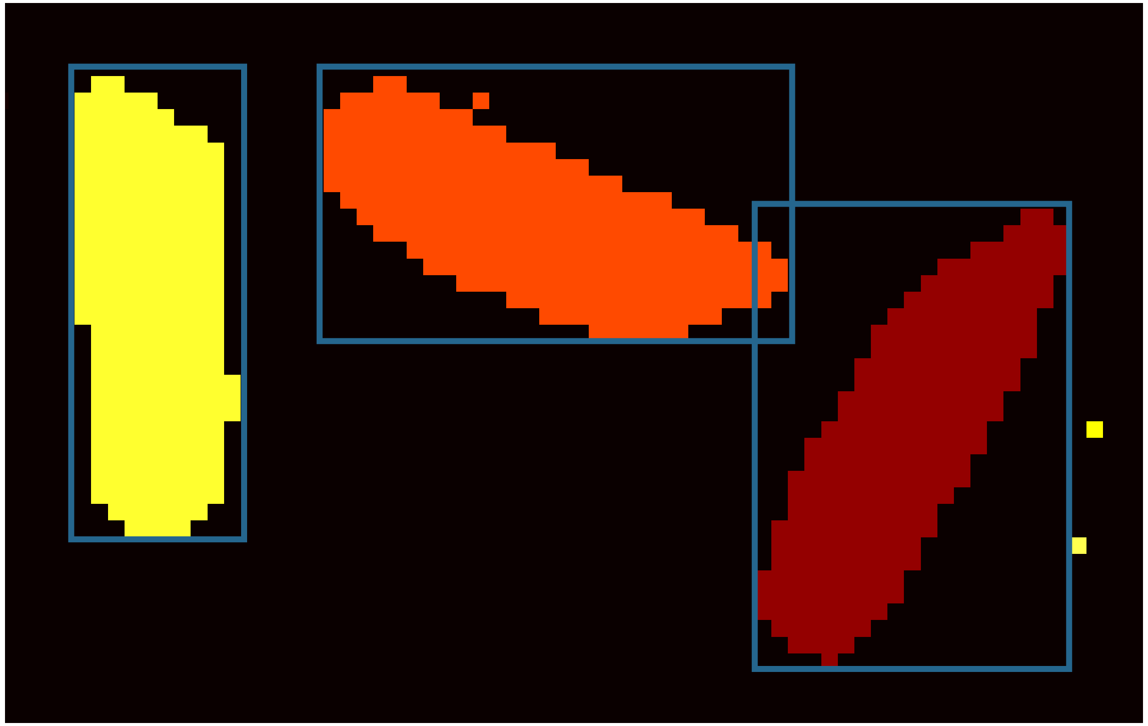
These provide information about size and orientation of the object



Too small, can be removed analyzing the area

Bounding Box vs Axis

These provide information about size and orientation of the object

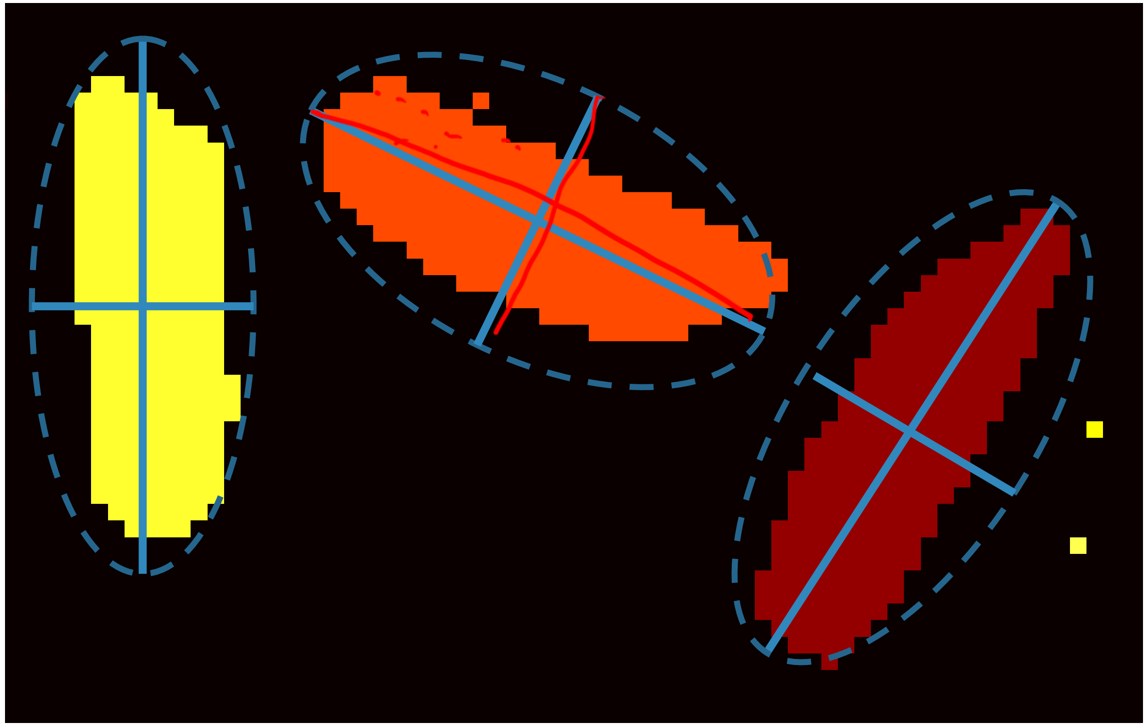


Bounding box allow to crop separate images for each component

(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

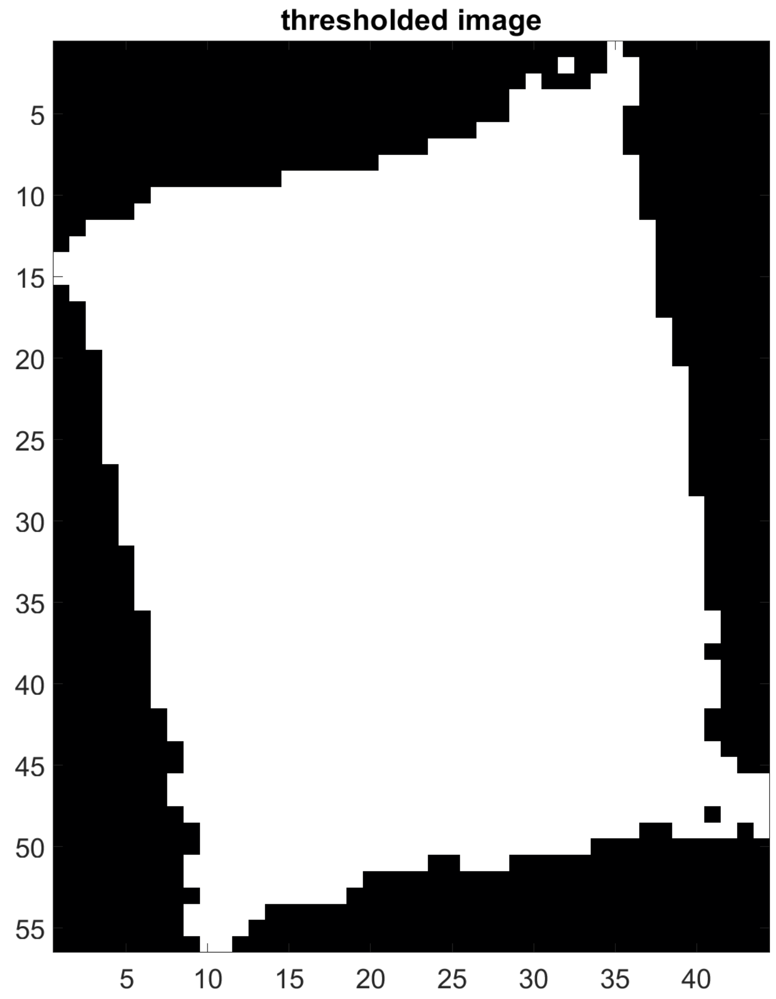


Blob axis are computer as axis the of the ellipse that has the same second-moments as the region.

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

Boundary Extraction – Morphological Gradient

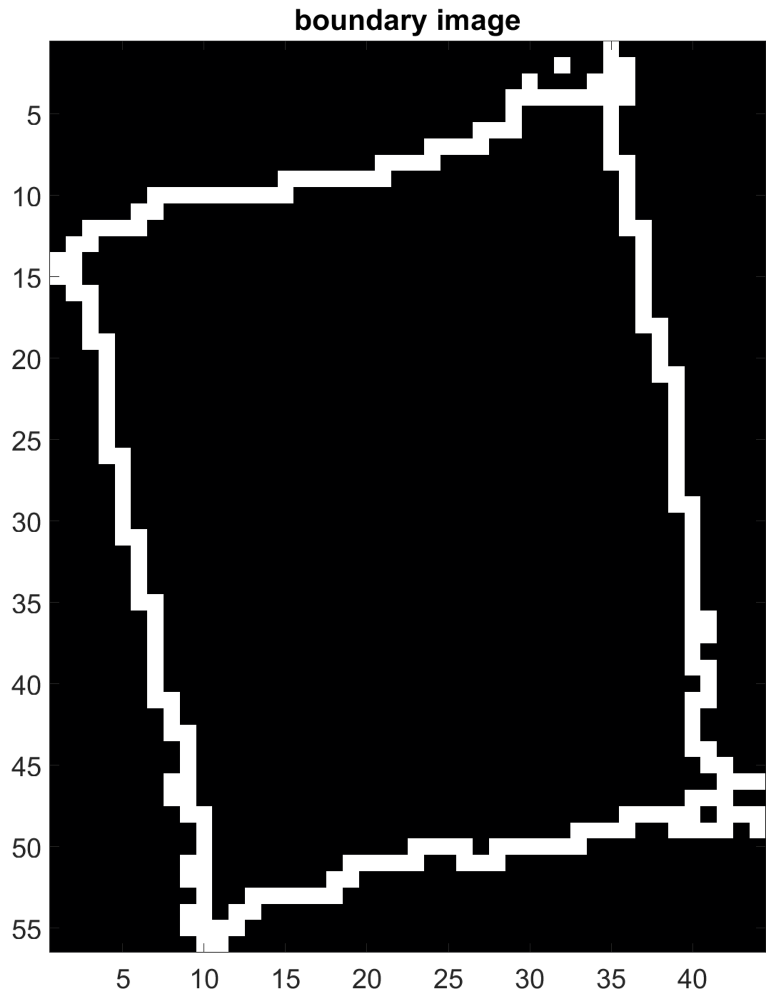
The simplest way to extract boundaries of an image is to subtract from a binary image its eroded version



$$T = I > \Gamma$$

Boundary Extraction – Morphological Gradient

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)]

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Edges, corners & features in images

Giacomo Boracchi, Luca Magri

luca.magri@polimi.it

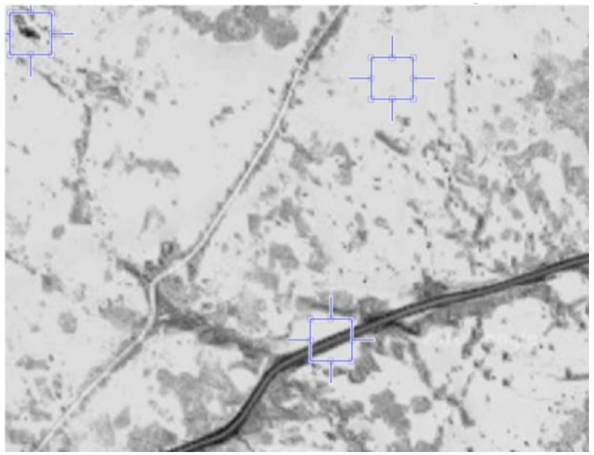
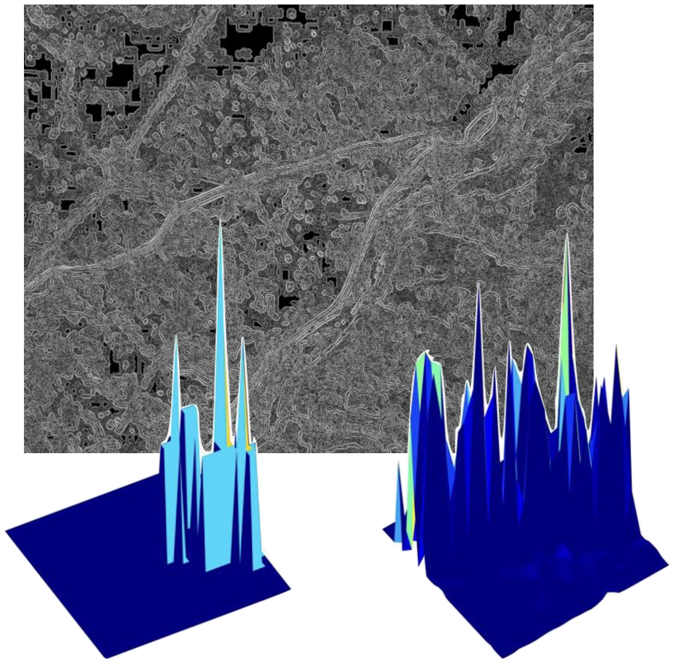
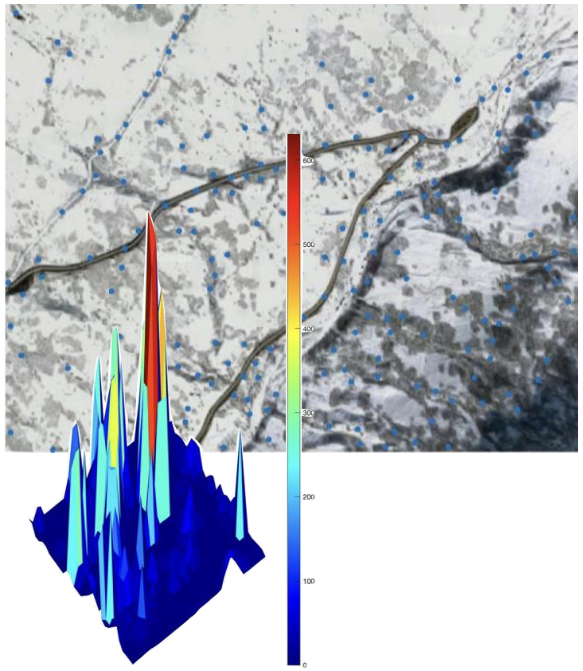
20878 - COMPUTER VISION AND IMAGE PROCESSING

Università Bocconi, Milano

Agenda for today: edges corners & features in images

- **Edge detection:** derivative filters
- The problem of geometric matching
    - **Corner detection:**
        - Moravec
        - Harris
    - Scale Invariant Feature Transform (a milestone in Computer Vision)

Suggested reading
Szeliski R.
Computer Vision
Ch. 7

Derivatives Estimation

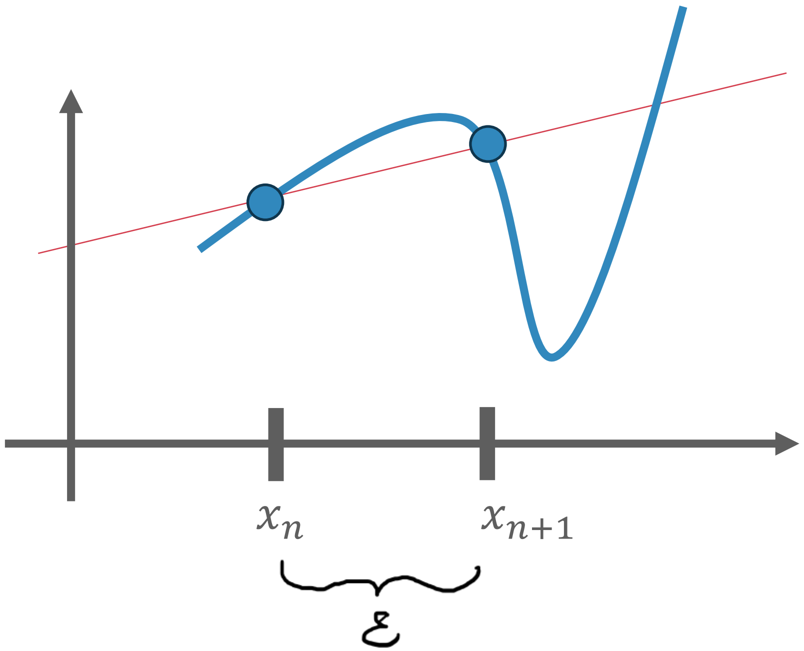
Differentiation and convolution

Recall the definition of derivative

$$\frac{\partial f(x_0)}{\partial x} = \lim_{\epsilon \to 0} \left(\frac{f(x_0 + \epsilon) - f(x_0)}{\epsilon} \right)$$

Now this is linear and shift invariant.

Therefore, in discrete domain, it will be represented as a convolution



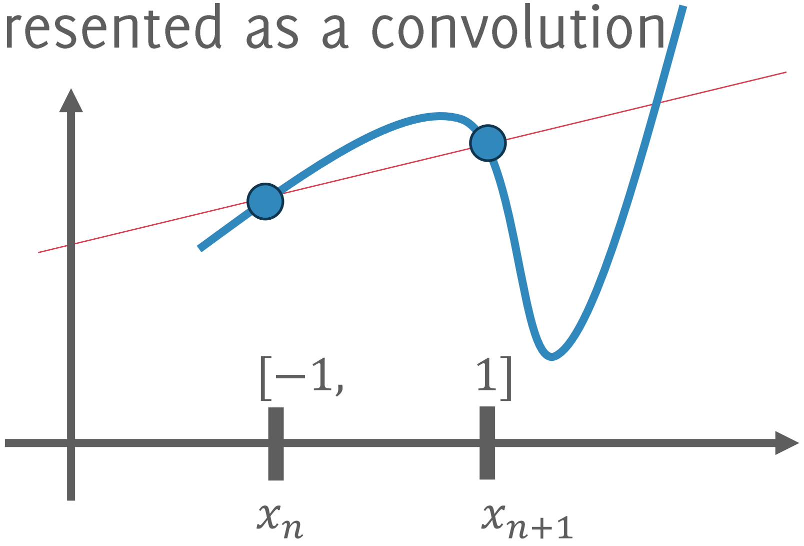
Differentiation and convolution

Recall the definition of derivative

$$\frac{\partial f(x_0)}{\partial x} = \lim_{\epsilon \to 0} \left(\frac{f(x_0 + \epsilon) - f(x_0)}{\epsilon} \right)$$

Now this is linear and shift invariant.

Therefore, in discrete domain, it will be represented as a convolution



We could approximate this as

$$\frac{\partial f(x_n)}{\partial x} \approx \frac{f(x_{n+1}) - f(x_n)}{\Delta x}$$

which is obviously a convolution against the Kernel [1 -1];

Finite Differences in 2D (discrete) domain

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \left(\frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon} \right)$$

$$\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0} \left(\frac{f(x, y+\varepsilon) - f(x,y)}{\varepsilon} \right)$$

Horizontal

$$\begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$\frac{\partial f(x_n, y_m)}{\partial x} \approx \frac{f(x_{n+1}, y_m) - f(x_n, y_m)}{\Delta x}$$

$$\frac{\partial f(x_n, y_m)}{\partial y} \approx \frac{f(x_n, y_{m+1}) - f(x_n, y_m)}{\Delta y}$$

