



extract something from image

LECTURE 08 IMAGE SEGMENTATION

Punnarai Siricharoen, Ph.D.

OBJECTIVES

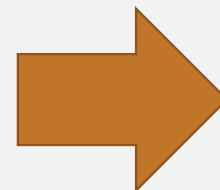
- To understand segmentation fundamentals and its challenges
- To understand and be able to apply segmentation techniques to solve simple to intermediate complexity of problems
- To identify the limitation of different segmentation techniques

TODAY'S CONTENT

- Segmentation techniques
 - Discontinuity
 - Similarity
 - Region-based segmentation
 - Other segmentation techniques

WHAT IS SEGMENTATION?

WHAT IS SEGMENTATION?



Ref: Filip Malmberg

SEGMENTATION

- finding groups of pixels that “go together” -> Cluster analysis.
- subdivides an image into its constituent regions or objects.
- The level to which the subdivision is carried depends on the problem being solved.
- Segmentation accuracy determines the eventual success or failure of computerized analysis procedures.

SEGMENTATION

- Accurate segmentation of objects of interest in an image greatly facilitates further analysis of these objects. For example, it allows us to:
 - Count the number of objects of a certain type.
 - Measure geometric properties (e.g., area, perimeter) of objects in the image.
 - Study properties of an individual object (intensity, texture, etc.)

semantic seg : ໂປ່ງປະກາດ

instance seg : ພັບຈຳນວນ

panoptic seg : uncountable noun

SEGMENTATION

- What would be correct segmentation of this image?



SEGMENTATION

- What would be correct segmentation of this image?



Until we specify the target



“Segment the orange car from the background”



“Segment all road signs from the background”

SEGMENTATION

- A segmentation can also be defined as a mapping from the set of pixels to some application dependent target set, e.g.
 - {Object, Background}
 - {Humans, Other objects}
 - {Healthy tissue, Tumors}
- To perform accurate segmentation, we (or our algorithms) need to somehow know how to differentiate between different elements of the target set.

BASIC SEGMENTATION

- Two main properties of intensity values:
 - **Discontinuity** -> partition an image based on abrupt changes in intensity, such as edges.
 - **Similarity** -> partition an image into regions what are similar according to a set of pre-defined criteria, such as thresholding, region growing and region splitting.

BASIC SEGMENTATION ALGORITHMS

Thresholding *Similarity*

- Based on pixel intensities (shape of histogram is often used for automation).

Edge-based *Discontinuity*

- Detecting edges that separate regions from each other.

Region-based *Similarity*

- Grouping similar pixels (with e.g. region growing or merge & split).

DISCONTINUITY

Using Discontinuity for segmentation

DETECTION OF DISCONTINUITIES

2

- Basic types of gray-level discontinuities in a digital image:
- 1 • Points, Lines, and Edges 3
- Look at discontinuities -> apply 3x3 mask into an image.

ω_1	ω_2	ω_3
ω_4	ω_5	ω_6
ω_7	ω_8	ω_9

Mask coefficients

$$R = \sum_{i=1}^{mn} \omega_i z_i$$

0	1	0
1	-4	1
0	1	0

	z_1	z_2	z_3	
	z_4	z_5	z_6	
	z_7	z_8	z_9	

Image pixels



Original
image



Laplacian
image

Spatial / Convolution filtering

flip

*ກຳພາດ ພວຍໄທມີຄວາມຄົບ ,
ແລ້ວເປັນ symmetric*

ω are mask coefficients
 z : values of image gray levels corresponding those coefficients
 mn : the total number of coefficients in the mask.

1

POINT DETECTION

- Detection of isolated points in an image can be detected using the mask below:

- The idea of an isolated point is that gray level is significantly different from its background.

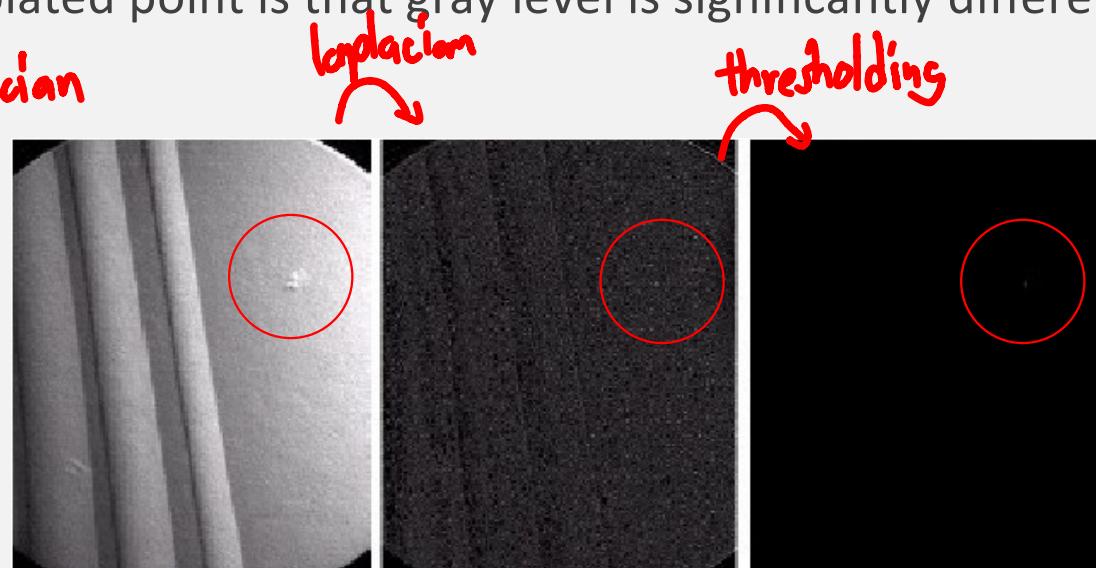
Laplacian

$$\begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline -1 & 8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

(a)

*non-Laplacian
zero'du
negative*

$$R | \geq T$$



T: nonneg. Threshold

R: convolved image (image * mask)

FIGURE 10.4
 (a) Point detection
 (Laplacian) mask.
 (b) X-ray image of turbine blade with a porosity.
 The porosity contains a single black pixel.
 (c) Result of convolving the mask with the image.
 (d) Result of using Eq. (10.2-8)
 showing a single point (the point was enlarged to make it easier to see). (Original image courtesy of X-TEK Systems, Ltd.)

2

LINE DETECTION

- The mask below will respond more strongly to lines (one pixel thick) oriented in horizontal, $+45^\circ$, vertical, and -45° directions, respectively.

