FAKULTAS ILMU KOMPUTER

Semester Gasal 2024/2025
Topic #7

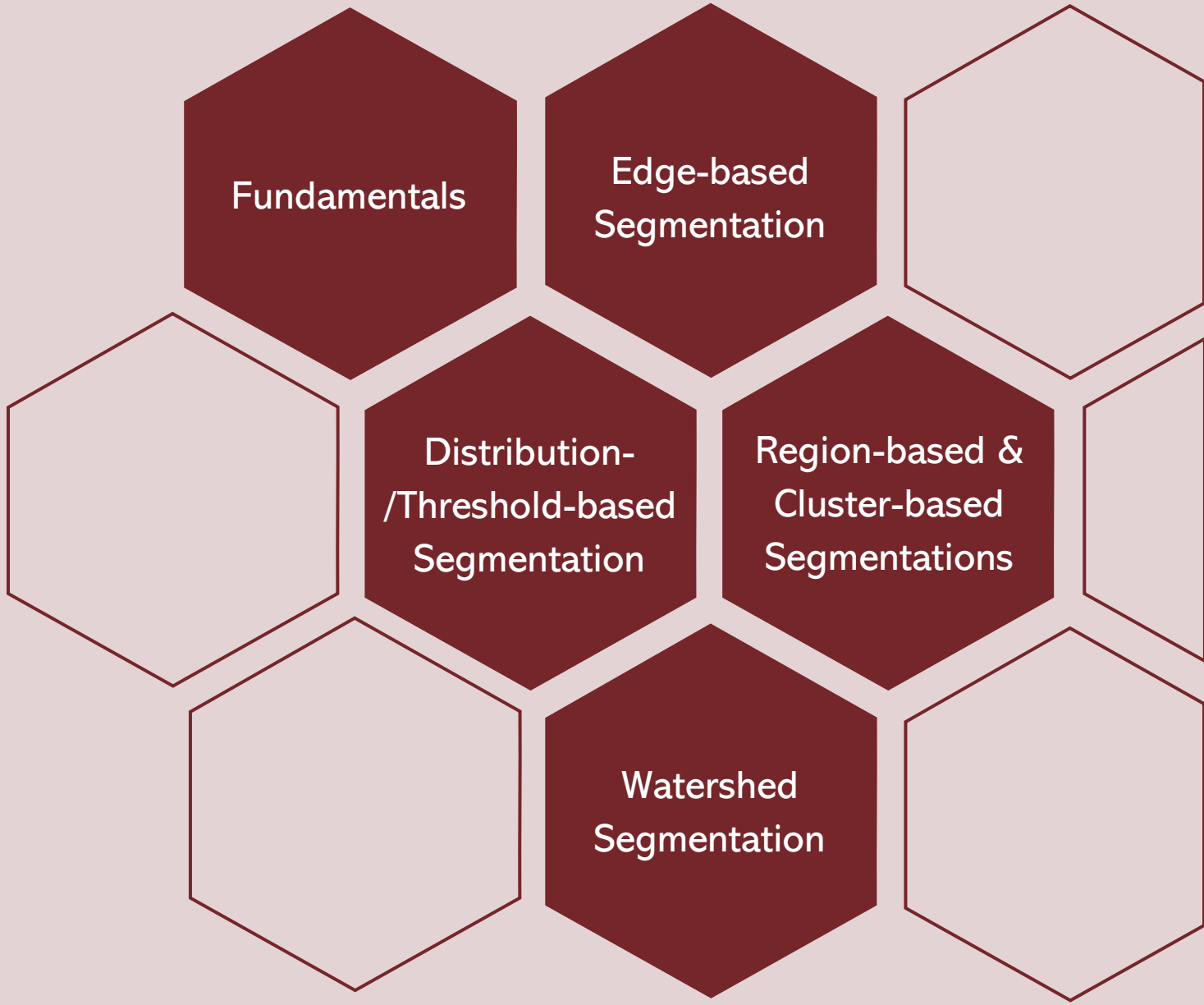
Image Segmentation

CSCE604133 Computer Vision
Faculty of Computer Science
Universitas Indonesia

Dr. Eng. Laksmita Rahadianti
Muhammad Febrian Rachmadi, Ph.D.
Dr. Dina Chahyati, Prof. Dr. Aniati M. Arymurthy

UNIVERSITAS INDONESIA
Veritas, Probitas, Iustitia
EST. 1849

Agenda

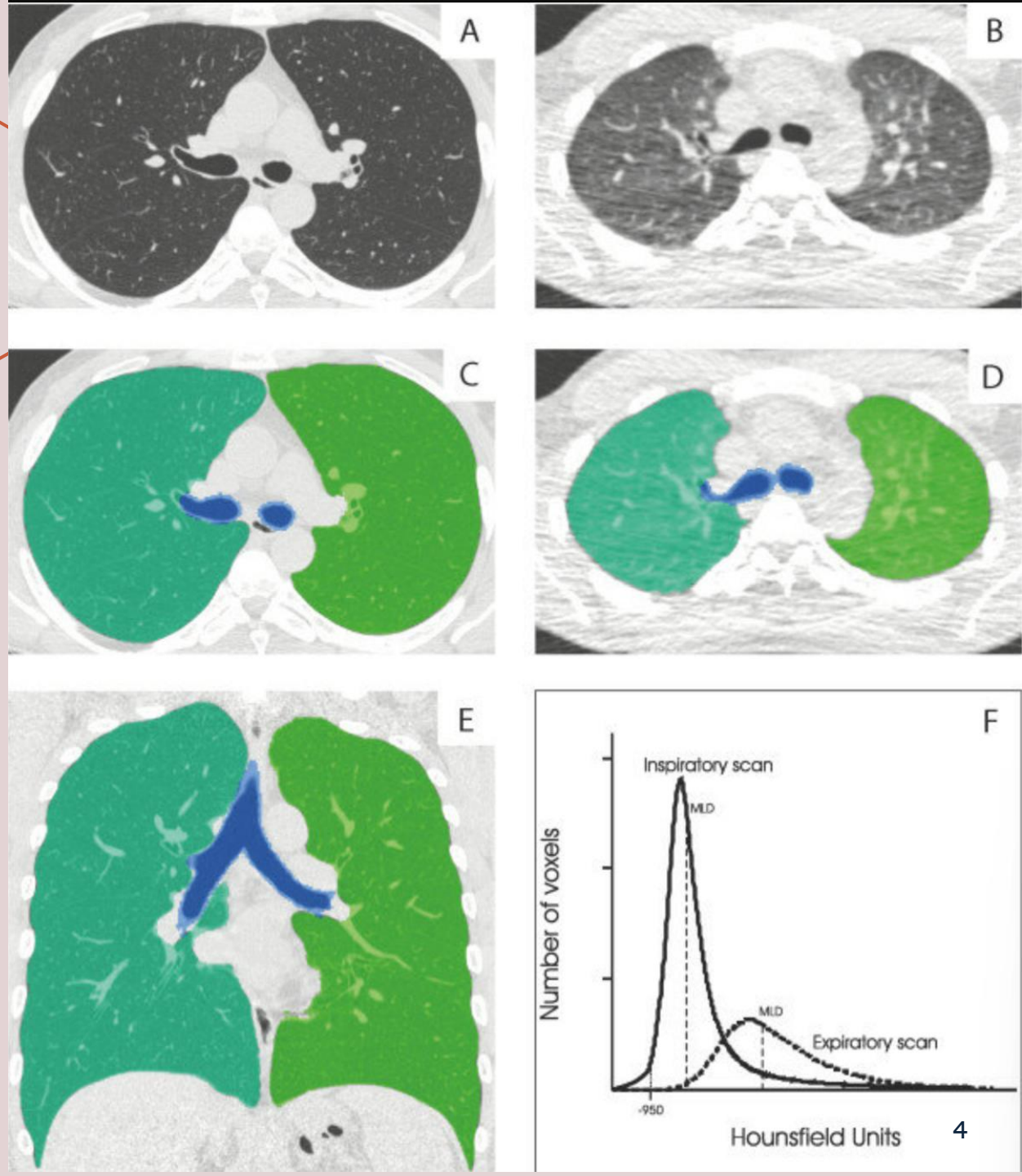

Fundamentals
Edge-based Segmentation
Distribution-/Threshold-based Segmentation
Region-based & Cluster-based Segmentations
Watershed Segmentation

Fundamentals of Image Segmentation

Section 1

Image Segmentation

- Image segmentation is a **recognition** technique that divides a digital image into subgroups called segments, thereby reducing its overall complexity and enabling the analysis and processing of each segment.
- Image segmentation technique involves assigning particular labels to pixels to identify objects, people, and other important elements.



https://mindy-support.com/news-post/what-is-image-segmentation-the-basics-and-key-techniques/
http://dx.doi.org/10.1186/1465-9921-14-59

Math Definition of Image Segmentation

- Let R represent the entire spatial region occupied by an image.
- Then, a segmentation algorithm will perform a partition of R into n subregions, $R_1, R_2, \ldots, R_n$.

n region nya itu harus konsisten (jadi harus semuanya punya hubungan (connected) kayak semuanya harus pake kacamata neighboursnya)

- Conditions for segmentation:
  a) Complete, i.e., every pixel must be in one of the regions.
  b) Points in a region are connected in some predefined sense (e.g., the points must be 8-connected)
  c) Regions must be disjoint.
  d) Properties must be satisfied by all pixels in a segmented region.
  e) Properties of pixels in two or more segmented regions must be different.

Types of Image Segmentation

- When can we say we have segmented the image?
  - When the regions of interest have been identified
  - It depends on the context and the task
- Image segmentation can be done in several different ways. Below you will find some of the most common techniques:
  1. **Semantic image segmentation:** This involves arranging the pixels in an image based on semantic class.
  2. **Instance segmentation:** This technique involves classifying pixels based on the instances of an object instead of classes.
  3. **Panoptic segmentation:** This is a newer technique and is often expressed as a combination of semantic and instance segmentation. It predicts the identity of each object, separating every instance of each object in the image. Jadi dia setiap kelas itu berbeda warna
- This lecture focuses on the **semantic image segmentation**.

Ini yang diajari di pelajaran CV

SEMANTIC IMAGE SEGMENTATION

INSTANCE SEGMENTATION

membedakan dari 2 instance yang labelnya berbeda.

PANOPTIC SEGMENTATION

https://mindy-support.com/news-post/what-is-image-segmentation-the-basics-and-key-techniques/

Principles of Image Segmentation

1. **Discontinuity:** Segment the image based on the abrupt changes in intensity.
2. **Similarity:** Segment the image based on similar characteristics.

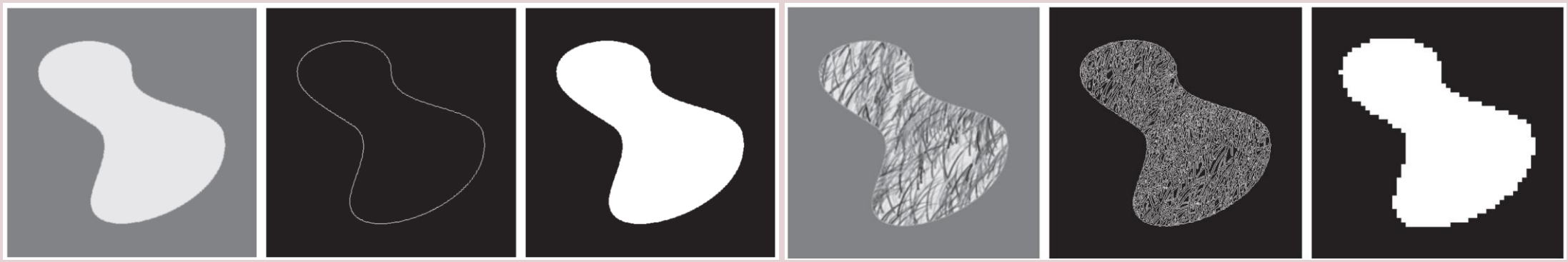
Edge-based Segmentation

Distribution-based & Threshold-based Segmentations

Region-based & Cluster-based Segmentations

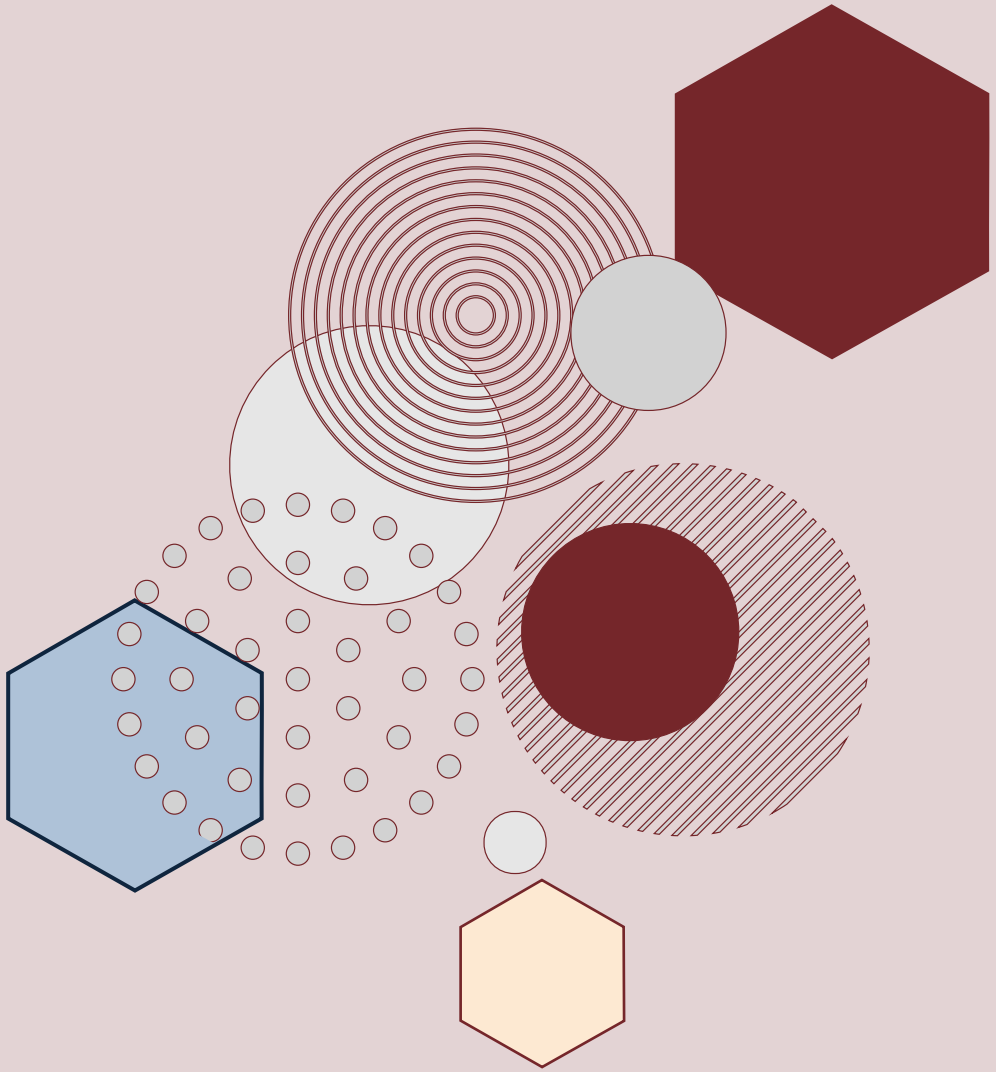
Watershed Segmentation

edge based segmentation: menentukan batas-batas (jadi yang diluar vs di dalam dianggap berbeda)



A region (constant intensity) | Boundary based on intensity discontinuities | Result of segmentation | Image of a texture region | Result of intensity discontinuity computations | Result of segmentation based on region properties

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."



Edge-based Segmentation

Section 2 – Fundamentals of Edge-based Segmentation

2.1 Point and Line Detections

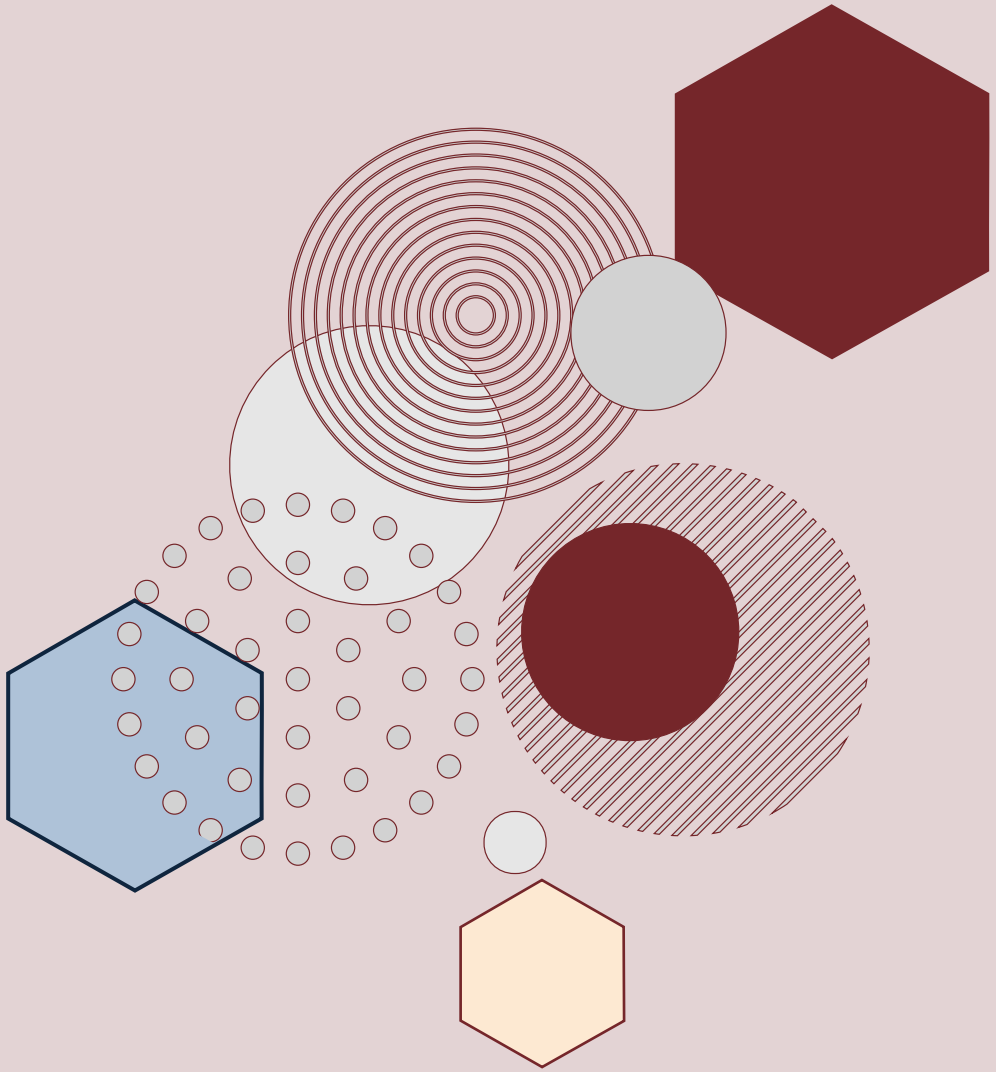
2.2 Edge Detection

- 2.2.1 Roberts, Prewitt, and Sobel Edge Detections
- 2.2.2 Canny Edge Detection

2.3 Post-processing Techniques After Edge Detection (Essential to Know, Supplementary)

Math Definition of Image Segmentation

- Let R represent the entire spatial region occupied by an image.
- Then, a segmentation algorithm will perform a partition of R into n subregions, $R_1, R_2, \ldots, R_n$.
- Conditions for segmentation:
  a) Complete, i.e., every pixel must be in one of the regions.
  b) Points in a region are connected in some predefined sense (e.g., the points must be 8-connected)
  c) Regions must be disjoint.
  d) Properties must be satisfied by all pixels in a segmented region.
  e) Properties of pixels in two or more segmented regions must be different.

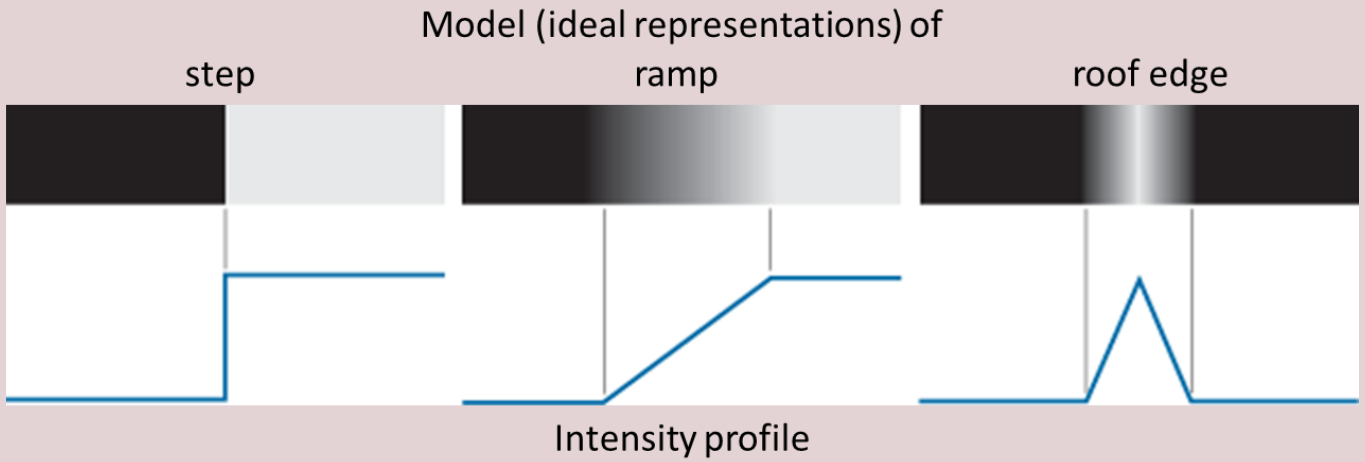


Edge-based Segmentation

Section 2.1 - Point and Line Detections

Edges in an Image

- **Edge:** a set of connected pixels that lie on the boundary between two regions



step: perubahnannya itu at one time

ramp: berubahnya pelan-pelan

roofedge: perubahannya intensitasnya tiba-tiba berubah pada saat naik/turun secara drastis

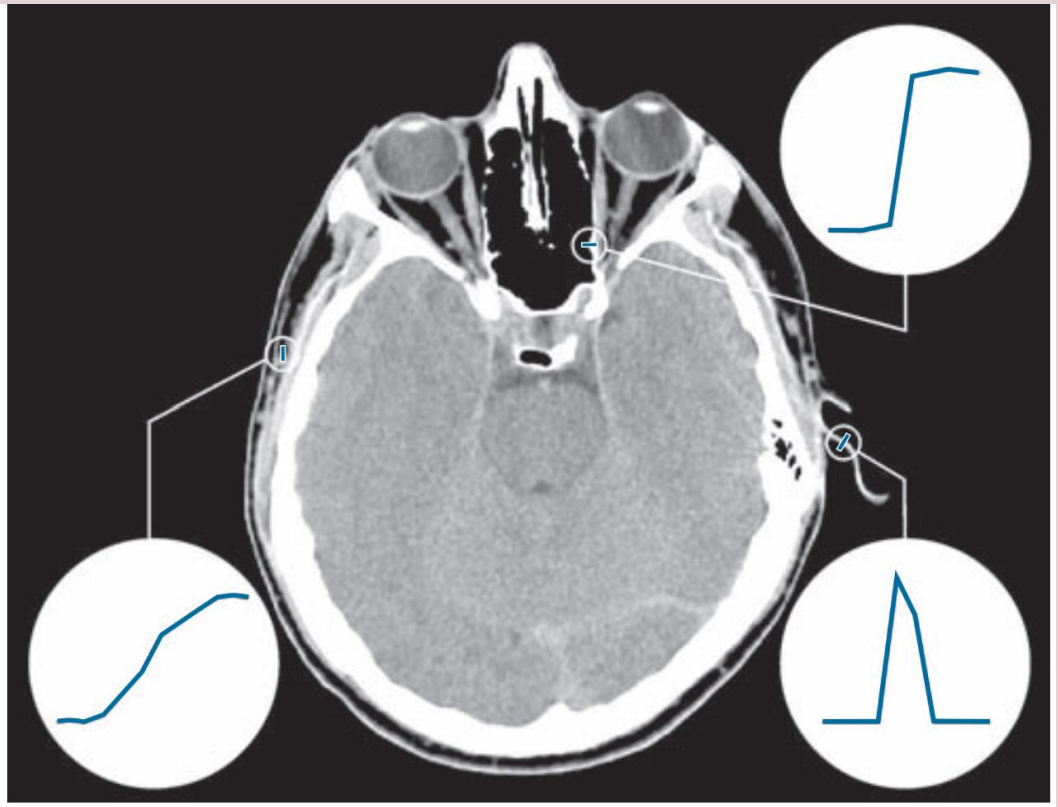


FIGURE 10.9 A 1508 × 1970 image showing (zoomed) actual ramp (bottom, left), step (top, right), and roof edge profiles. The profiles are from dark to light, in the areas enclosed by the small circles. The ramp and step profiles span 9 pixels and 2 pixels, respectively. The base of the roof edge is 3 pixels. (Original image courtesy of Dr. David R. Pickens, Vanderbilt University.)