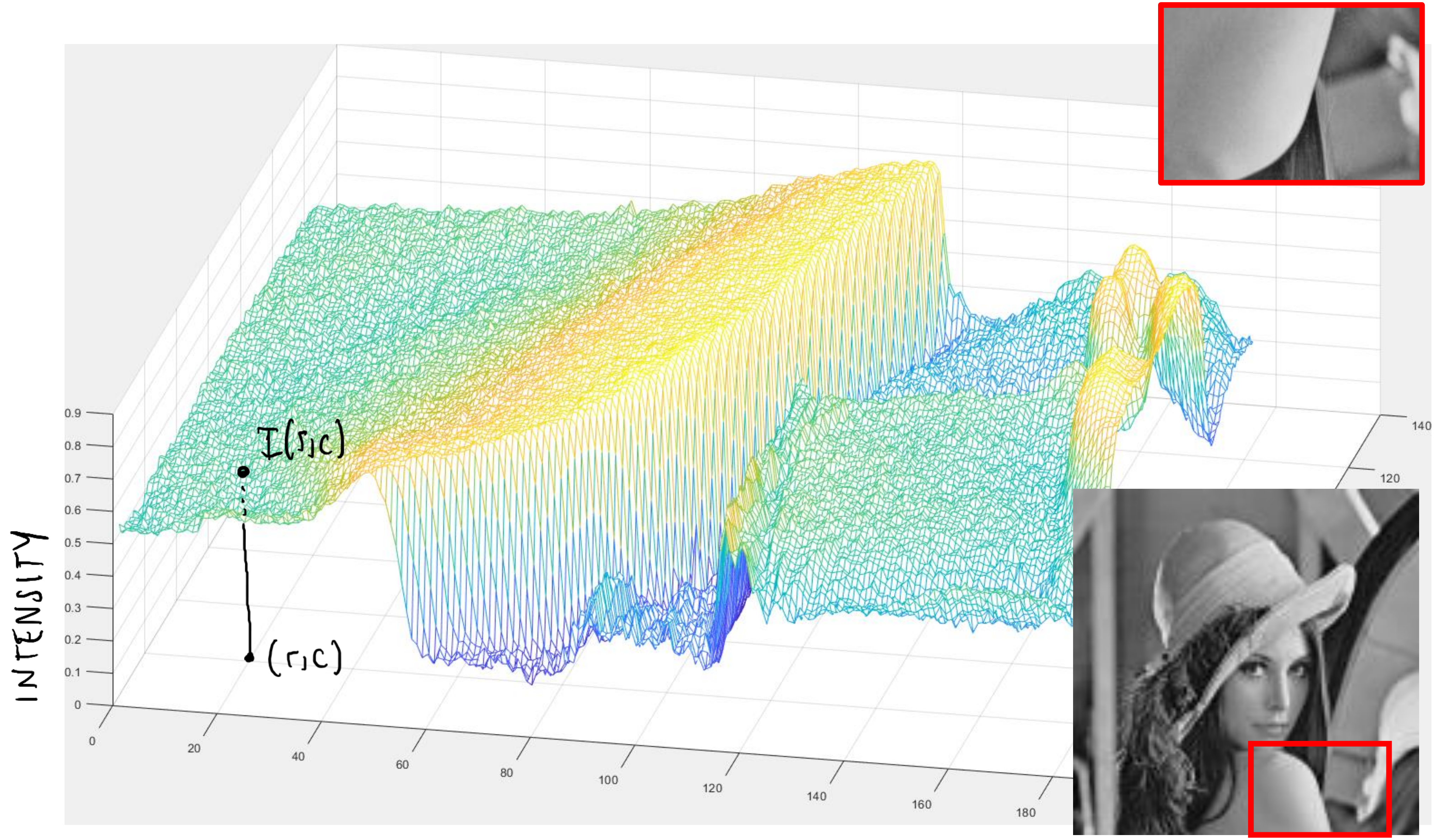
Vertical

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Discrete Approximation

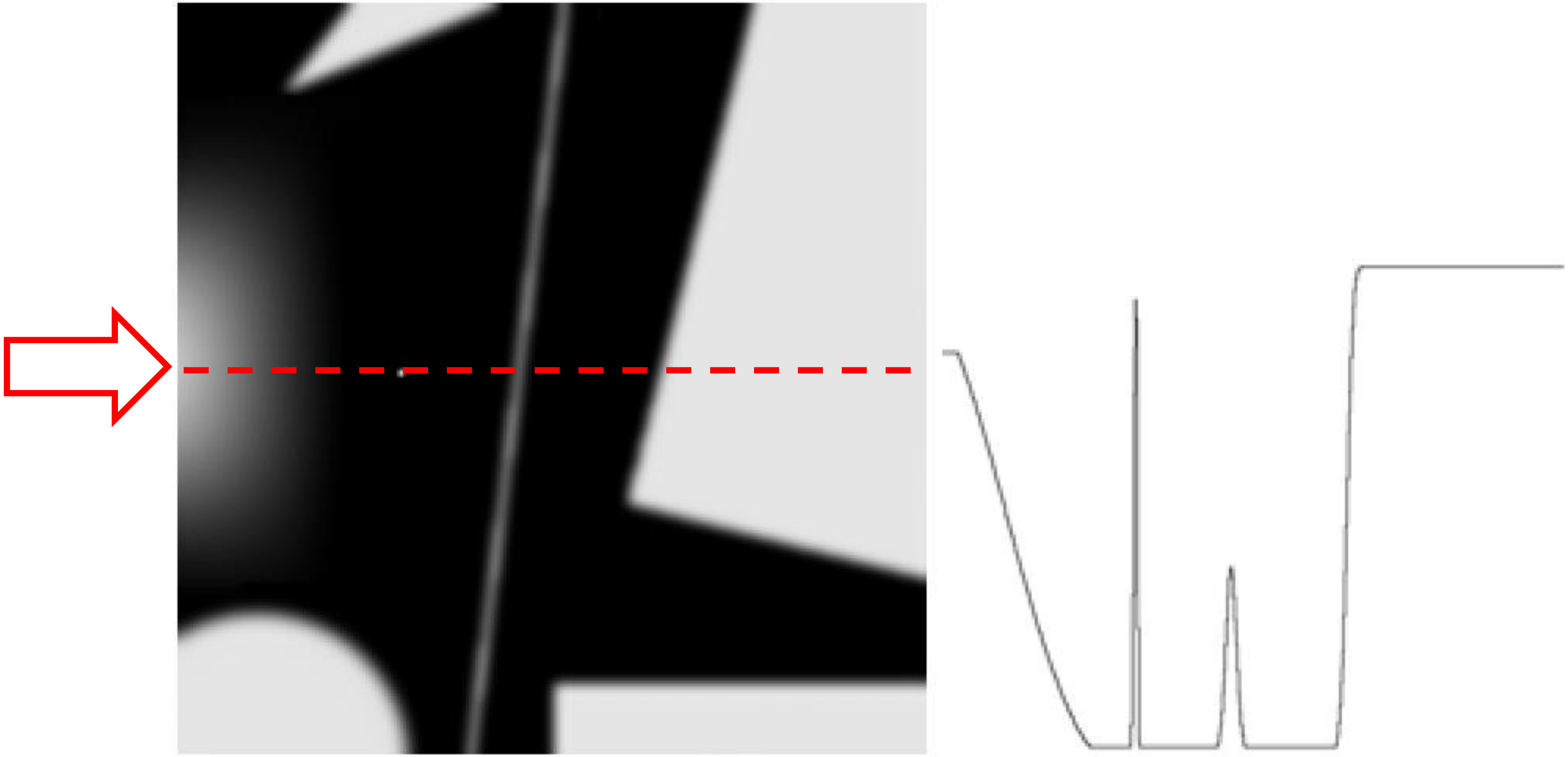
Convolution Kernels

Think of an image as a 2d, real-valued function



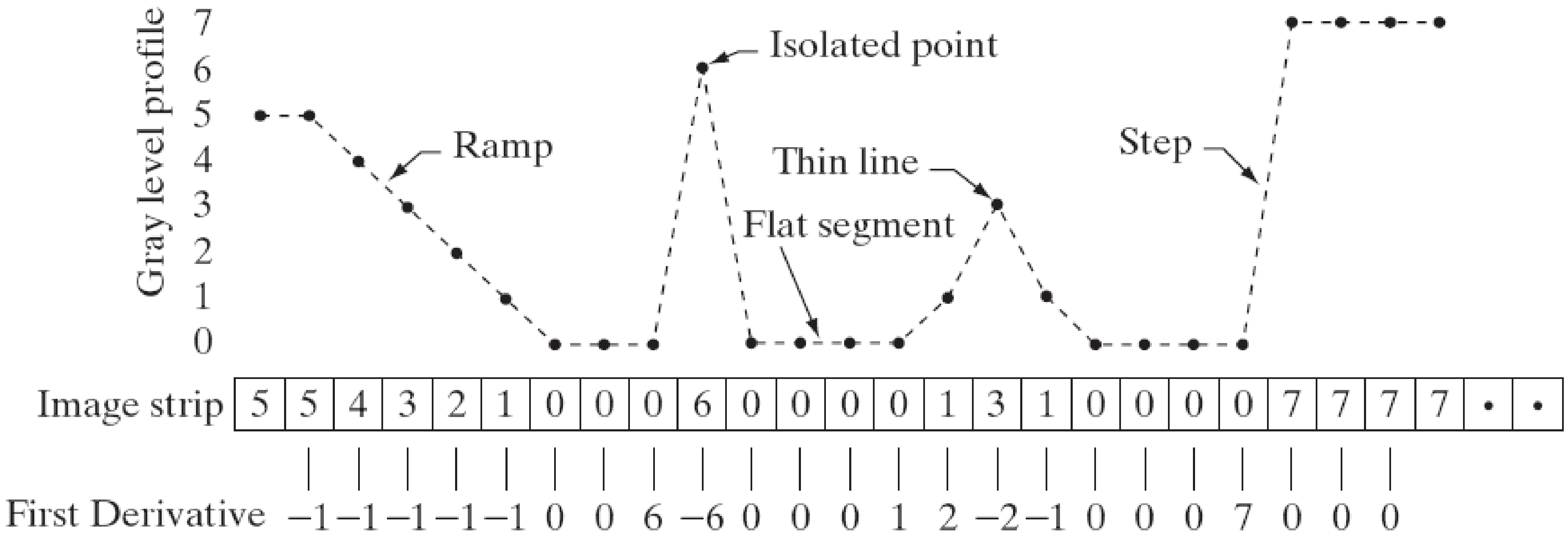
A 1D Example

Take a line on a grayscale image



A 1D Example (II)

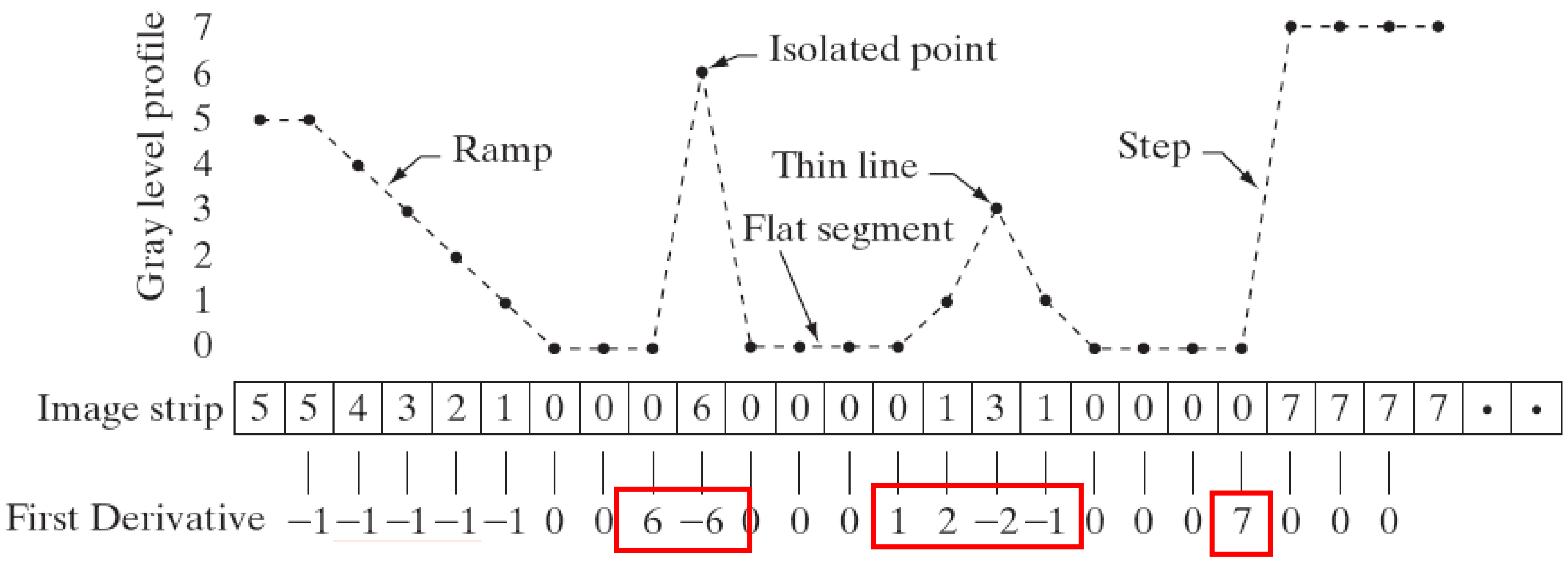
Filter the image values by a convolution against the filter [1 -1]



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

Derivatives

Derivatives are used to **highlight intensity discontinuities** in an image and to deemphasize regions with slowly varying intensity levels



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

Differentiating Filters

The derivatives can be also computed using centered filters:

$$f_x(x) = f(x-1) - f(x+1)$$

Such that the horizontal derivative is:

$$f_x = f \otimes \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

While the vertical derivative is:

$$f_y = f \otimes \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}^T$$

Classical Operators: Prewitt

Horizontal derivative

$$s = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad dx = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad h_x = s \circledast dx = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Smooth

Differentiate

Vertical derivative

$$s = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad dy = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad h_y = s \circledast dy = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

Classical Operators: Sobel

Horizontal derivative

$$s = \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 1 & 1 \end{bmatrix} \qquad dx = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad h_x = s \circledast dx = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Smooth   Differentiate

Vertical derivative

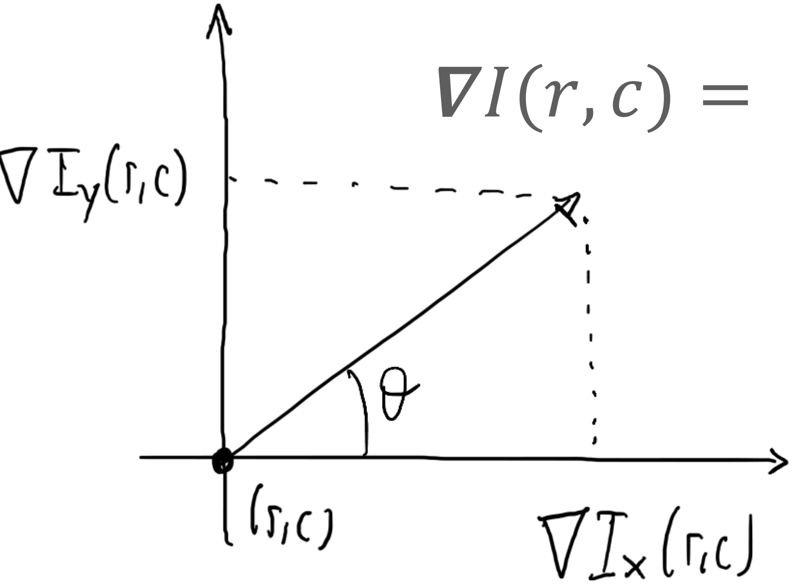
$$s = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & 1 \end{bmatrix} \qquad dy = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \qquad h_y = s \circledast dy = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

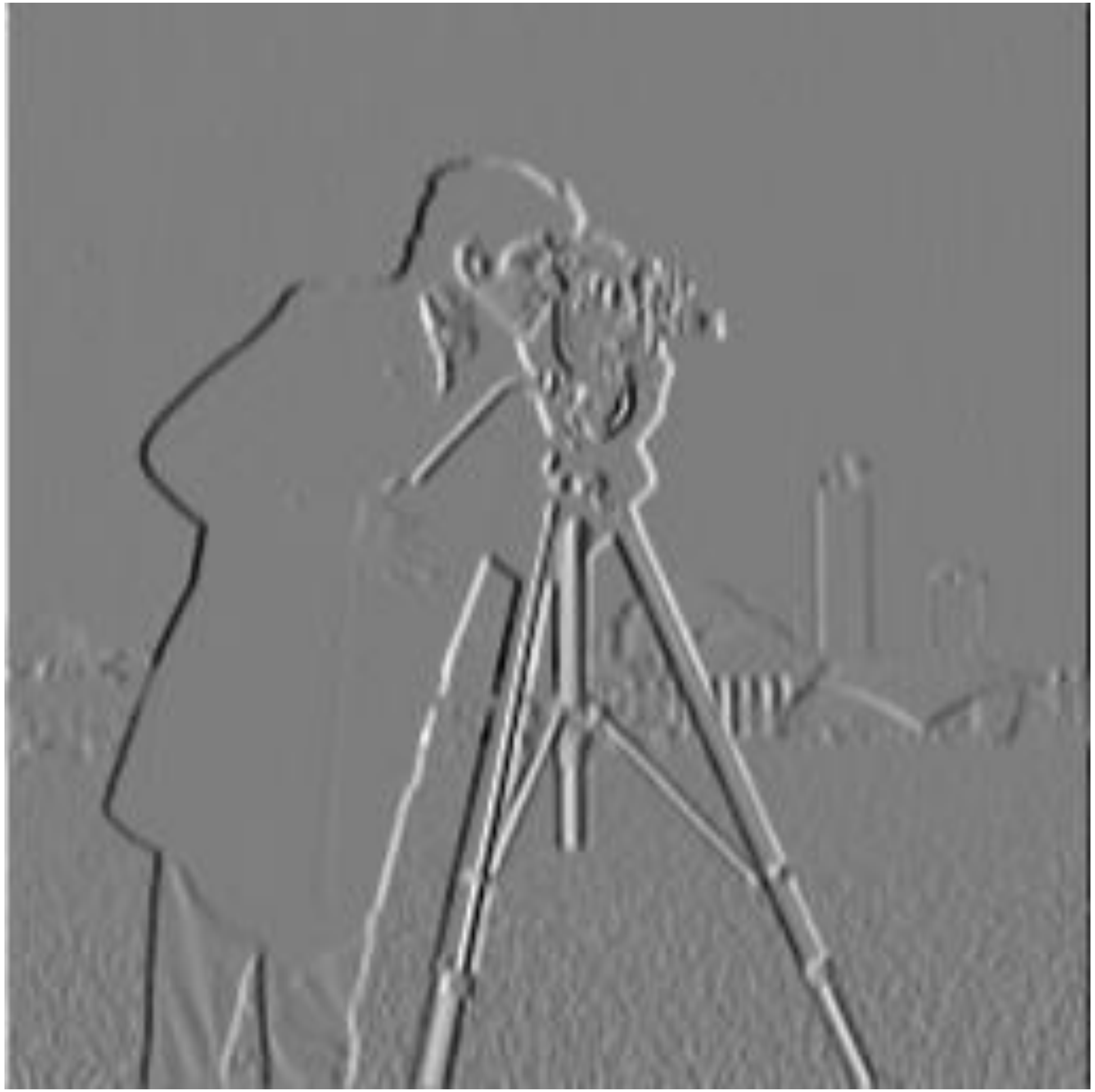
Another famous test image - cameraman

Horizontal Derivatives using Sobel

$$\nabla I_x = (I \circledast d_x)$$

$$\nabla I(r,c) = \begin{bmatrix} \nabla I_x(r,c) \\ \nabla I_y(r,c) \end{bmatrix}$$

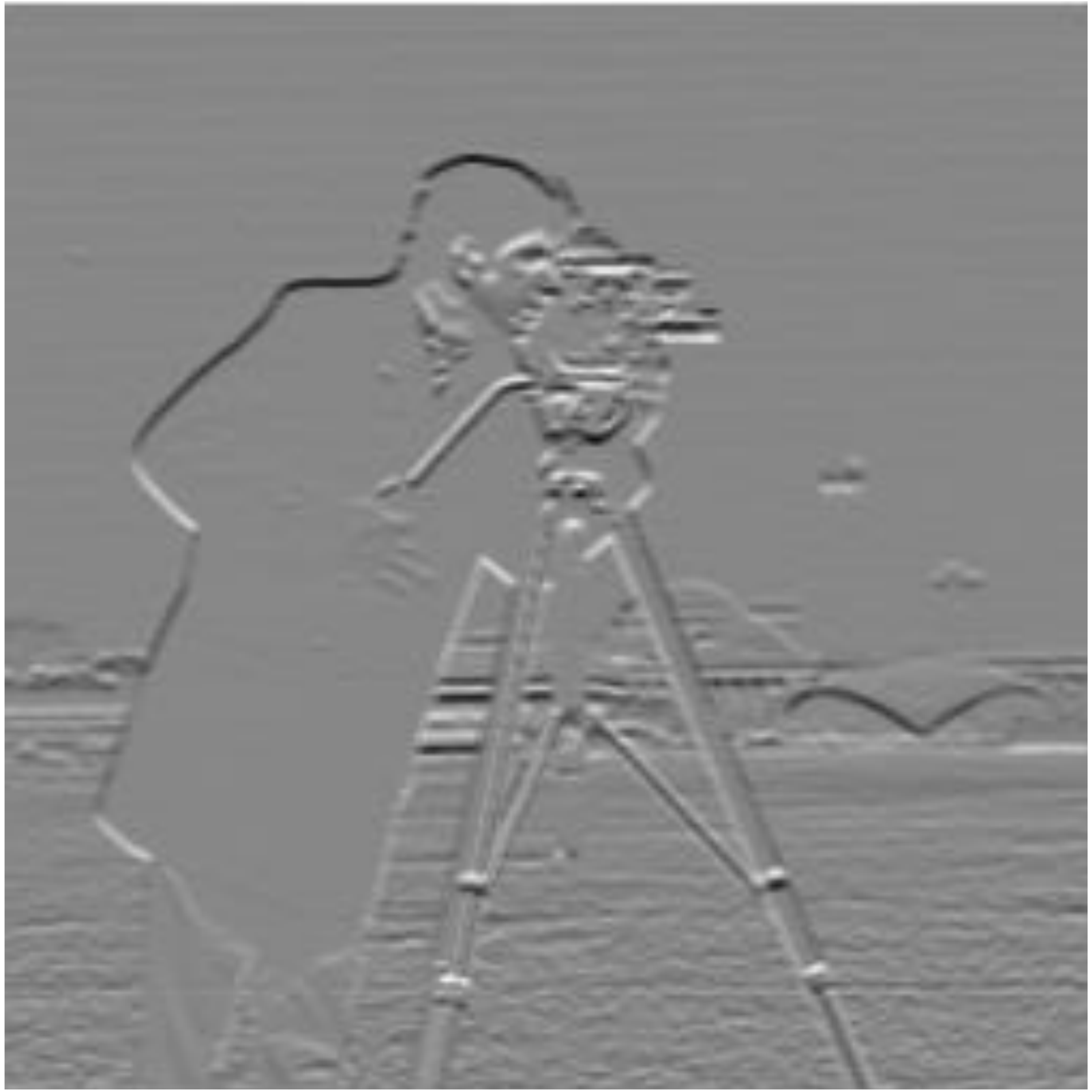




Vertical Derivatives using Sobel

$$\nabla I_y = (I \circledast d_y)$$
$$d_y = d_x^{\top}$$

$$\nabla I(r,c) = \begin{bmatrix} \nabla I_x(r,c) \\ \nabla I_y(r,c) \end{bmatrix}$$



Gradient Magnitude

$$\|\nabla I\| = \sqrt{(I \circledast d_x)^2 + \left(I \circledast d_y\right)^2}$$

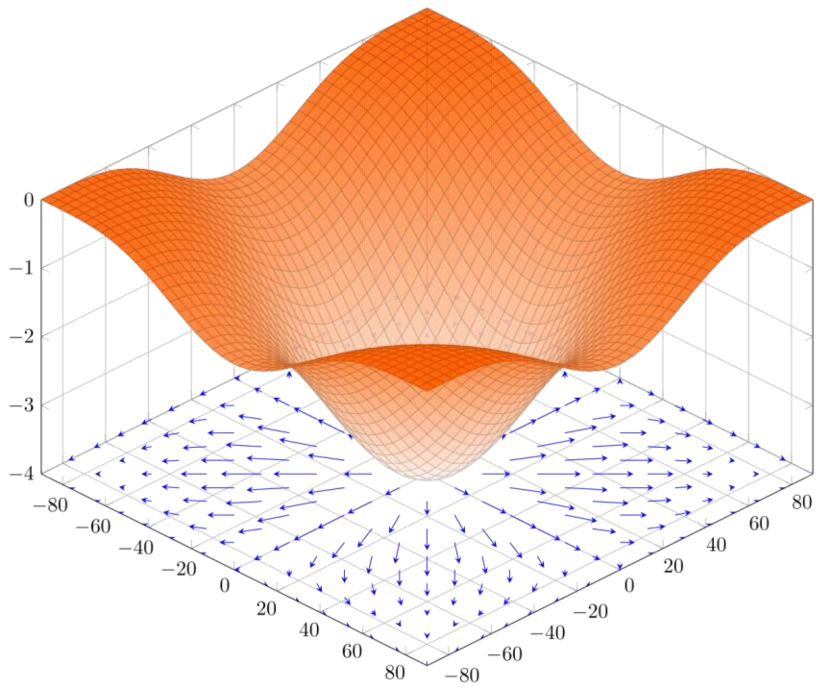
$$\boldsymbol{\nabla} I(r,c) = \begin{bmatrix} \nabla I_x(r,c) \\ \nabla I_y\,(r,c) \end{bmatrix}$$