Line masks:

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

$+45^\circ$

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

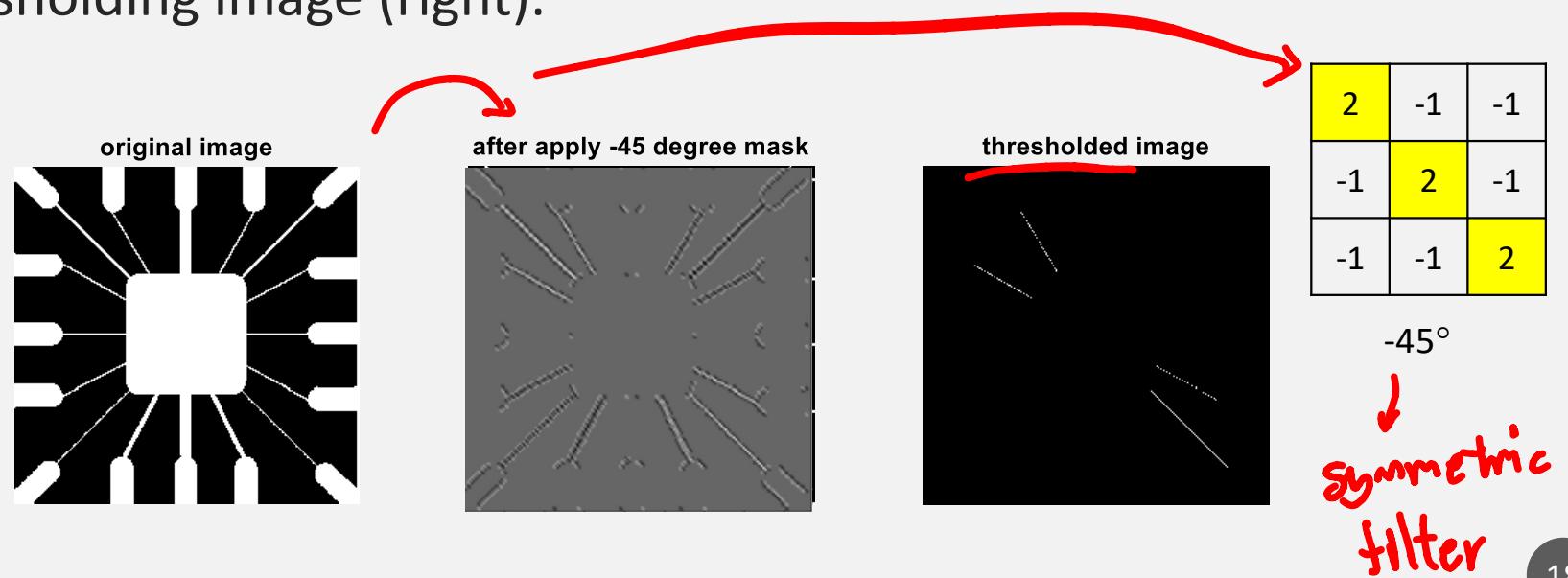
2	-1	-1
-1	2	-1
-1	-1	2

-45°

2

LINE DETECTION

- Digitized (binary) portion of a wire-bond mask for an electronic circuit (left).
- Absolute value of results after processing with -45° line detector (middle).
- Result of thresholding image (right).



3

EDGE DETECTION

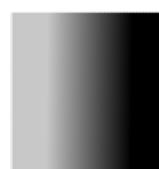
- Intuitively, an edge is a set of connected pixels that lie on the boundary between two regions.
- A reasonable definition of “edge” requires the ability to measure gray-level transitions in a meaningful way.

Detecting sharp & local changes in intensity.

Step



Ramp



Line



Point



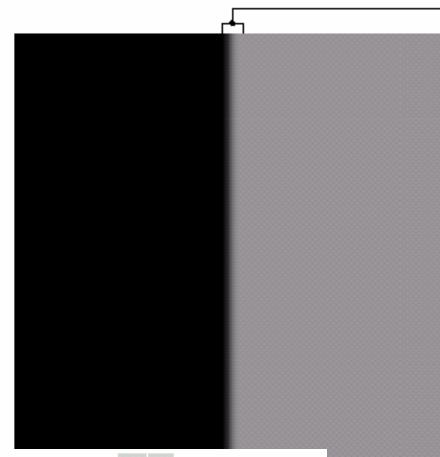
Ref: Filip Malmberg

3

EDGE DETECTION

a b

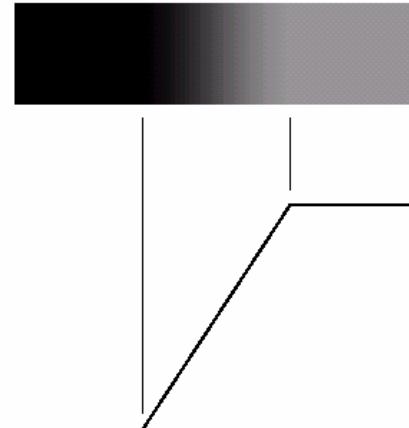
FIGURE 10.6
 (a) Two regions separated by a vertical edge.
 (b) Detail near the edge, showing a gray-level profile, and the first and second derivatives of the profile.



Model of an ideal digital edge



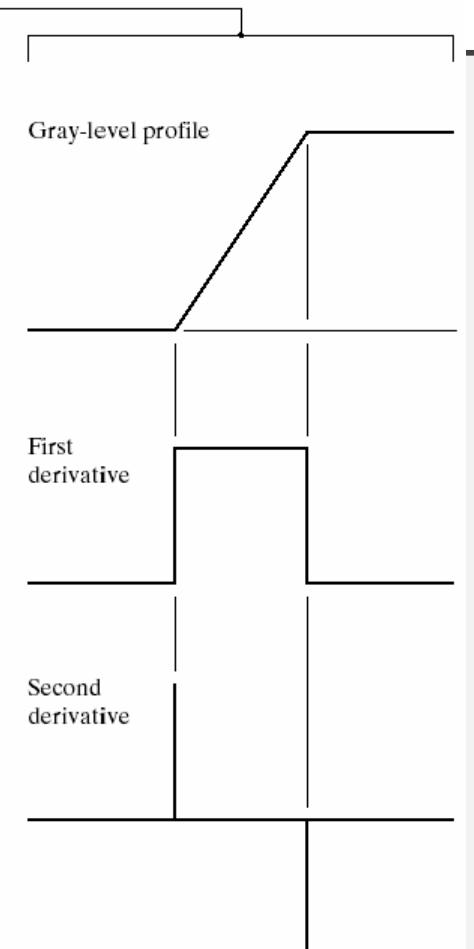
Model of a ramp digital edge



Gray-level profile of a horizontal line through the image

a b

FIGURE 10.5
 (a) Model of an ideal digital edge.
 (b) Model of a ramp edge. The slope of the ramp is proportional to the degree of blurring in the edge.