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

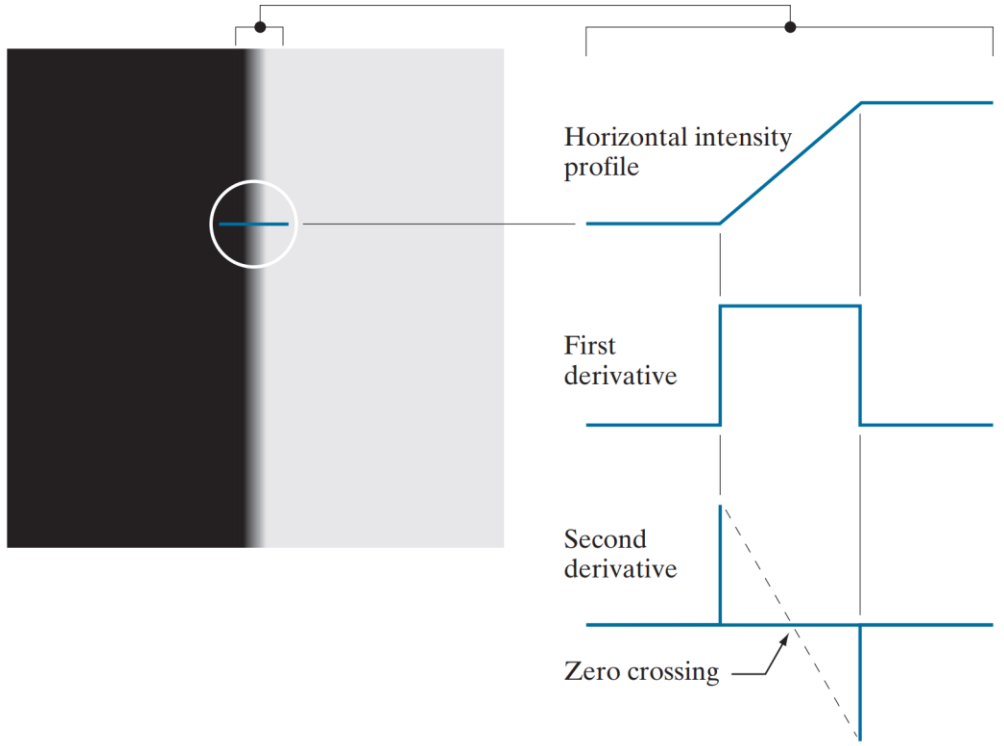
Analyzing an Edge using Derivatives

- Edges in an image can be detected by using derivatives.
- **Second-order derivatives** have a more robust response to finer detail, such as thinner lines, and **produce a double-edge response.**
  - **Double-edge response:** The sign of the second derivative determines the change in intensity, i.e., negative (light to dark) or positive (dark to light).

Untuk menghitung edge itu bisa pakai turunan kedua

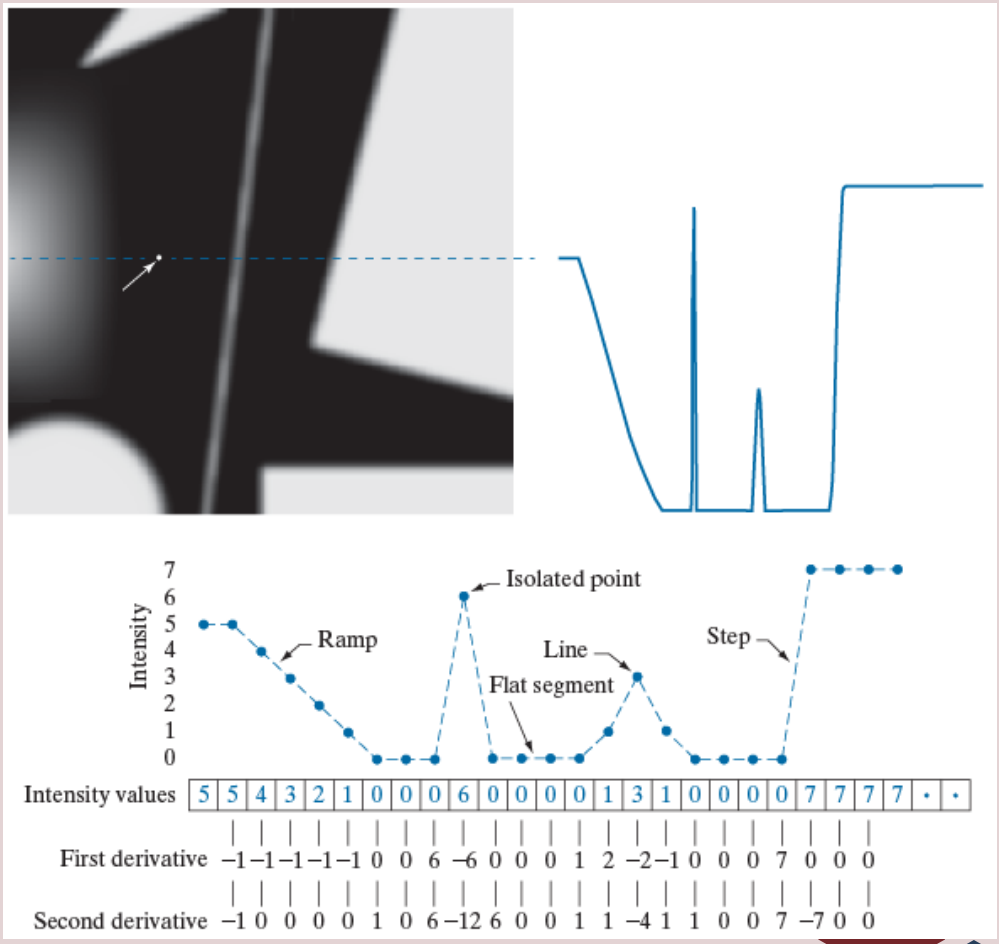
a b

FIGURE 10.10
(a) Two regions of constant intensity separated by an ideal ramp edge. (b) Detail near the edge, showing a horizontal intensity profile, and its first and second derivatives.



Derivatives of A Digital Function

- Derivatives of a digital function are defined in terms of ***finite differences***.
- Approximation of ***the first derivative*** must be:
  - Zero in areas of constant intensity;
  - Nonzero at the onset of an intensity step/ramp;
  - Nonzero at points along an intensity ramp.
- Approximation of ***the second derivative*** must be:
  - Zero in areas of constant intensity;
  - Nonzero at the onset and end of an intensity step/ramp;
  - Zero at points along an intensity ramp.



Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

[Math] Approximation to Derivatives of 1D Function

- Approximation to the n-order derivative at an arbitrary point x of a function $f(x)$ can be obtained by expanding the function $f(x+\Delta x)$ into a Taylor series about x:

$$f(x+\Delta x) = f(x) + \Delta x \frac{\partial f(x)}{\partial x} + \frac{(\Delta x)^2}{2!}\frac{\partial^2 f(x)}{\partial^2 x} + \frac{(\Delta x)^3}{3!}\frac{\partial^3 f(x)}{\partial^3 x} + \cdots = \sum_{n=0}^{\infty} \frac{(\Delta x)^n}{n!}\frac{\partial^n f(x)}{\partial^n x} \quad \text{(Eq. 10–1)}$$

  where Δx is the separation between samples of f.

- Note, $\Delta x = 1$ is for the sample preceding x, and $\Delta x = -1$ for the sample following x.

0	1	2	3	4	5	6

$$f(x+1) = f(x) + \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial^2 x} + \frac{1}{3!}\frac{\partial^3 f(x)}{\partial^3 x} + \cdots = \sum_{n=0}^{\infty} \frac{1}{n!}\frac{\partial^n f(x)}{\partial^n x} \quad \text{(Eq. 10–2)}$$

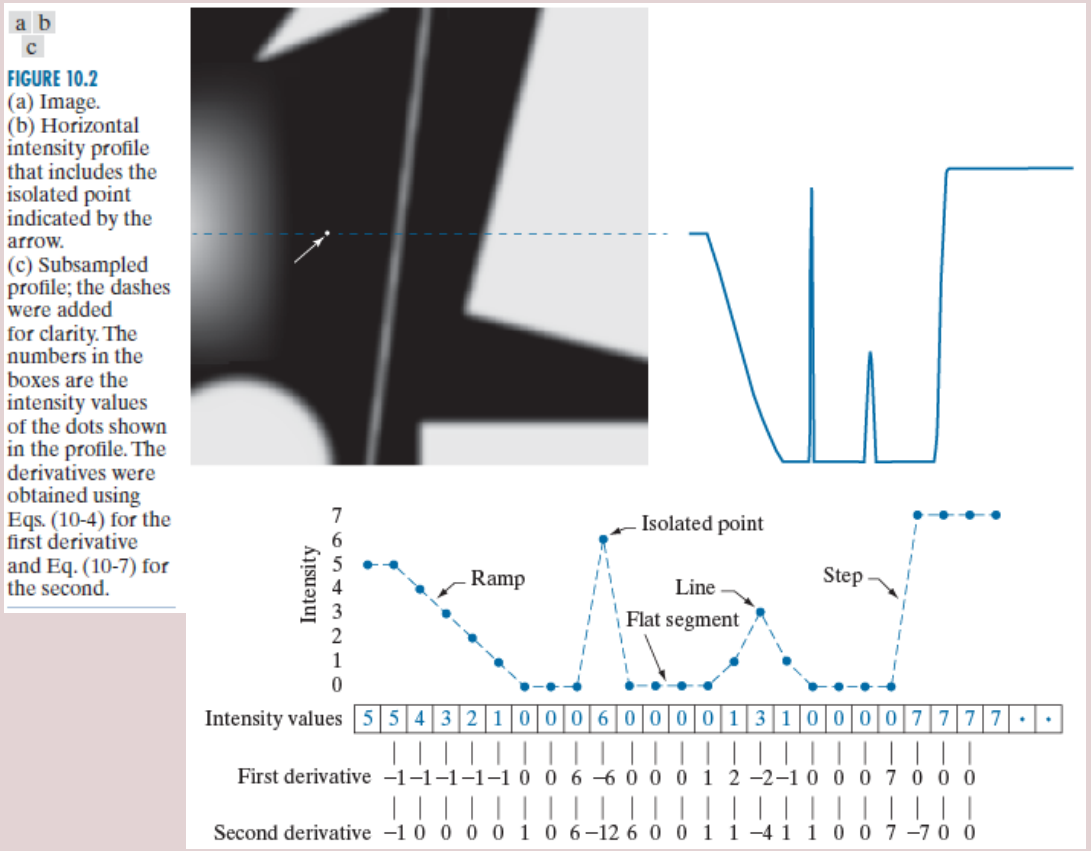
$$f(x-1) = f(x) - \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial^2 x} - \frac{1}{3!}\frac{\partial^3 f(x)}{\partial^3 x} - \cdots = \sum_{n=0}^{\infty} \frac{-1}{n!}\frac{\partial^n f(x)}{\partial^n x} \quad \text{(Eq. 10–3)}$$

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

[Math] Intensity Differences using the Terms of Taylor Series

$$f(x+1) = f(x) + \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial^2 x} + \frac{1}{3!}\frac{\partial^3 f(x)}{\partial^3 x} + \cdots \quad (10-2)$$

$$f(x-1) = f(x) - \frac{\partial f(x)}{\partial x} + \frac{1}{2!}\frac{\partial^2 f(x)}{\partial^2 x} - \frac{1}{3!}\frac{\partial^3 f(x)}{\partial^3 x} - \cdots \quad (10-3)$$



A. An approximation of **the first-order derivative**:

1. The **forward difference** is obtained from (Eq. 10-2):

$$\frac{\partial f(x)}{\partial x} = f'(x) = f(x+1) - f(x)$$

2. The **backward difference** is obtained from (Eq. 10-3):

$$\frac{\partial f(x)}{\partial x} = f'(x) = f(x) - f(x-1)$$

3. The **central difference** is obtained by subtracting (Eq. 10-3) from (Eq. 10-2)

$$\frac{\partial f(x)}{\partial x} = f'(x) = \frac{f(x+1) - f(x-1)}{2}$$

B. An approximation of **the second order derivative** is obtained by adding (Eq. 10-2) and (Eq. 10-3):

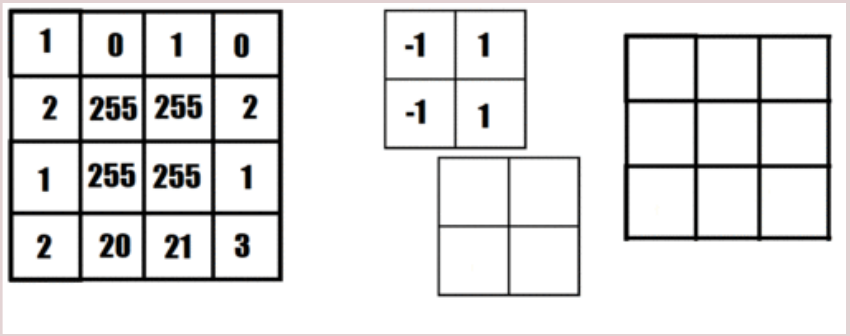
$$\frac{\partial^2 f(x)}{\partial^2 x} = f''(x) = f(x+1) - 2f(x) + f(x-1)$$

Computing the First and Second Derivatives

Kita bisa meneragkan suatu pixel

- Spatial **convolution** is the approach of choice for computing the first and second derivates at every pixel location in an image.

https://www2.cs.uregina.ca/~dbd/cs831/notes/neural-networks/conv-neural-networks/



Remember:
- Negative (light to dark)
- Positive (dark to light)

- The response of the filter at the center point of the (3x3) kernel is

$$Z = w_1 z_1 + w_2 z_2 + \cdots + w_9 z_9 = \sum_{k=1}^{9} w_k z_k$$

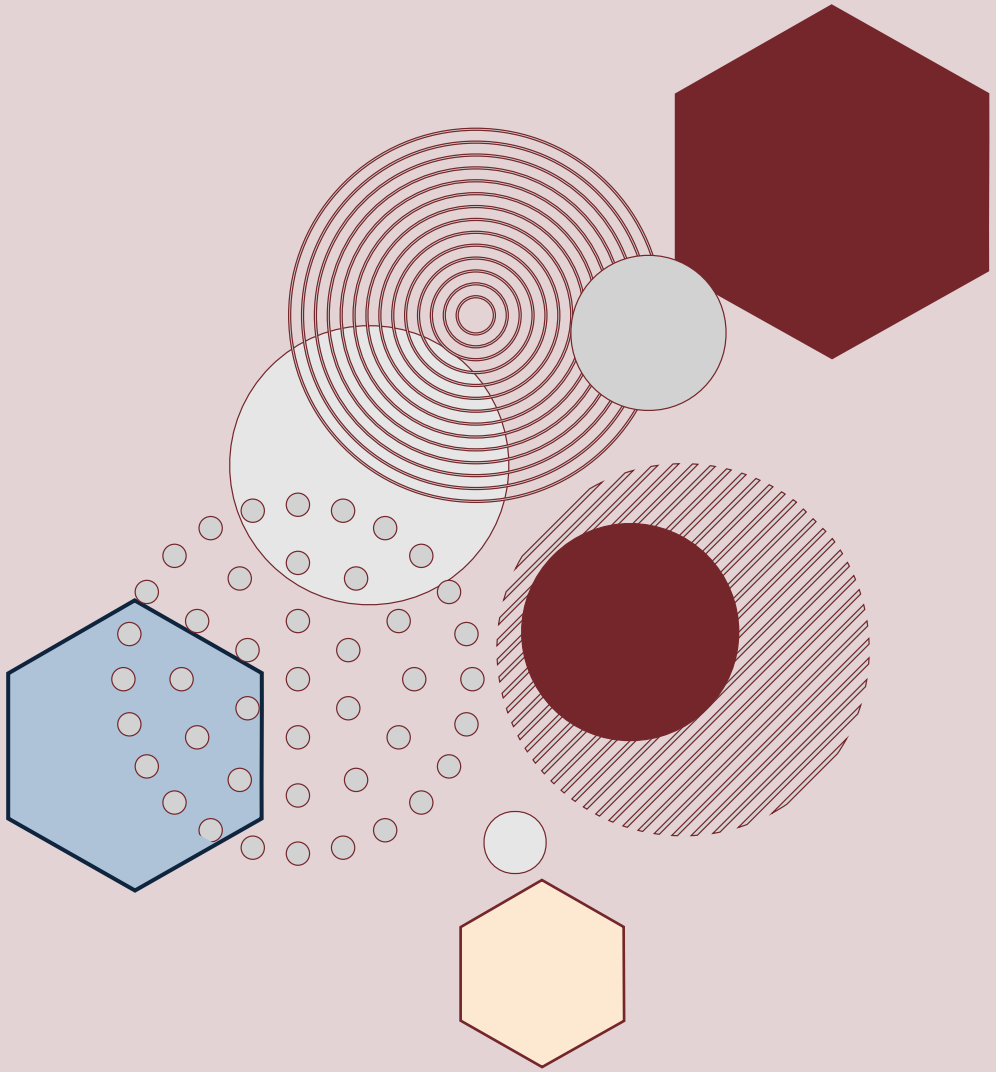
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

FIGURE 10.3 A general 3×3 spatial filter kernel. The w's are the kernel coefficients (weights).

where z_k is the intensity of the pixel whose spatial location corresponds to the location of the k^{th} kernel coefficient.

- In the next sections, we will learn several convolution kernels commonly used for detecting edges in an image.

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."



Edge-based Segmentation

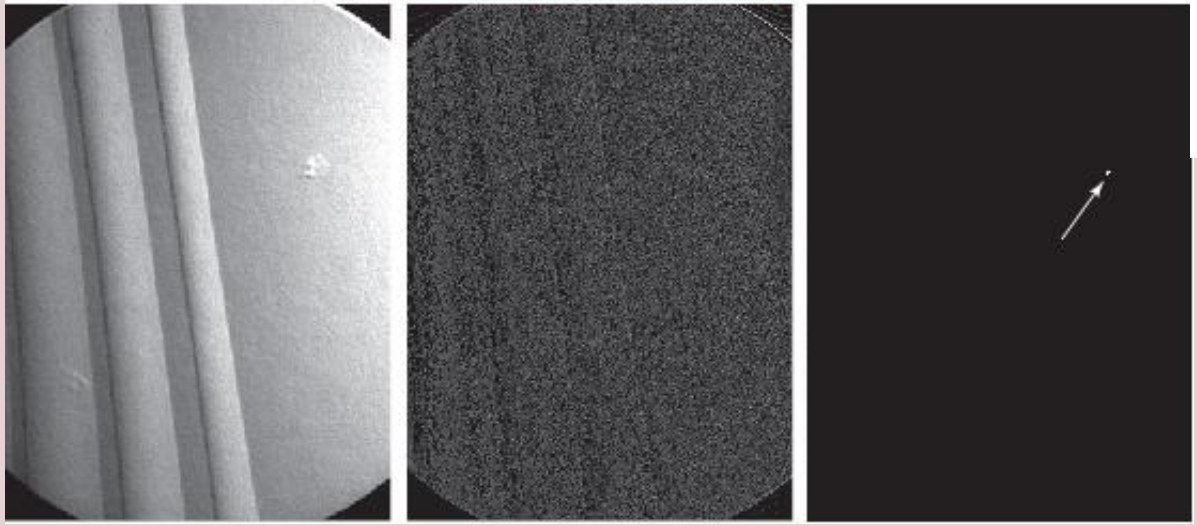
Section 2.2 - Edge Detection

Detection of Isolated Points

Apabila kita sampling suatu titik yang memiliki perbedaan intensitas, maka dia akan berbeda dengan neighbors lainnya)

- Detection of (isolated) points could be performed by using a **Laplacian kernel**.

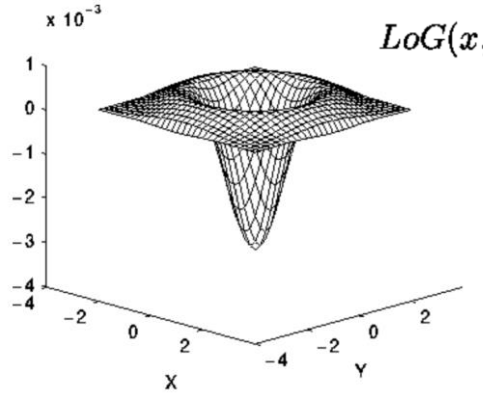
1	1	1
1	−8	1
1	1	1



a
b c d

FIGURE 10.4
(a) Laplacian kernel used for point detection. (b) X-ray image of a turbine blade with a porosity manifested by a single black pixel. (c) Result of convolving the kernel with the image. (d) Result of using Eq. (10-15) was a single point (shown enlarged at the tip of the arrow). (Original image courtesy of X-TEK Systems, Ltd.)