The Gradient Orientation

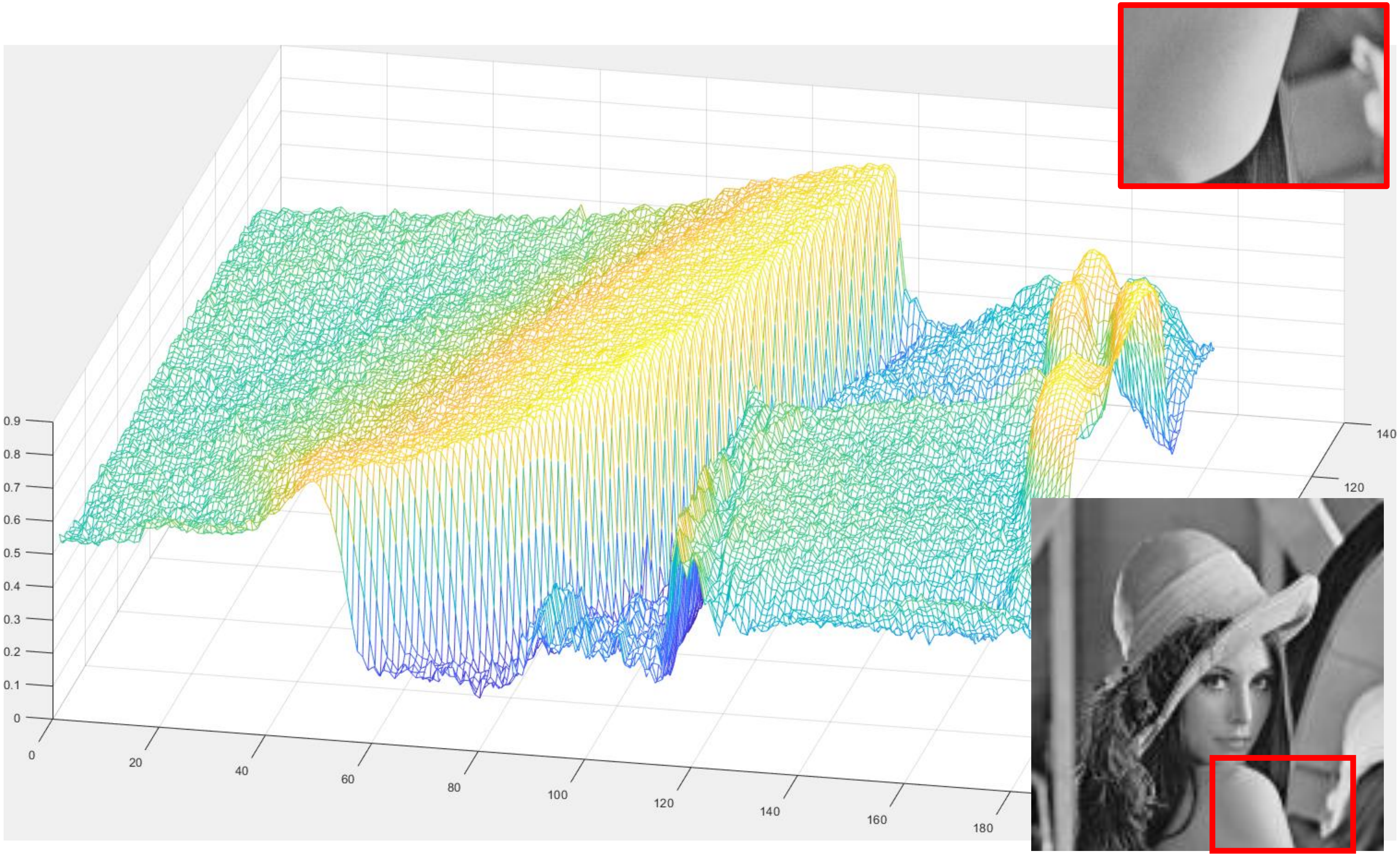
Like for continuous function, the gradient in each pixel points at the **steepest growth/decrease direction.**

$$\angle \nabla I(r,c) = \operatorname{atand}\left(\frac{\nabla I_y(r,c)}{\nabla I_x(r,c)}\right) = \operatorname{atand}\left(\frac{\left(I \circledast d_y\right)(r,c)}{\left(I \circledast d_x\right)(r,c)}\right)$$

The gradient norm indicates how strong is the intensity variation in the gradient direction



Think of an image as a 2d, real-valued function



The Image Gradient

Image Gradient is the gradient of a real-valued 2D function

$$\nabla I(r,c) = \begin{bmatrix} I \circledast d_x \\ I \circledast d_y \end{bmatrix}(r,c)$$

where principal derivatives are computed through convolution against the derivative filters (e.g. Prewitt)

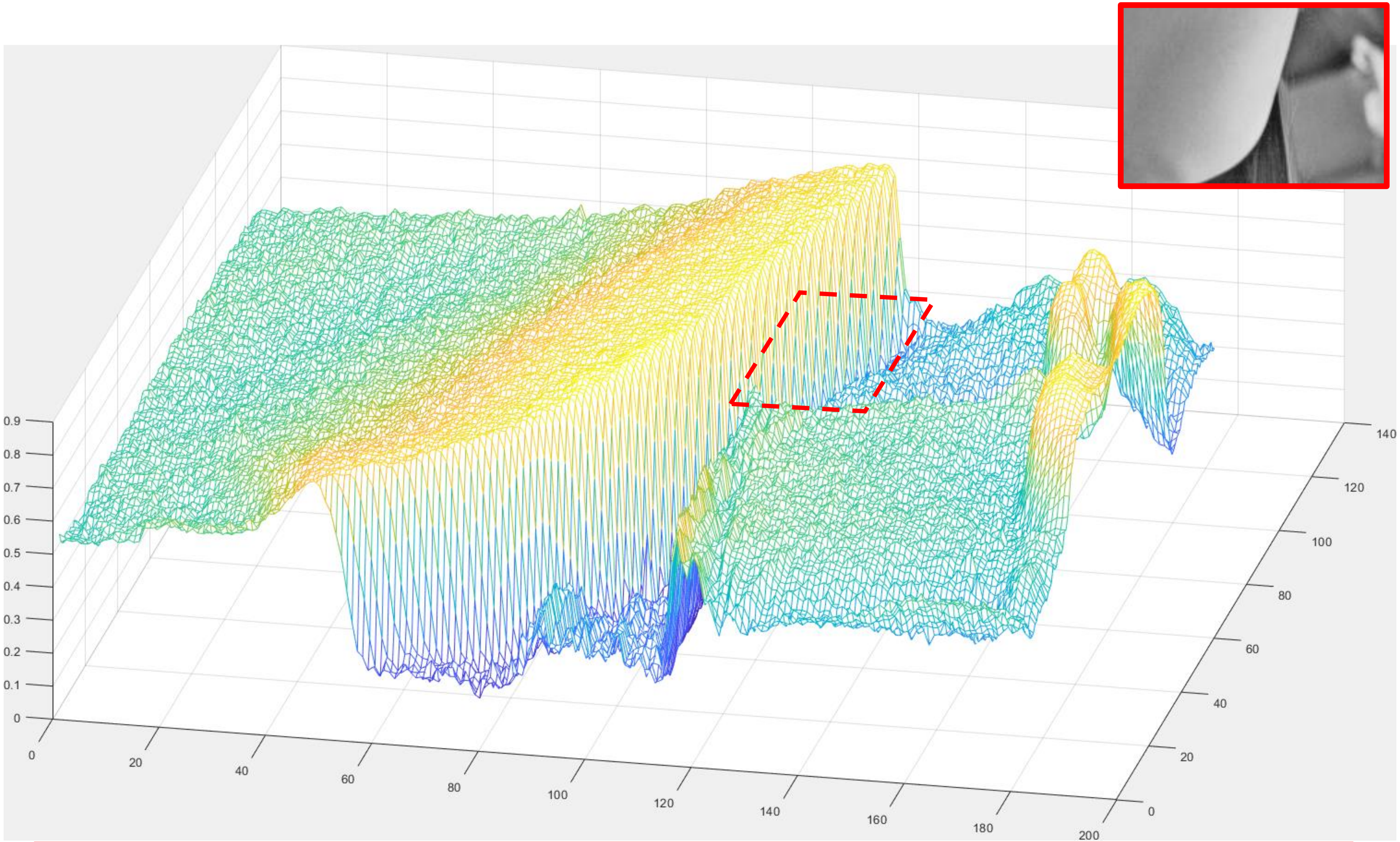
$$dx = \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, \qquad dy = dx'$$

Image gradient behaves like the gradient of a function:

$|\nabla I(r,c)|$ is large where there are large variations

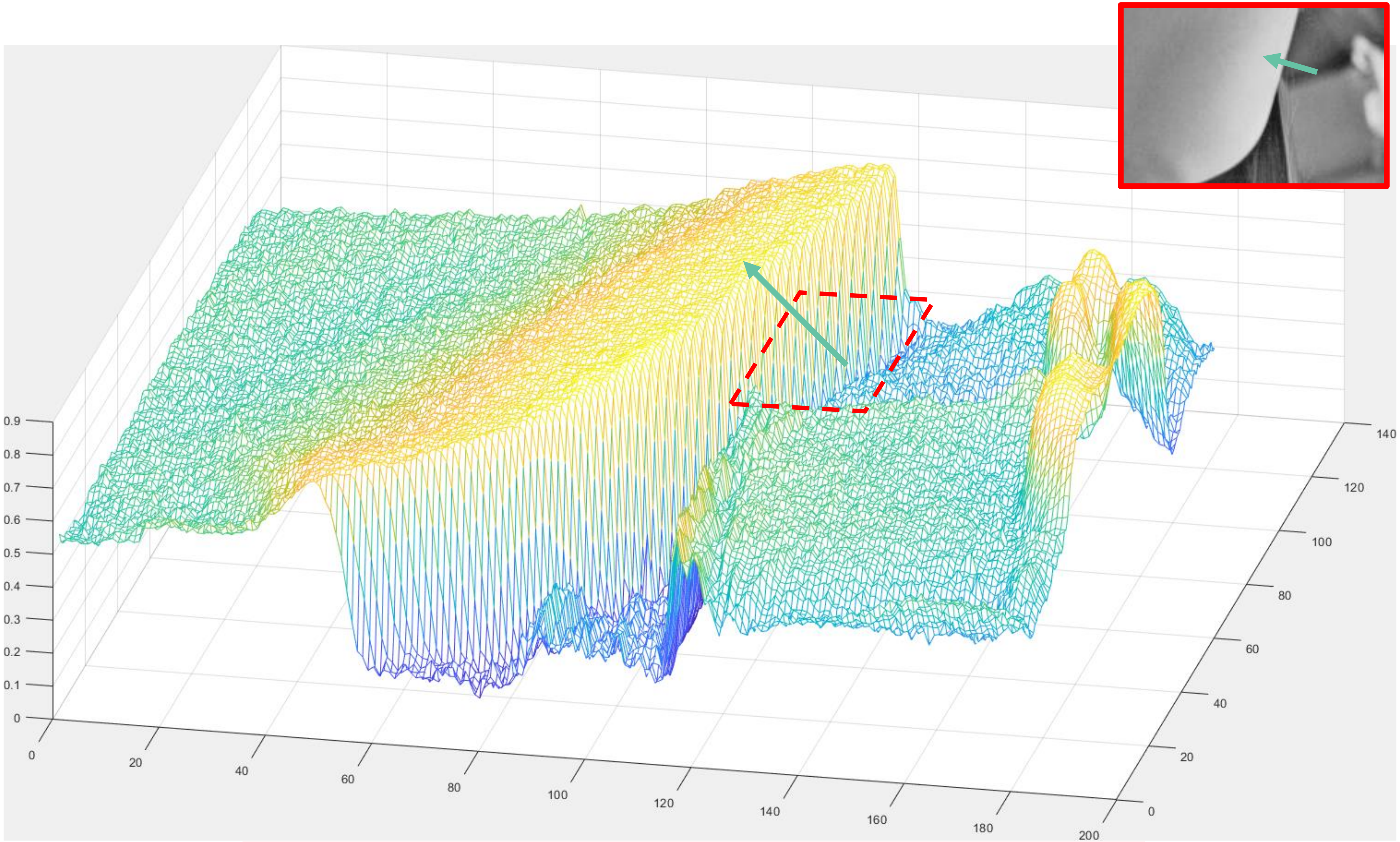
$\angle \nabla I(r,c)$ is the direction of the steepest variation

Think of an image as a 2d, real-valued function



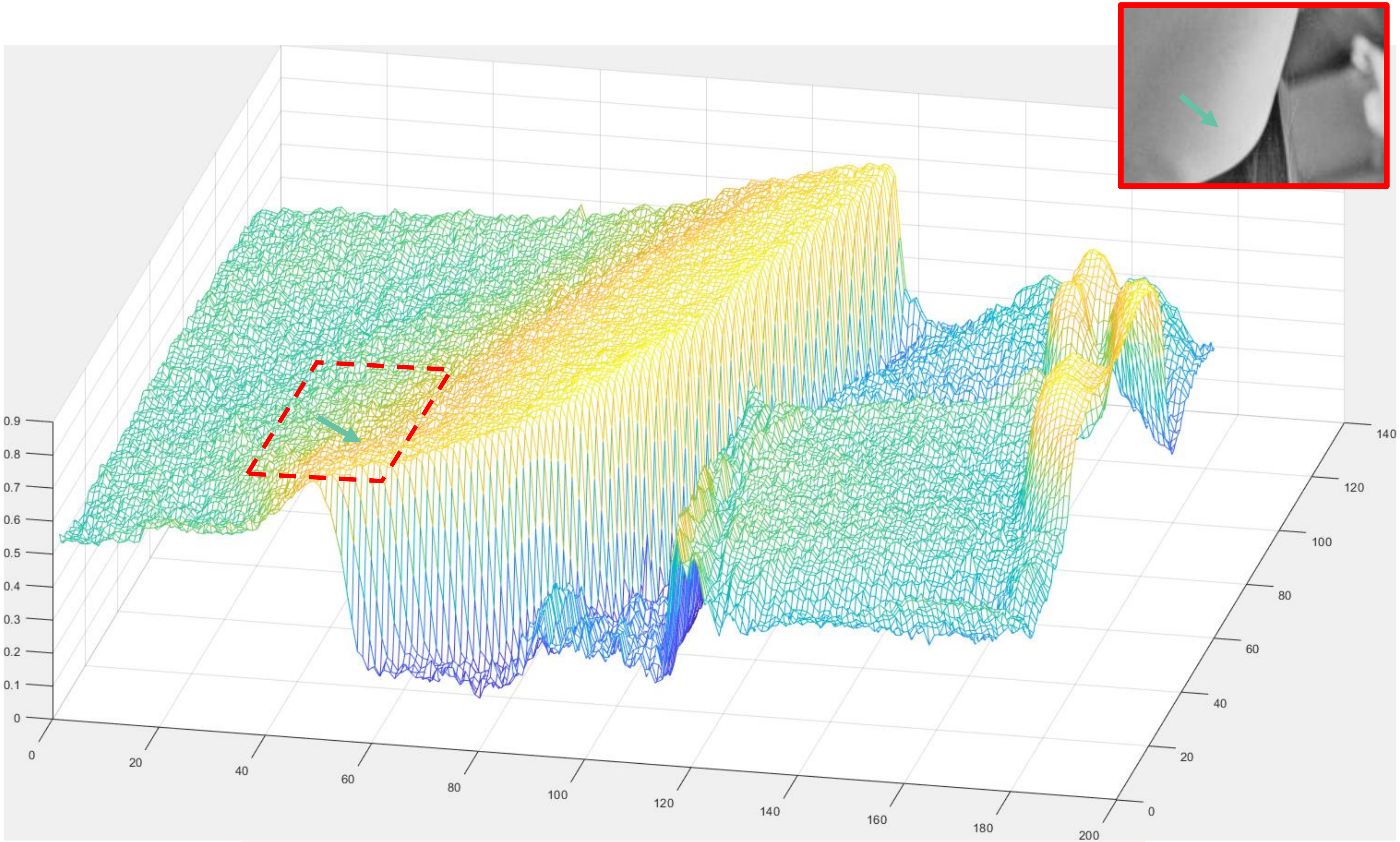
Local spatial transformations are defined over neighborhood like this

Think of an image as a 2d, real-valued function



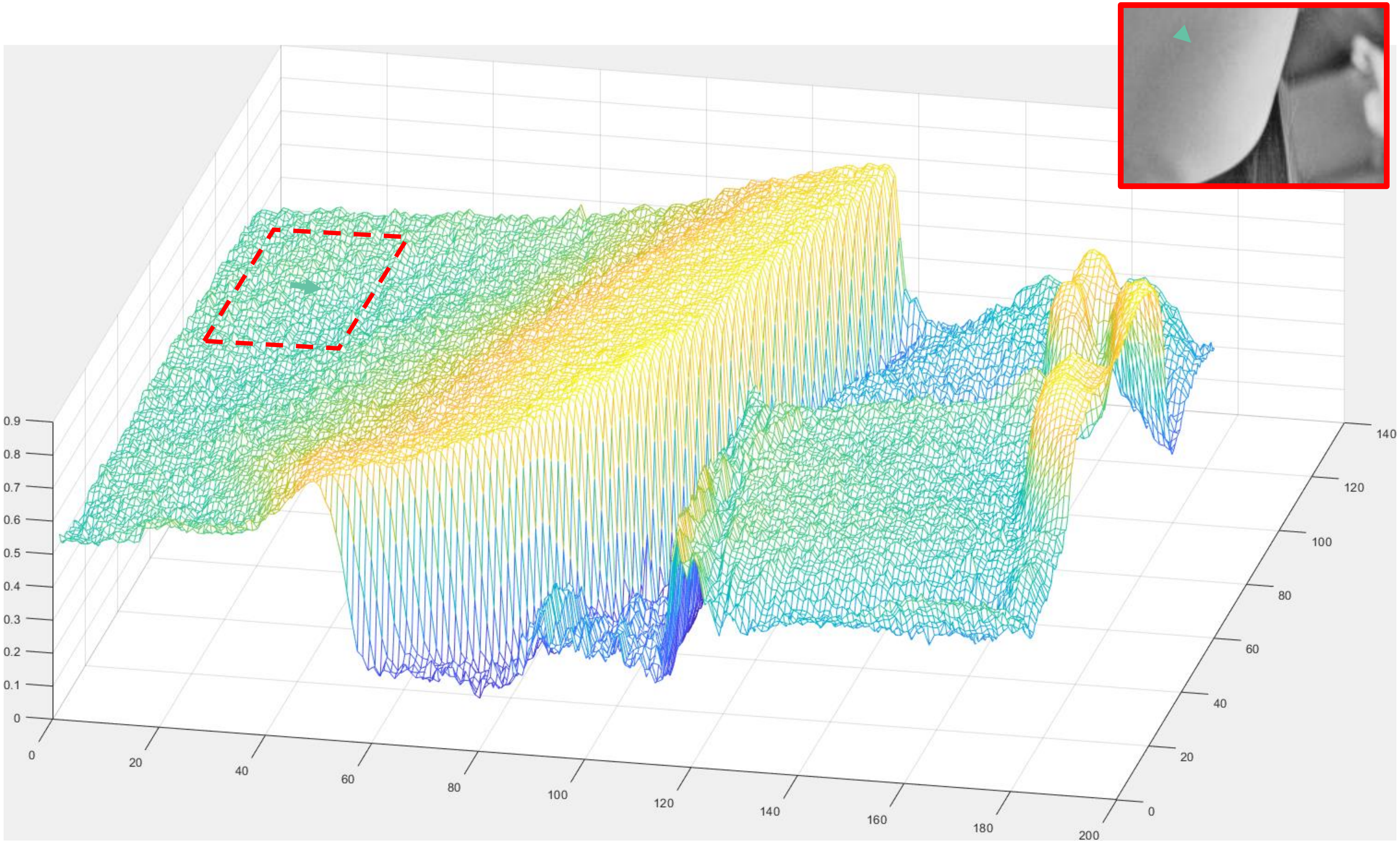
What about the gradient in this neighborhood?

Think of an image as a 2d, real-valued function



What about the gradient in this neighborhood?

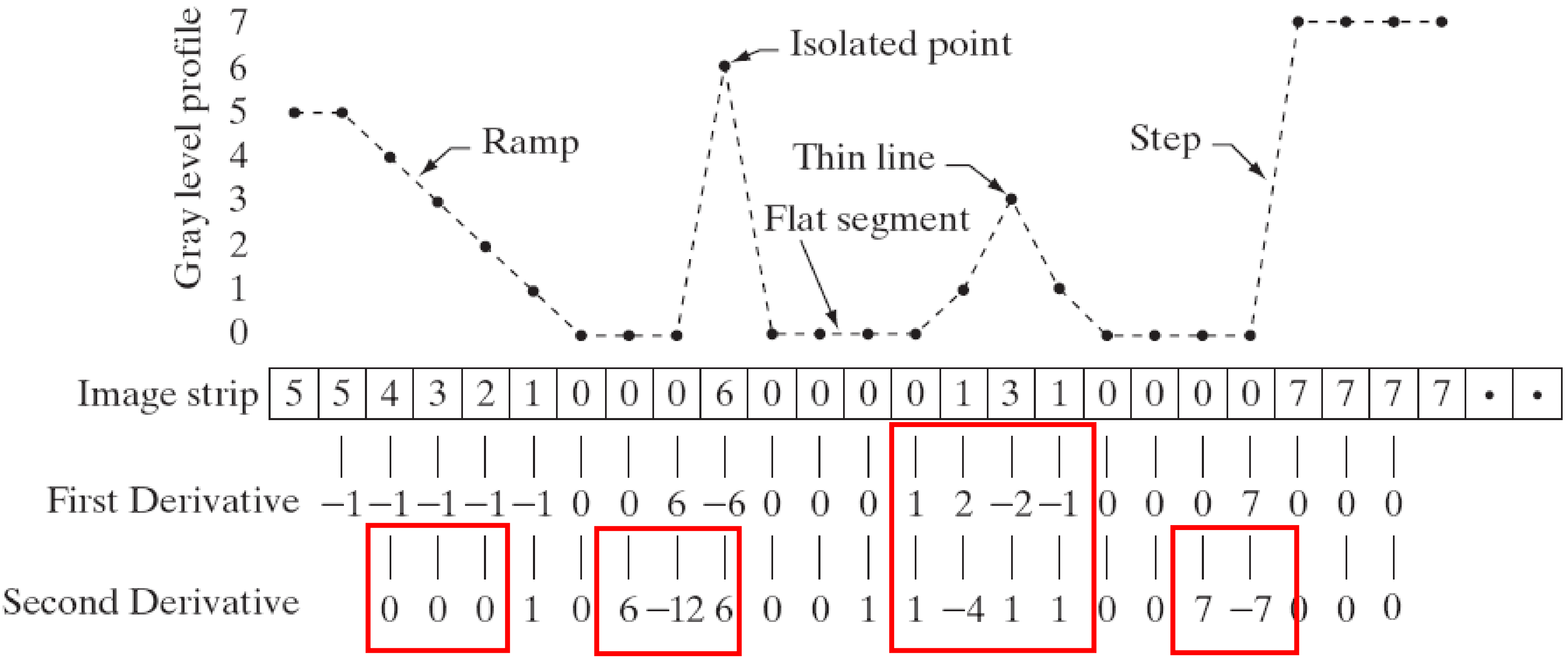
Think of an image as a 2d, real-valued function



Higher Order Derivatives

Derivatives

Derivatives are used to highlight intensity discontinuities in an image and to deemphasize regions with slowly varying intensity levels