3

EDGE DETECTION

- First derivatives:
- The gradient of an image $f(x,y)$ at location (x,y) is defined as the vector

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\nabla f \approx |G_x| + |G_y|$$

- A weight value of 2 is used to achieve some smoothing by giving more importance to the center point, called the Sobel operators.

Gx (edges at x-direction)
Gy (edges in y-direction)

Image from Gonzalez & Woods, Digital Image Processing

a
b
c
d
e
f
g

FIGURE 10.8
A 3×3 region of an image (the z 's are gray-level values) and various masks used to compute the gradient at point labeled z_5 .

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-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

Gx Prewitt Gy

-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Sobel

3

EDGE DETECTION

- First derivatives:
- The Prewitt and Sobel operators are among the most used in practice for computing digital gradients.
- Prewitt masks are simpler to implement than the Sobel masks, but the latter have slightly superior noise-suppression characteristics, an important issue when dealing with derivatives.

Might be opposite in the program

Gx (edges at x-direction)

Gy (edges in y-direction)

Image from Gonzalez & Woods, Digital Image Processing

a
b c
d e
f g

FIGURE 10.8
A 3×3 region of an image (the z 's are gray-level values) and various masks used to compute the gradient at point labeled z_5 .

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

-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

Gx

Prewitt

Gy

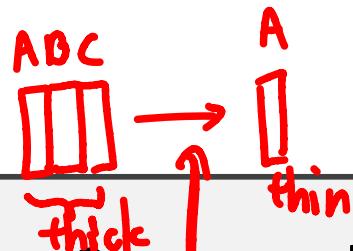
-1	-2	-1	-1	0	1
0	0	0	-2	0	2
1	2	1	-1	0	1

Sobel

3

EDGE DETECTION

- The Canny method has been developed by John F. Canny in 1986
- The Canny method finds edges by looking for local maxima of the gradient of I.
① can detect thin line
- This method uses two thresholds to detect strong and weak edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges.
- Steps:
 1. Noise Reduction (remove noise with 5x5 Gaussian filter)
② reduce effect of noise avg, blur
 2. Finding Intensity Gradient of the Image - Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction
Spatial : reduce noise

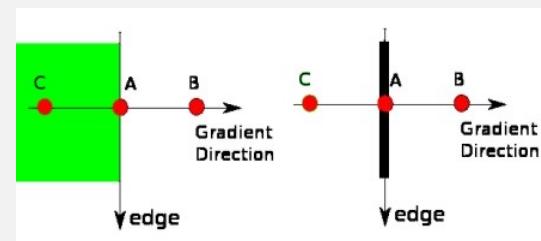


3 EDGE DETECTION

- The Canny method - multi stage algorithm

3. Non-maximum Suppression – A full scan of image is done to remove any unwanted pixels which may not constitute the edge. Point A will be compared with B and C.

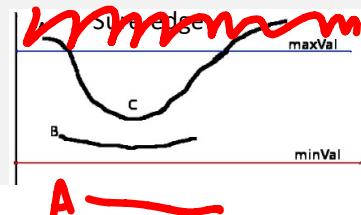
- if A is local maxima, A will be considered for the next stage.
- If not, suppressed. Remaining only **thin edges**.



A compared to C --- Yes
B > minVal but same with C – Not Edges

- 4. Hysteresis Thresholding** – double thresholds

- Removes pixels whose gradients lower than the threshold noises on the assumption that edges are long lines
- Edge tracking by hysteresis - Weak edges that are connected to strong edges will be actual/real edges



only sure edge will be used : Double hysteresis

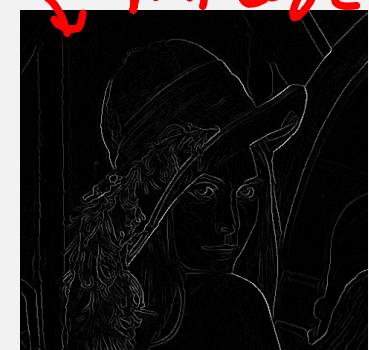
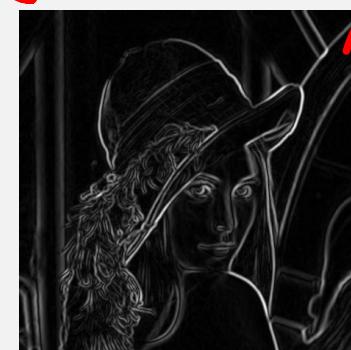
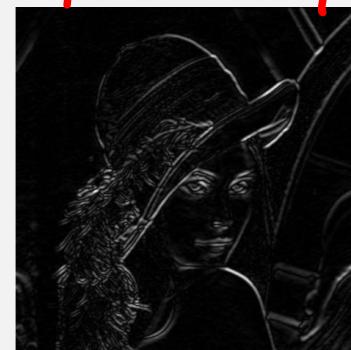
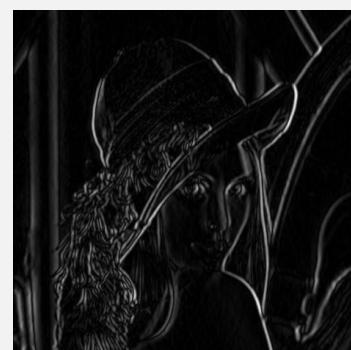
we finally get is strong edges in the image

24

Edge tracking : C 1st, B 2nd, 1st: C 1st is a sure edge

3

EDGE DETECTION



Gaussian Blur

only sure edge



Double hysteresis



Edge tracking

Sobel

$$|G_x| + |G_y|$$

thin edge

Canny Edge detection

<http://www.justin-liang.com/tutorials/canny/>

3

EDGE DETECTION

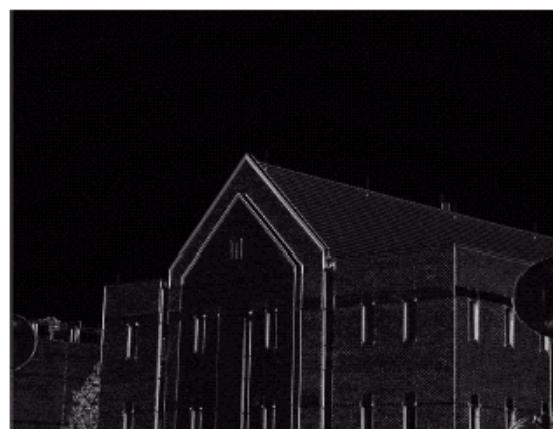
a	b
c	d

FIGURE 10.10

(a) Original image. (b) $|G_x|$, component of the gradient in the x -direction.