Laplacian/Laplacian of Gaussian Filter:
https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm



$$LoG(x,y) = -\frac{1}{\pi\sigma^4}\left[1 - \frac{x^2+y^2}{2\sigma^2}\right]e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$Z = w_1z_1 + w_2z_2 + \cdots + w_9z_9 = \sum_{k=1}^{9} w_kz_k$$

$$g(x,y) = \begin{cases} 1 & if\ |Z(x,y)| > T \\ 0 & otherwise \end{cases}$$

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Line Detection

bisa mencari angle-angle atau kemunculan gatid tertentu.

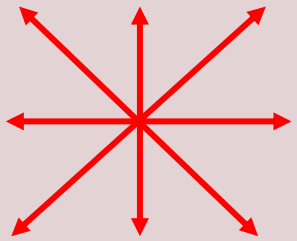
- We also can use the same Laplacian kernel for line detection.

0	0	0
0	0	0
0	0	0
255	255	255
255	255	255
255	255	255
0	0	0
0	0	0
0	0	0

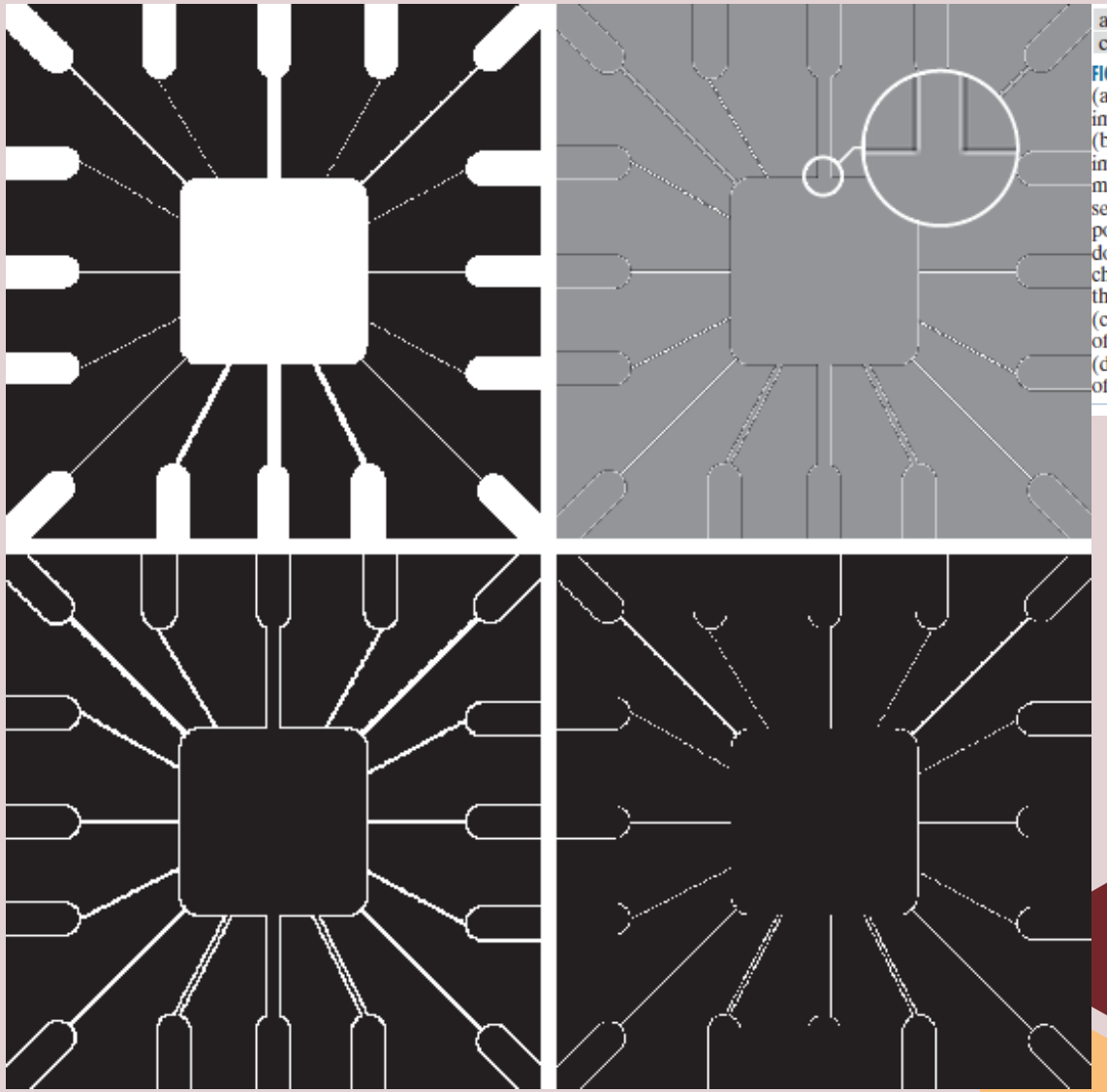
→

0
765
-765
0
-765
765
0

1	1	1
1	−8	1
1	1	1



Remember:
- Negative (light to dark)
- Positive (dark to light)



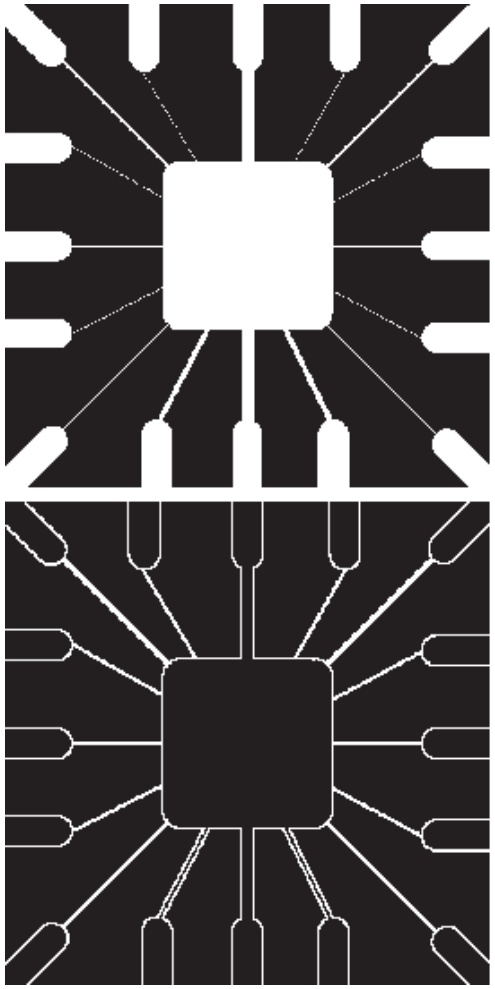
a b
c d

FIGURE 10.5
(a) Original image.
(b) Laplacian image; the magnified section shows the positive/negative double-line effect characteristic of the Laplacian.
(c) Absolute value of the Laplacian.
(d) Positive values of the Laplacian.

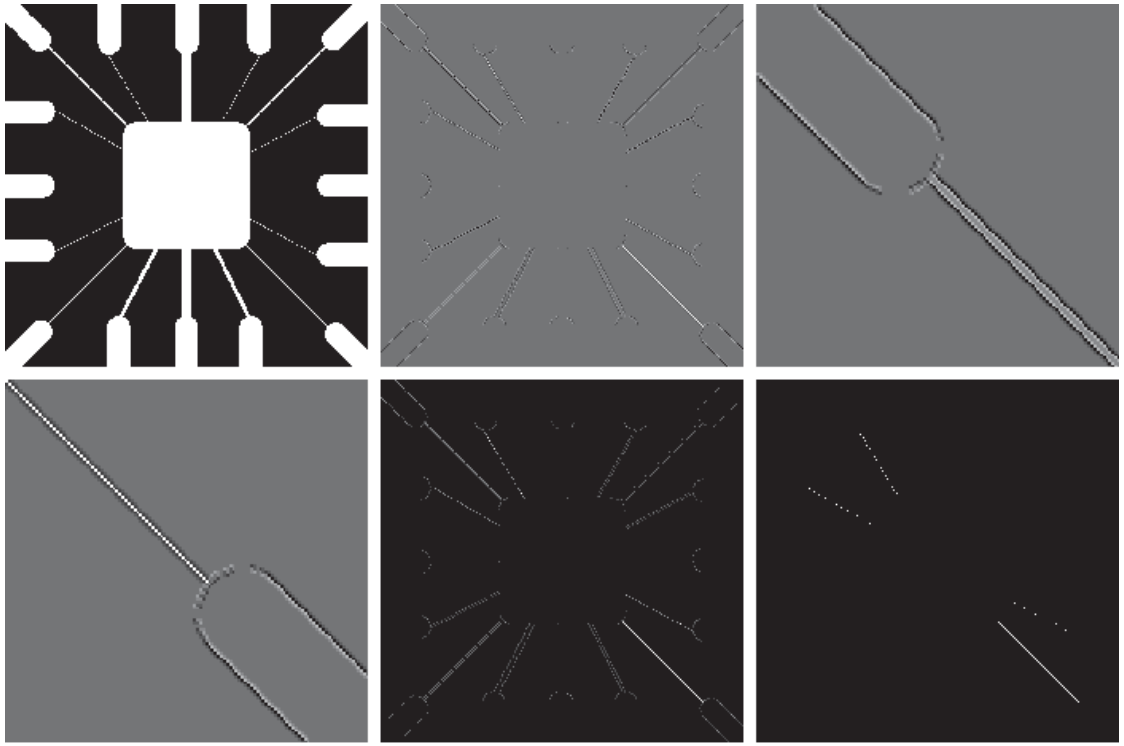
Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Detecting Lines in Specific Directions

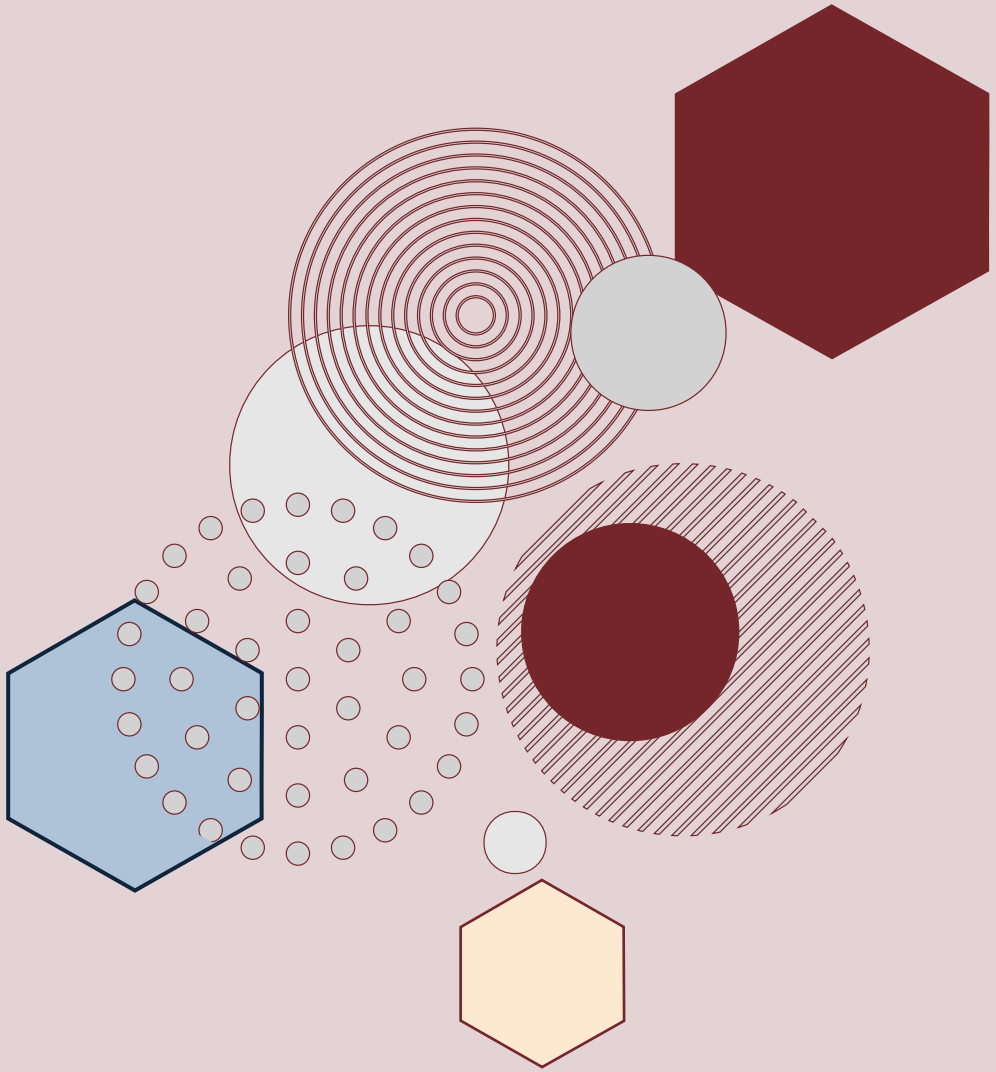
1	1	1
1	−8	1
1	1	1



−1	−1	−1	2	−1	−1	−1	2	−1	−1	−1	2
2	2	2	−1	2	−1	−1	2	−1	−1	2	−1
−1	−1	−1	−1	−1	2	−1	2	−1	2	−1	−1
Horizontal			+45°			Vertical			−45°		



Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

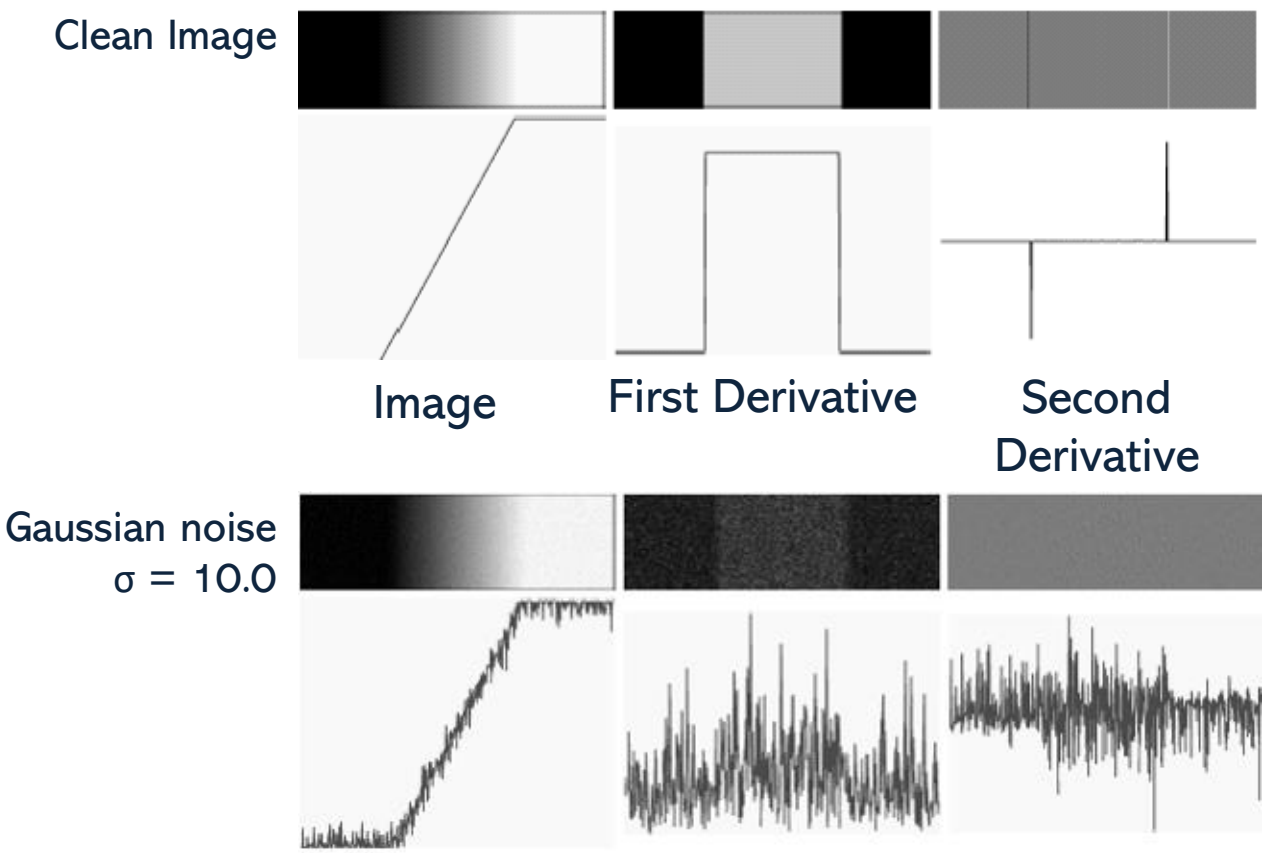


Edge-based Segmentation

Section 2.2.1 - Roberts, Prewitt, and Sobel Edge Detections

Edge with Some Noises

- **The problem:** Edge detection is only based on the intensity values, so there will be some problems when there are some noises in an image.



Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Some Important Notes about Edge Detection

- Edge detection is very sensitive to spatial noises.
- The flow processes in edge detections:
    1. Image smoothing for noise reduction
    2. Detection of edge points
    3. Edge localization

[Math] Gradient Operators

- The tool of choice for finding edge magnitude and direction at an arbitrary location (x, y) of an image, f, is the *gradient* denoted by vector ∇f.
- The gradient : $\nabla f(x,y) \equiv \text{grad}[f(x,y)] \equiv \begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix} = \begin{bmatrix} \partial f(x,y)/\partial x \\ \partial f(x,y)/\partial y \end{bmatrix}$
- The magnitude : $M(x,y) = \|\nabla f(x,y)\| = \sqrt{g_x^2(x,y) + g_y^2(x,y)}$
- The direction : $\alpha(x,y) = \tan^{-1}\begin{bmatrix} g_x(x,y) \\ g_y(x,y) \end{bmatrix}$

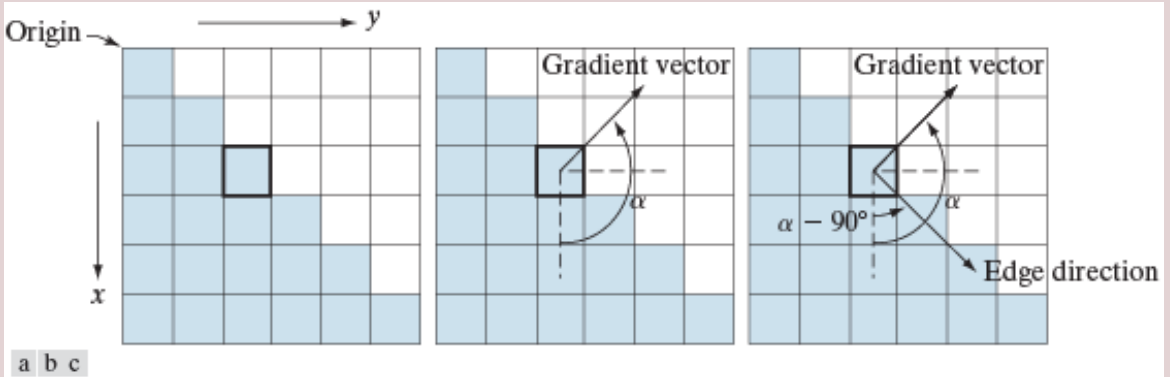
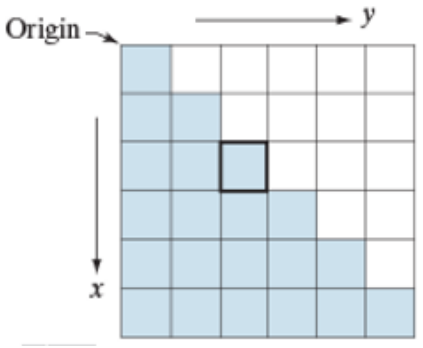


FIGURE 10.12 Using the gradient to determine edge strength and direction at a point. Note that the edge direction is perpendicular to the direction of the gradient vector at the point where the gradient is computed. Each square represents one pixel. (Recall from Fig. 2.19 that the origin of our coordinate system is at the top, left.)