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

Second Order Derivatives

The Laplacian of the second order derivative is defined as

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

where

$$\frac{\partial^2 I}{\partial x^2} = I(x+1, y) + I(x-1, y) - 2I(x, y)$$

$$\frac{\partial^2 I}{\partial y^2} = I(x, y-1) + I(x, y+1) - 2I(x, y)$$

Thus,

$$\nabla^2 I = I(x+1, y) + I(x-1, y) + I(x, y-1) + I(x, y+1) - 4I(x, y)$$

It’s a linear operator -> it can be implemented as a convolution

TODO: prove that the second order derivatve is like this

Filter for Digital Laplacian

The Laplacian of the second order derivative is defined as

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

0	1	0
1	-4	1
0	1	0

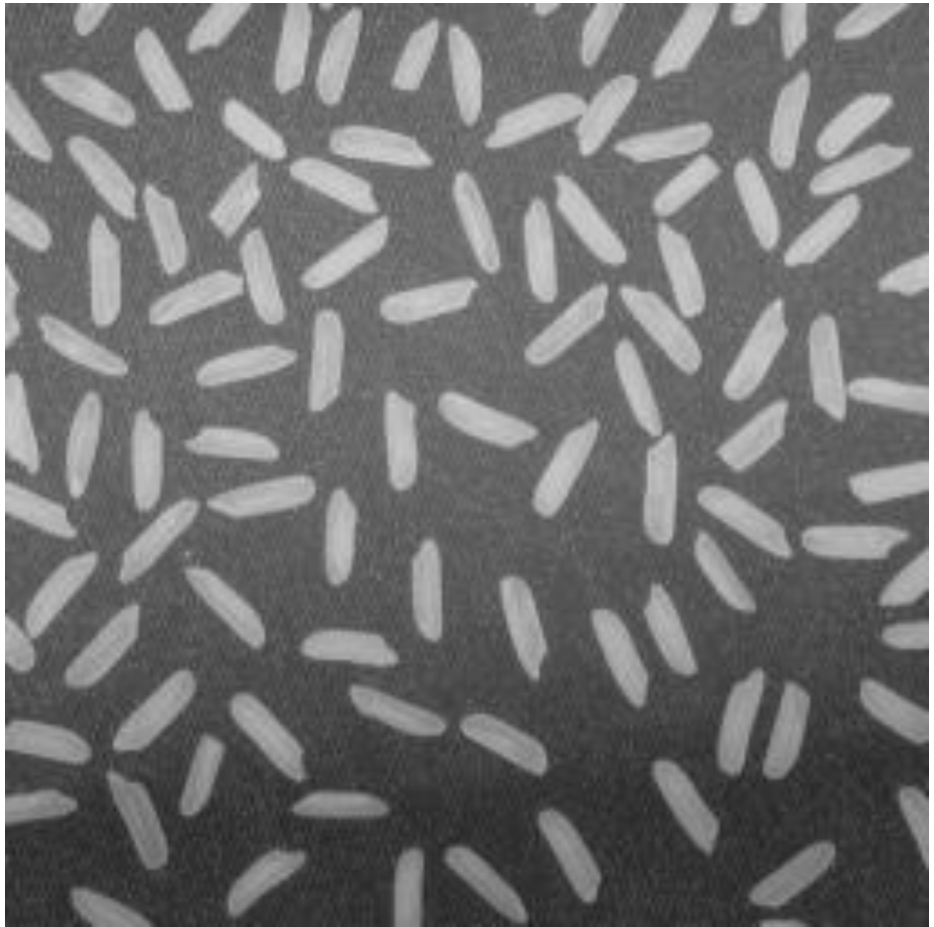
Standard definition, inviariant to 90° rotation

1	1	1
1	-8	1
1	1	1

Alternative definition, inviariant to 45° rotation

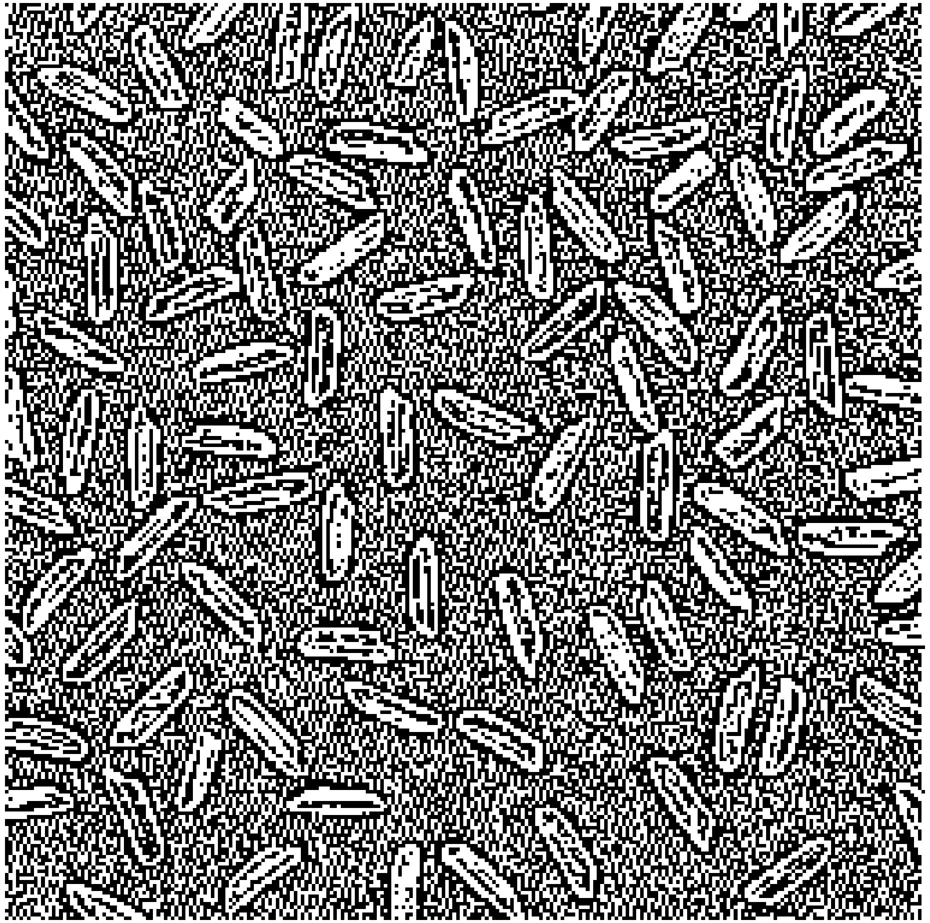
The Laplacian: Image Sharpening

The Laplacian of an image have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background.



The Laplacian: Image Sharpening

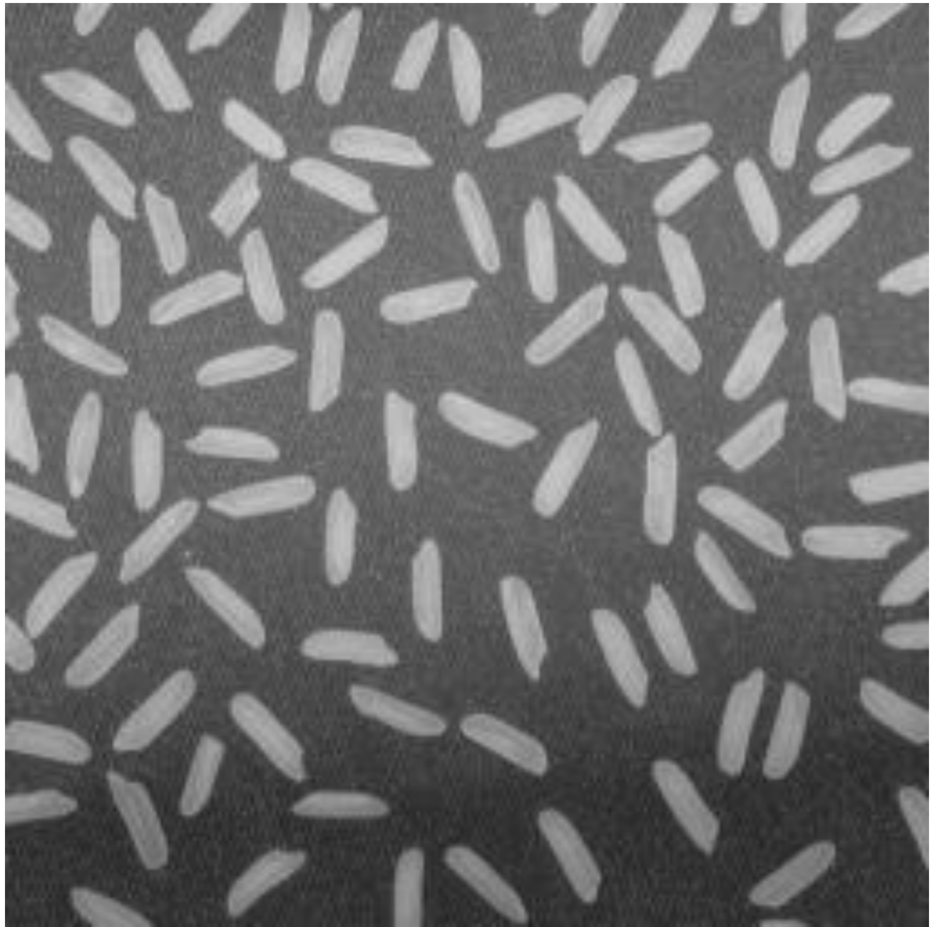
The Laplacian of an image have grayish edge lines and other discontinuities, all superimposed on a dark, featureless background.



The Laplacian: Image Sharpening

Background features can be "recovered" simply by adding the Laplacian image to the original (provided suitable rescaling)

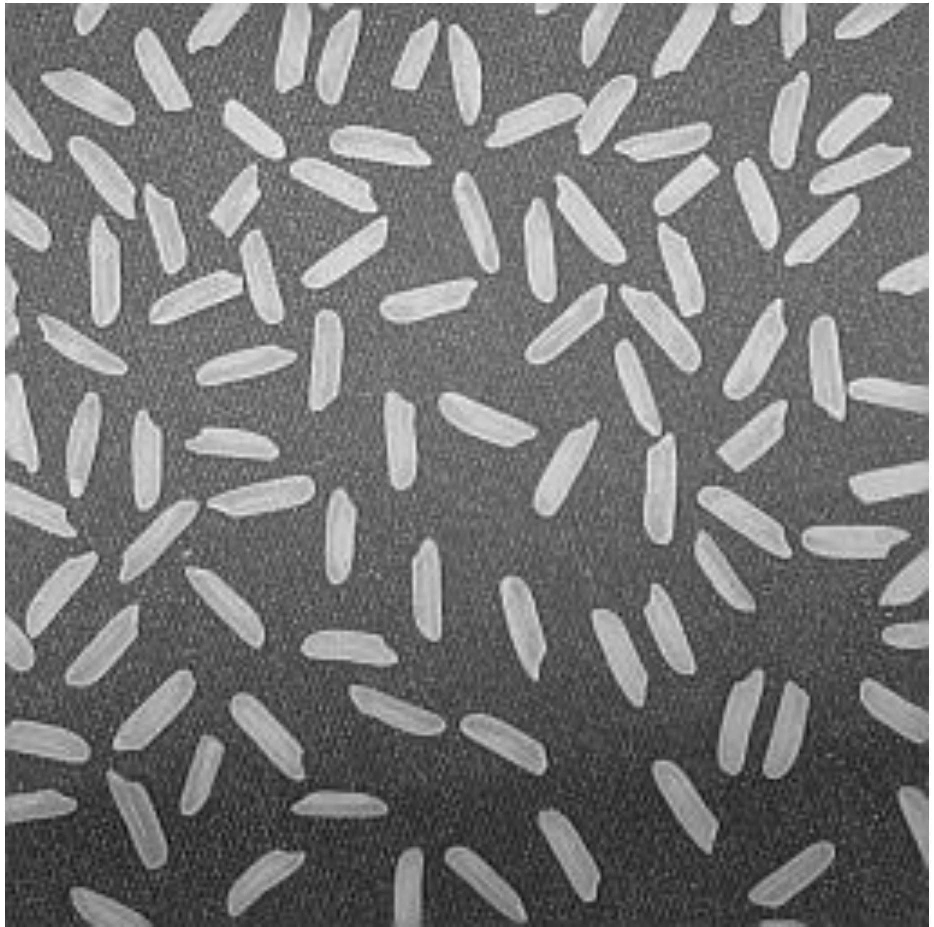
$$G(r,c) = I(r,c) + k[\nabla^2 I(r,c)]$$



The Laplacian: Image Sharpening

Background features can be "recovered" simply by adding the Laplacian image to the original (provided suitable rescaling)

$$G(r,c) = I(r,c) + k[\nabla^2 I(r,c)]$$

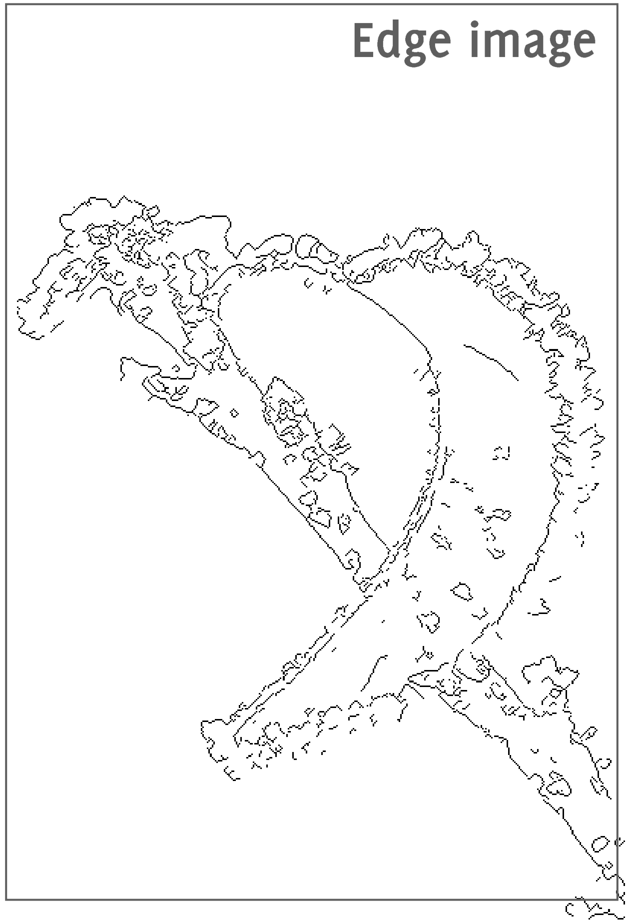


Edges in Images

Edge Detection in Images

Goal: **Automatically** find the contour of objects in a scene.

What For: Edges are significant for scene understanding, enhancement compression…