(c) $|G_y|$, component in the y -direction.

(d) Gradient image, $|G_x| + |G_y|$.



3

EDGE DETECTION

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

FIGURE 10.13
Laplacian masks
used to
implement
Eqs. (10.1-14) and
(10.1-15),
respectively.

- **Second derivatives:**
- The Laplacian of a 2-D function (x,y) is a second-order derivative defined as

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

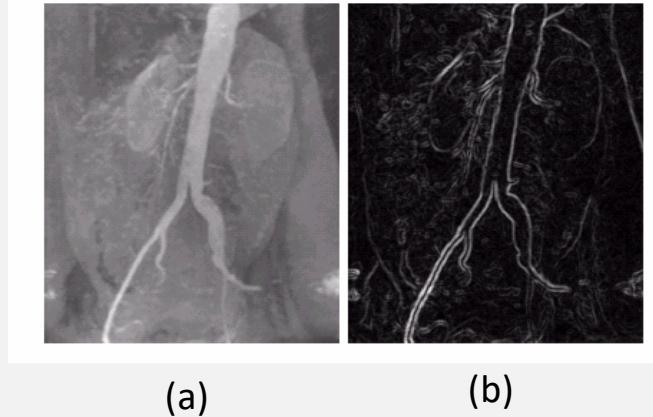
- As a second-order derivative, the Laplacian typically is unacceptably sensitive to noise.
- **Laplacian of a Gaussian (LoG)**
 - Laplacian combined with smoothing as a precursor to finding edges via zero crossings.

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

3

EDGE DETECTION

- **Second derivatives:**
- One straightforward approach for approximating zero crossings is to threshold the LoG image by setting all its positive values to, say, white, and all negative values to black.



(a)

(b)

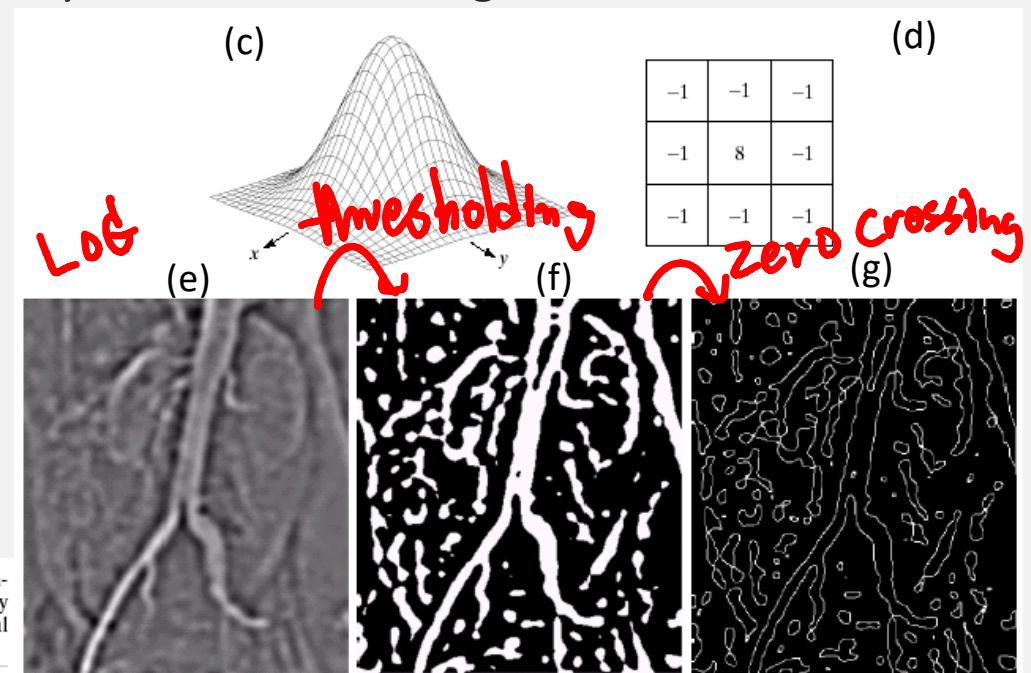


FIGURE 10.15 (a) Original image. (b) Sobel gradient (shown for comparison). (c) Spatial Gaussian smoothing function. (d) Laplacian mask. (e) LoG. (f) Thresholded LoG. (g) Zero crossings. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

3

EDGE DETECTION

Python

```

img = cv2.imread('cameraman.tif',0)

#canny
img_canny = _____

#sobel
img_sobelx = cv2.Sobel(img,cv2.CV_8U,1,0,ksize=3)
img_sobely = cv2.Sobel(img,cv2.CV_8U,0,1,ksize=3)
img_sobel = _____

#prewitt
kernelx = np.array([[-1,-1,-1],[0,0,0],[1,1,1]])
kernely = _____
img_prewittx = cv2.filter2D(img, -1, kernelx)
img_prewitty = _____
img_prewitt = _____

```

```

void cv::Canny ( InputArray image,
                OutputArray edges,
                double threshold1,
                double threshold2,
                int apertureSize = 3,
                bool L2gradient = false

```

image 8-bit input image.
edges output edge map; single channels 8-bit image, which has the same size as image
threshold1 first threshold for the hysteresis procedure.
threshold2 second threshold for the hysteresis procedure.
apertureSize aperture size for the Sobel operator.
L2gradient a flag, indicating whether a more accurate L_2 norm = $\sqrt{(dI/dx)^2 + (dI/dy)^2}$



3

EDGE DETECTION

```
#roberts  
kernelx = _____  
kernely = _____  
img_robertsx = _____  
img_robertsy = _____  
img_roberts = _____  
  
#Laplacian  
laplacian = _____  
img_laplacian = _____
```

Display:

```
titles = ['Original Image','Canny','Sobel','Prewitt','Roberts','Laplacian']  
images = [img, img_canny, img_sobel, img_prewitt, img_roberts, img_log]  
for i in range(6):  
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')  
    plt.title(titles[i])  
    plt.xticks([]),plt.yticks([])  
plt.show()
```

```
void cv::Canny ( InputArray image,  
                 OutputArray edges,  
                 double threshold1,  
                 double threshold2,  
                 int apertureSize = 3,  
                 bool L2gradient = false  
 )
```

image 8-bit input image.
edges output edge map; single channels 8-bit image, which has the same size as image
threshold1 first threshold for the hysteresis procedure.
threshold2 second threshold for the hysteresis procedure.
apertureSize aperture size for the Sobel operator.
L2gradient a flag, indicating whether a more accurate L_2 norm = $\sqrt{(dI/dx)^2 + (dI/dy)^2}$