[Math] Gradient Operators – The Kernels

- For the partial derivatives $\partial f/\partial x$ and $\partial f/\partial y$, we typically use a forward or centered finite difference.
- Using forward difference:
  - $g_x(x,y) = {}^{\partial f(x,y)}/_{\partial x} = f(x+1,y) - f(x,y)$ → $\begin{bmatrix}-1\\1\end{bmatrix}$
  - $g_y(x,y) = {}^{\partial f(x,y)}/_{\partial y} = f(x,y+1) - f(x,y)$ → $\begin{bmatrix}-1 & 1\end{bmatrix}$



- Exercises!
  - Calculate the following matrices by using the two kernels! (* is convolution)

$$\begin{bmatrix}0&0&1&1&0&0\\0&0&1&1&0&0\end{bmatrix} * \begin{bmatrix}-1 & 1\end{bmatrix} = \begin{bmatrix}&&&&&\\&&&&&\end{bmatrix}$$

$$\begin{bmatrix}0&0&1&1&0&0\\0&0&1&1&0&0\end{bmatrix} * \begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}&&&&&\end{bmatrix}$$

$$\begin{bmatrix}0&0\\0&0\\1&1\\1&1\\0&0\\0&0\end{bmatrix} * \begin{bmatrix}-1\\1\end{bmatrix} = \begin{bmatrix}&\\&\\&\\&\\&\end{bmatrix}$$

$$\begin{bmatrix}0&0\\0&0\\1&1\\1&1\\0&0\\0&0\end{bmatrix} * \begin{bmatrix}-1 & 1\end{bmatrix} = \begin{bmatrix}\\\\\\\\\\\end{bmatrix}$$

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Basic Gradient Operators for Edge Detection

- When diagonal edge direction is of interest, we need 2D kernels.
- Some basic edge detectors:
  - Roberts
  - Prewitt
  - Sobel

−1	0	0	−1
0	1	1	0

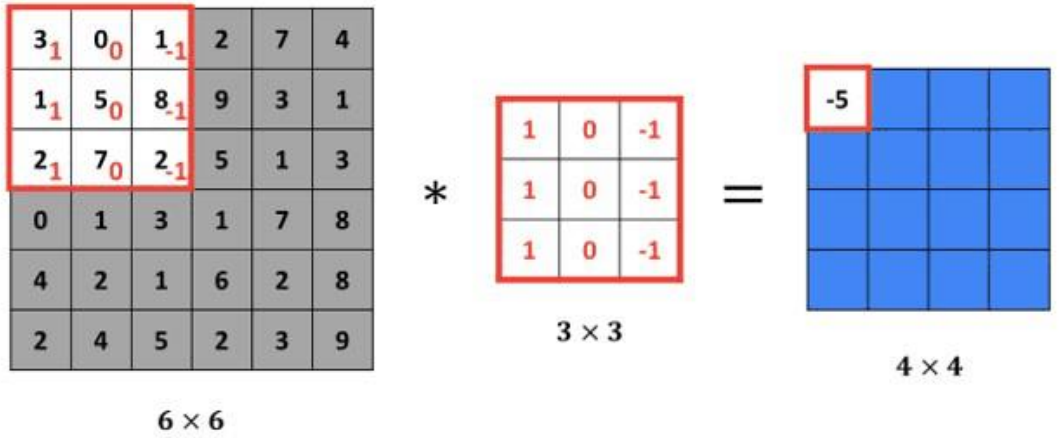
Roberts

−1	−1	−1	−1	0	1
0	0	0	−1	0	1
1	1	1	−1	0	1

Prewitt

−1	−2	−1	−1	0	1
0	0	0	−2	0	2
1	2	1	−1	0	1

Sobel



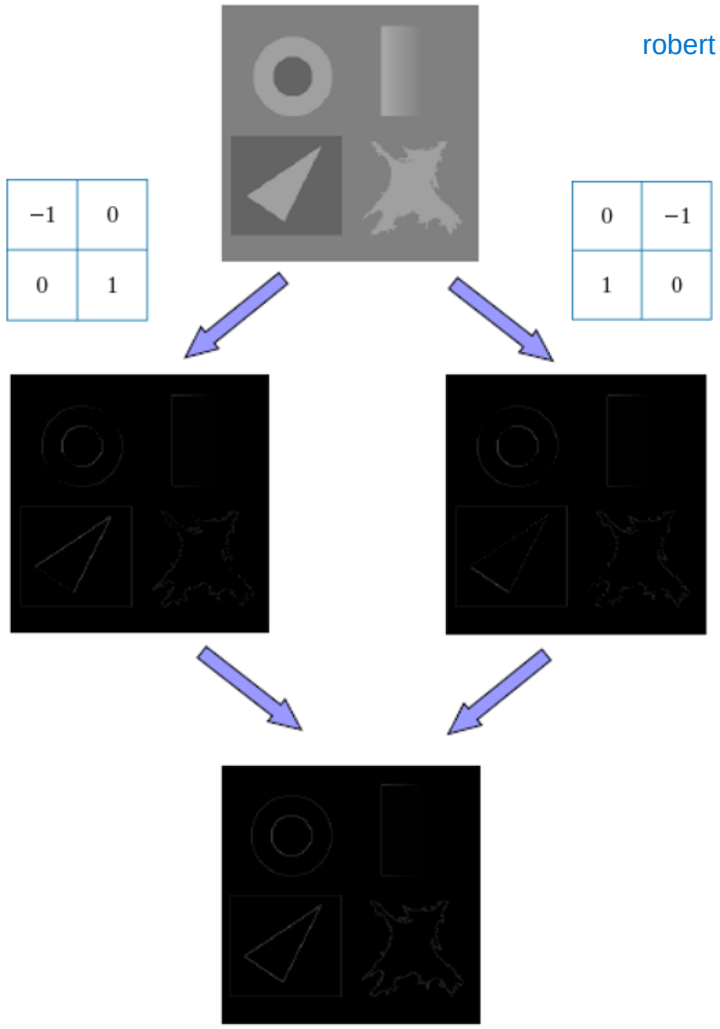
$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times -1 + 8 \times -1 + 2 \times -1 = -5$$

https://datahacker.rs/004-how-to-smooth-and-sharpen-an-image-in-opencv/

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

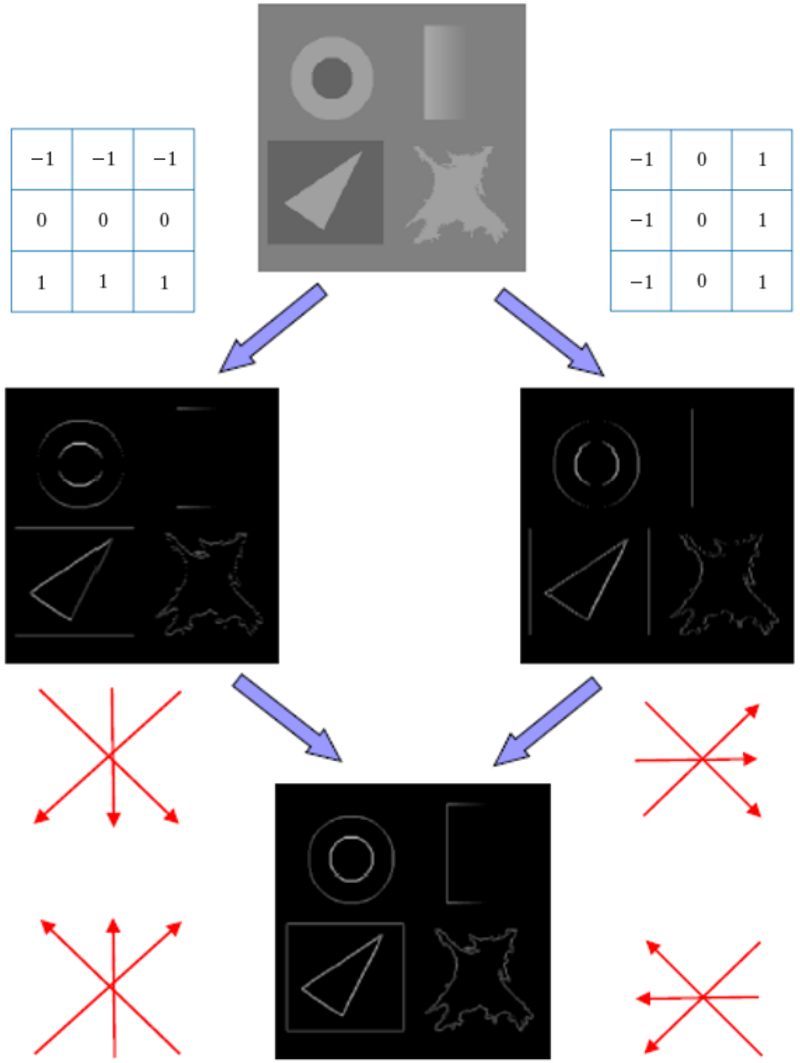
Roberts Gradient Operator

robert iitu 0 sama 1



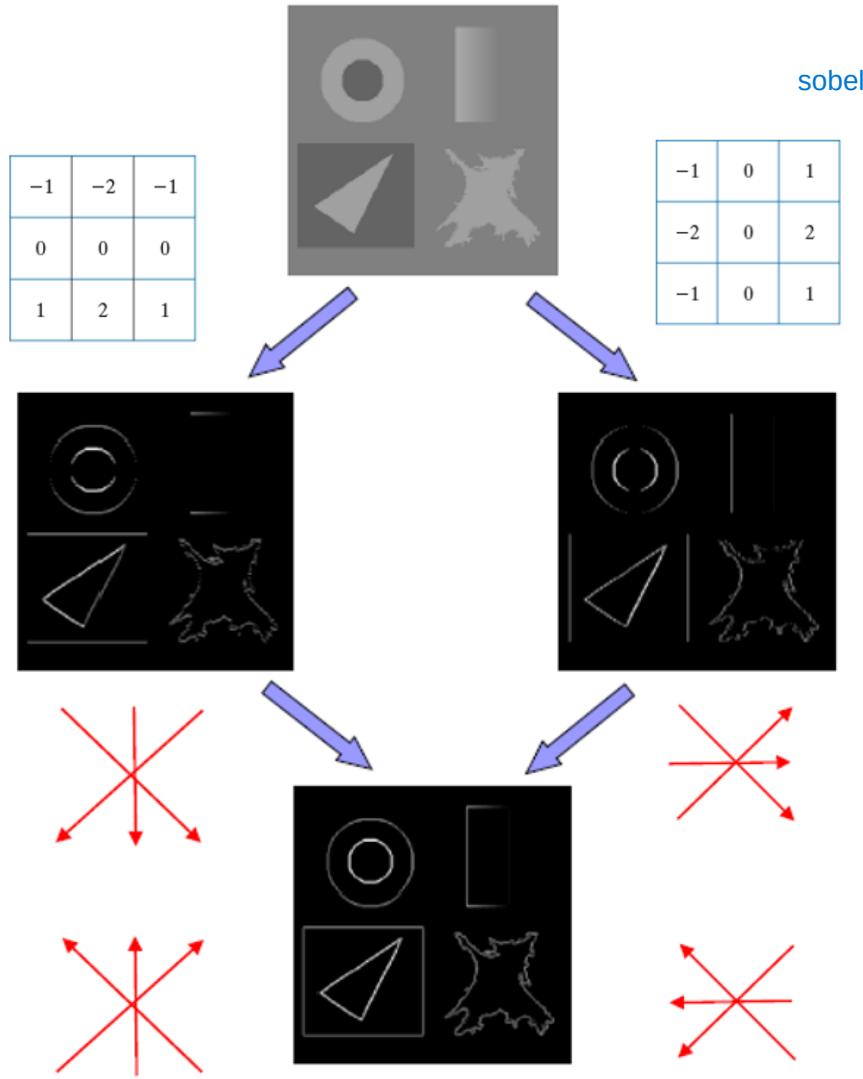
Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Prewitt Gradient Operator



Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Sobel Gradient Operator



sobel ada 2

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Comparison of Different Gradient Operators

kalau datanya cuku 2x2, maka bisa pake robert

Kalau misalnya 2x2 itu tidak cukup dan mau lebih mendatail

kita bisa pake prewitt/sobel.

Original

Roberts

Prewitt

Sobel

Sobel Edge Detection

Original Image

ini noisy karena disetiap change yang kecil. Dia akan mendeteksi semua edge.

The gradient image $|g_x|$ in the x-direction

The gradient image $|g_y|$ in the y-direction

The gradient image $|g_x|+|g_y|$

A bit noisy

Sobel Edge Detection + Smoothing (pre-processing)

Original Image

+ prior smoothing

ini untuk hilangin noise.

The gradient image $|g_x|$ in the x-direction

The gradient image $|g_y|$ in the y-direction

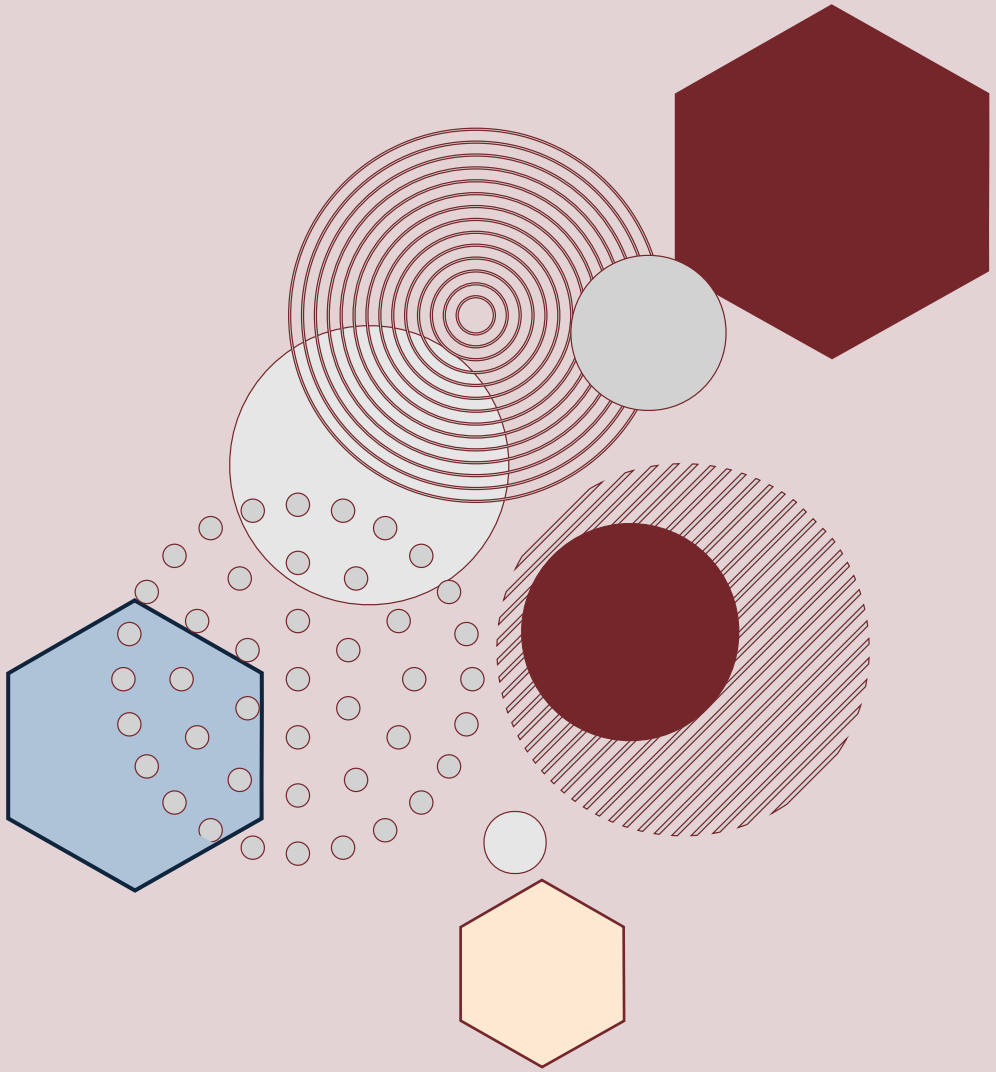
The gradient image $|g_x|+|g_y|$

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Sobel Edge Detection + Thresholding (post-processing)

- To determine which edges are “significant”, it is also possible to threshold the edge image.

Nah kalau misalnya mau cari edge yang signifikan, kita bisa set thresholding agar kita menentukan itu edge atau engga.



Edge-based Segmentation

Section 2.2.2 - Canny Edge Detection

Canny Edge Detector

- A multi-step algorithm that can detect edges with noise suppressed at the same time (John F. Canny, 1987).

- Main Objectives:
  a) Low error rate: Finding all edges and nothing but edges.
  b) Localization of edges: Distance between edges found and actual edges should be minimized.
  c) Single response: No multiple edge pixels when only a single edge exists.

- Canny edge detector is superior in general to the edge detectors discussed thus far.

Computation of Canny Edge Detector

1. Smooth the input image with a Gaussian filter

$$G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \qquad f_s(x,y) = G(x,y) * f(x,y)$$

2. Compute the gradient magnitude and angle images

$$M(x,y) = \sqrt{g_x^2 + g_y^2} \qquad \alpha(x,y) = \tan^{-1}\frac{g_y}{g_x}$$

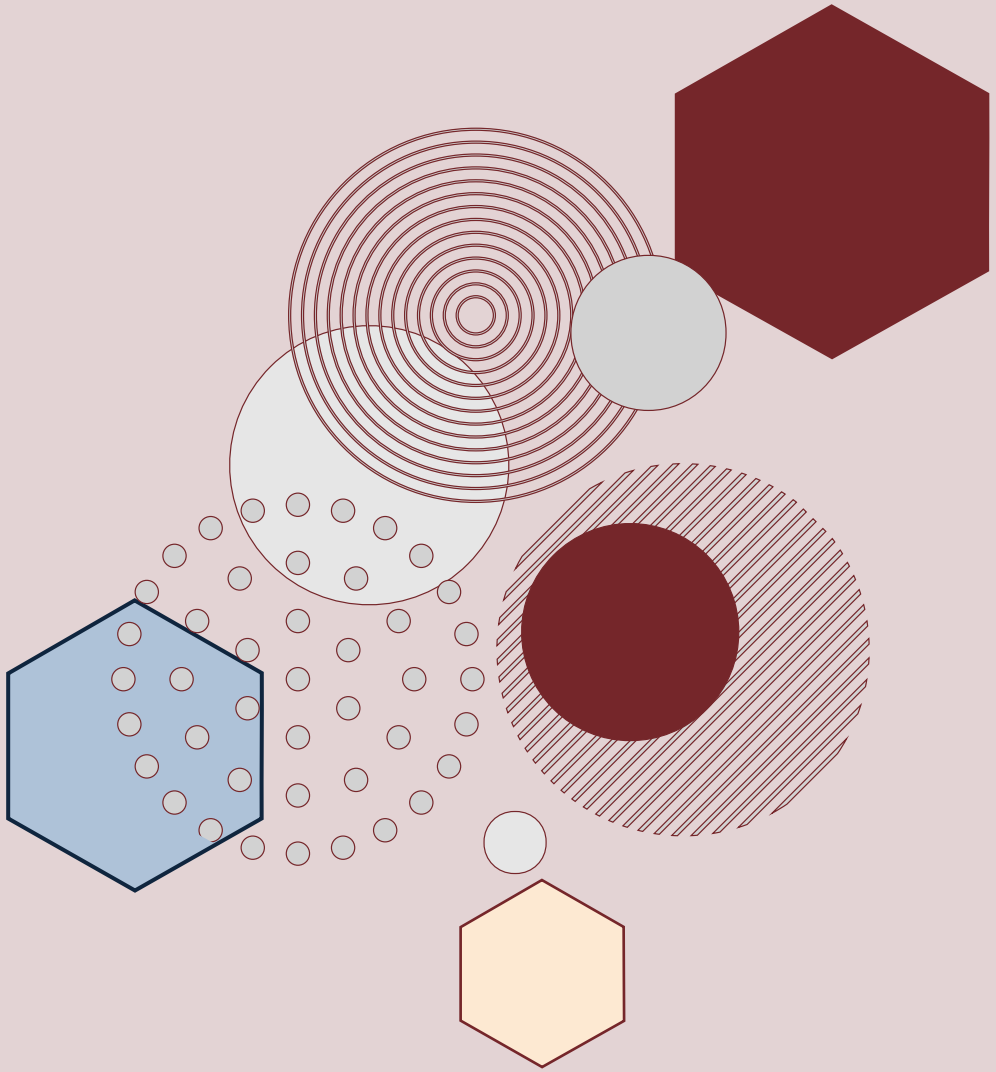
3. Apply non-maximum suppression to the gradient magnitude
   - Get rid of spurious response to edge detection
4. Use double thresholding to determine potential edges
5. Track edge by using connectivity analysis (*hysteresis*) to link edges
   - To avoid false edges

More details:
https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html

Illustration of Canny Edge Detection

a b
c d

FIGURE 10.25
(a) Original image of size 834×1114 pixels, with intensity values scaled to the range $[0, 1]$.
(b) Thresholded gradient of the smoothed image.
(c) Image obtained using the Marr-Hildreth algorithm.
(d) Image obtained using the Canny algorithm. Note the significant improvement of the Canny image compared to the other two.

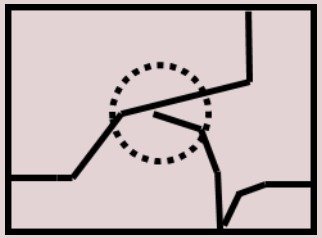


Edge-based Segmentation

Section 2.3 - Post-processing Techniques After Edge Detection (Essential to Know, Supplementary)

Why post-processing methods are needed?

- Ideally, edge detection should yield sets of pixels lying only on edges.
- In practice, these pixels seldom characterize edges completely because of noise, nonuniform illumination, and other effects that introduce discontinuities.
- Linking algorithms are designed to assemble edge pixels into meaningful edges and/or region boundaries.
- Two kinds of linking algorithms:
  a) Local (neighborhood) processing
     - We know the gradient's magnitude (M) and its angel (α) of an edge, so we can close the neighboring gaps that do not exceed a specified length L.
  b) Global processing
     - Global processing is applicable when we work in an unstructured environment (i.e., all we have is an edge map and no knowledge about where objects of interest might be. One example is the Hough Transform.



Post-processing Techniques After Edge Detection (Essential to Know, Supplementary)

2.3.1 Hough Transform

- Line: https://docs.opencv.org/4.x/d6/d10/tutorial_py_houghlines.html
- Circle: https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html

2.3.2 Contours

- https://docs.opencv.org/4.x/d3/d05/tutorial_py_table_of_contents_contours.html
- https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/

2.3.3 Convex hull

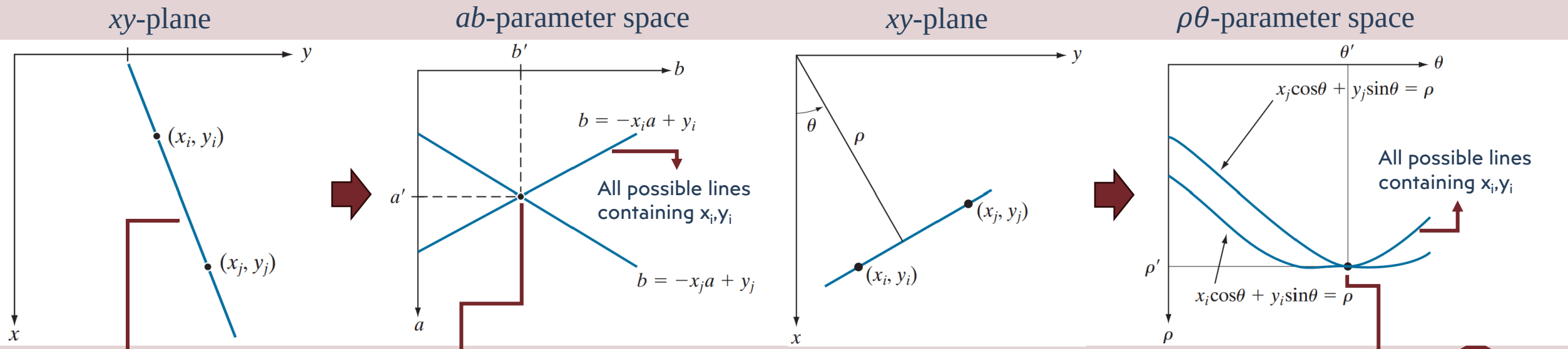
- https://www.geeksforgeeks.org/convex-hull-algorithm/
- https://learnopencv.com/convex-hull-using-opencv-in-python-and-c/

2.3.3 Other contours

- https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html

Hough Transform (Lines)

- The Hough Transform is a popular technique for detecting any shape if you can represent it mathematically. It can detect a shape even if it is broken or distorted a bit.
- A line can be represented as y=ax+b or in a parametric form, as $x.\cos\theta+y.\sin\theta=\rho$ where ρ is the perpendicular distance from the origin to the line, and θ is the angle formed by this perpendicular line and the horizontal axis measured counter-clockwise.