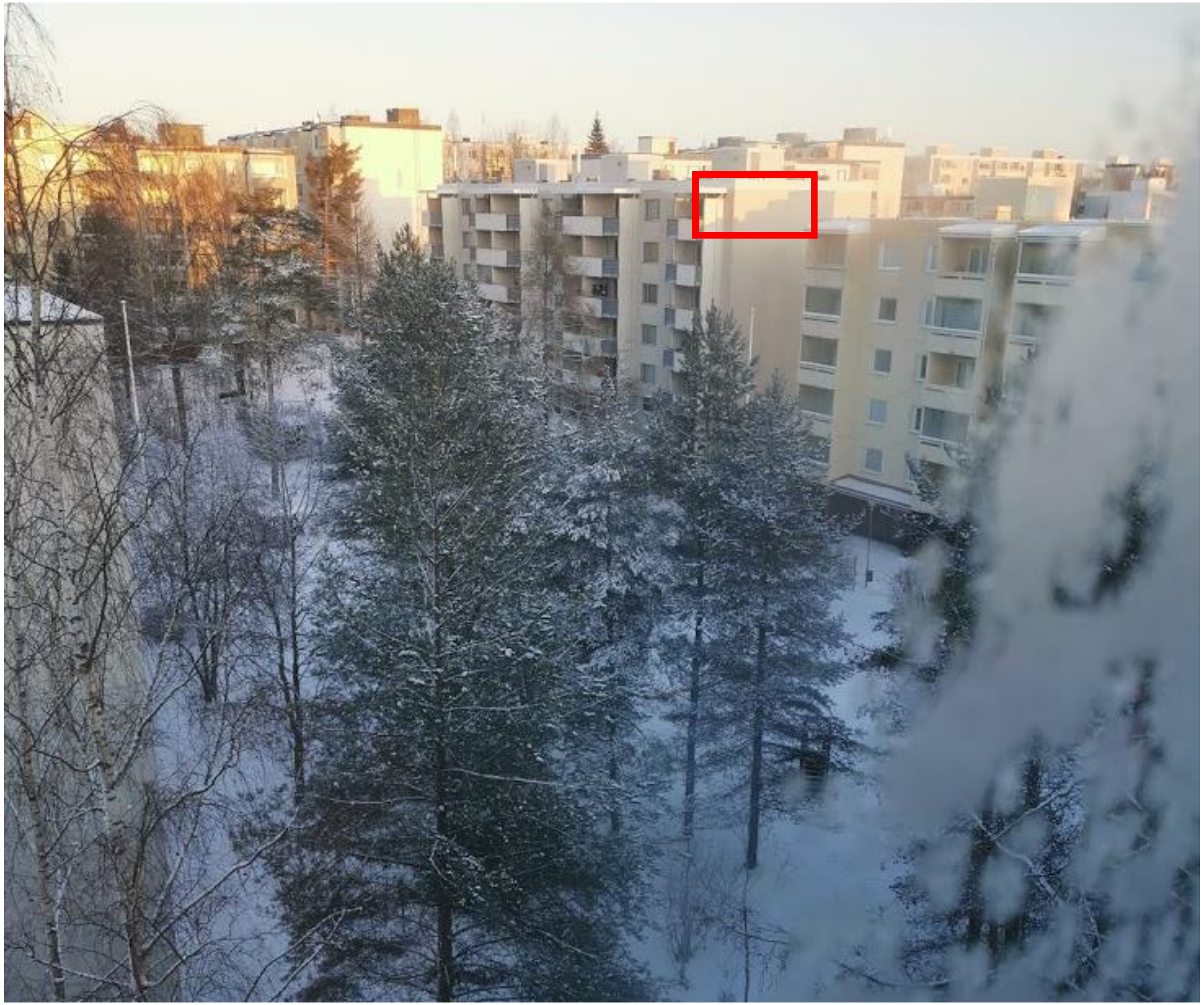


Typically the edge mask is «flipped», 1 at edges and 0 elsewhere

Edges in Images

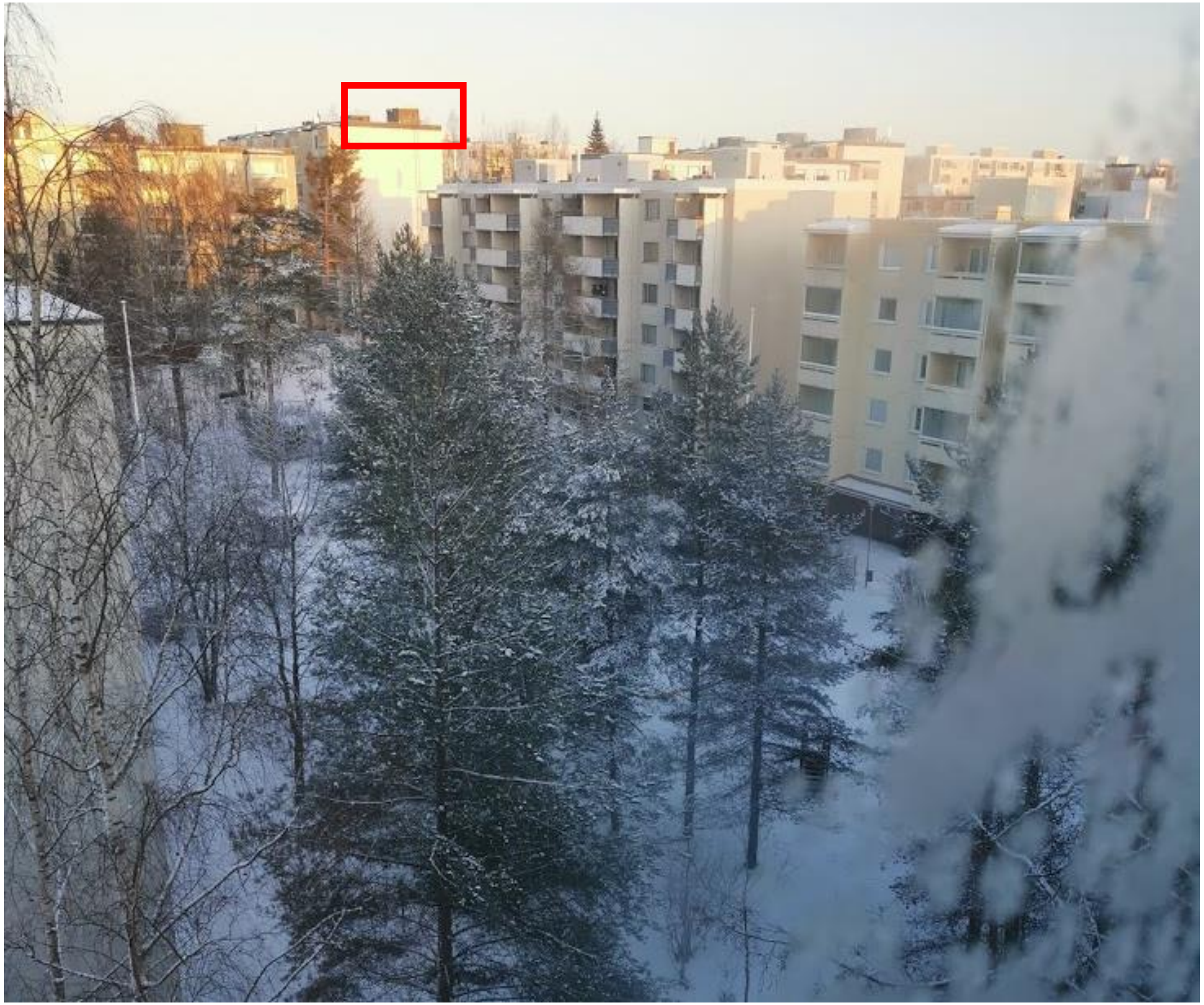
Depth discontinuities

Edges in Images

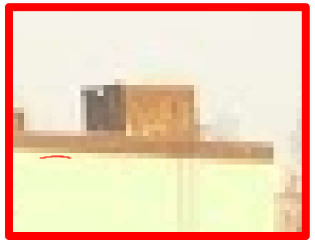


Shadows

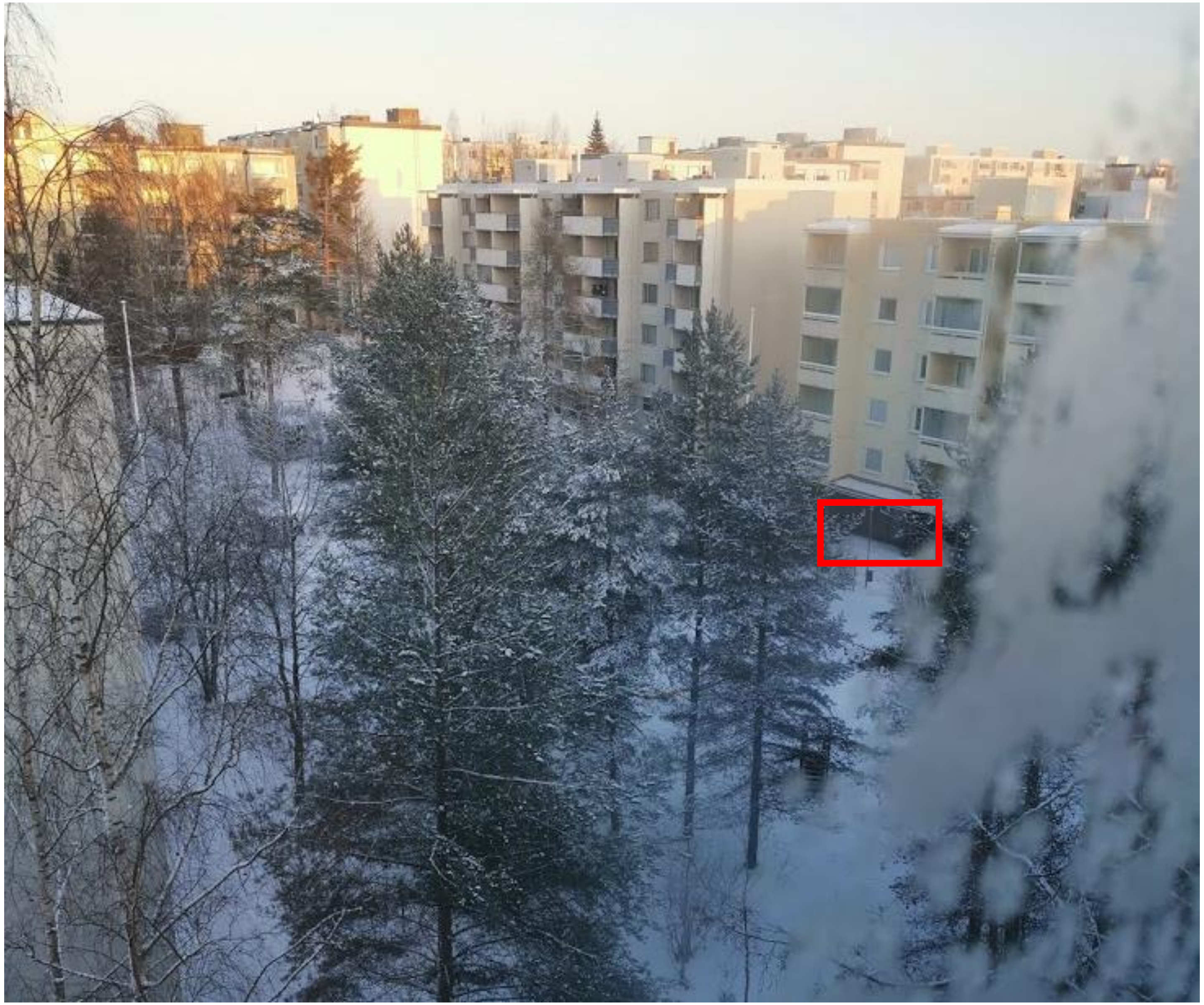
Edges in Images



Discontinuities in the surface color, Color changes



Edges in Images



Discontinuities in the surface normal

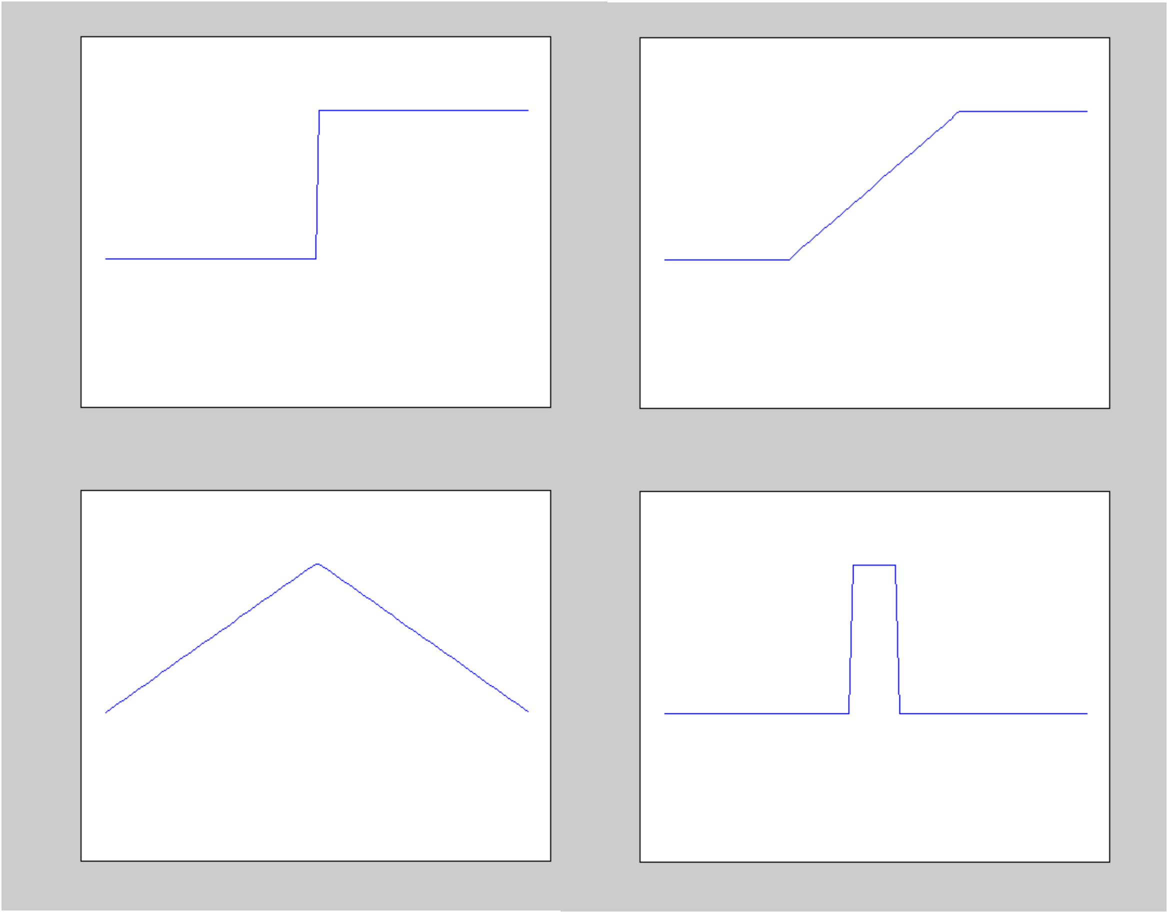
What is an Edge

Lets define an edge to be a **discontinuity** in image intensity function.

Several Models

- Step Edge
- Ramp Edge
- Roof Edge
- Spike Edge

They can be thus detected as **discontinuities** of image **Derivatives**



Edge Detection

Gradient Magnitude and edge detectors

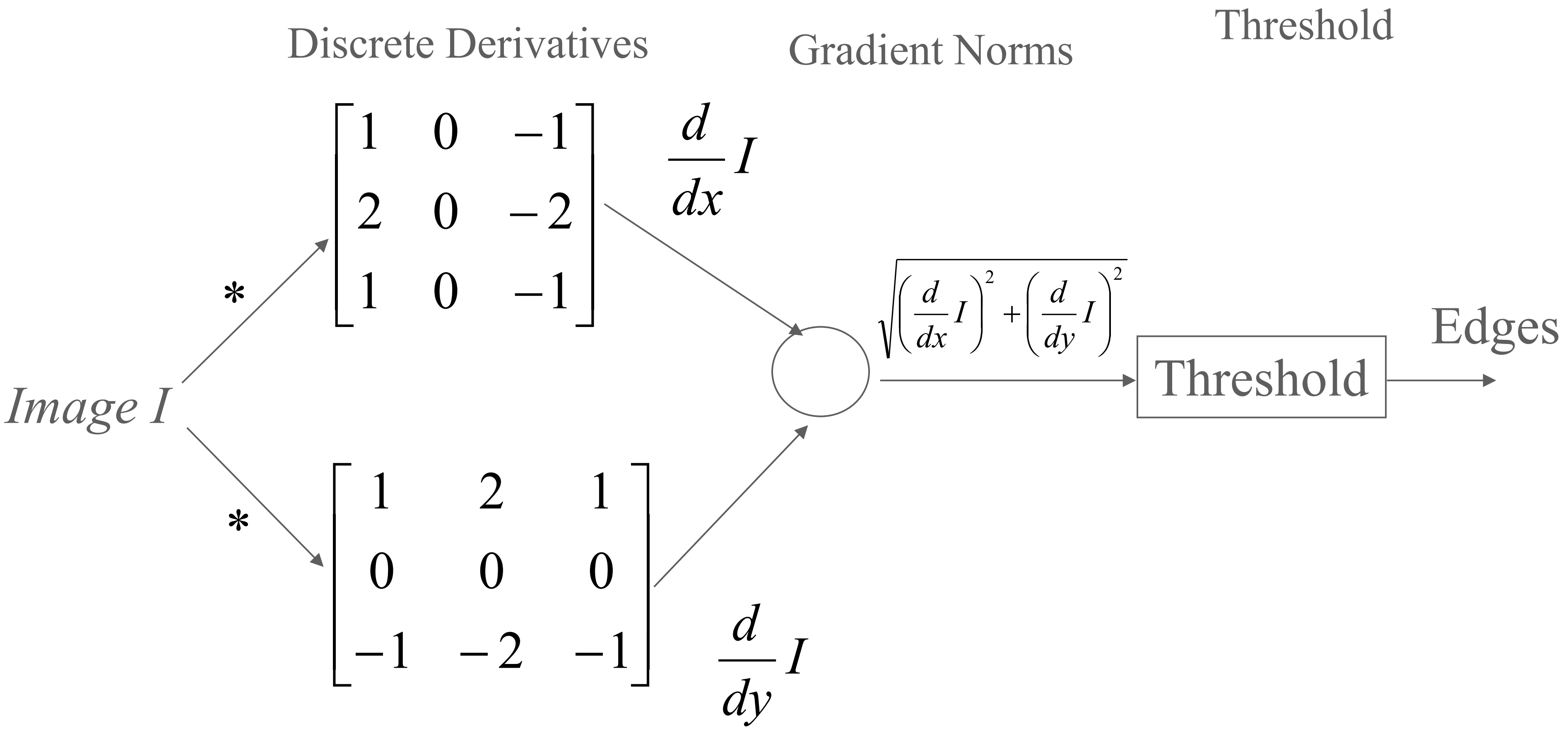
Gradient Magnitute is not a binary image

We can see edges but we cannot identify them, yet

$$||\nabla I|| = \sqrt{(I \circledast d_x)^2 + \left(I \circledast d_y\right)^2}$$

Detecting Edges in Image

Sobel Edge Detector

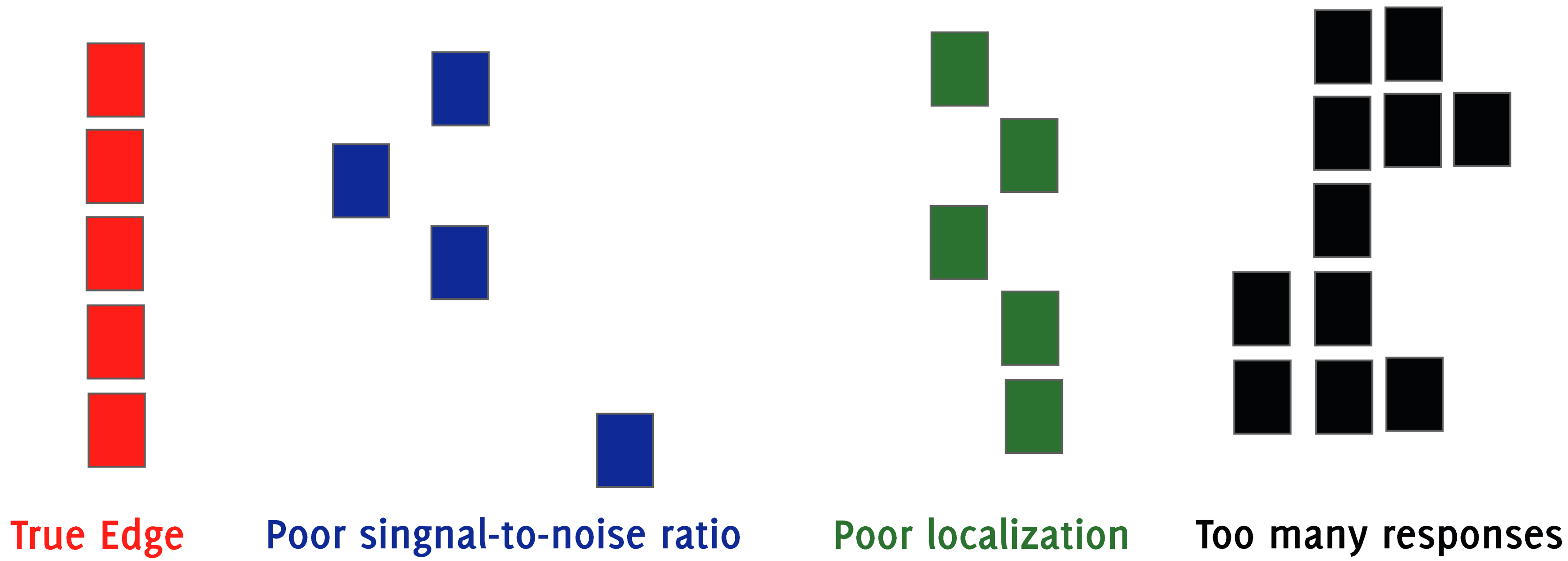


Canny Edge Detector Criteria

Good Detection: The optimal detector must minimize the probability of false positives as well as false negatives.

Good Localization: The edges detected must be as close as possible to the true edges.

Single Response Constraint: The detector must return one point only for each edge point. similar to good detection but requires an ad-hoc formulation to get rid of multiple responses to a single edge



Canny Edge Detector

It is characterized by 3 important steps

- Convolution with smoothing Gaussian filter before computing image derivatives
- Non-maximum Suppression
- Hysteresis Thresholding

J. Canny "A Computational Approach to Edge Detection" IEEE PAMI vol 8, no. 6, Nov. 1986 http://perso.limsi.fr/Individu/vezien/PAPIERS_ACS/canny1986.pdf

Canny Edge Detector

Smooth by Gaussian (smoothing regulated by σ)

$$S = G_\sigma * I \qquad G_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Compute x and y derivatives

$$\Delta S = \left[\frac{\partial}{\partial x} S \quad \frac{\partial}{\partial y} S \right]^T = \left[S_x \quad S_y \right]^T$$

Compute gradient magnitude and orientation

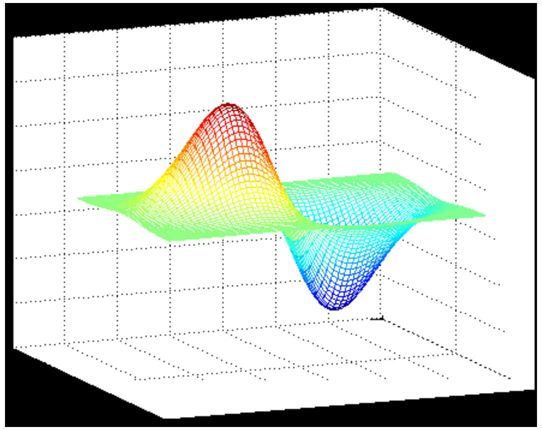
$$|\Delta S| = \sqrt{S_x^2 + S_y^2} \qquad \theta = \tan^{-1} \frac{S_y}{S_x}$$

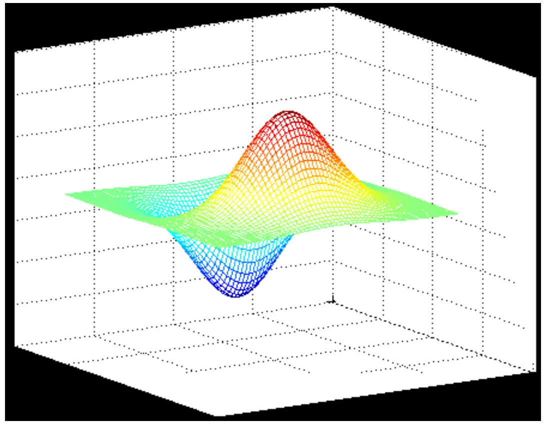
Canny Edge Operator (derivatives)

$$\Delta S = \Delta\left(G_\sigma * I\right) = \Delta G_\sigma * I$$

$$\Delta G_\sigma = \left[\frac{\partial G_\sigma}{\partial x} \quad \frac{\partial G_\sigma}{\partial y}\right]^T$$

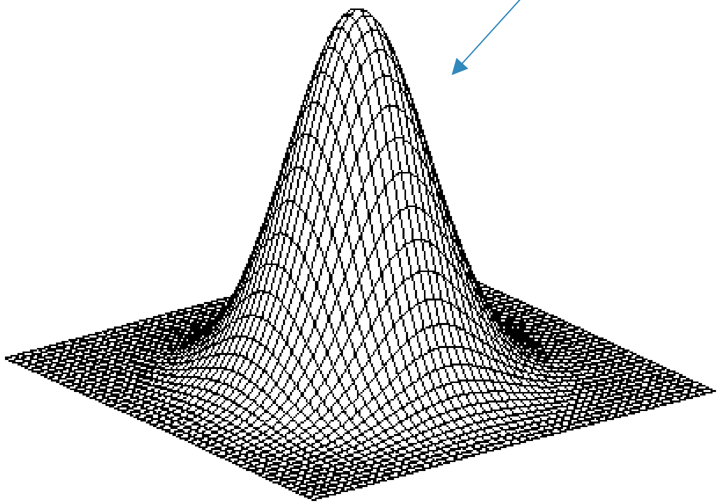
$$\Delta S = \left[\frac{\partial G_\sigma}{\partial x} * I \quad \frac{\partial G_\sigma}{\partial y} * I\right]^T$$