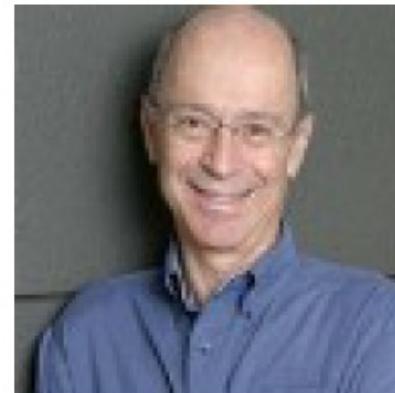
4

HOUGH TRANSFORM

- Hough transform is generally used to detect whether a set of pixels lie on curves or lines of some specified objects.
- Patented by Paul Hough in 1962. The version of the method as it is used today was invented by Richard Duda and Peter Hart in 1972.



Richard Duda



Peter Hart

4

HOUGH TRANSFORM

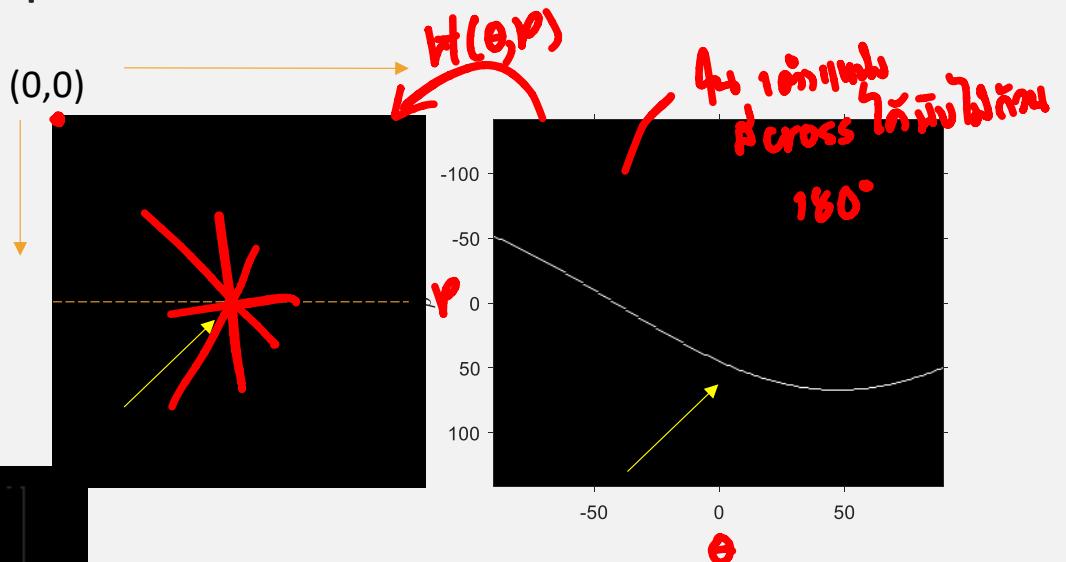
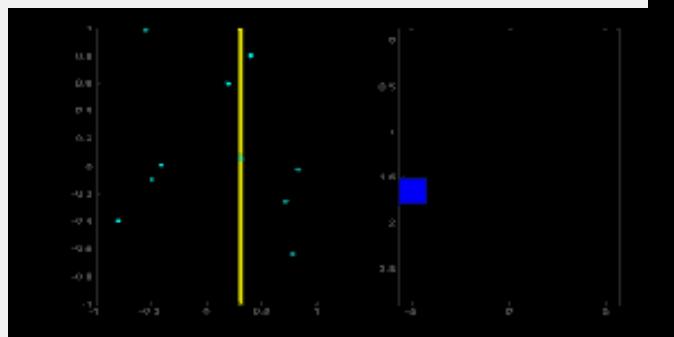
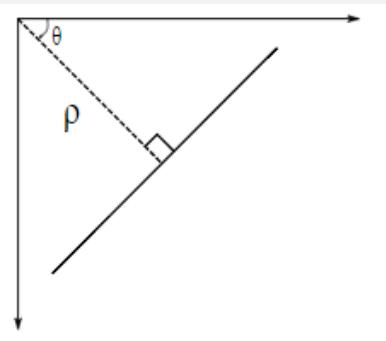
detect line

- Hough Line transform is to find every possible line that could fit a set of pixels in an image.

$$y = mx + c$$



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

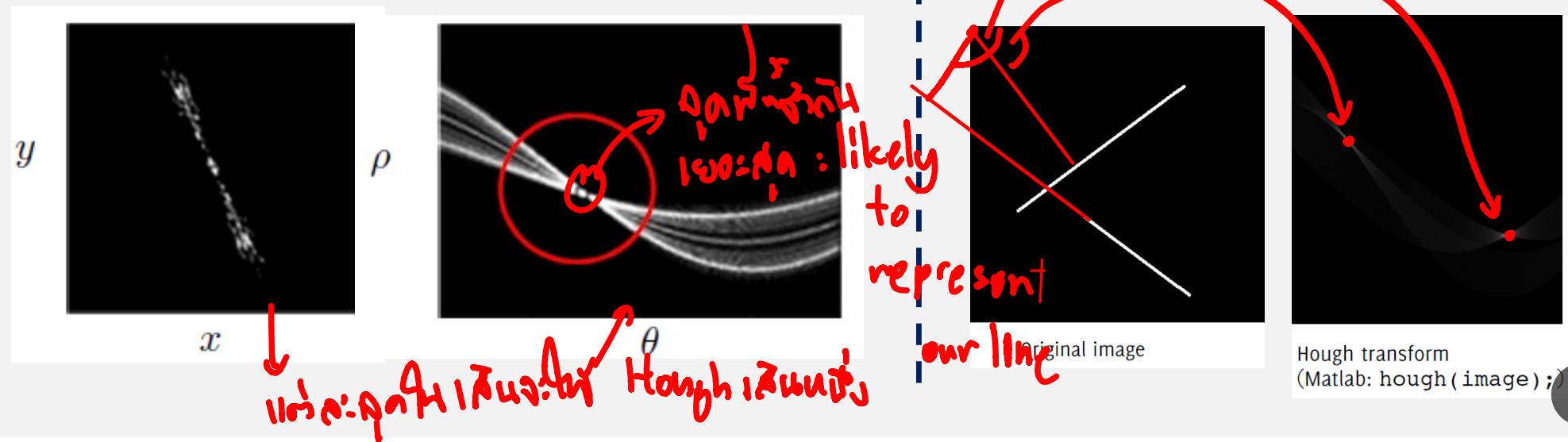


Ref. https://opencv-python-tutorial.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html