$y = ax + b$

All points in the line have the same a and b parameters

b = -ax + y

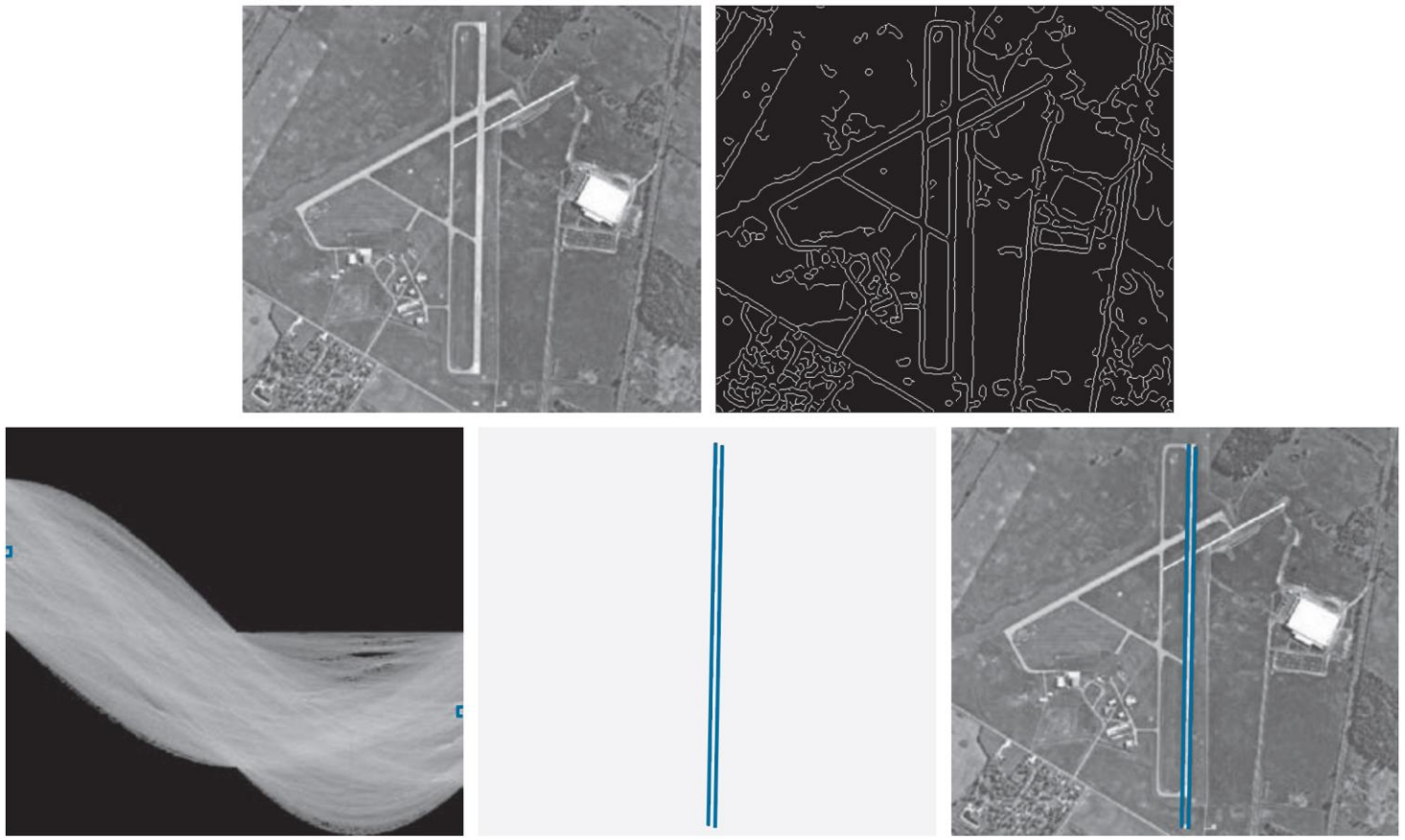
The line containing both x_i,y_i and x_i,y_i at the same time

$x.\cos\theta + y.\sin\theta = \rho$

The line containing both x_i,y_i and x_i,y_i at the same time

Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Multiple Lines from Hough Transform



a b
c d e

FIGURE 10.31 (a) A 502×564 aerial image of an airport. (b) Edge map obtained using Canny's algorithm. (c) Hough parameter space (the boxes highlight the points associated with long vertical lines). (d) Lines in the image plane corresponding to the points highlighted by the boxes. (e) Lines superimposed on the original image.

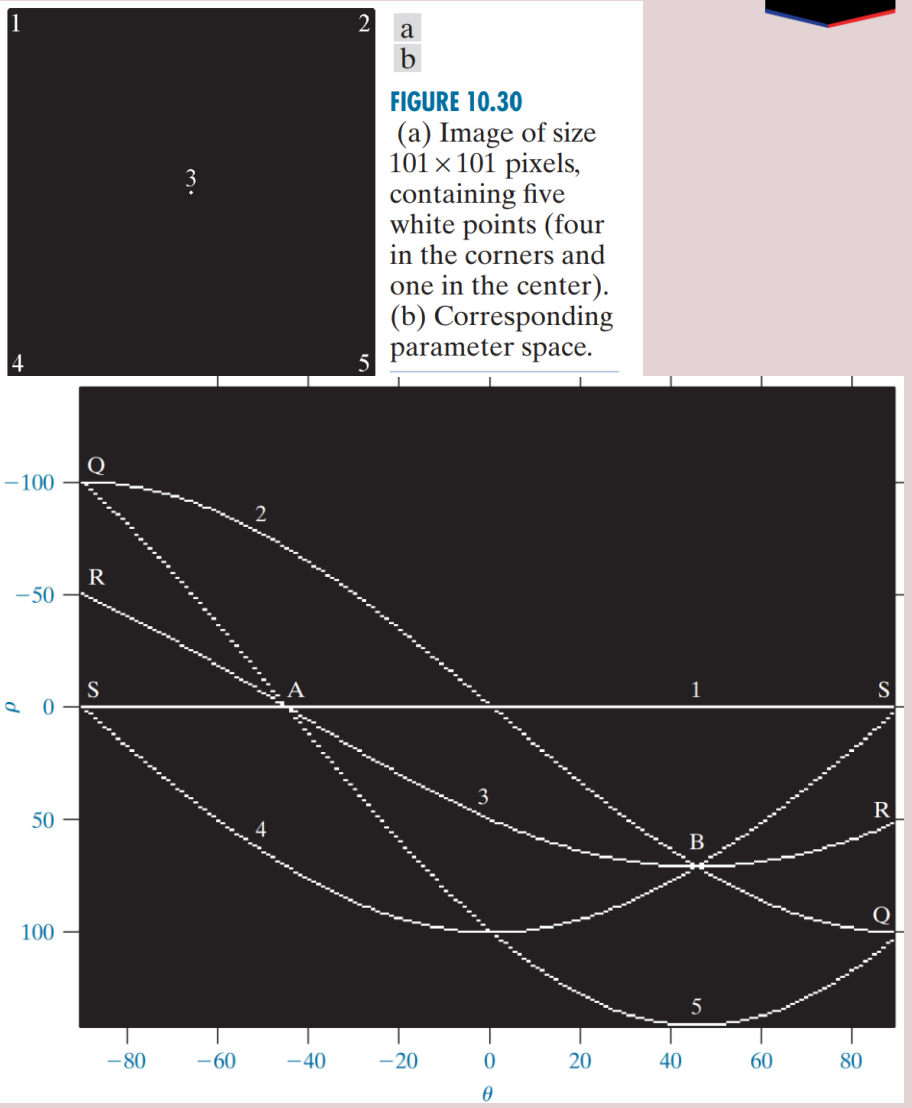
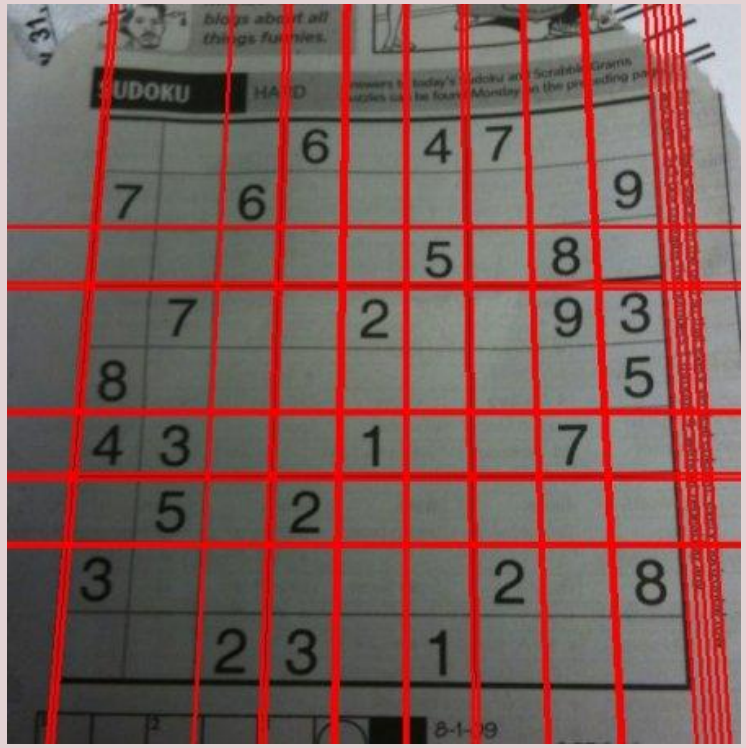


FIGURE 10.30 (a) Image of size 101×101 pixels, containing five white points (four in the corners and one in the center). (b) Corresponding parameter space.

HoughLines in OpenCV

```
import cv2 as cv
import numpy as np

img = cv.imread(cv.samples.findFile('sudoku.png'))
gray = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
edges = cv.Canny(gray,50,150,apertureSize = 3)

lines = cv.HoughLines(edges,1,np.pi/180,200)
for line in lines:
 rho,theta = line[0]
 a = np.cos(theta)
 b = np.sin(theta)
 x0 = a*rho
 y0 = b*rho
 x1 = int(x0 + 1000*(-b))
 y1 = int(y0 + 1000*(a))
 x2 = int(x0 - 1000*(-b))
 y2 = int(y0 - 1000*(a))

 cv.line(img,(x1,y1),(x2,y2),(0,0,255),2)

cv.imwrite('houghlines3.jpg',img)
```



https://docs.opencv.org/4.x/d6/d10/tutorial_py_houghlines.html

Hough Transform (Circles)

- A circle is represented mathematically as $(x-x_{center})^2+(y-y_{center})^2=r^2$ where (x_{center},y_{center}) is the center of the circle, and r is the radius of the circle. From the equation, we can see we have 3 parameters, so we need a 3D accumulator for the Hough transform, which would be highly ineffective.

```
import numpy as np
import cv2 as cv

img = cv.imread('opencv-logo-white.png', cv.IMREAD_GRAYSCALE)
assert img is not None, "file could not be read, check with os.path.exists()"
img = cv.medianBlur(img,5)
cimg = cv.cvtColor(img,cv.COLOR_GRAY2BGR)

circles = cv.HoughCircles(img,cv.HOUGH_GRADIENT,1,20,
 param1=50,param2=30,minRadius=0,maxRadius=0)

circles = np.uint16(np.around(circles))
for i in circles[0,:]:
 # draw the outer circle
 cv.circle(cimg,(i[0],i[1]),i[2],(0,255,0),2)
 # draw the center of the circle
 cv.circle(cimg,(i[0],i[1]),2,(0,0,255),3)

cv.imshow('detected circles',cimg)
cv.waitKey(0)
cv.destroyAllWindows()
```

image

https://docs.opencv.org/4.x/da/d53/tutorial_py_houghcircles.html

Contours

kita bisa detect contorus yang mempunya warana yang sama dengan jntensstis,

Garis-garis yang putus-putus belum tentnu contorus.