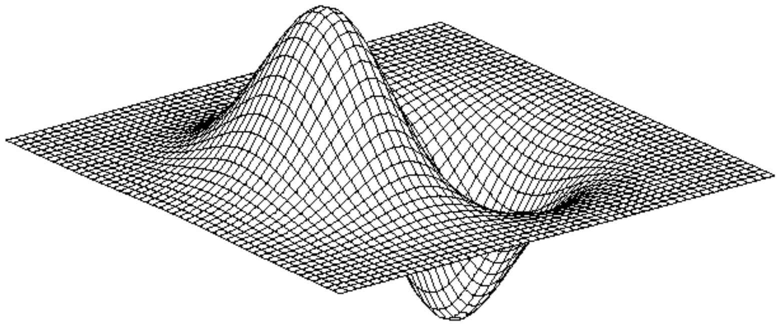


Convolution is associative

$I \otimes (g \otimes dx)$



$* \quad [1 \quad 0 \quad -1] =$

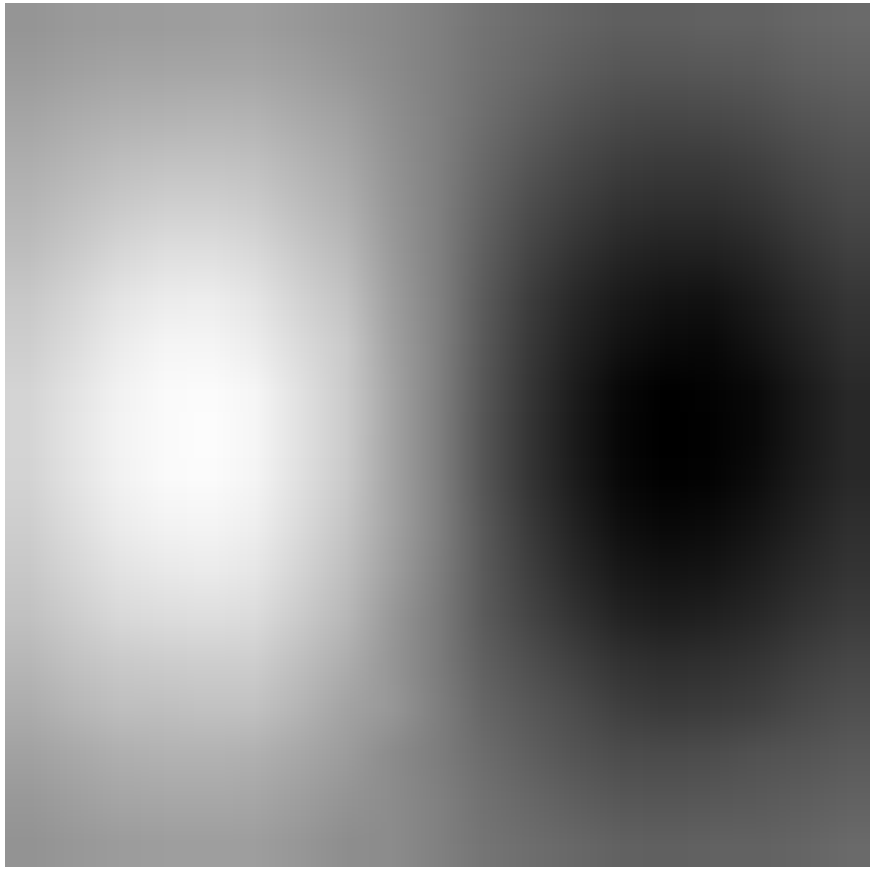


2D-Gaussian

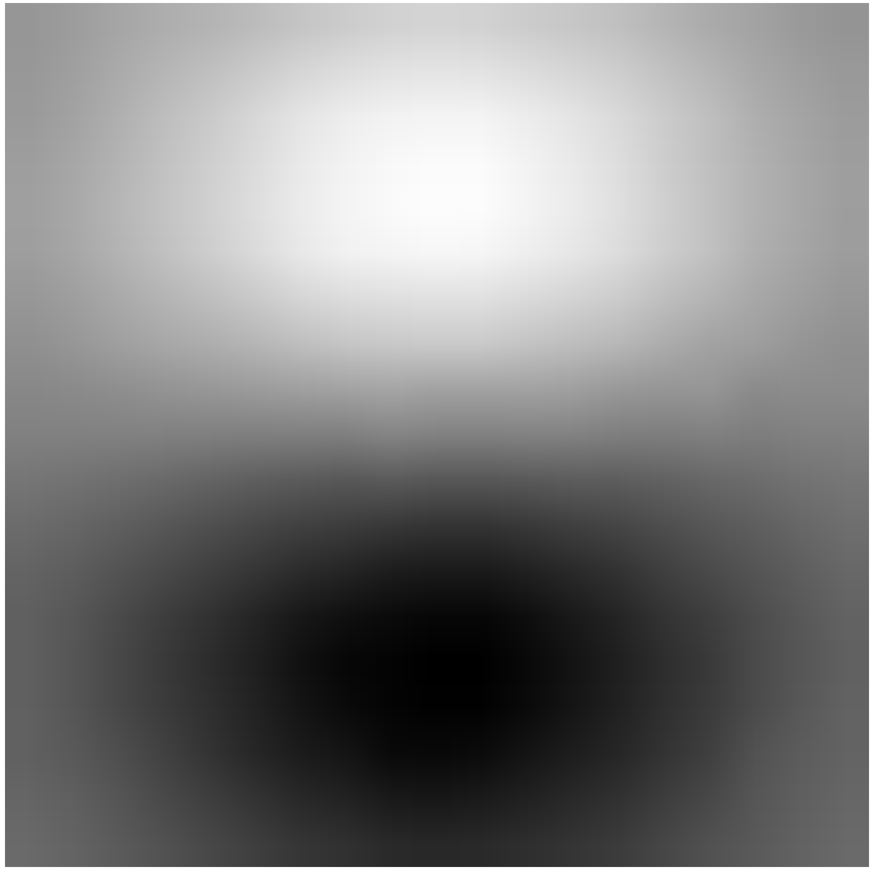
x - derivative

Gaussian Derivative Filters

The amount of smoothing is regulated by a parameter σ



x-direction

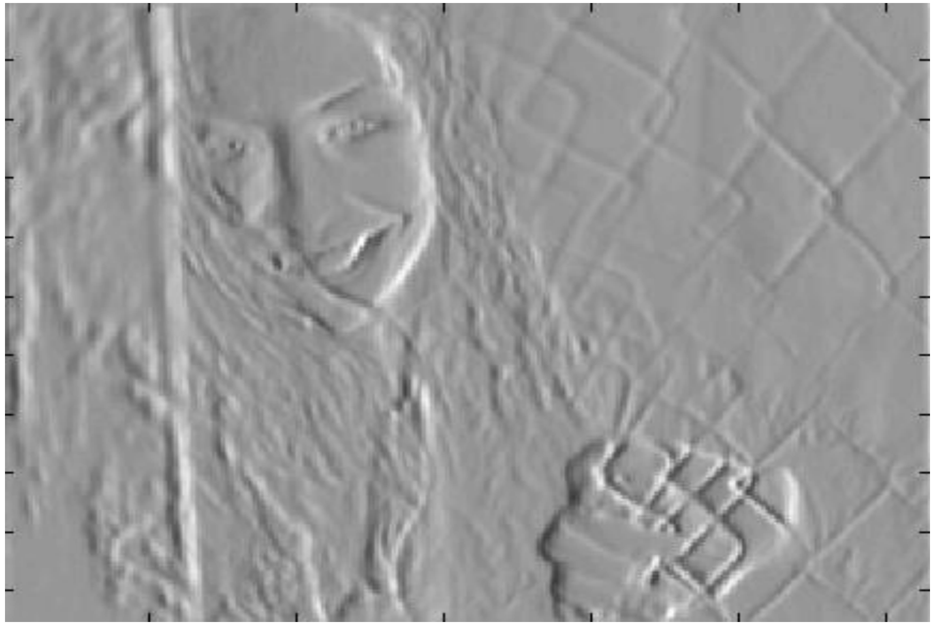


y-direction

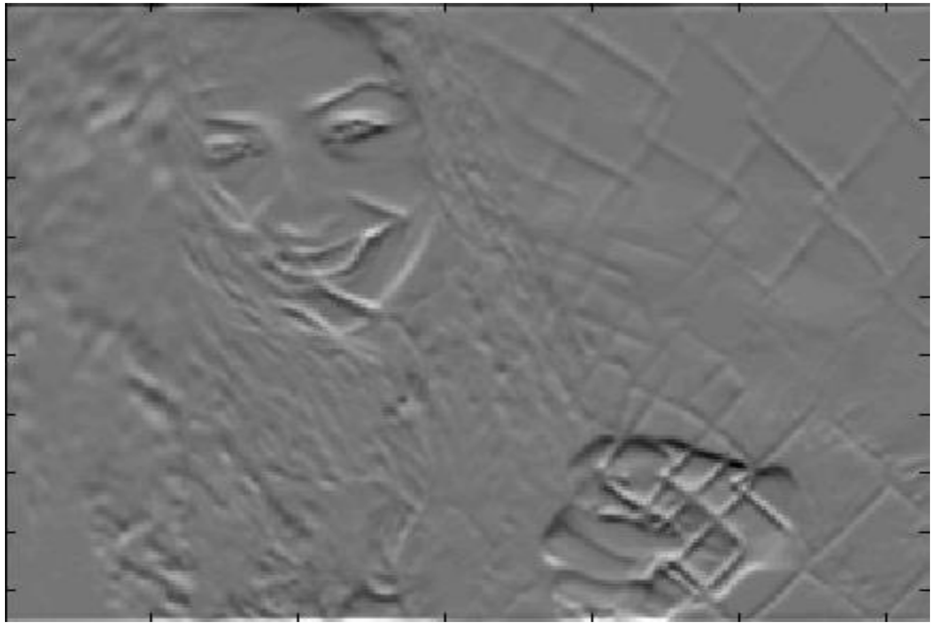
Canny Edge Detector

I

S_x



S_y



Canny Edge Detector

$$|\Delta S| = \sqrt{S_x^2 + S_y^2}$$

Gradient Magnitude

I

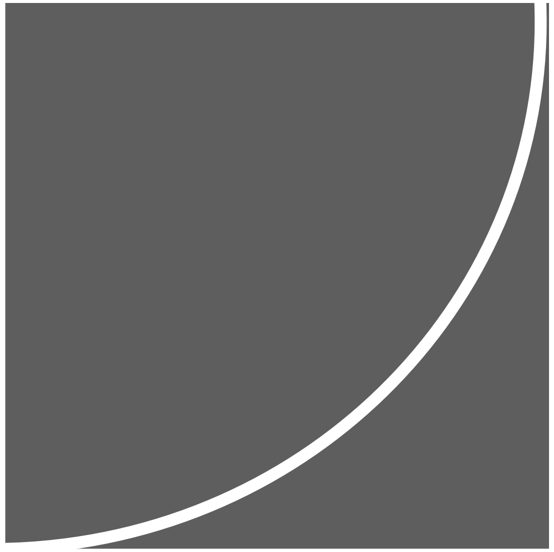
$$|\Delta S| \geq Threshold = 25$$

Thresholded Gradient Magnitude

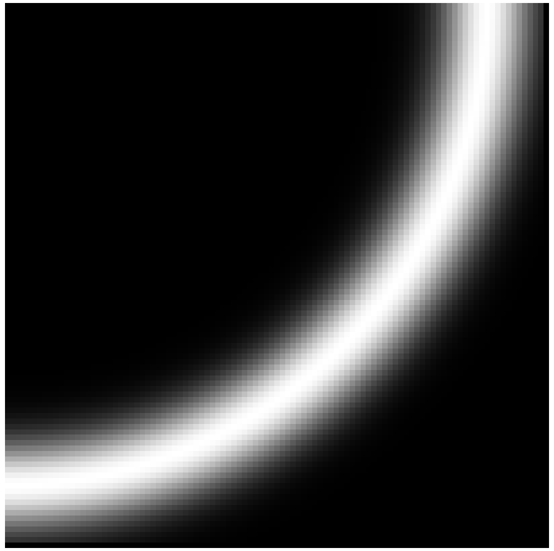
Non-Maximum Suppression: The Idea

We wish to determine the points along the curve where the gradient magnitude is largest.

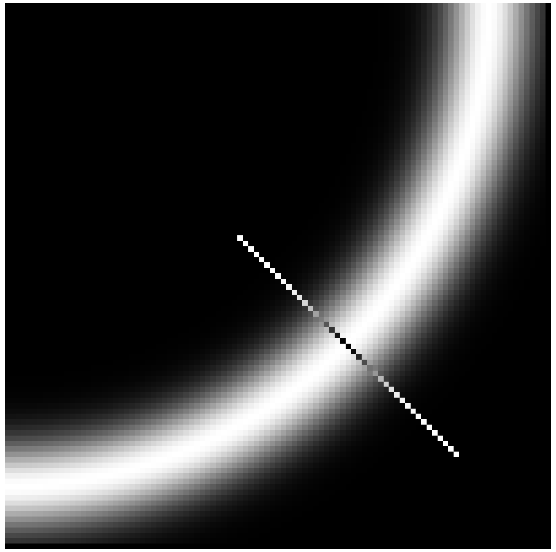
Non-maximum suppression: we look for a maximum along a slice orthogonal to the curve. These points form a 1D signal.



Original Image

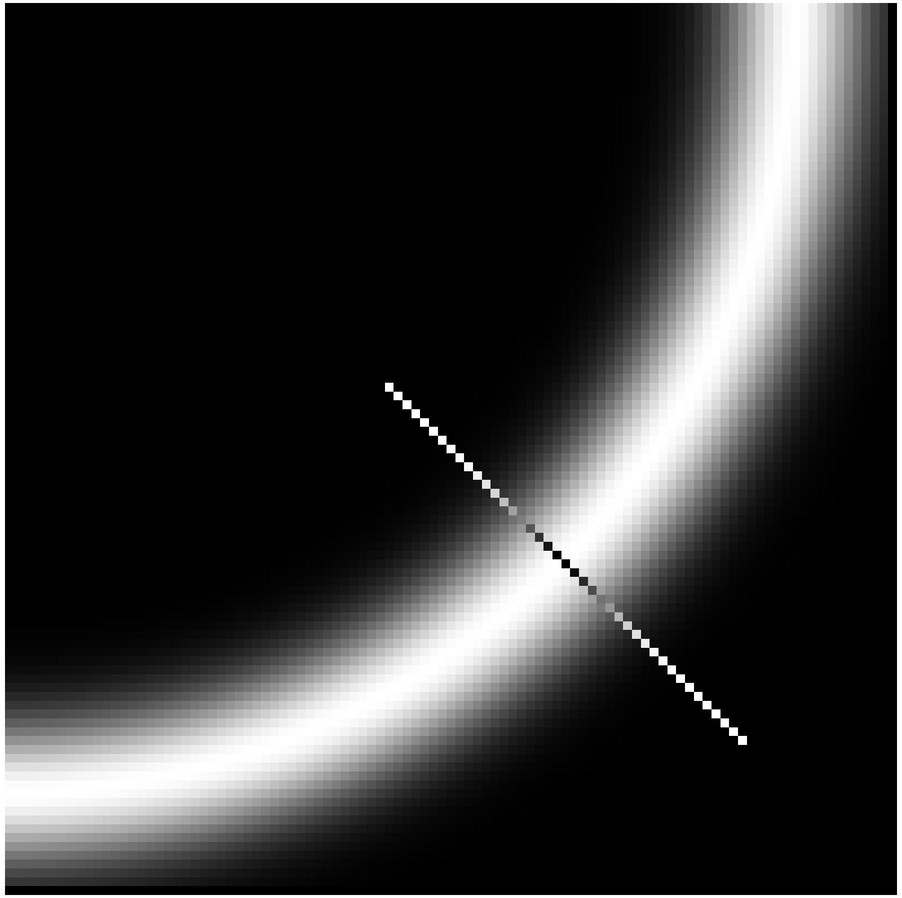


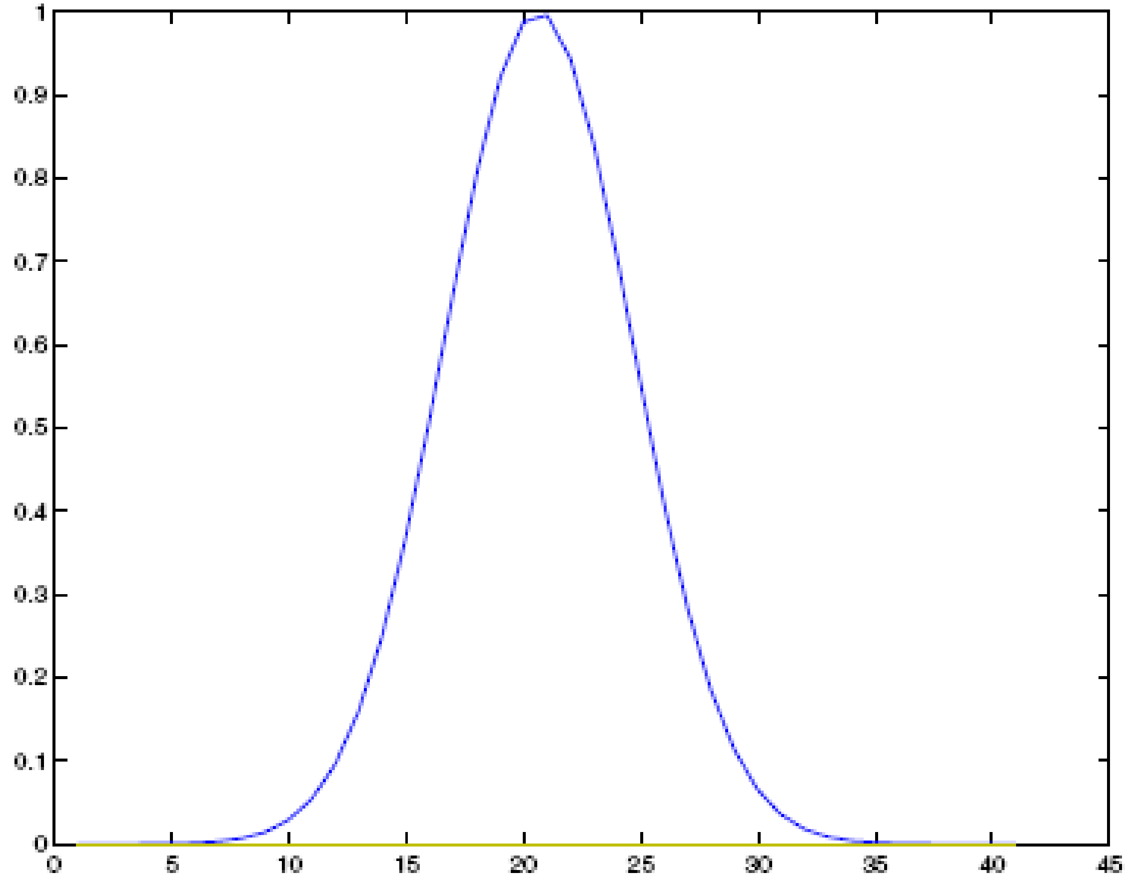
Gradient Magnitude (before thresholding)



Segment orthogonal

Non-Maximum Suppression



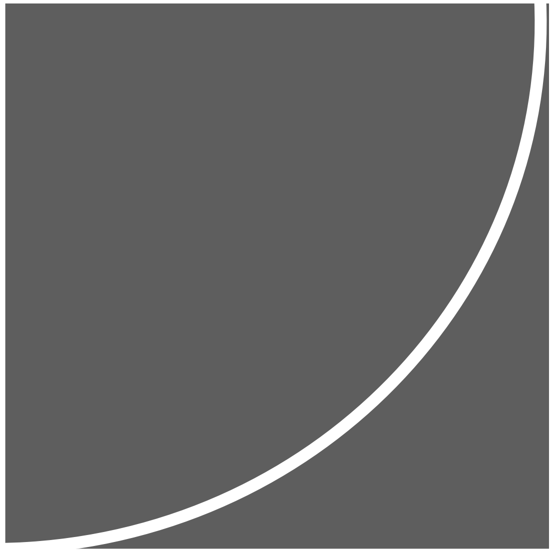


Non-Maximum Suppression: The Idea

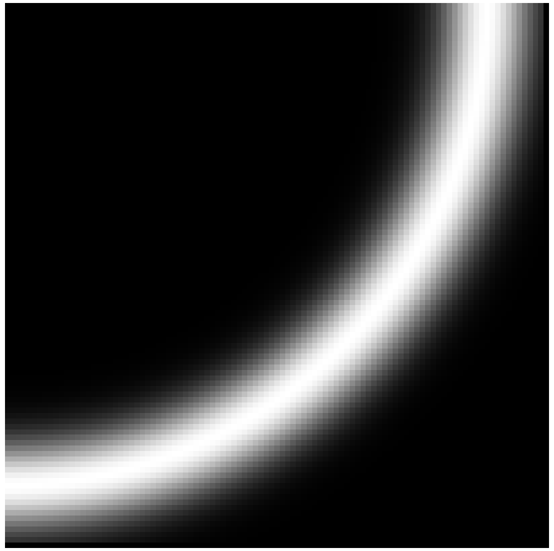
There are two issues:

i. **which slice to select to extract the maximum?**

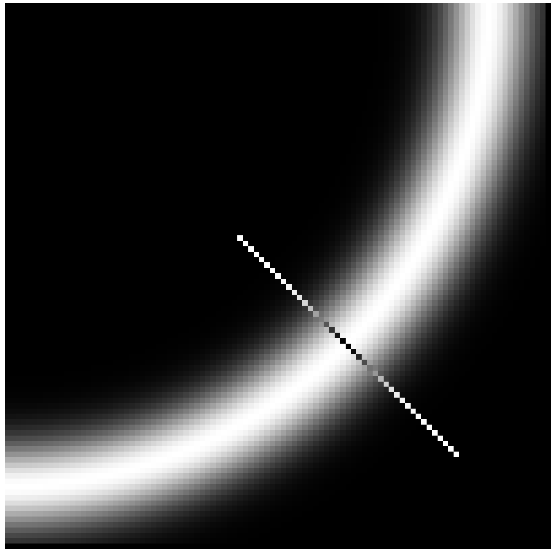
ii. once an edge pixel has been found, **which pixel to test next?**



Original Image

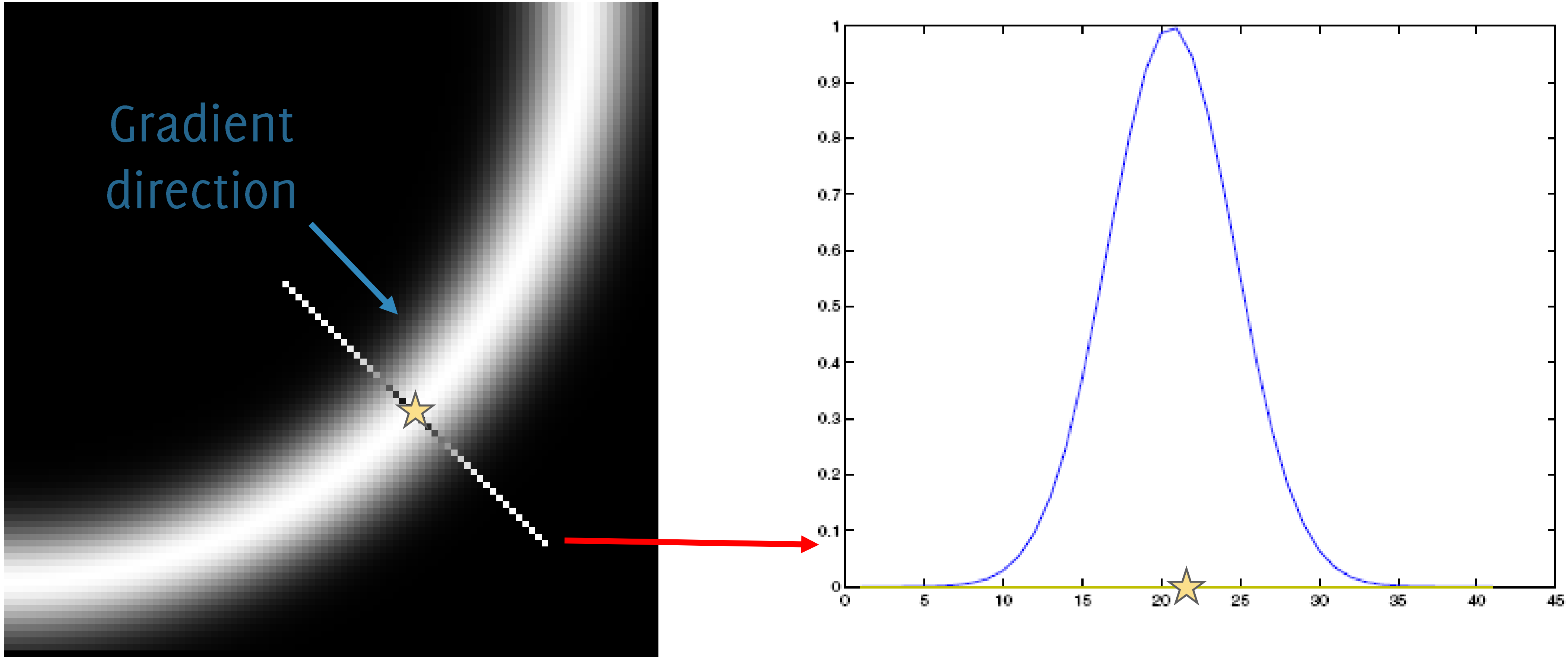


Gradient Magnitude (after thresholding)



Segment orthogonal

Non-Maximum Suppression – Idea (II)



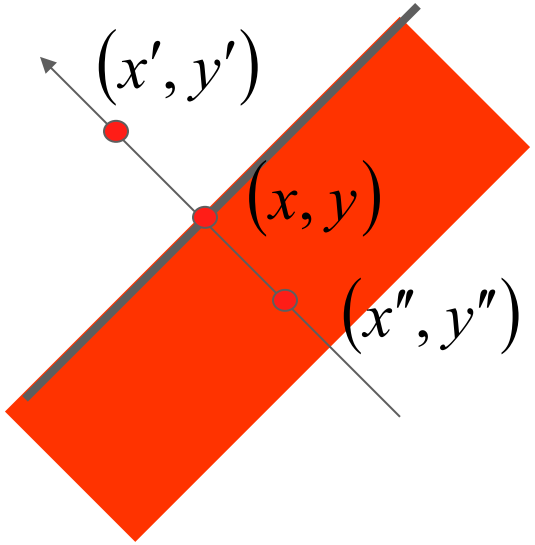
In each pixel, **the gradient indicates the direction of the steepest variation**: thus, the gradient is orthogonal to the edge direction (no variation along the edge). We have to consider pixels on a segment following the gradient direction

The intensity profile along the segment. We can easily identify the location of the maximum.