4

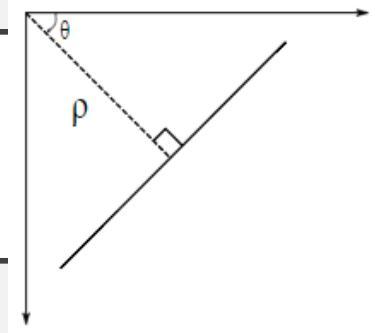
HOUGH TRANSFORM

- The intersection point is usually the maximum magnitude resulted from the Hough transform.
- Performed after Edge Detection



4

HOUGH TRANSFORM

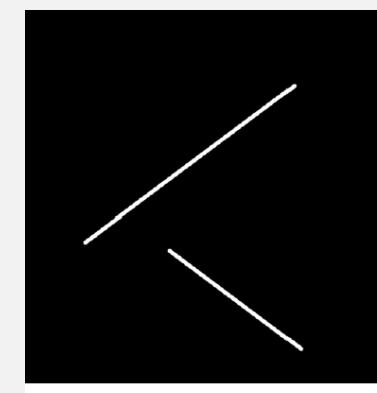


- **Hough Line Algorithm:**
- Typically use a different parameterization
 - ρ is the perpendicular distance from the line to the origin
 - θ is the angle this perpendicular makes with the x axis
- Basic Hough transform algorithm
 1. Initialize $H[\rho, \theta] = 0$ (H is an accumulator array)
 2. for each edge point $I[x, y]$ in the image

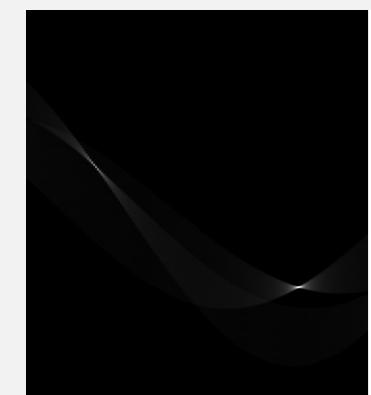
for $\theta = 0$ to 180

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

$$H[\rho, \theta] += 1$$
 3. Find the value(s) of (ρ, θ) where $H[\rho, \theta]$ is maximum
 4. The detected line in the image is given by



Original image

Hough transform
(Matlab: `hough(image)` ;)

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

HOUGH TRANSFORM [PYTHON]

- HoughLines

Python: `cv2.HoughLines(image, rho, theta, threshold[, lines[, srn[, stn]]]) → lines`

Parameters: • **image** – 8-bit, single-channel binary source image. The image may be modified by the function.

- **lines** – Output vector of lines. Each line is represented by a two-element vector (ρ, θ) . ρ is the distance from the coordinate origin $(0, 0)$ (top-left corner of the image). θ is the line rotation angle in radians ($0 \sim$ vertical line, $\pi/2 \sim$ horizontal line).
- **rho** – Distance resolution of the accumulator in pixels.
- **theta** – Angle resolution of the accumulator in radians.
- **threshold** – Accumulator threshold parameter. Only those lines are returned that get enough votes ($> threshold$).
- **srn** – For the multi-scale Hough transform, it is a divisor for the distance resolution **rho**. The coarse accumulator distance resolution is **rho** and the accurate accumulator resolution is **rho/srn**. If both **srn=0** and **stn=0**, the classical Hough transform is used. Otherwise, both these parameters should be positive.
- **stn** – For the multi-scale Hough transform, it is a divisor for the distance resolution **theta**.

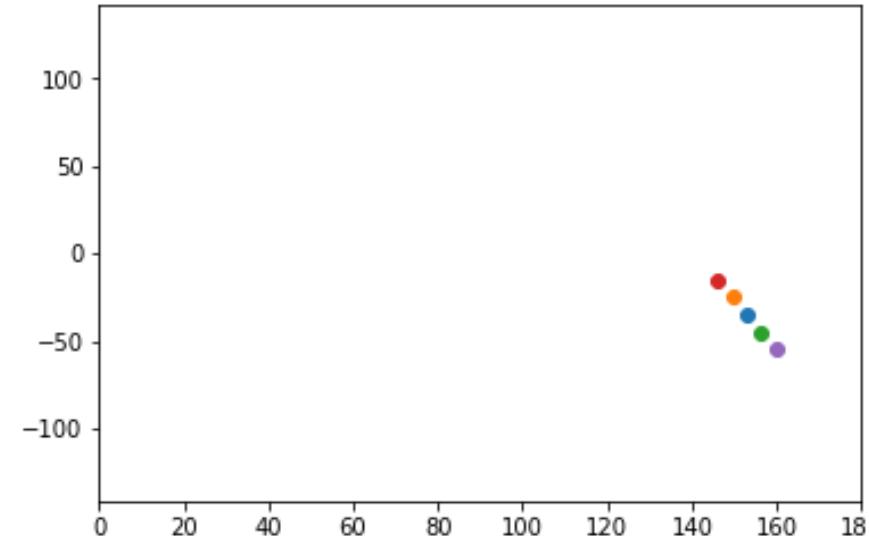
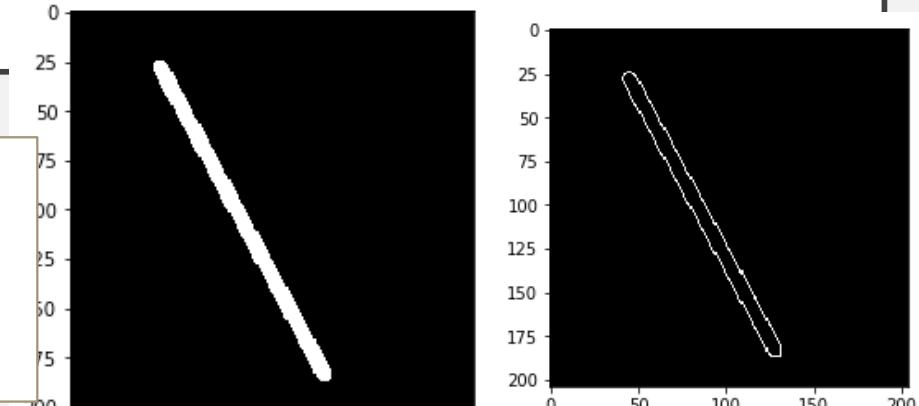
https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html