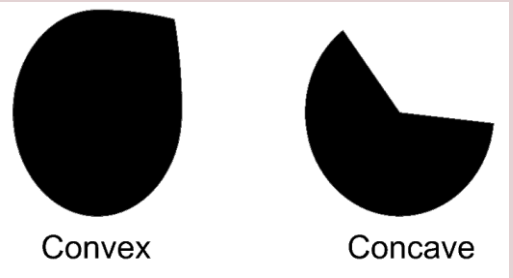
- Contours can be explained simply as a curve joining all the continuous points (along the boundary), having the same color or intensity.
- For better accuracy, use binary images. So before finding contours, apply threshold or canny edge detection. The contours are a useful tool for shape analysis and object detection and recognition.

```python
import cv2
import numpy as np

# Let's load a simple image with 3 black squares
image = cv2.imread('C://Users//gfg//shapes.jpg')
cv2.waitKey(0)

# Grayscale
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)

# Find Canny edges
edged = cv2.Canny(gray, 30, 200)
cv2.waitKey(0)

# Finding Contours
# Use a copy of the image e.g. edged.copy()
# since findContours alters the image
contours, hierarchy = cv2.findContours(edged,
    cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

cv2.imshow('Canny Edges After Contouring', edged)
cv2.waitKey(0)

print("Number of Contours found = " + str(len(contours)))

# Draw all contours
# -1 signifies drawing all contours
cv2.drawContours(image, contours, -1, (0, 255, 0), 3)

cv2.imshow('Contours', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

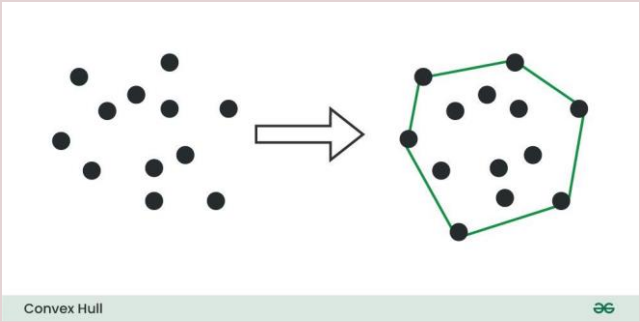
- https://docs.opencv.org/4.x/d3/d05/tutorial_py_table_of_contents_contours.html
- https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/

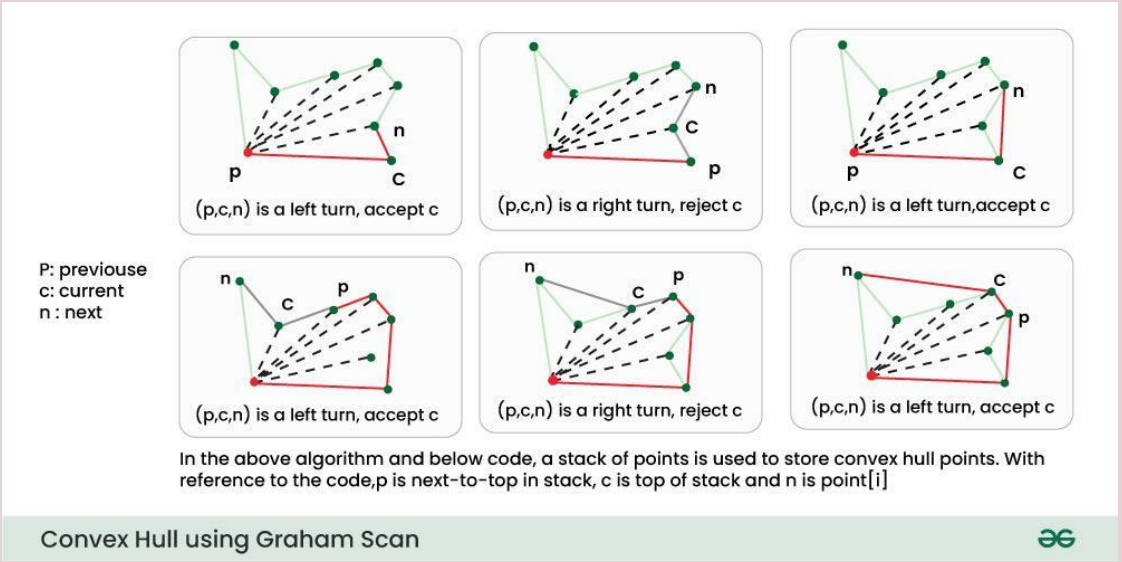
Convex hull

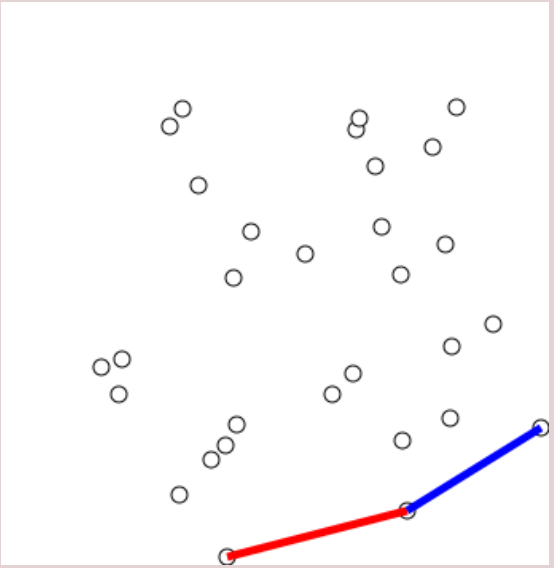
mirip kayak contours



- A convex object is one with no interior angles greater than 180 degrees.
- Hull means the exterior or the shape of the object.
- Convex Hull will look similar to contour approximation, but it is not (they may provide the same results in some cases). There are several methods to find the convex hull.





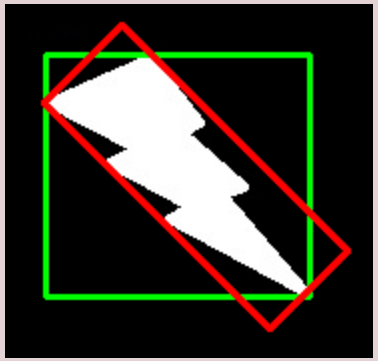


- https://www.geeksforgeeks.org/convex-hull-algorithm/
- https://learnopencv.com/convex-hull-using-opencv-in-python-and-c/
- https://commons.wikimedia.org/wiki/File:GrahamScanDemo.gif

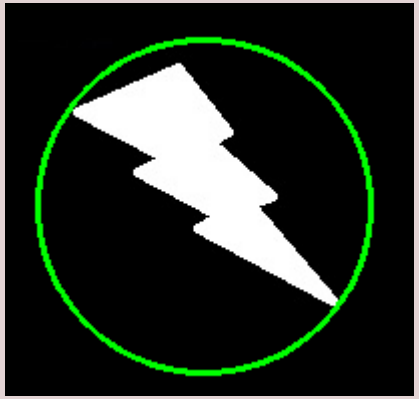
Other contours

- https://docs.opencv.org/4.x/dd/d49/tutorial_py_contour_features.html

Straight Bounding and Rotated Rectangles

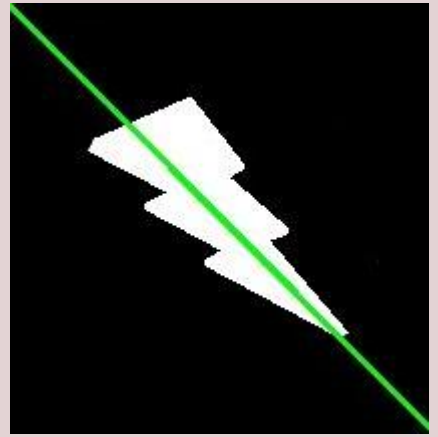


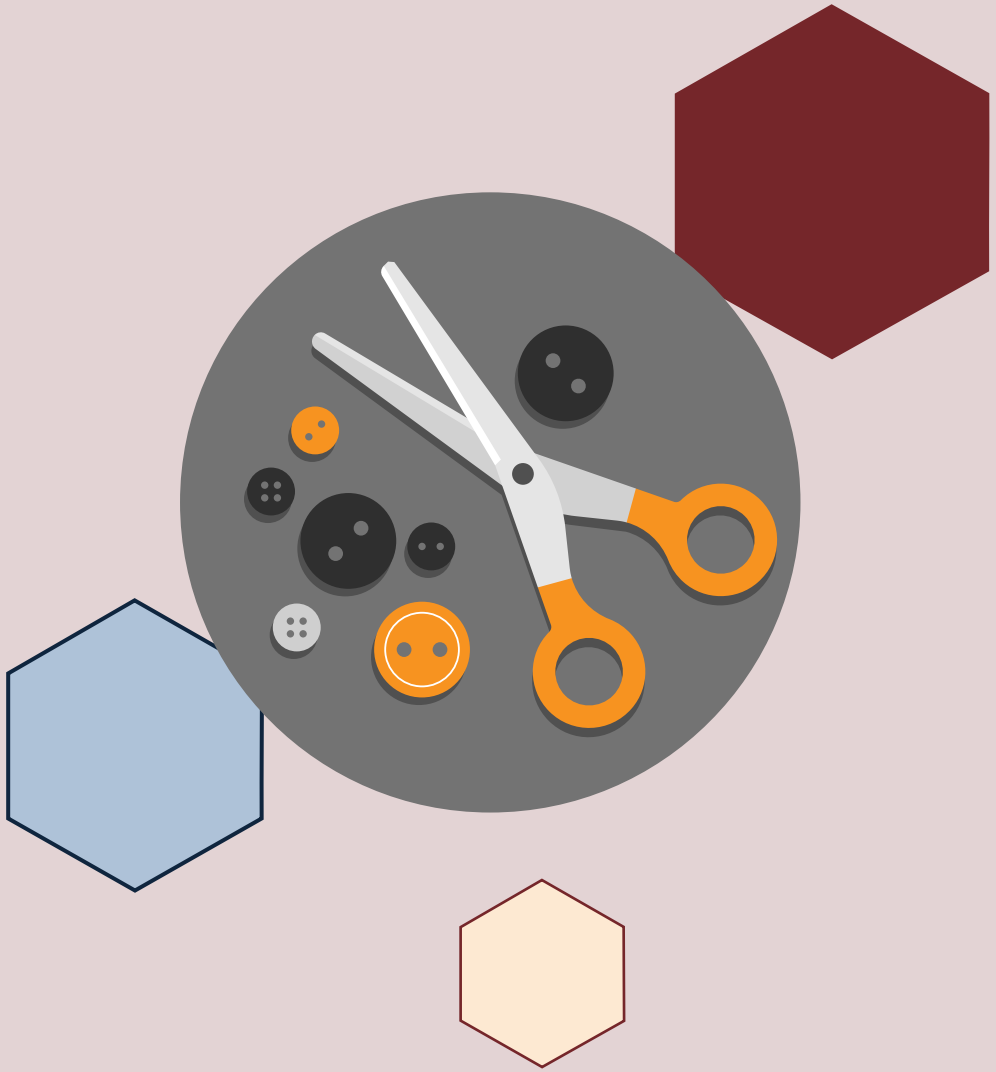
Minimum Enclosing Circle



Fitting an Ellipse

Fitting a Line





Distribution-/Threshold-based Segmentation

Section 3 - Distribution-/Threshold-based Segmentation

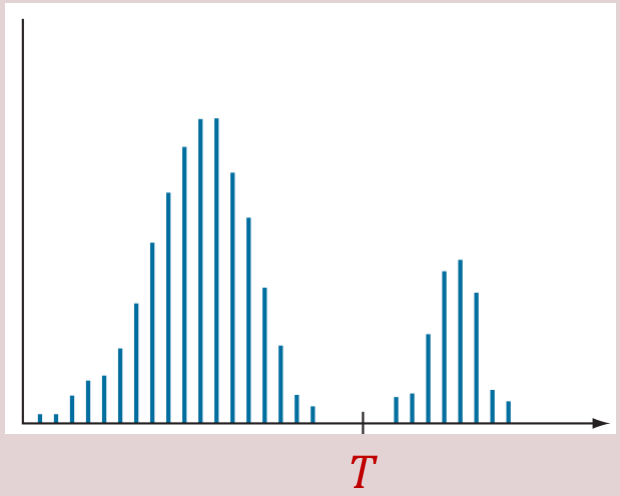
3.1 Optimum Global Thresholding – Otsu's Method

Histogram Thresholding (1/2)

- Suppose we have an intensity histogram

kita bisa nentuin T nya .

Kalau f(x, y) > T makanya dia bakal seperate image ke light, sebaliknya down.



- We can separate the image into ‘light’ and ‘dark’ areas with threshold T

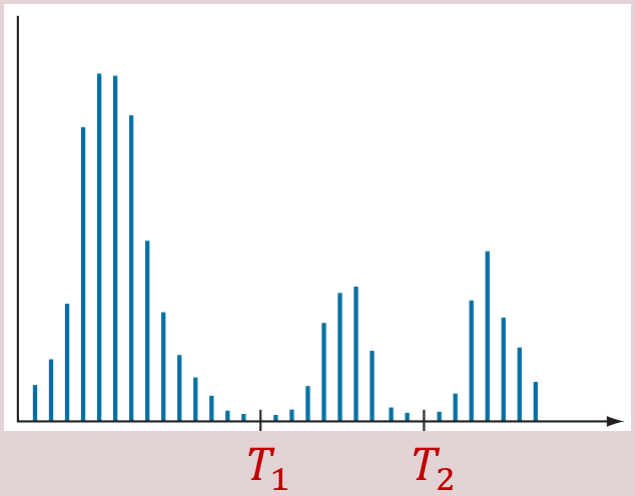
$$g(x,y) = \begin{cases} 1, & if\ f(x,y) > T \\ 0, & if\ f(x,y) \leq T \end{cases}$$

- If T is constant over image $f(x,y)$ → global thresholding
- If T varies over image → variable thresholding
  - If T varies based on location→ local/regional thresholding

Histogram Thresholding (2/2)

- Suppose we have a different intensity histogram

Kita juga bisa multiple thresholding dimana teridi dari 2 T.



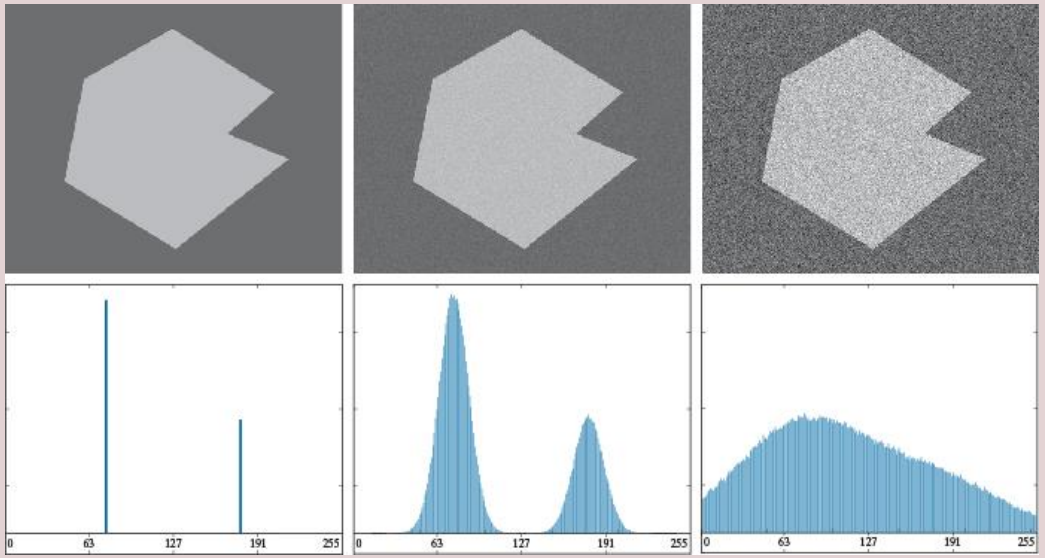
- We can separate the image uniform areas with thresholds T_i

$$g(x,y) = \begin{cases} a, & if\ f(x,y) > T_2 \\ b, & if\ T_1 < f(x,y) \leq T_2 \\ c, & if\ f(x,y) \leq T_1 \end{cases}$$

Potential Problems

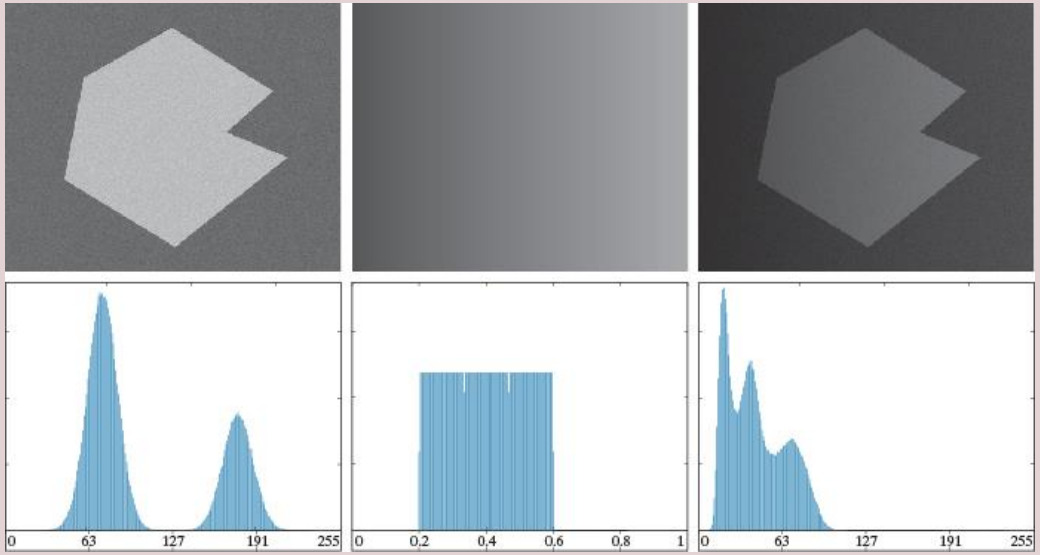
kalo gk ada noise thresholdnya jelas ada 2

- Noise in Images

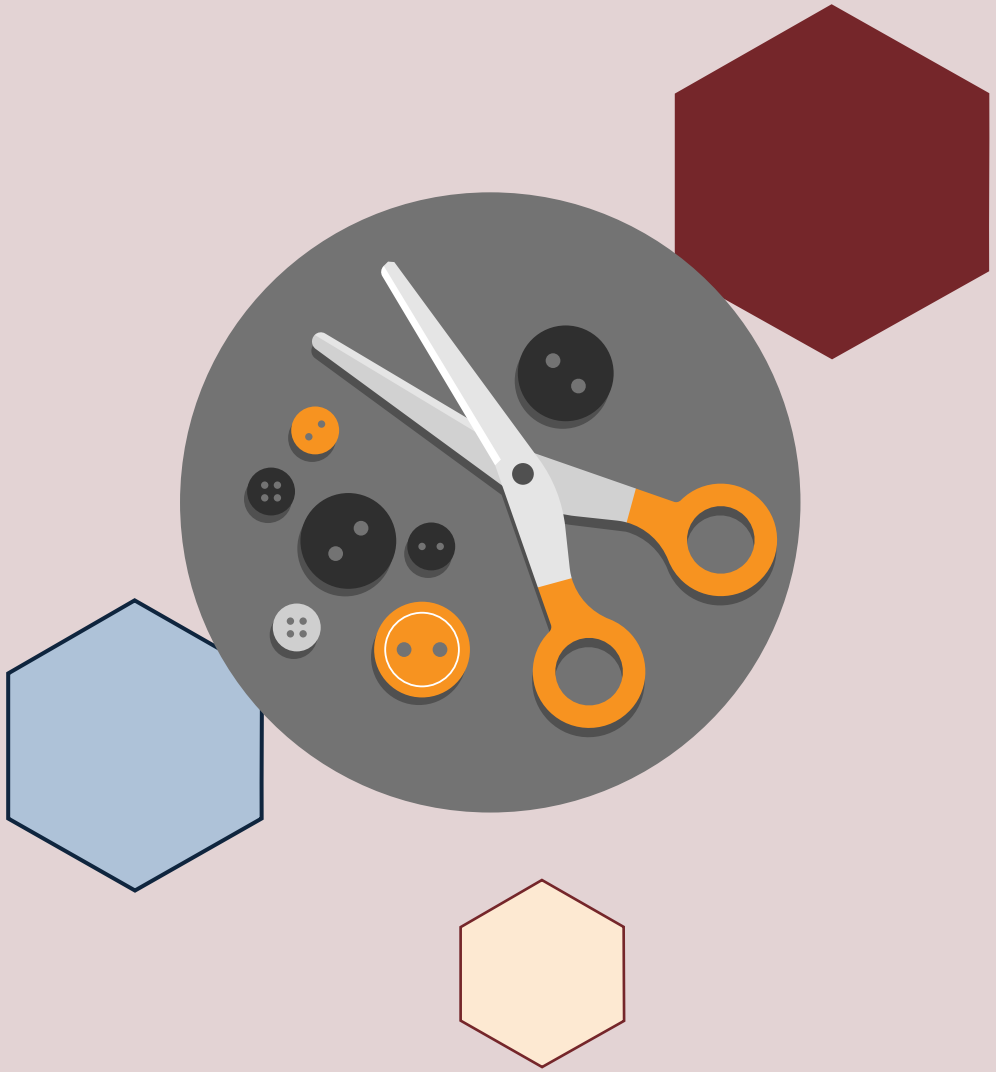


kalau thresholdnya itu ada noise dikit, bisa dilihat htresholdnya udah banyak bgt candidatenya.

- Poor Illumination



Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."



Distribution-/Threshold-based Segmentation

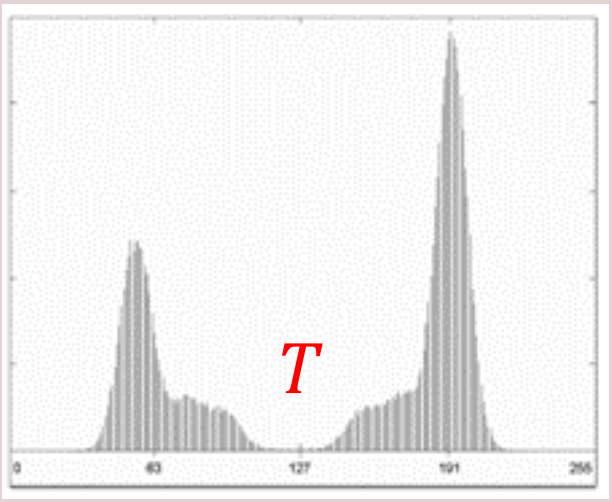
Section 3.1 - Optimum Global Thresholding – Otsu's Method

Basic Global Thresholding

- A method proposed by Nobuyuki Otsu (1979).
- Otsu's thresholding method is a binarization problem

Kita bisa pakai ottsu threshold untuk dapetin nilai T nya

Original Image



Histogram

Thresholded Image

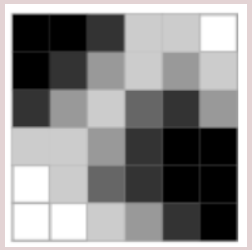
Which value of T should we take?

- The idea is to find the *ideal* threshold value to **minimize within-class variance** (and, at the same time, maximize between-class variance)

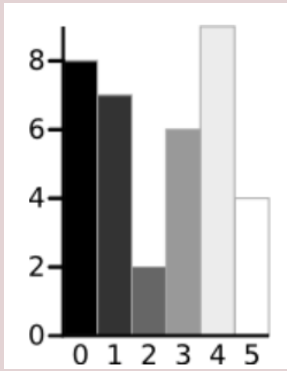
Gonzalez, R. C., and R. E. Woods. "Digital Image Processing, 4th Global Edition."

Otsu's Method – Computation (1/5)

- If we had 6 levels of intensity



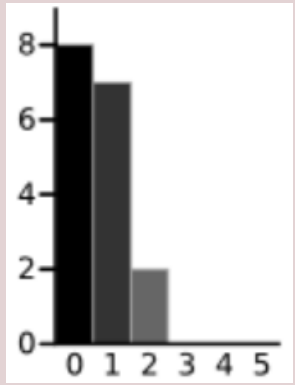
Image



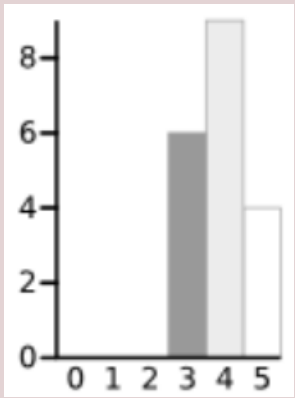
Histogram

- Suppose we take the threshold $T = 3$

Background



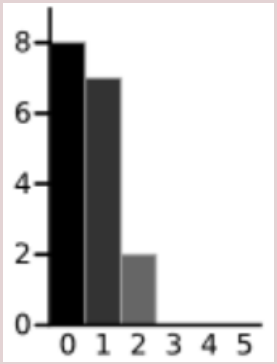
Foreground

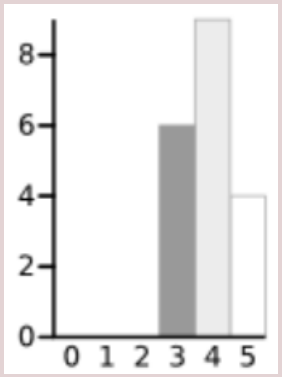


http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Otsu's Method – Computation (2/5)

- For these two classes

Background 

Foreground 

- Compute for L-1 bins
  - Probability (weight) of class

$$w_b = \frac{\sum_{i=0}^{t-1} p(i)}{N} \qquad w_f = \frac{\sum_{i=t}^{L-1} p(i)}{N}$$

N: total number of pixels
$p(i)$: frequency of intensity i
w_k : Weight of class k with threshold t

  - Variance of class

$$\sigma_b^2 = \sum_{i=0}^{t-1} (i - \mu_b)^2 \, p(i) \qquad \sigma_f^2 = \sum_{i=t}^{L-1} (i - \mu_f)^2 \, p(i)$$

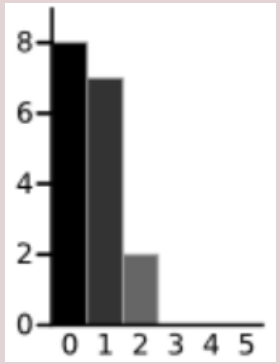
μ_k: Mean of class k

http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

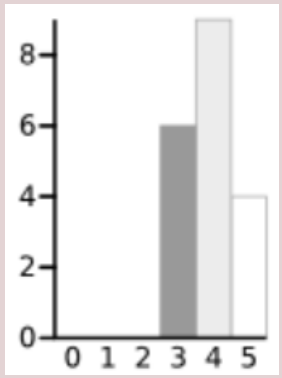
Otsu's Method – Computation (3/5)

- For these two classes

Background



Foreground



$$w_b = \frac{\sum_{i=0}^{t-1} p(i)}{N} \qquad w_f = \frac{\sum_{i=t}^{L-1} p(i)}{N}$$

- Compute for L-1 bins
  - Inter-class variance

$$\sigma_W^2 = w_b \sigma_b^2 + w_f \sigma_f^2$$