Non-Maximum Suppression - Threshold

Suppress the pixels in 'Gradient Magnitude Image' which are not local maximum

$$M(x,y)=\begin{cases}|\Delta S|(x,y) & \text{if } |\Delta S|(x,y)>|\Delta S|(x',y') \\ & \&\, |\Delta S|(x,y)>|\Delta S|(x'',y'') \\ 0 & \text{otherwise}\end{cases}$$



(x', y') and (x'', y'') are the neighbors of (x, y) in $|\Delta S|$

These have to be taken on a line along the gradient direction in (x, y)

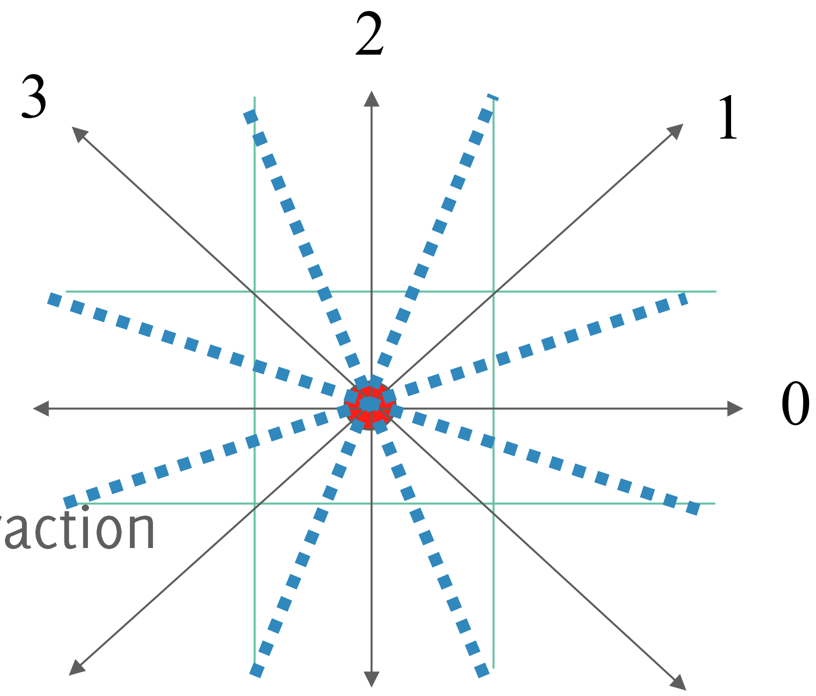
Non-Maximum Suppression: Quantize Gradient Directions

In practice the gradient directions are quantized according to 4 main directions, each covering 45° (orientation is not considered)

- Thus, only diagonal, horizontal, vertical line segments are considered

We consider 4 quantized directions 0,1,2, 3

$$\theta(\boldsymbol{x_0}) = \text{atan}\left(\frac{\partial/_{\partial y}\, I(\boldsymbol{x_0})}{\partial/_{\partial x}\, I(\boldsymbol{x_0})}\right)$$



Orientation is irrelevant since this is meant for segment extraction

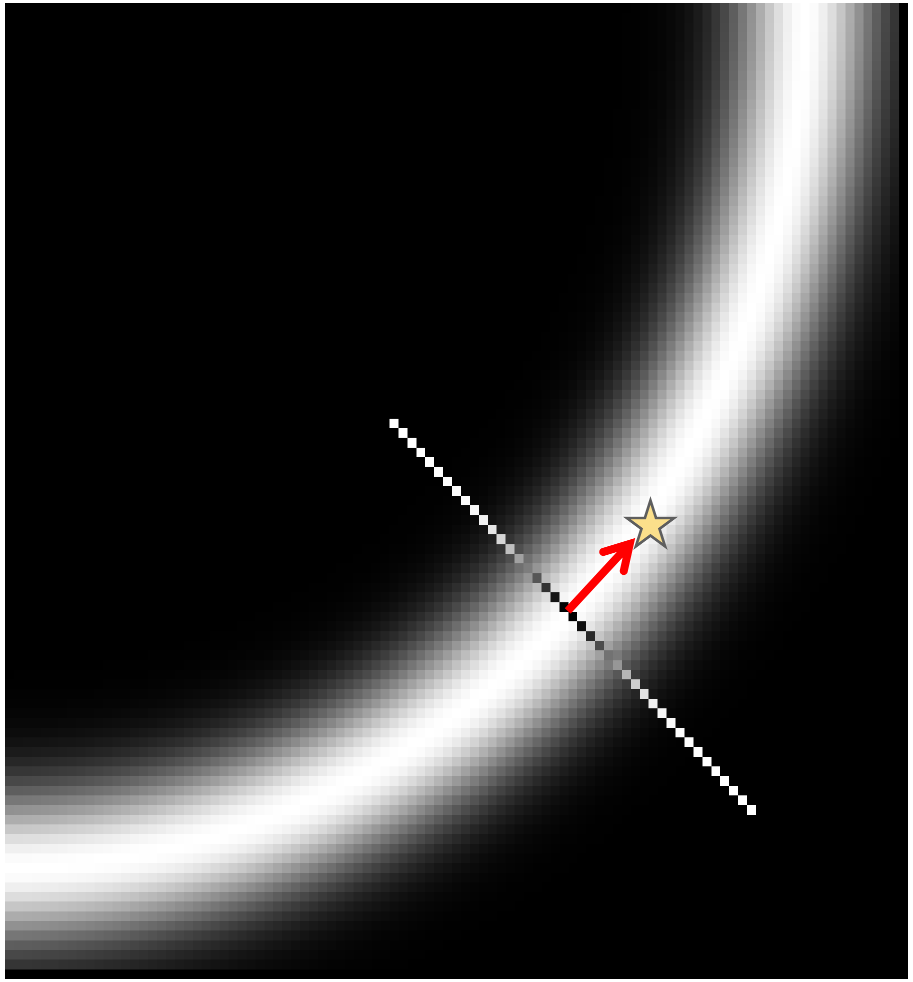
Tracking the edge direction

The direction orthogonal to the gradient follows the edge

Once a local maxima is found, **we consider the direction orthogonal to the gradient in that pixel,**

The direction is quantized as for extracting the 1D segment for nonmaximum suppression

We move one step in the quantized direction to determine another point where to extract 1D segments



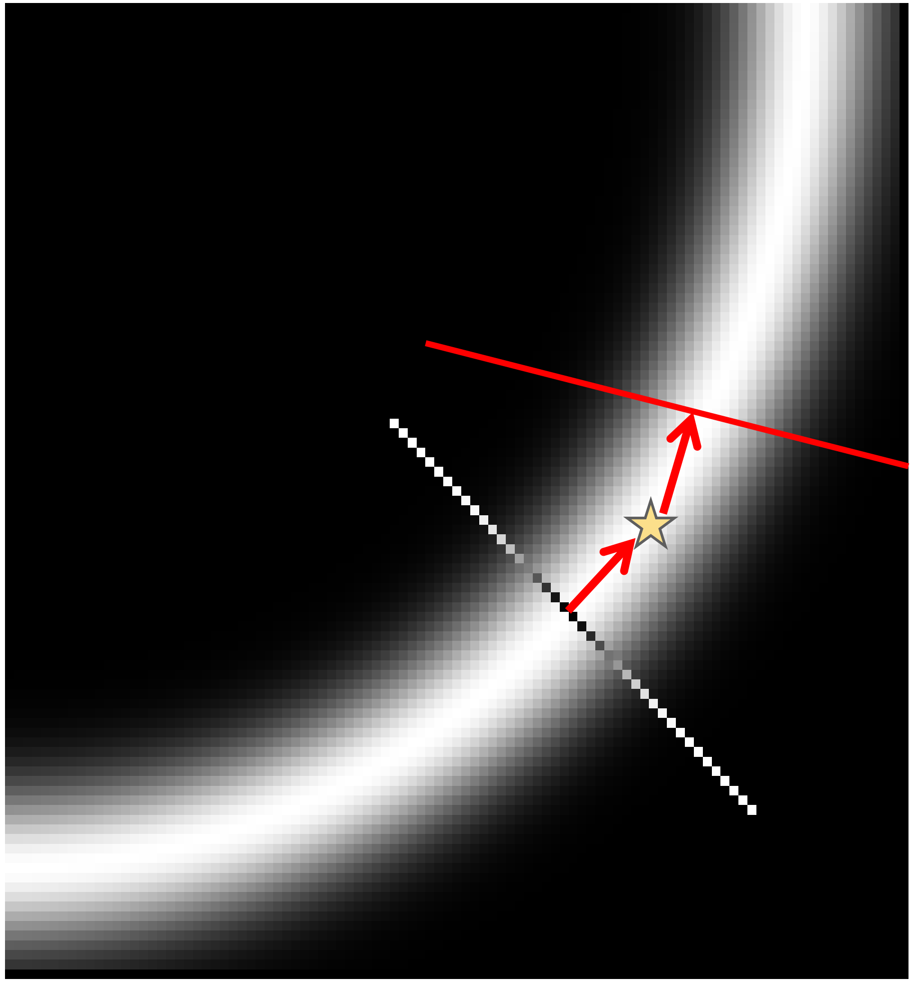
Tracking the edge direction

The direction orthogonal to the gradient follows the edge

Once a local maxima is found, **we consider the direction orthogonal to the gradient in that pixel,**

The direction is quantized as for extracting the 1D segment for nonmaximum suppression

We move one step in the quantized direction to determine another point where to extract 1D segments



Non-Maximum Suppression

$$|\Delta S| = \sqrt{S_x^2 + S_y^2}$$

M

Results from nonmaximum suppression

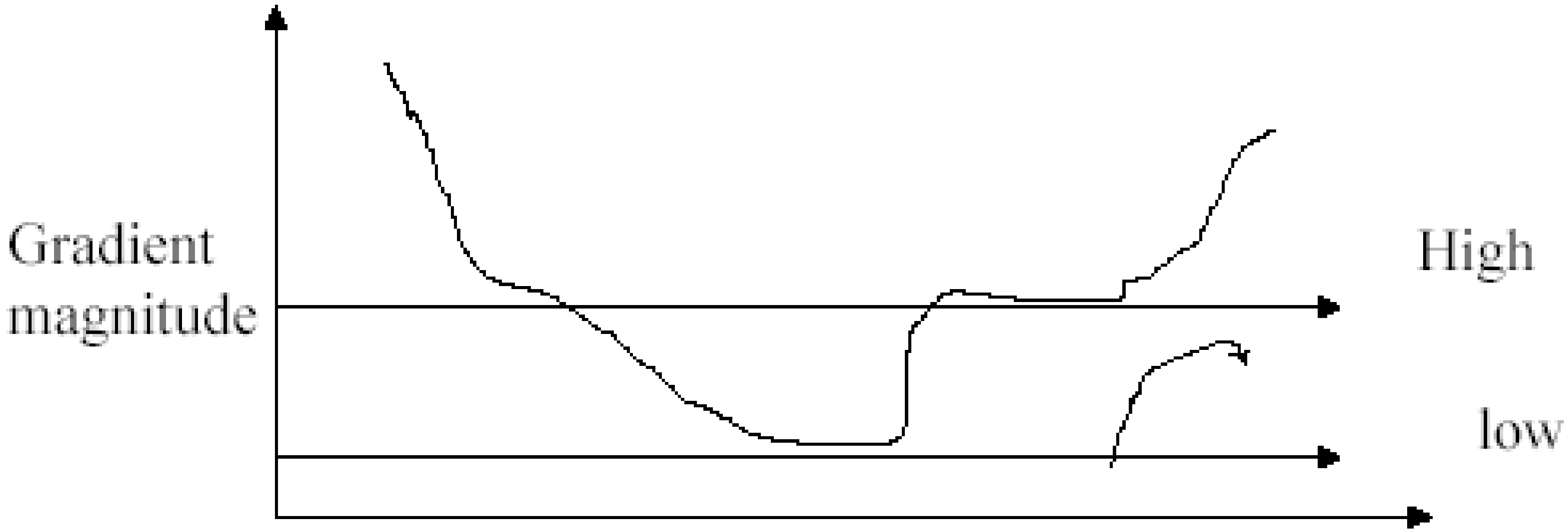
$$M \geq Threshold = 25$$

Hysteresis Thresholding

Use of two different threshold High and Low for

- For new edge starting point
- For continuing edges

In such a way the edges continuity is preserved



Hysteresis Thresholding

If the gradient at a pixel is **above 'High' threshold,**

- declare it an **'edge pixel'**.

If the gradient at a pixel is **below 'Low' threshold**

- declare it a **'non-edge-pixel'**.

If the gradient at a pixel is **between 'Low' and 'High' thresholds**

- then declare it an **'edge pixel'** if and only if can be directly **connected** to an 'edge pixel' or connected via pixels between 'Low' and ' High'.