HOUGH TRANSFORM [PYTHON]

```
#HoughLines  
img = cv2.imread('line.png')  
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)  
edges = cv2.Canny(gray, 50, 150, apertureSize = 3)  
lines = cv2.HoughLines(edges, 10, np.pi/180, 100)
```

To display:

```
#Change radian to degree  
linesdegree = lines.copy()  
linesdegree[:,0:1,1:2] = linesdegree[:,0:1,1:2]*180/np.pi  
  
# use scatter plot  
for i in range(0,len(linesdegree)):  
    plt.scatter(linesdegree[i,0:1,1:2],linesdegree[i,0:1,0:1])  
  
# set x,y limits  
plt.xlim(0, 180)  
plt.ylim(-100*np.sqrt(2), 100*np.sqrt(2))
```



https://docs.opencv.org/2.4/modules/imgproc/doc/feature_detection.html

4

OTHER HOUGE TRANSFORM

- The equation has three parameters for circle Hough transform– x_0 , y_0 , ρ
- Hough circle / lines -> opencv
- Hough circle / lines / ellipse -> scikit-image

Analytic Form	Parameters	Equation
Line	ρ, θ	$xcos\theta + ysin\theta = \rho$
Circle	x_0, y_0, ρ	$(x - x_0)^2 + (y - y_0)^2 = \rho^2$
Parabola	x_0, y_0, ρ, θ	$(y - y_0)^2 = 4\rho(x - x_0)$
Ellipse	x_0, y_0, a, b, θ	$(x - x_0)^2/a^2 + (y - y_0)^2/b^2 = 1$

EXERCISE #2 HOUGH LINE TRANSFORM

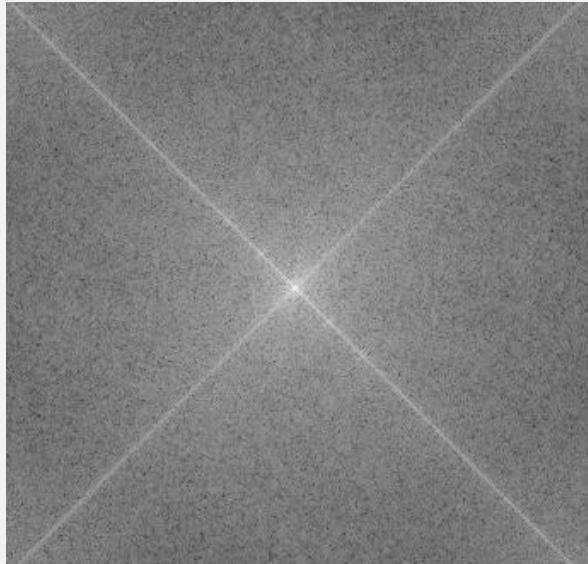
- Exercise #2 Find main lines in this image and describe the lines:
 - What are the main ρ (s), and θ (s) of this image?



clock2.jpg

ASSIGNMENT

- Find main lines in this image and describe the lines:

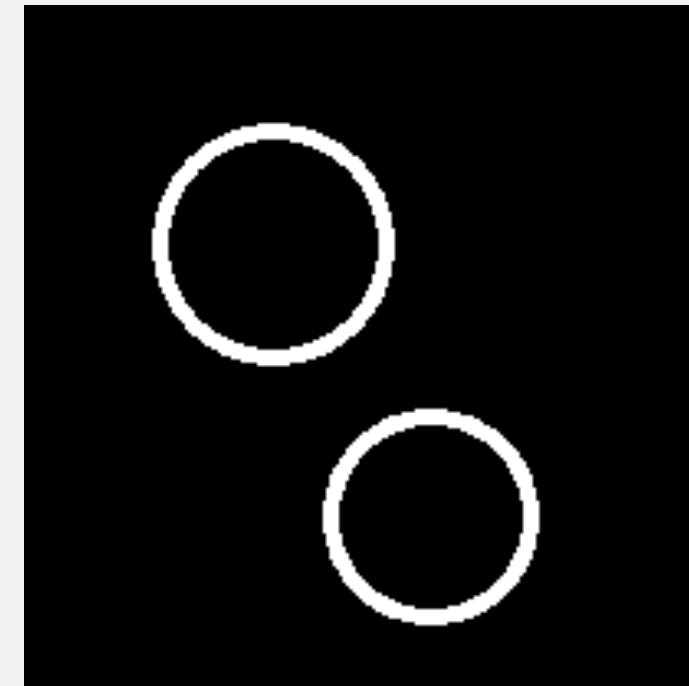


findLine.png

EXERCISE #3 HOUGH CIRCLE

- Exercise #3 Read circle.png to detect circle using Hough transform

```
◆ HoughCircles()  
  
void cv::HoughCircles ( InputArray image,  
                      OutputArray circles,  
                      int method,  
                      double dp,  
                      double minDist,  
                      double param1 = 100 ,  
                      double param2 = 100 ,  
                      int minRadius = 0 ,  
                      int maxRadius = 0  
                    )  
  
Python:  
cv.HoughCircles( image, method, dp, minDist[, circles[, param1[, param2[, minRadius[, maxRadius]]]]] ) -> circles
```



SIMILARITY

Using Similarity for segmentation

THRESHOLDING

- Isolate objects from their background.
- The success of these operations depends very much on choosing an appropriate threshold level.
 - Too low -> reduce object size
 - Too high -> include extraneous background
- Automatic method for choosing a best threshold is needed!!!!