w_k: weight/probability of class k
σ_k^2: variance of class k

$$\sigma_b^2 = \sum_{i=0}^{t-1} (i - \mu_b)^2 \, p(i)$$

$$\sigma_f^2 = \sum_{i=t}^{L-1} (i - \mu_f)^2 \, p(i)$$

- Find threshold t where inter-class variance is minimum

http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

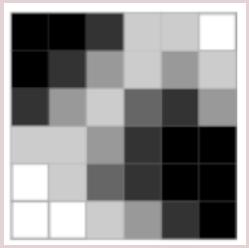
Otsu's Method – Computation (4/5)

- **Background:**
  - Weight = W_b = (8+7+2)/36 = 0.4722
  - Mean = μ_b = [(0x8)+(1x7)+(2x2)]/17 = 0.6471
  - Variance = σ_b^2 = [((0-0.6471)^{2}x8)+((1-0.06471)^{2}x7)+((2-0.6471)^{2}x2)]/17
    = [(0.4187x8) + (0.1246x7) + (1.8304x2)]/17 = 0.4637
- **Foreground:**
  - Weight = W_f = (6+9+4)/36 = 0.5278
  - Mean = μ_f = [(3x6)+(4x9)+(5x4)]/19 = 0.3.8947
  - Variance = σ_f^2 = [((3-3.8947)^{2}x6)+((4-3.8947)^{2}x9)+((5-3.8947)^{2}x4)]/19
    = [(4.8033x6) + (0.0997x9) + (4.8864x4)]/19 = 0.5152
- **Find threshold T where within-class variance is minimum**

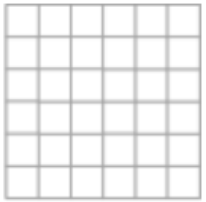
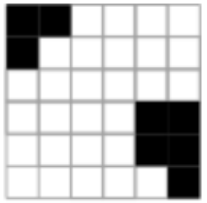
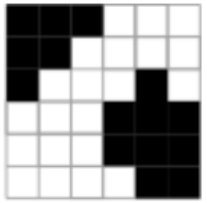
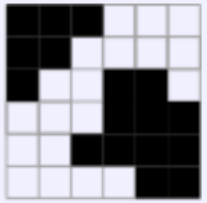
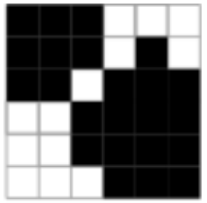
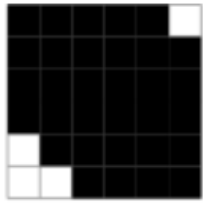
http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Otsu's Method – Computation (5/5)

- For our 6-intensity image



Image

Threshold	T=0	T=1	T=2	T=3	T=4	T=5
						
Within Class Variance	σ^2_W = 3.1196	σ^2_W = 1.5268	σ^2_W = 0.5561	σ^2_W = 0.4909	σ^2_W = 0.9779	σ^2_W = 2.2491

- Minimum intra-class (within-class) variance at $T = 3$
- Between-class variance: $\sigma_B^2 = W_b W_f (\mu_b - \mu_f)^2$

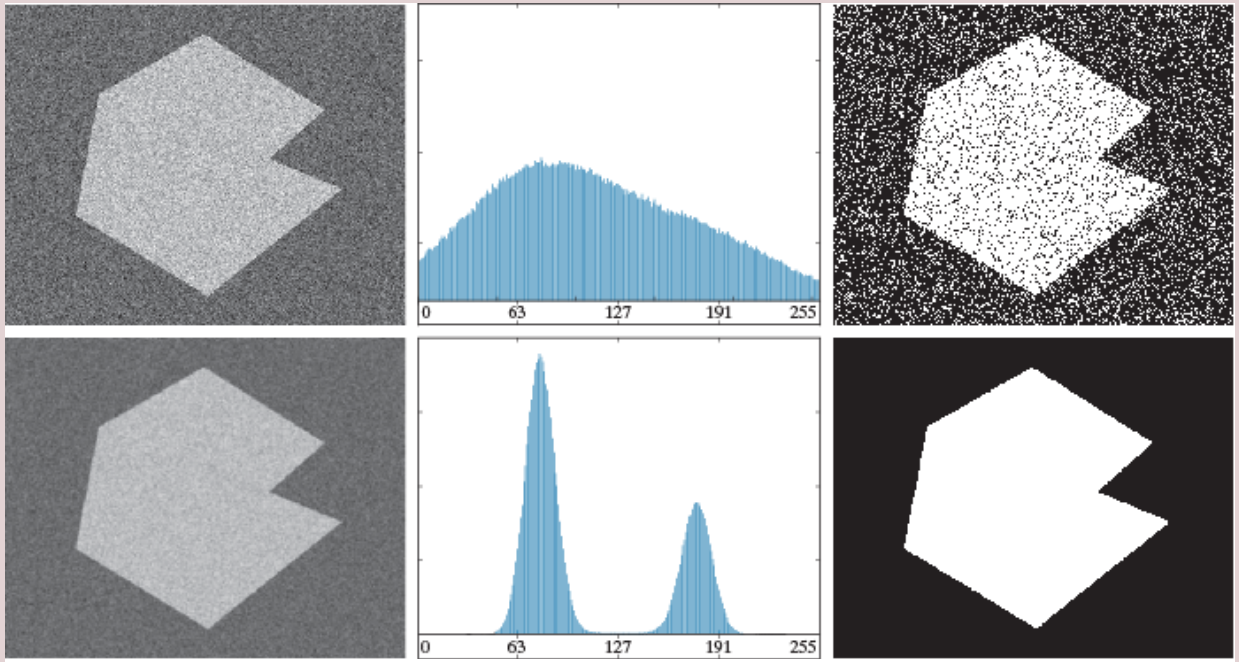
  For $T = 3$, $\sigma_B^2 = 0.4722 \times 0.5278(0.6471 - 3.8947)^2 = 2.6286$ (maximum)

http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html

Otsu's Method in Action

Kalau awal-awal kan gara2 noise, kita gak bisa untuk dapetin treshold, cuman kalau pakai ottsu kita bisa, dan nanti gamarnya tinggal di transfomr (pakai dilasi/erosi, dll)

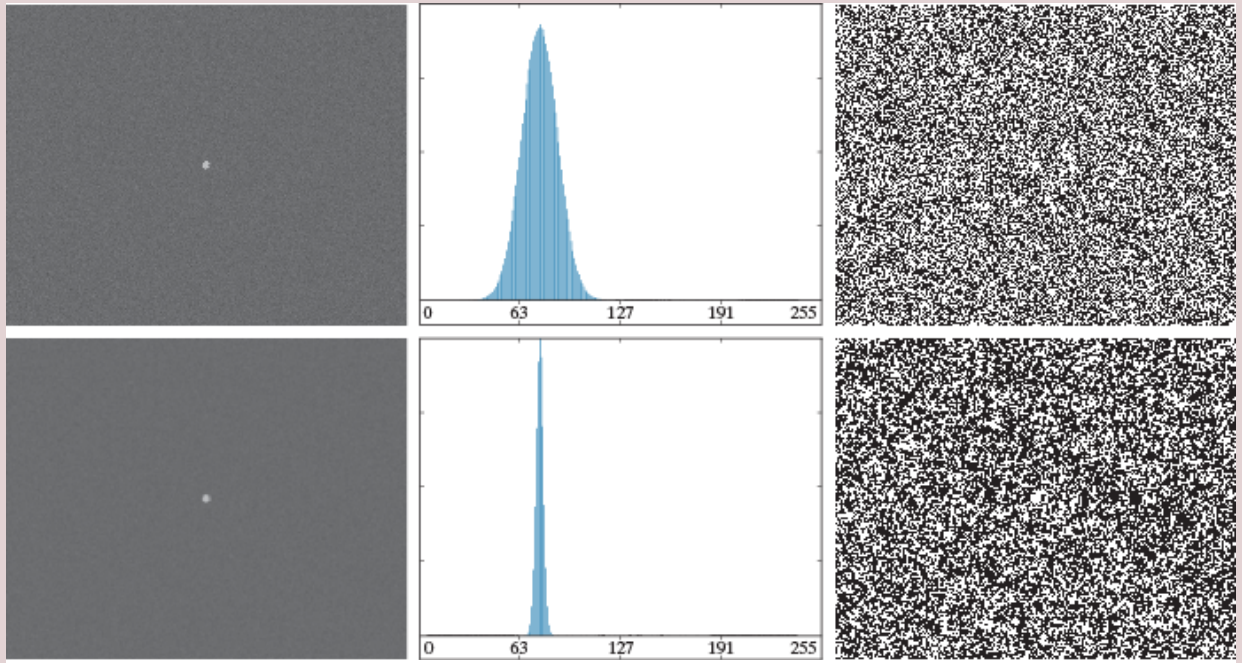
Successful Case



a b c
d e f

FIGURE 10.37 (a) Noisy image from Fig. 10.33(c) and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging kernel and (e) its histogram. (f) Result of thresholding using Otsu's method.

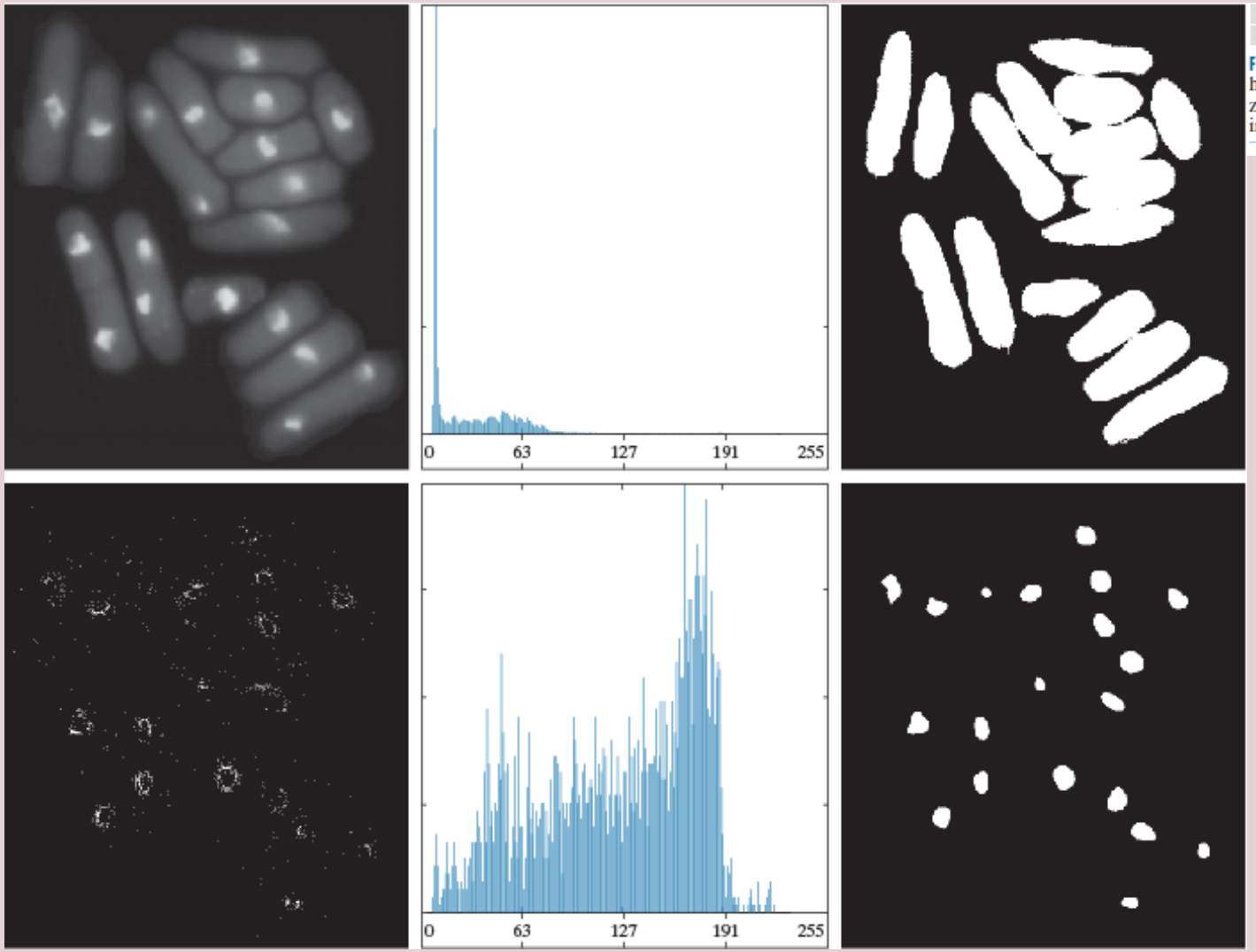
Failed Case



a b c
d e f

FIGURE 10.38 (a) Noisy image and (b) its histogram. (c) Result obtained using Otsu's method. (d) Noisy image smoothed using a 5×5 averaging kernel and (e) its histogram. (f) Result of thresholding using Otsu's method. Thresholding failed in both cases to extract the object of interest. (See Fig. 10.39 for a better solution.)

Edge Detection to Improve Otsu's Method

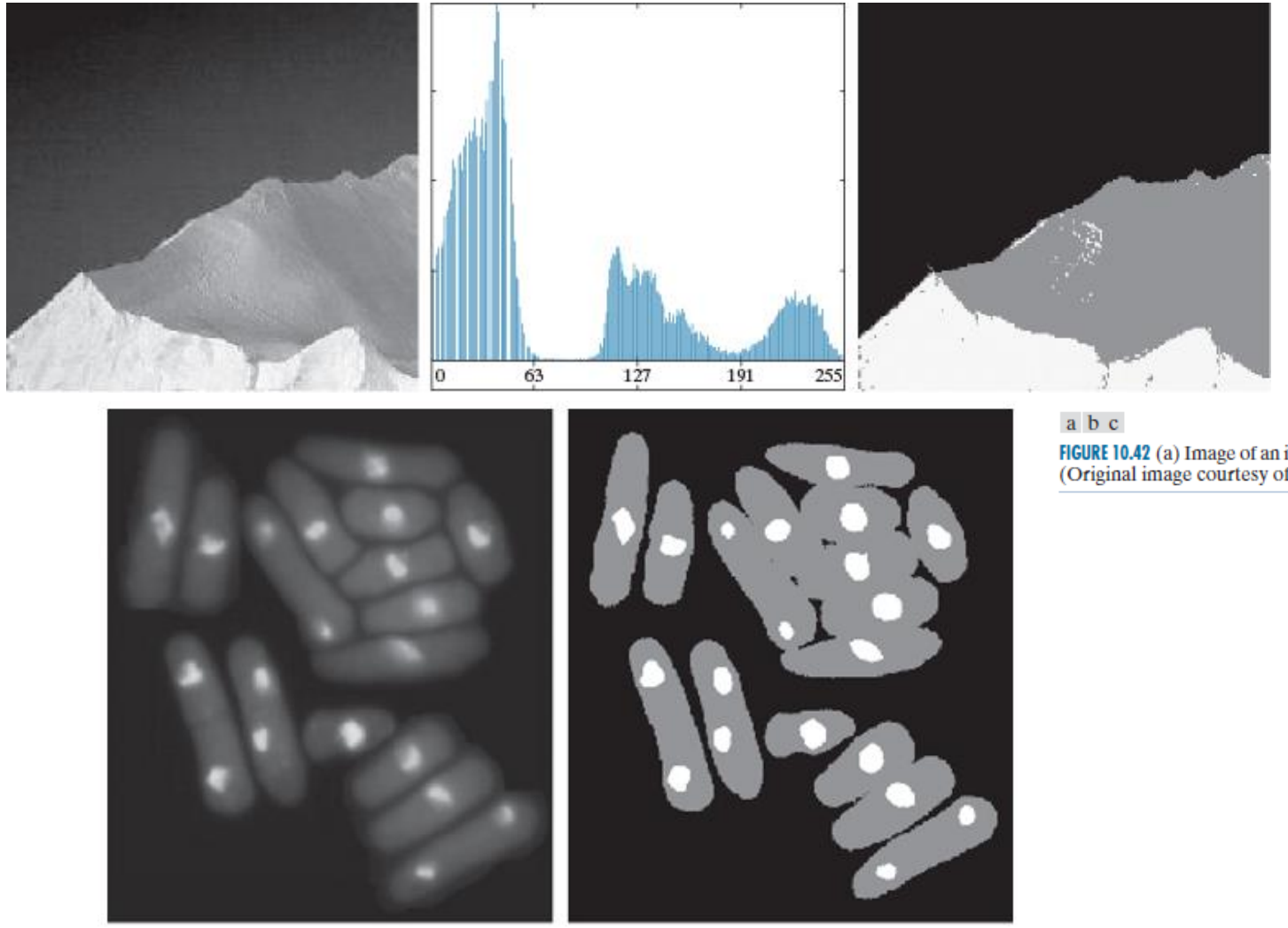


a b c
d e f

FIGURE 10.40 (a) Image of yeast cells. (b) Histogram of (a). (c) Segmentation of (a) with Otsu's method using the histogram in (b). (d) Mask image formed by thresholding the absolute Laplacian image. (e) Histogram of the non-zero pixels in the product of (a) and (d). (f) Original image thresholded using Otsu's method based on the histogram in (e). (Original image courtesy of Professor Susan L. Forsburg, University of Southern California.)

ottsu bisa membedakan beberapa class

Multiple Global Thresholding using Otsu's Method



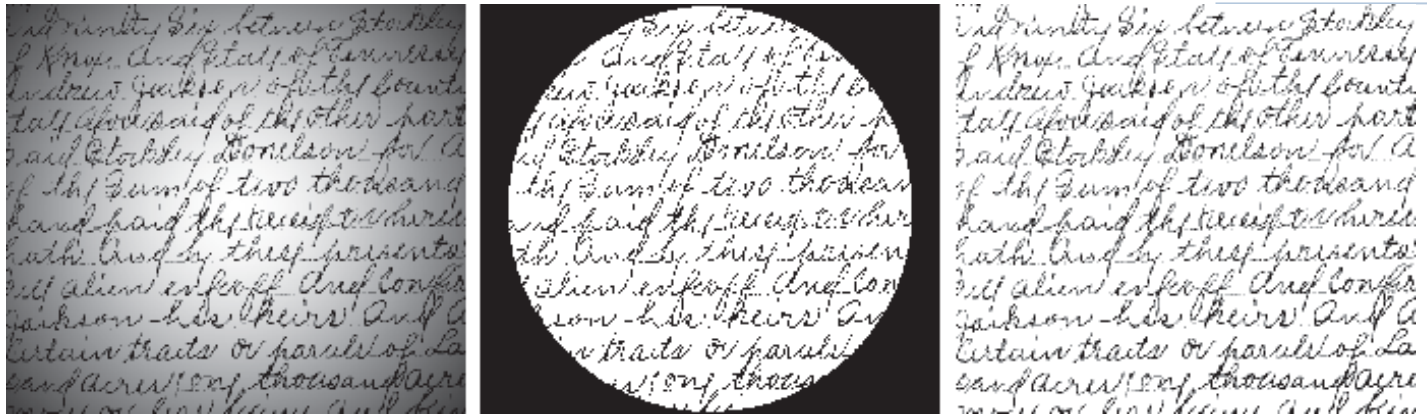
a b c

FIGURE 10.42 (a) Image of an iceberg. (b) Histogram. (c) Image segmented into three regions using dual Otsu thresholds. (Original image courtesy of NOAA.)

[Supplementary] Variable Thresholding

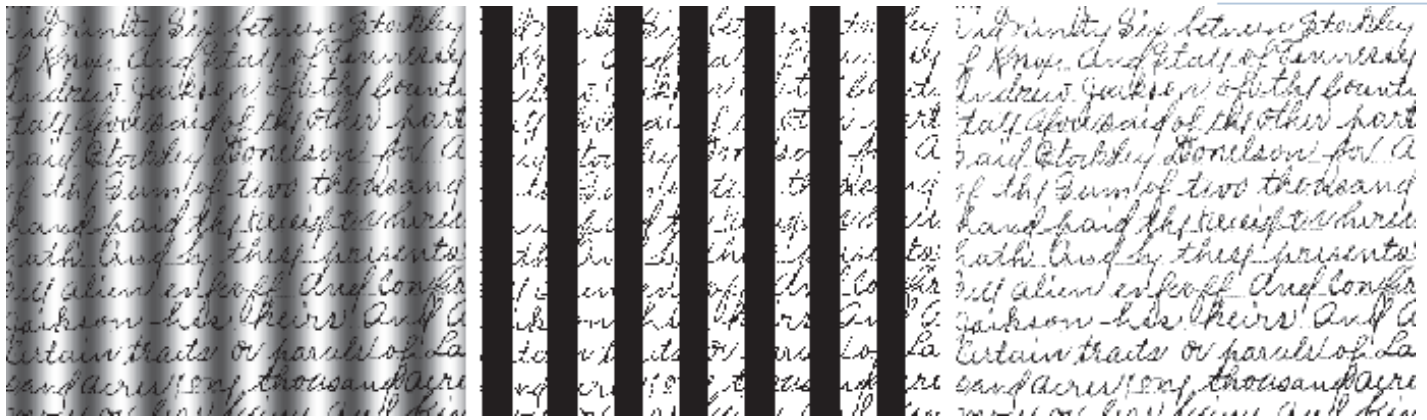
a b c

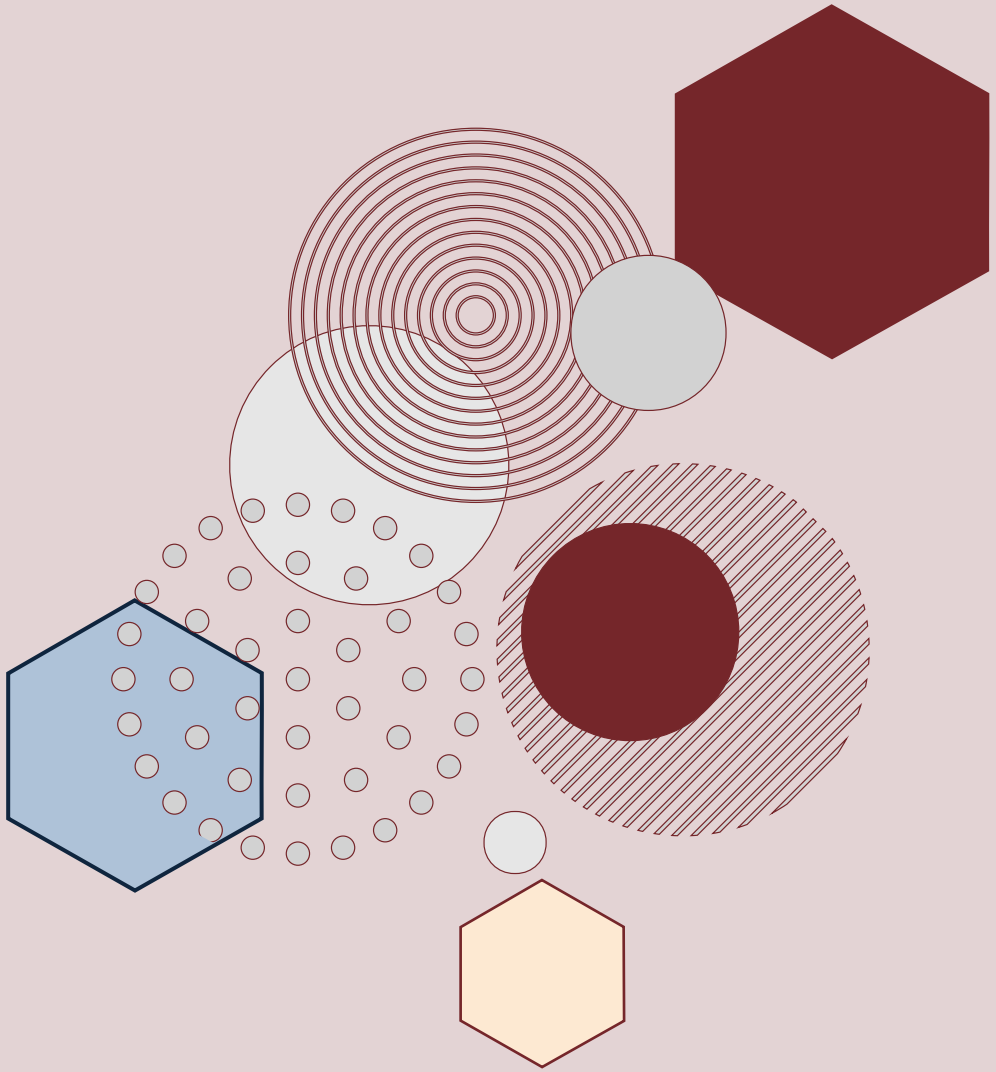
FIGURE 10.44 (a) Text image corrupted by spot shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages.



a b c

FIGURE 10.45 (a) Text image corrupted by sinusoidal shading. (b) Result of global thresholding using Otsu's method. (c) Result of local thresholding using moving averages..





Region-based and Cluster-based Segmentations

Section 3 - Region-based & Cluster-based Segmentations

3.1 Region Growing
3.2 Region Splitting and Merging
3.3 Clustering

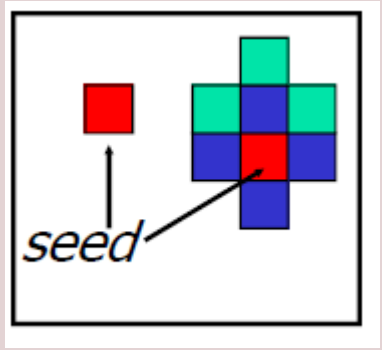
Region Growing

kita bisa buat sebuah region, dimana kita liat tentangganya 4 neightbor atau 8 neighbour untuk dapaeting buat sebuah region

- Bottom up:
  1. Determine some seed pixels m (Initialization)
  2. Check homogeneity using 4- or 8-neighbors
  3. Using a criteria of uniformity

$$\max_{P \in R} |f(P) - m| < T$$

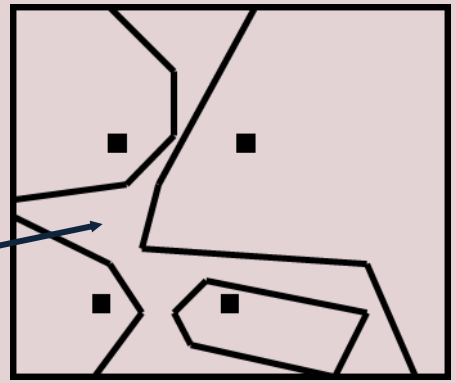
  4. If similar to seed
     - Merge with region
     - Else, Do not merge with the region



- Problems: *Belum tentu menghasilkan wilayah-wilayah yang bersambungan*.

diisni drea tidak dilabel

belum tentu untuk mendapatkan wilayah yang bersambungan



Region Growing in Action

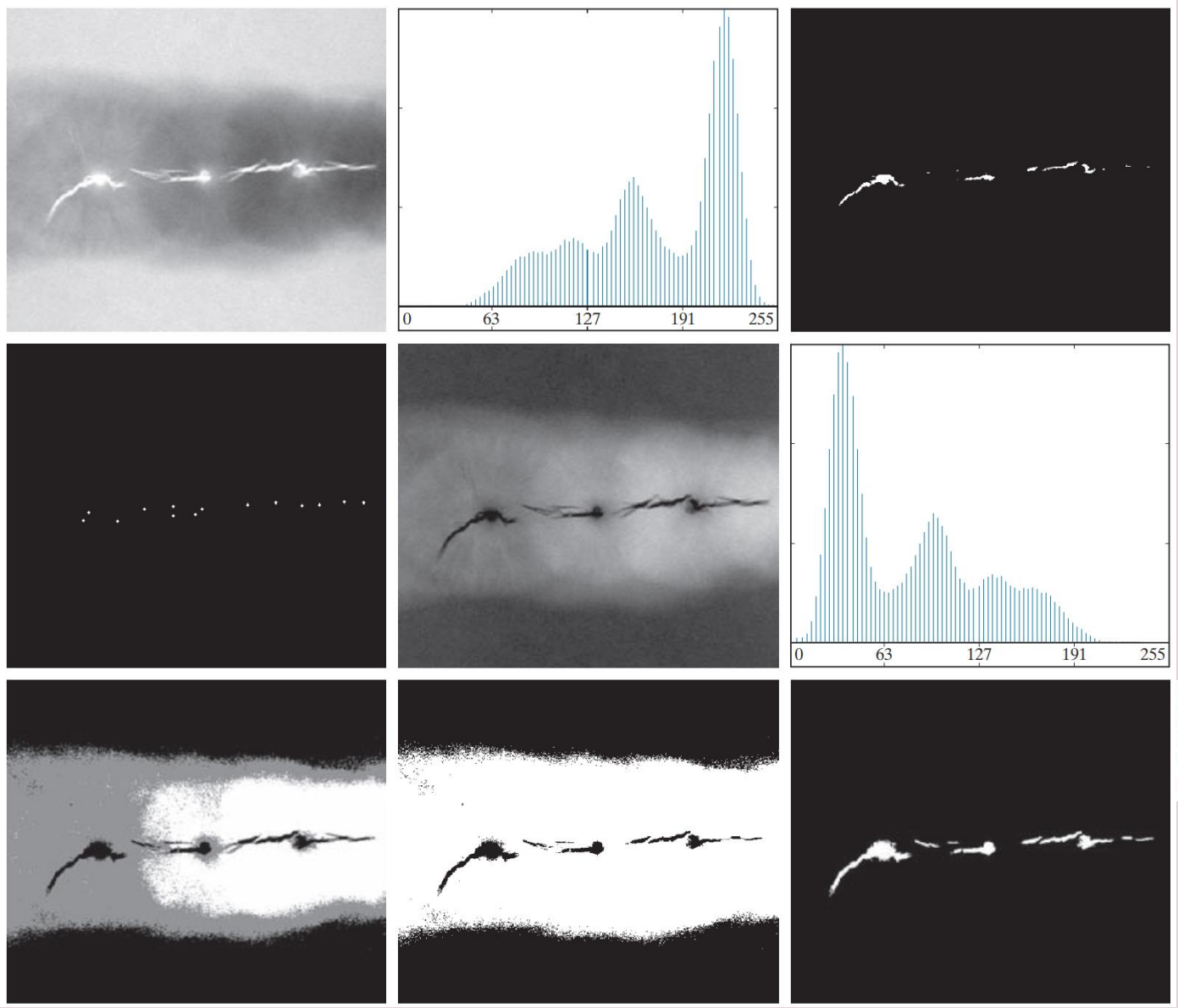
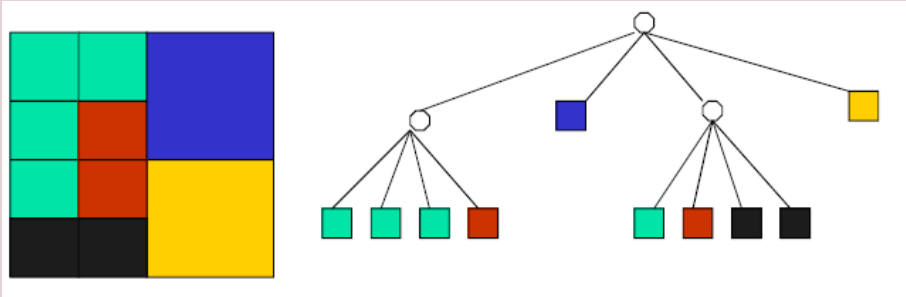


Figure 10.46 (a) X-ray image of a defective weld. (b) Histogram. (c) Initial seed image. (d) Final seed image (the points were enlarged for clarity). (e) Absolute value of the difference between the seed value (255) and (a). (f) Histogram of (e). (g) Difference image thresholded using dual thresholds. (h) Difference image thresholded with the smallest of the dual thresholds. (i) Segmentation result obtained by region growing. (Original image courtesy of X-TEK Systems, Ltd.)

Region Splitting and Merging

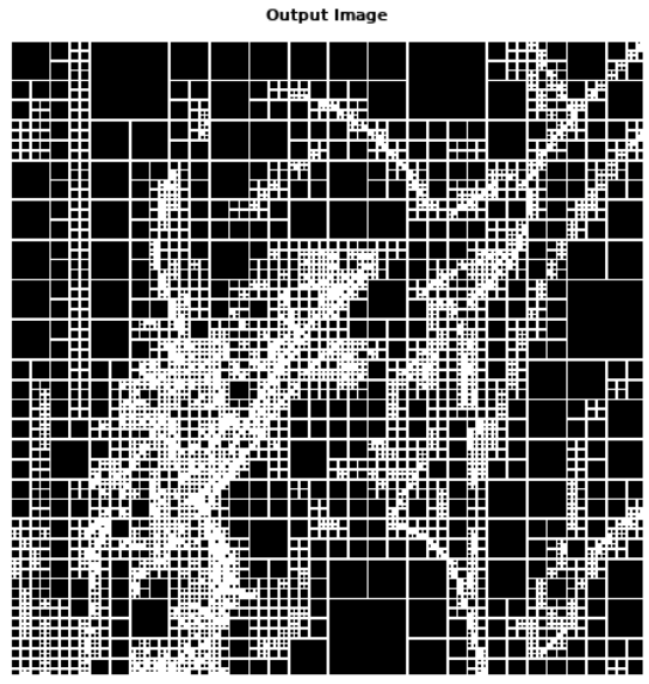
- Top-Down Splitting:
  1. Using a quad tree
  2. Using a criteria of uniformity (such as variance)
  3. If variance is too high
     1. Split from region
     2. Else, leave with region
     - Note: If only splitting is used, the final partition normally contains adjacent regions with identical properties.

kalo sama di merge.

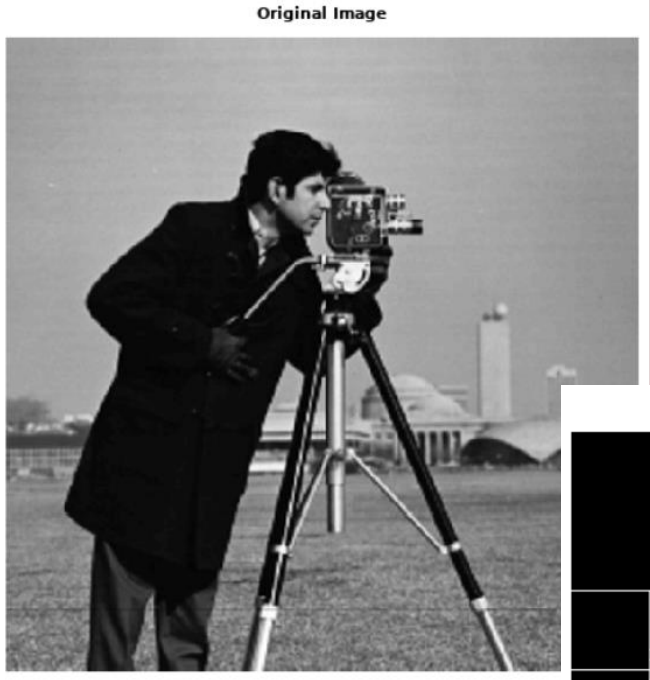


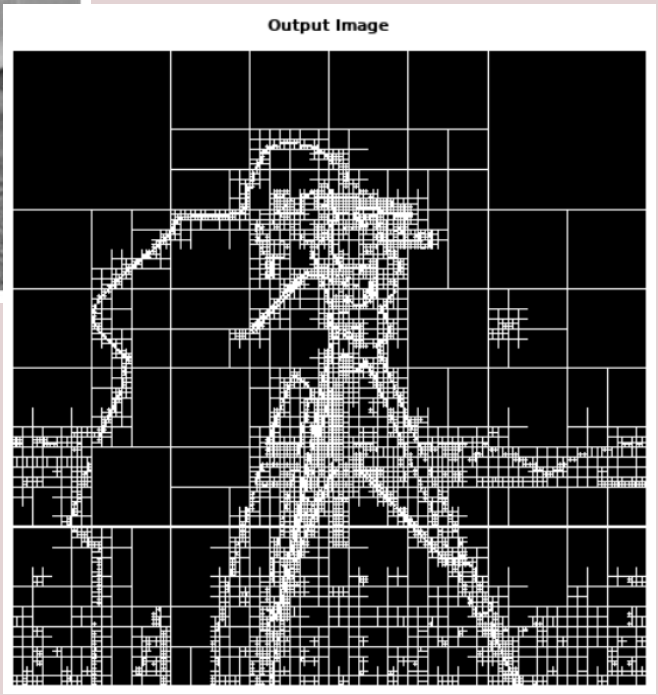
- Bottom-Up Merging:
  1. When no further splitting is possible, merge any adjacent regions for which the criteria of uniformity (i.e., property) is the same/similar (e.g., thresholding).
  2. Stop when no further merging is possible.

Region Splitting and Merging in Action



ini hasilnya region splitting (dbagi jadi 4 by 4. Dan didapatkan hasil segmentasinya seperti ini:





Images are from: https://nirantak.com/image-processing-region-splitting

Notes about Region Splitting

- Algorithm:
  - Define the criterion to be used for homogeneity
  - Split the image into equal size regions
  - Calculate homogeneity for each region
  - If the region is homogeneous, then merge it with neighbors
  - The process is repeated until all regions pass the homogeneity test
- There are several ways to define homogeneity, some examples are:
  - Uniformity - The region is homogeneous if its gray scale levels are constant or within a given threshold.
  - Local mean vs global mean - If the mean of a region is greater than the mean of the global image, then the region is homogeneous
  - Variance - The gray level variance is defined as

$$\sigma^2 = (1/(N-1)) \sum_{(r,c)\epsilon R} [I(r,c) - \bar{I}]^2$$

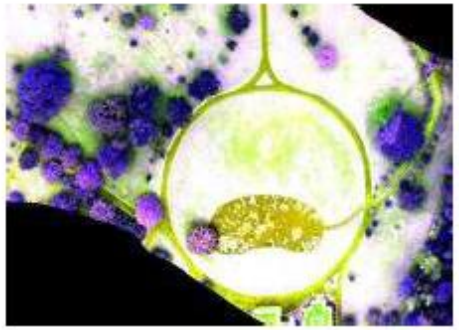
$$\bar{I} = (1/N) \sum_{(r,c)\epsilon Region} I(r,c)$$

Images are from: https://nirantak.com/image-processing-region-splitting

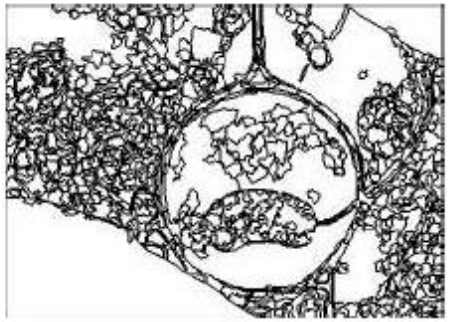
Clustering

disini akan masuk ke cluster yang sama walaupun perbedaan tempat nya jauh.

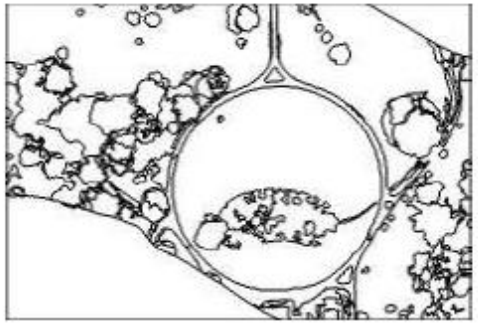
Seharusnya, hanya yang regionnya berdekatan yang di clustering Untuk menjawab hal itu ada namanya SLIC.



Citra CASI Kebun Raya Bogor
(PT. The Map Indonesia Data)



Jumlah cluster banyak



Jumlah cluster sedikit

(Source: Wiweka H.)

LLM: cari perbedaan area yang ada di gambar, dia cari kesamaan di darea edges.

Fitur-fitur dapat digunakan dala m uniform criteria (kita tergantung, definisi clustering kita seperti apa).

Apakah kriteria nya itu mau dari warna, edges, tekstur, intensitas, dll.

- How do we decide which pixels to group together?
  - Similarity (color, intensity)
  - Distance
- Several methods that can be used:
  - K-means clustering,
  - Simple Linear Iterative Clustering (SLIC)

Ada region yang dari kecil di gedin.

Ada juga region yang pecah-pecah

Kita disini melakukan clustering. Setiap vector RGB yang mirip-mirip akan di clustering.

Kita bisa pakai teknik clustering, contohnya K-Means.

Simple Linear Iterative Clustering (SLIC)

dia measure pixel distance sama color difference.

- Uses 2 measures joined with weights:
  - Color difference→ to find similar colors
  - Pixel distance → to find near pixels

Achanta, Radhakrishna, et al. "SLIC superpixels compared to state-of-the-art superpixel methods." *IEEE transactions on pattern analysis and machine intelligence* 34.11 (2012): 2274-2282.

Algorithm 1 SLIC superpixel segmentation

/ Initialization */*

Initialize cluster centers $C_k = [l_k, a_k, b_k, x_k, y_k]^T$ by sampling pixels at regular grid steps S.

Move cluster centers to the lowest gradient position in a 3×3 neighborhood.

Set label $l(i) = -1$ for each pixel i.

Set distance $d(i) = \infty$ for each pixel i.

repeat

  / Assignment */*

  for each cluster center C_k **do**

    for each pixel i in a $2S \times 2S$ region around C_k **do**

      Compute the distance D between C_k and i.

      if $D < d(i)$ **then**

        set $d(i) = D$

        set $l(i) = k$

      end if

    end for

  end for

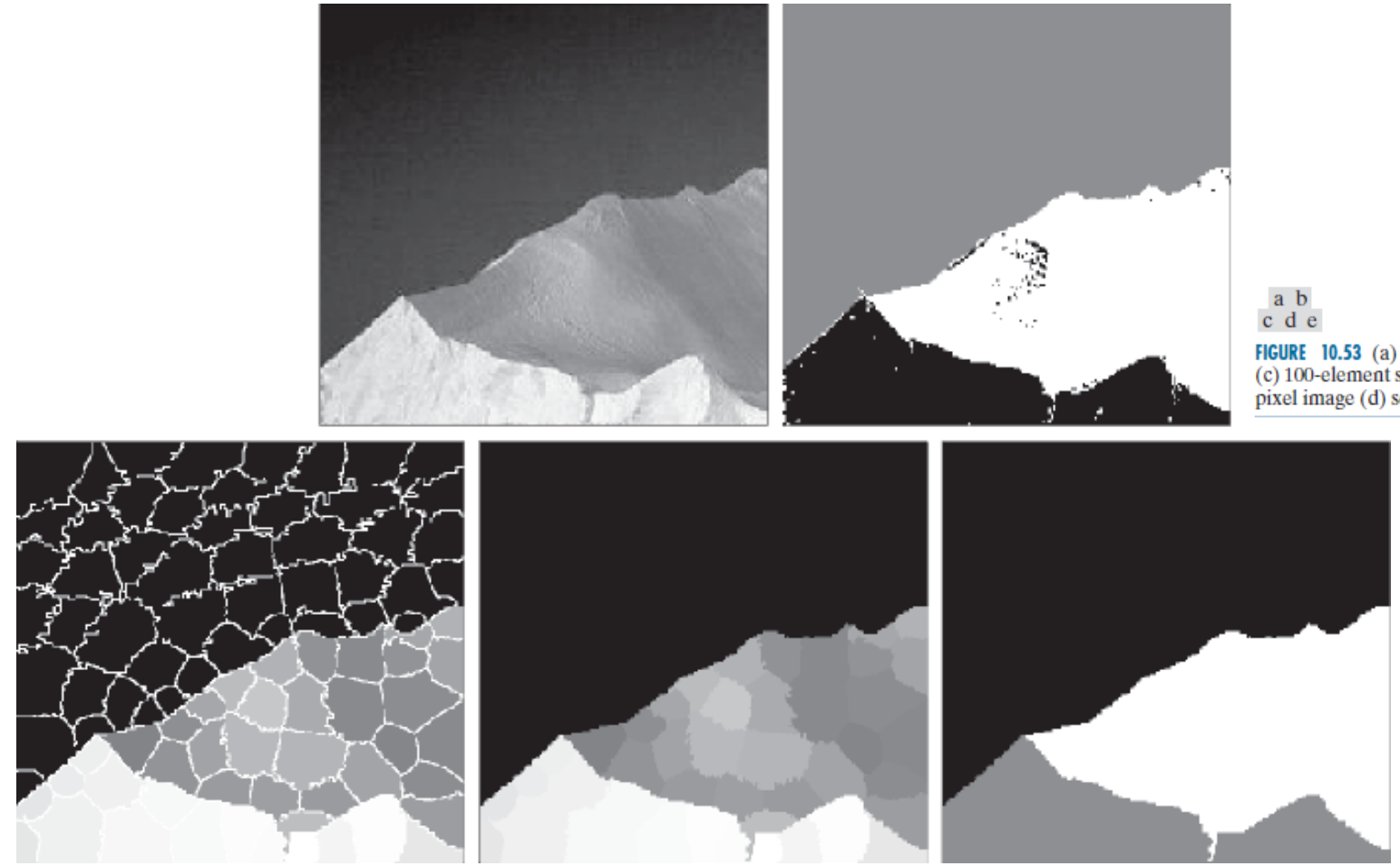
  / Update */*

  Compute new cluster centers.

  Compute residual error E.

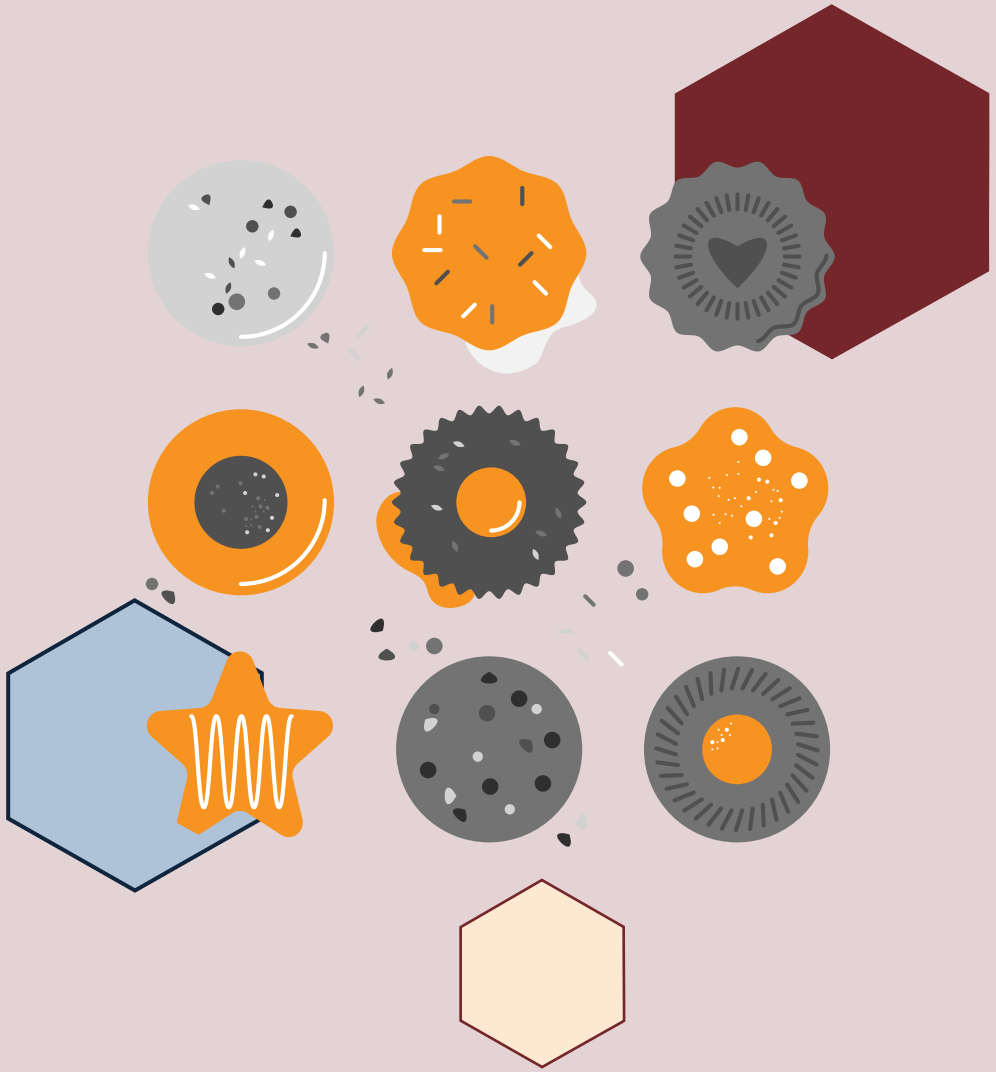
until $E \leq$ threshold

Clustering for Segmentation in Action



a b
c d e

FIGURE 10.53 (a) Image of size 533 × 566 (301,678) pixels. (b) Image segmented using the *k*-means algorithm. (c) 100-element superpixel image showing boundaries for reference. (d) Same image without boundaries. (e) Superpixel image (d) segmented using the *k*-means algorithm. (Original image courtesy of NOAA.)



Watershed Segmentation

Section 4 - Watershed Segmentation

Watershed Segmentation

- Any grayscale image can be viewed as a topographic surface, with high-intensity denoting peaks and hills and low-intensity denoting valleys.
    1. You start filling every isolated valley (local minima) with different colored water (labels).
    2. As the water rises, depending on the peaks (gradients) nearby, water from different valleys, obviously with different colors will start to merge.
    3. To avoid that, you build barriers in the locations where water merges.
    4. You continue the work of filling water and building barriers until all the peaks are underwater.
    5. Then, the barriers you created give you the segmentation result.

Apabila intensitas sebuah gambar kita anggap sebagai contour

dimana intensitas tinggi dia tinggi, dan intensitas rendah dia rendah.

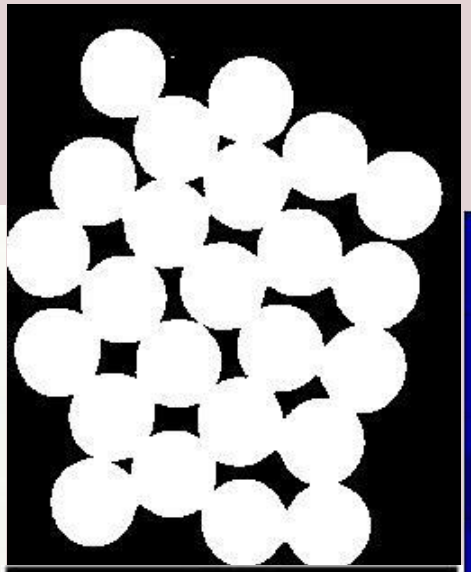
Maka dia kalau tinggi, diisi air sampe tumpah. Nah intinya dia itu akan terus memasukkan segmentation

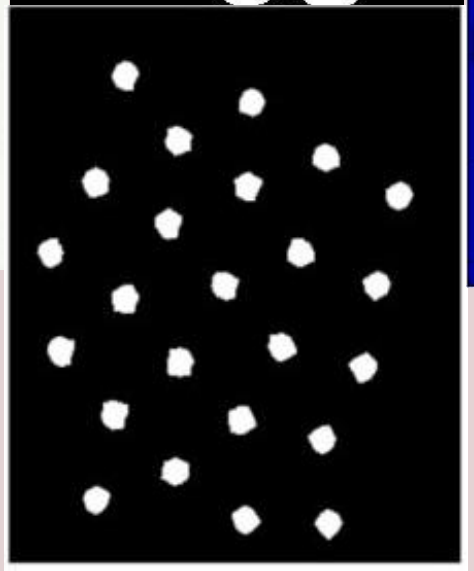
https://docs.opencv.org/4.x/d3/db4/tutorial_py_watershed.html

Watershed Segmentation using OpenCV

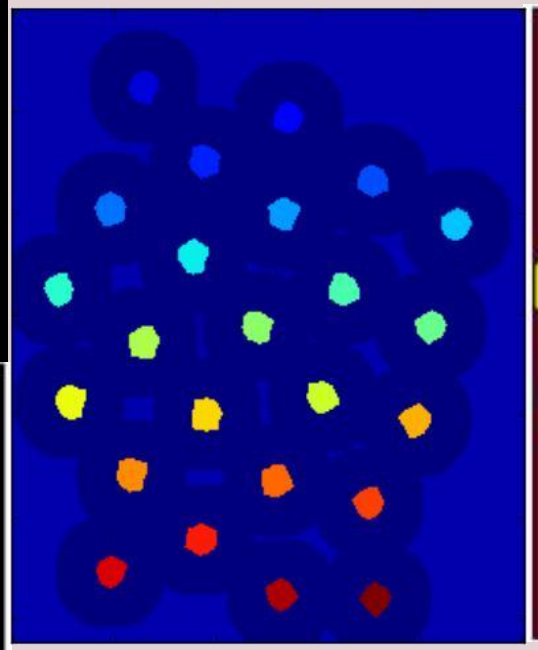
Threshold

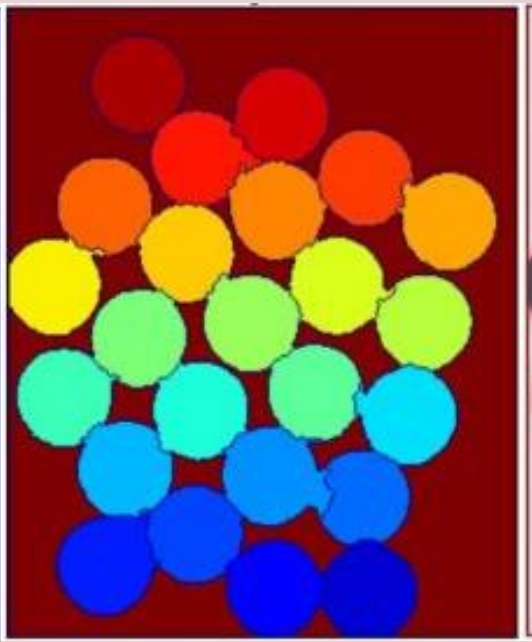
Image





Threshold + Erosion





https://docs.opencv.org/4.x/d3/db4/tutorial_py_watershed.html

Evaluasi sekarang pake IoU (Intersection Over Union), atau intinya pakai Jaccard Similarity.

Yang lebih baik adalah menggunakan Dice Coeficcient. Jaid untuk mengurangi komponensasi/penalty, basically di Dice Coeficcient, dia pakai F1 Score (equivalensi).

Thank you!

CSCE604133 Computer Vision
Faculty of Computer Science
Universitas Indonesia

Dr. Eng. Laksmita Rahadianti
Muhammad Febrian Rachmadi, Ph.D.
Dr. Dina Chahyati, Prof. Dr. Aniati M. Arymurthy