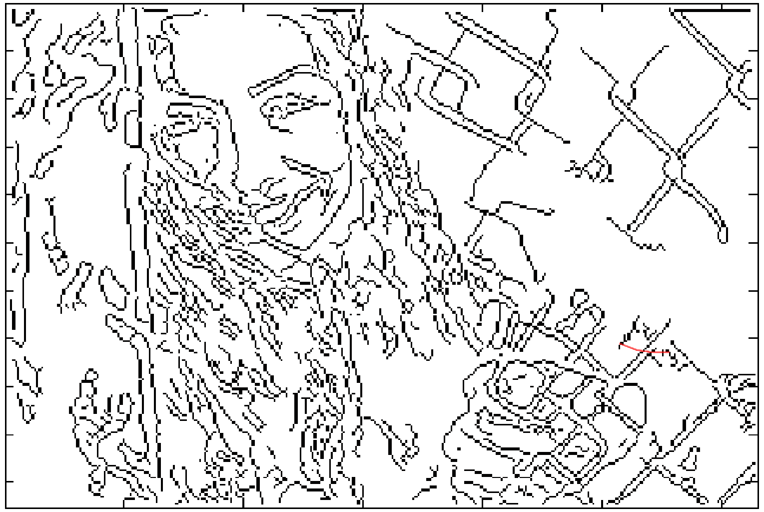
Hysteresis Thresholding

M

$M \geq Threshold = 25$

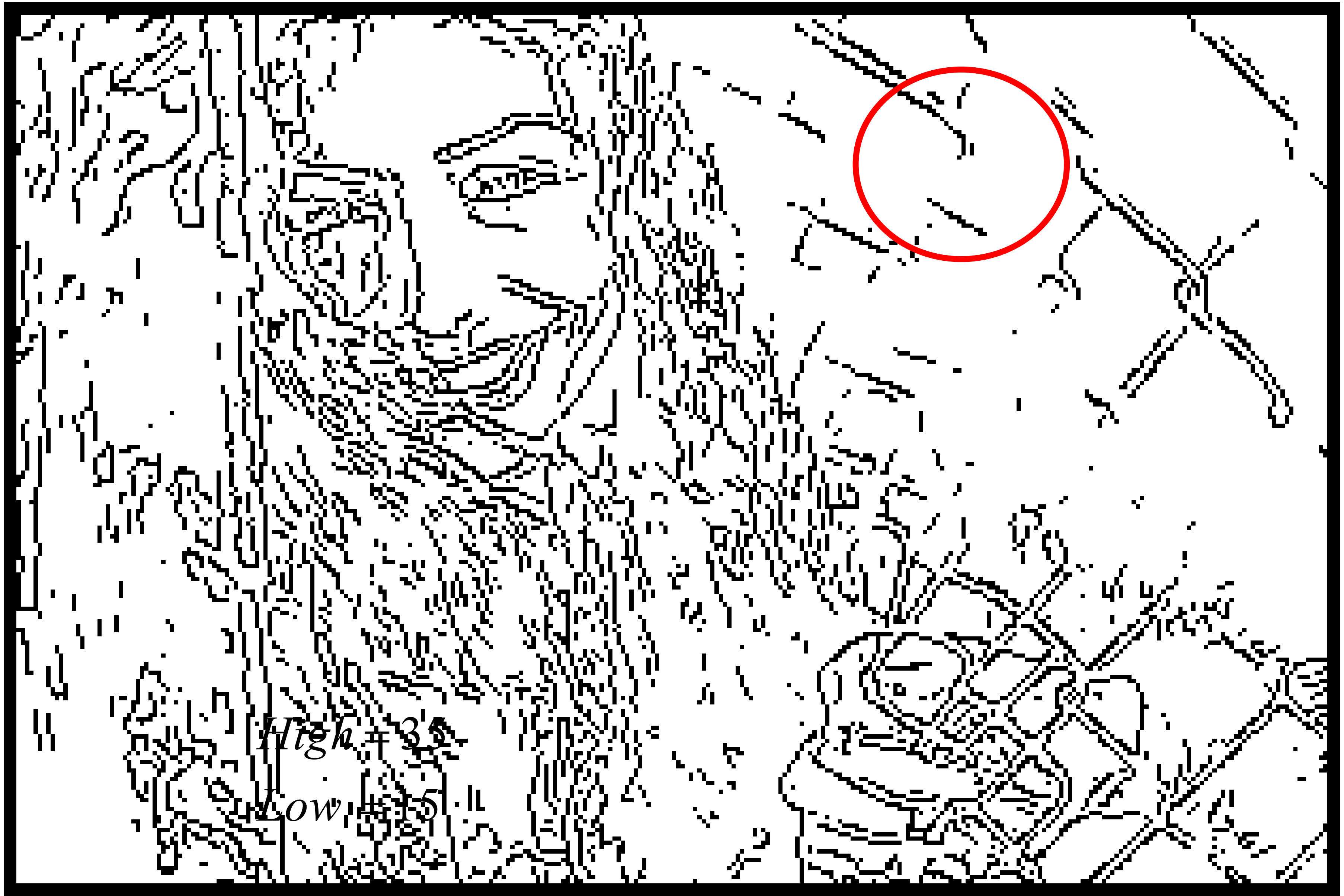
$High = 35$

$Low = 15$



Hysteresis Thresholding

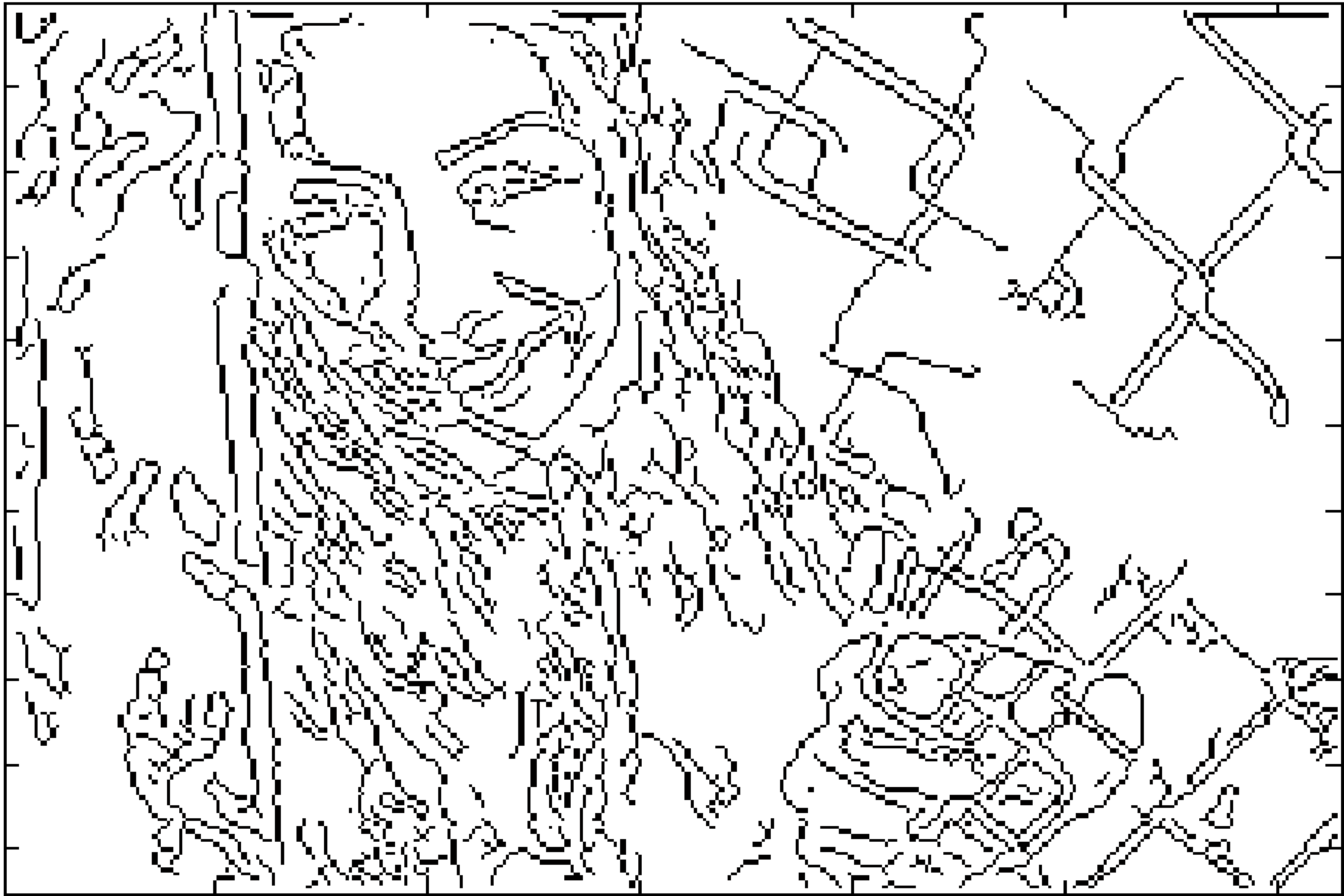
$$M \geq Threshold = 25$$



Hysteresis Thresholding

$High = 35$

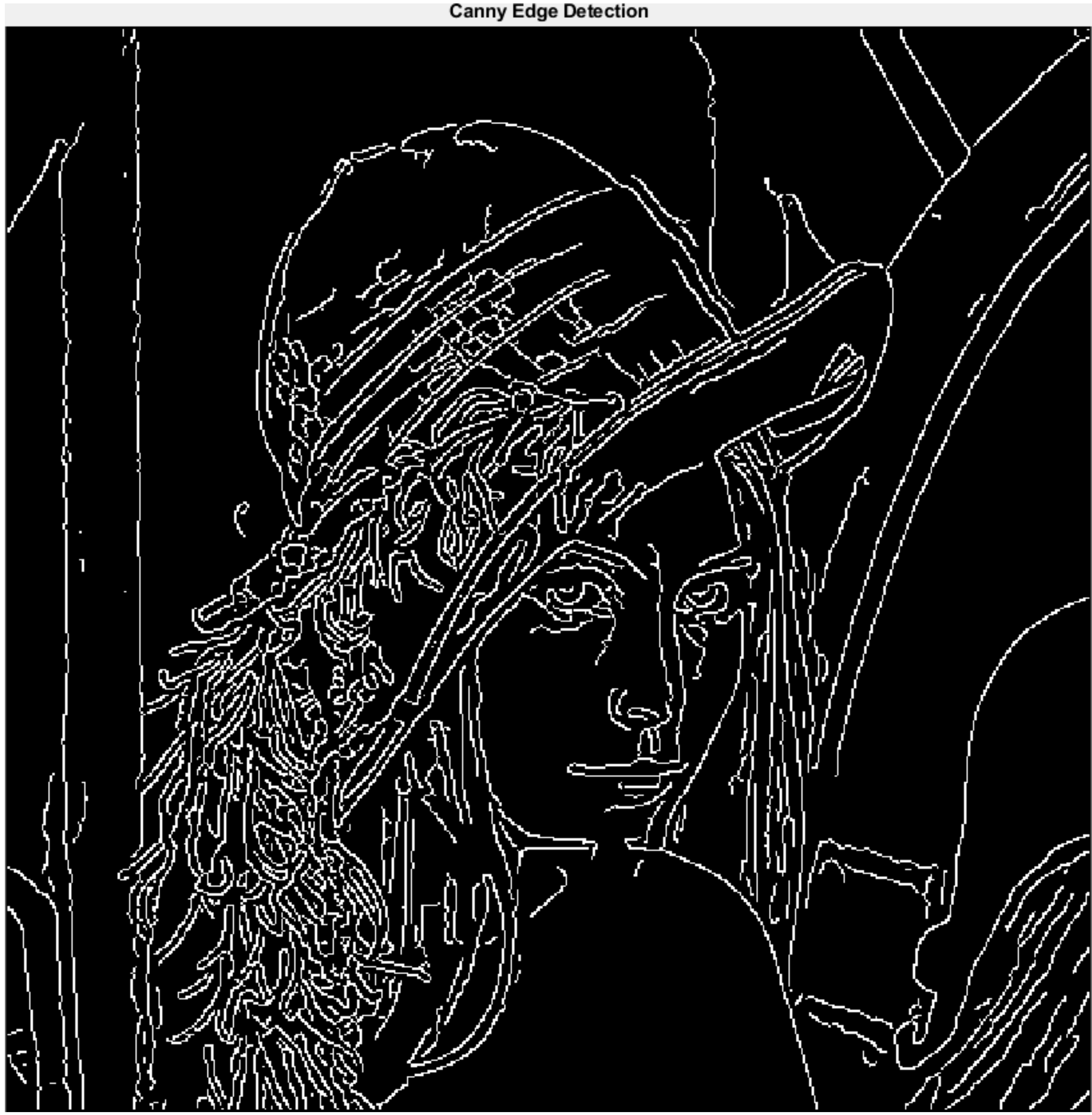
$Low = 15$



Canny Edge Detection

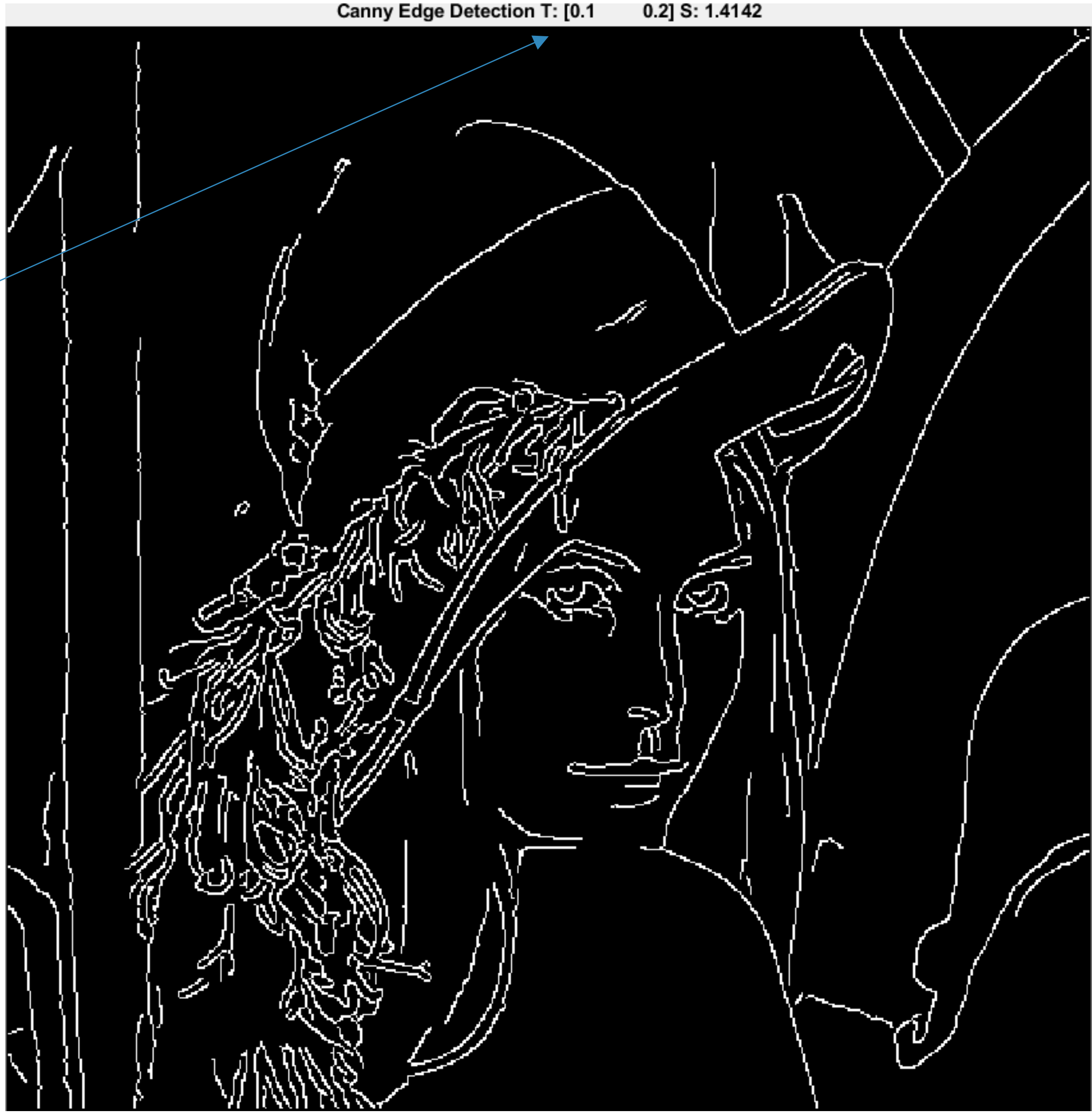
Canny Edge Detection

$\sigma_o = \sqrt{2}$



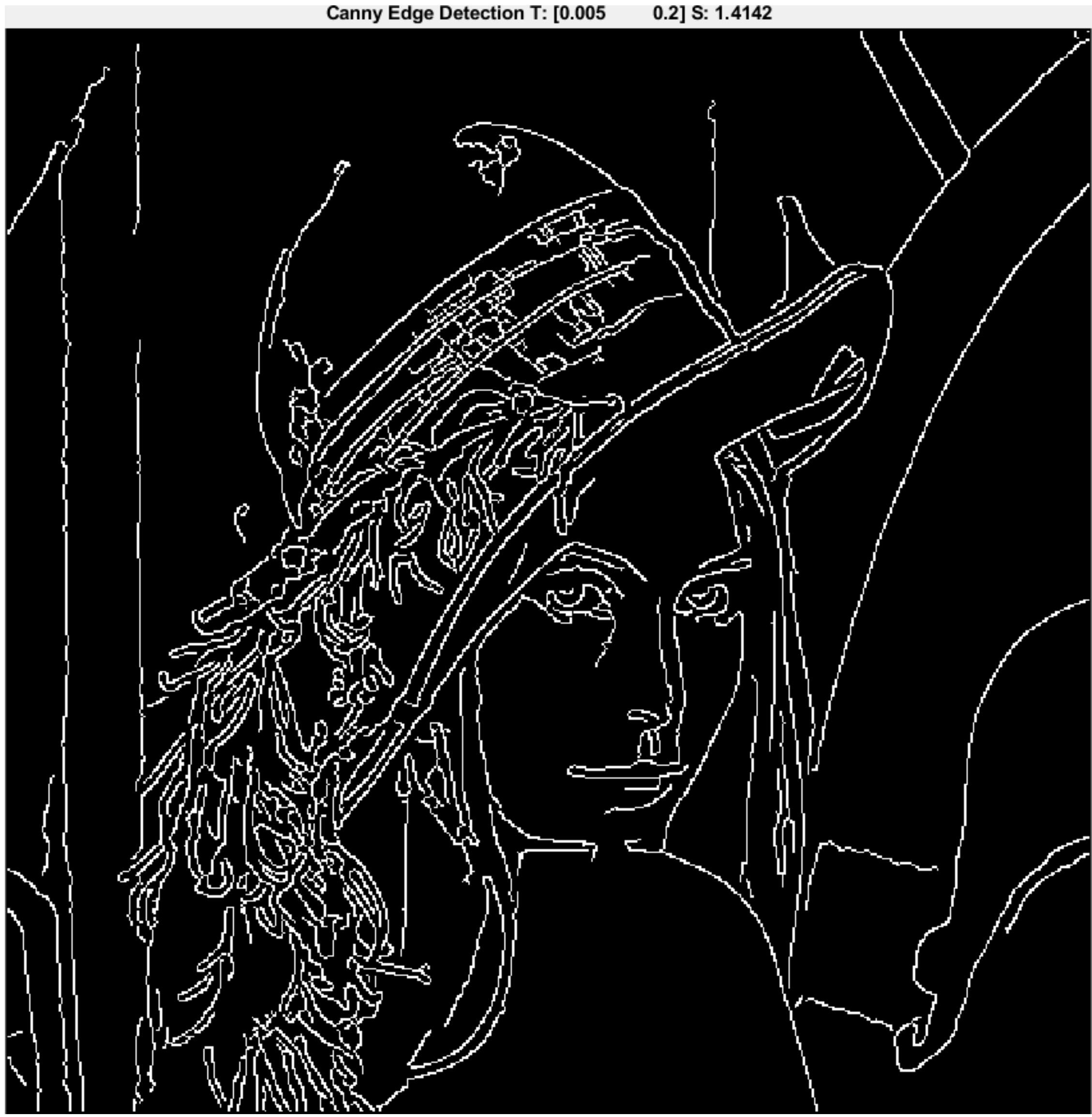
Canny Edge Detection – changing hysteresis thresholds

Threshold: [Low, High], Sigma



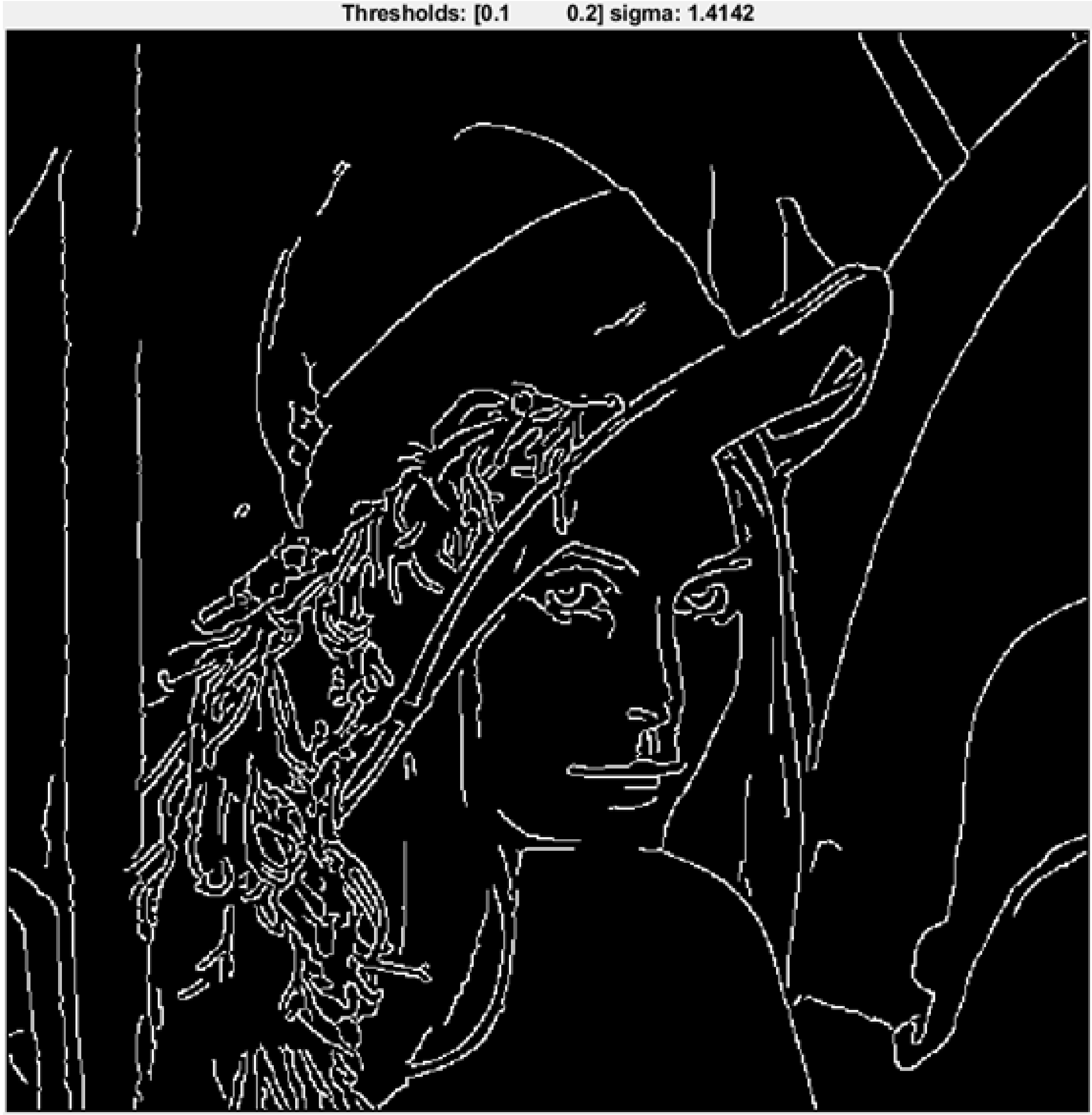
Canny Edge Detection – changing hysteresis thresholds

Decreasing the low threshold extends the length of existing edges



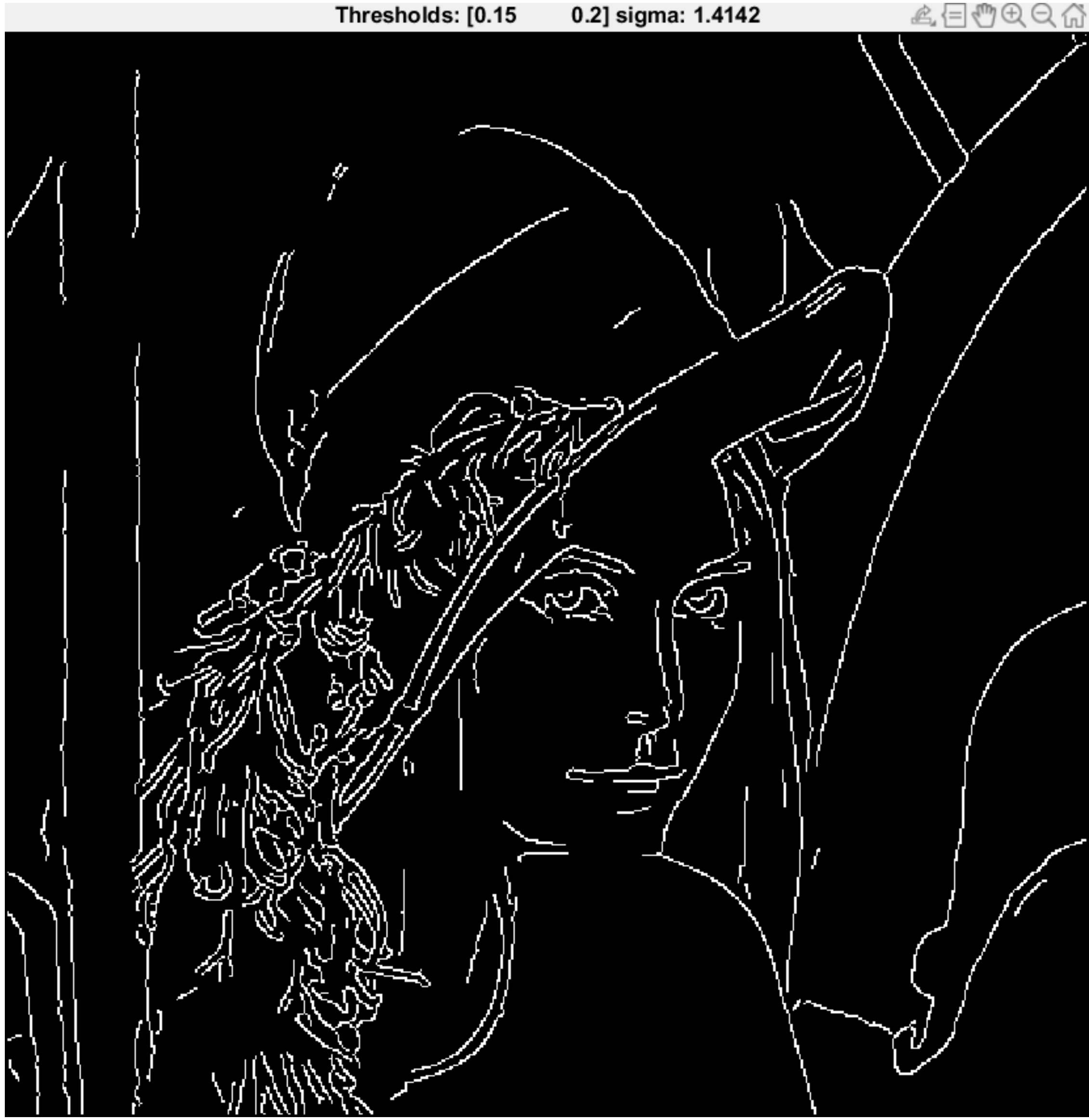
Canny Edge Detection – changing hysteresis thresholds

Reference thresholds



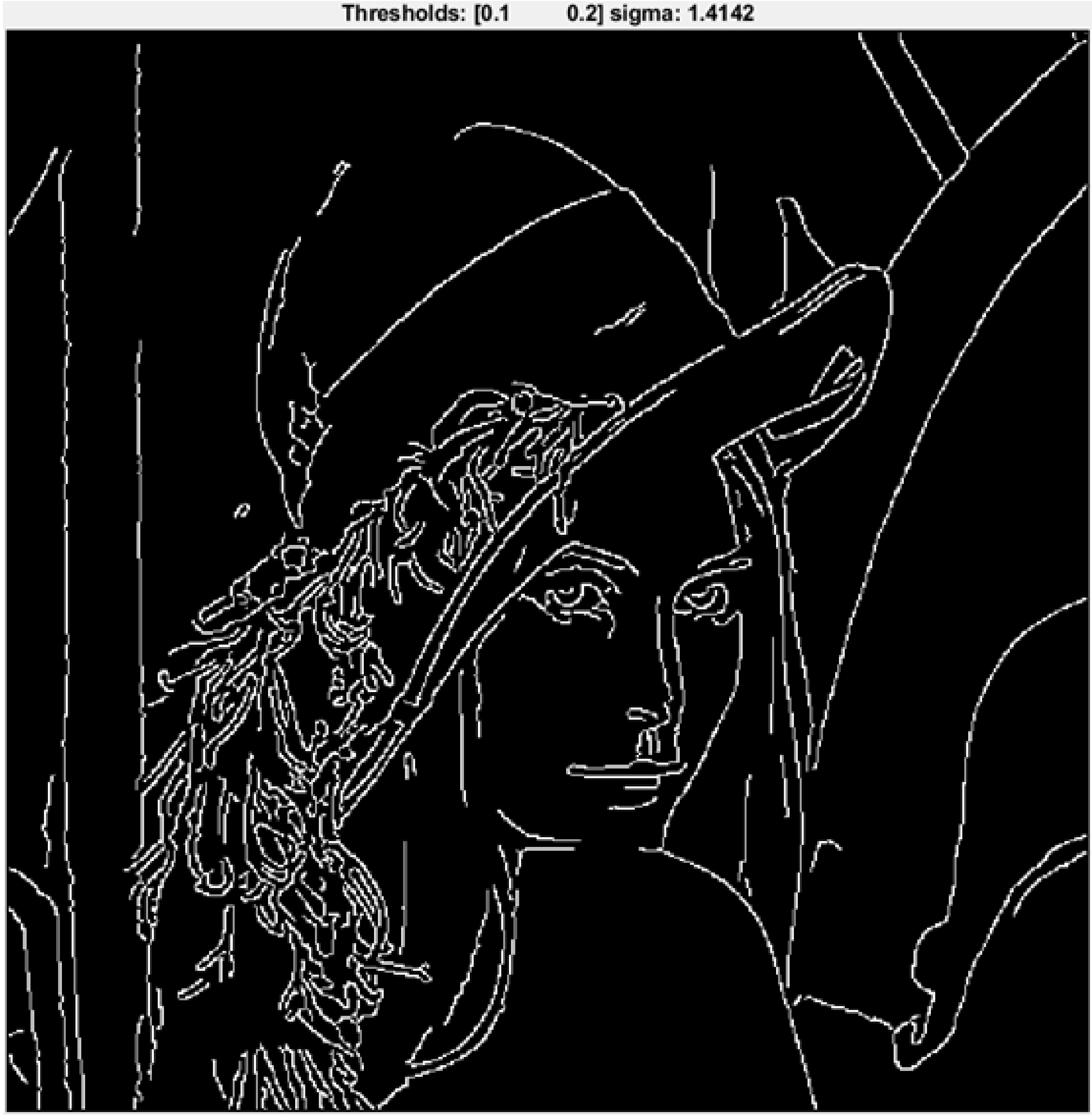
Canny Edge Detection – changing hysteresis thresholds

Increasing the low threshold shorten edges



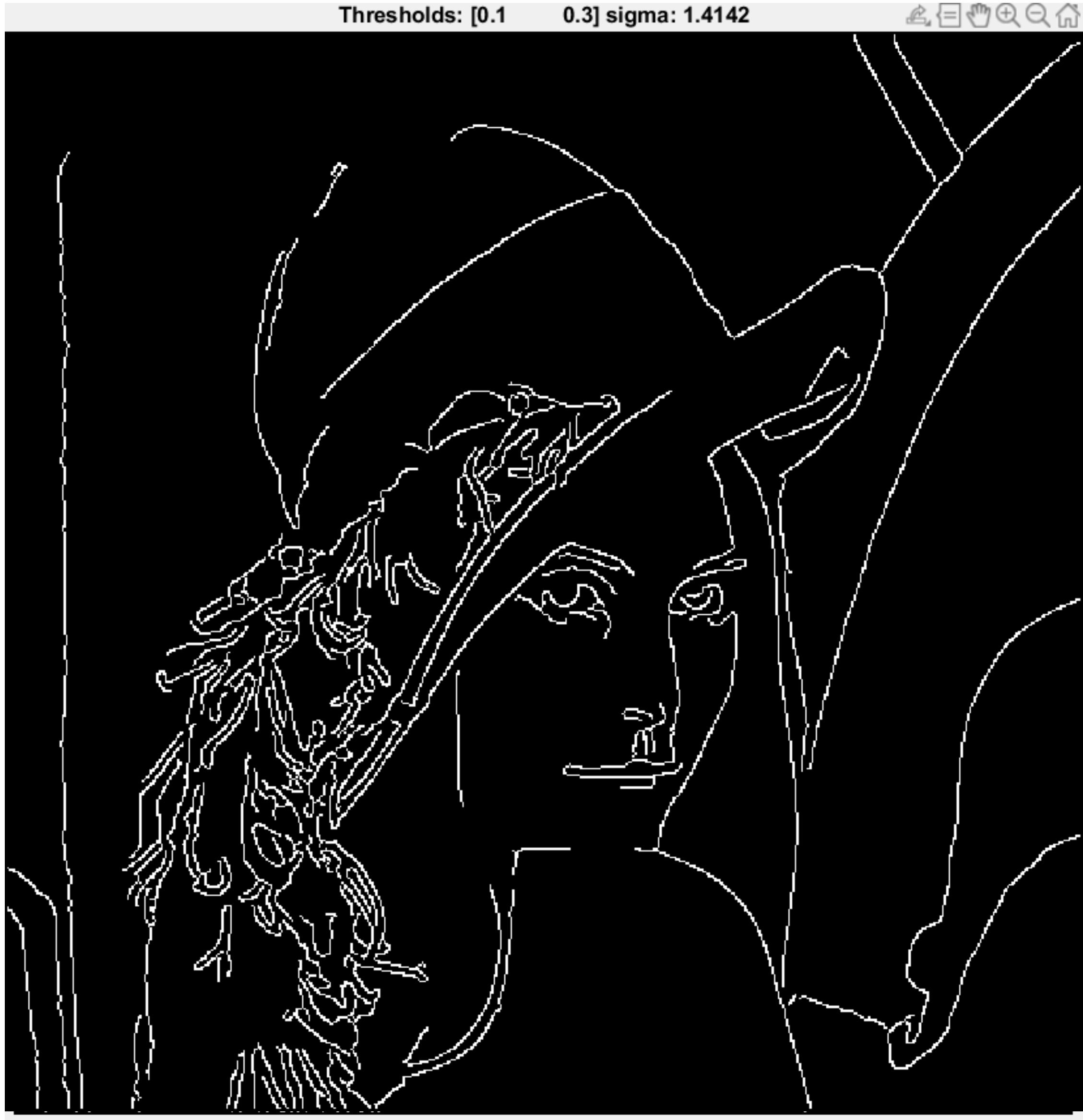
Canny Edge Detection – changing hysteresis thresholds

Reference thresholds



Canny Edge Detection – changing hysteresis thresholds

Increasing the high threshold reduces the number of edges



https://bigwww.epfl.ch/demo/ip/demos/edgeDetector/