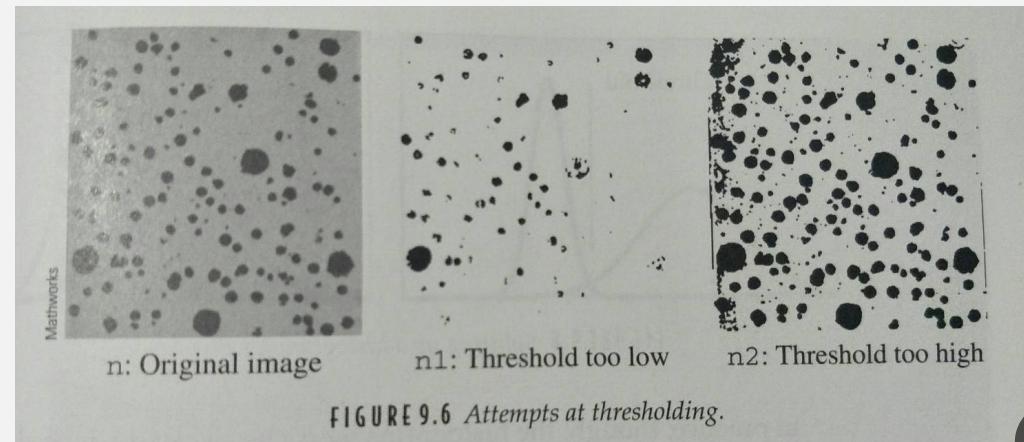
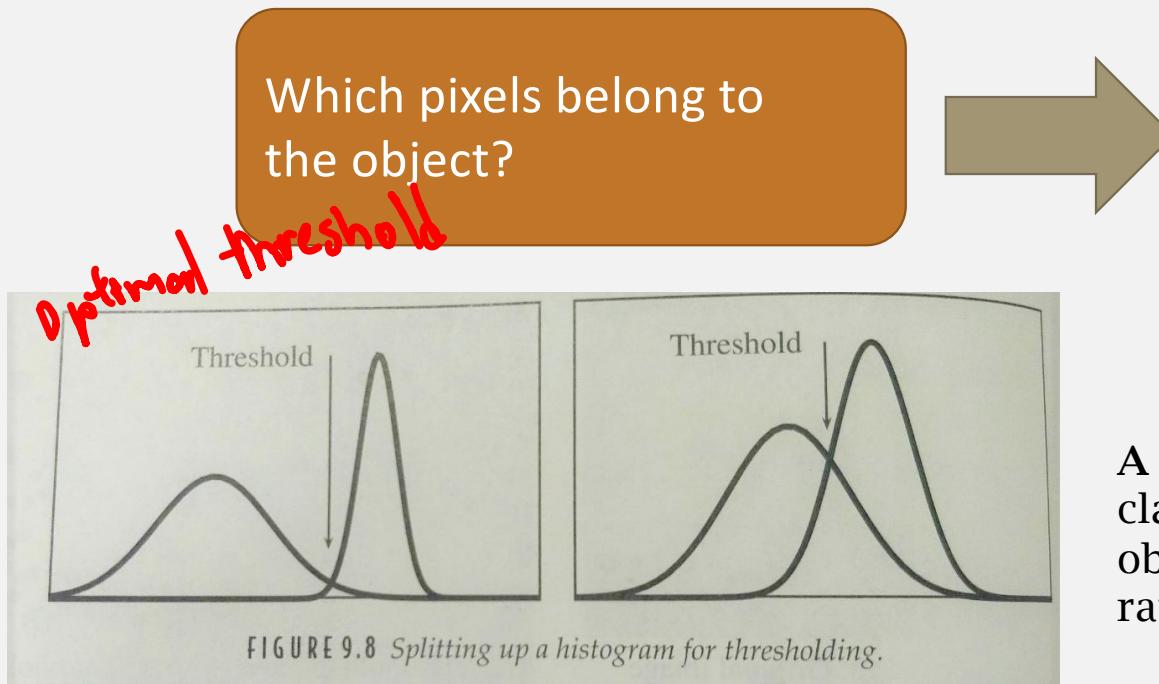


FIGURE 9.6 Attempts at thresholding.

THRESHOLDING

- Good threshold:



A **threshold T** , a gray level intensity, classifies every pixel as belonging to objects (foreground) or background. (Or rather, {dark objects, bright objects}).

THRESHOLDING

① Global threshold

- The same value is used for the whole image.

Optimal global threshold

- Based on the shape of the current image histogram. Searching for valleys, Gaussian distribution etc.

② Local (or dynamic) threshold

- The image is divided into non-overlapping sections, which are thresholded one by one.

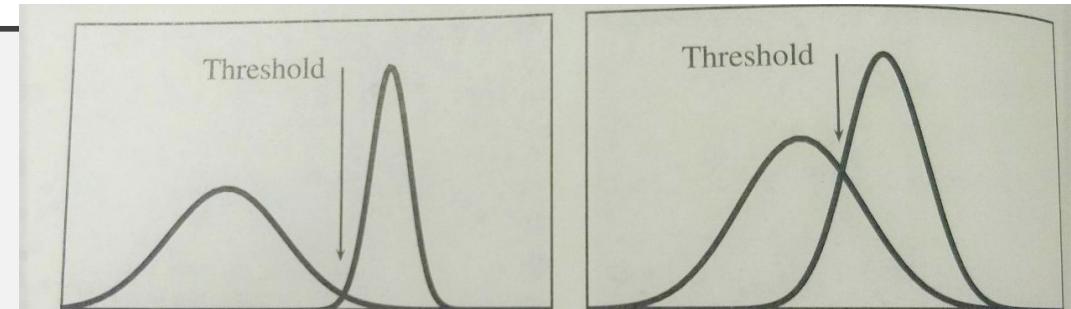
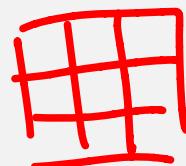
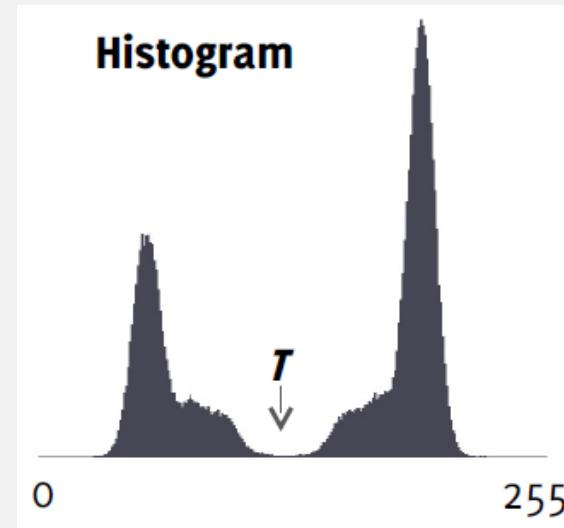


FIGURE 9.8 Splitting up a histogram for thresholding.



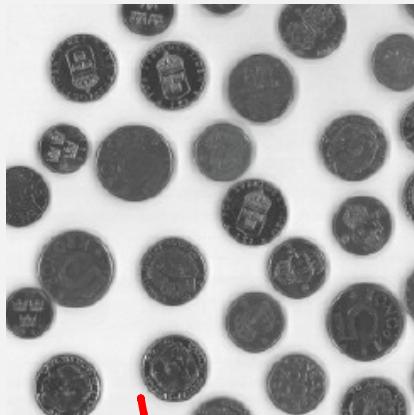
THRESHOLDING

- We chose a threshold T midway between the two gray value distributions.



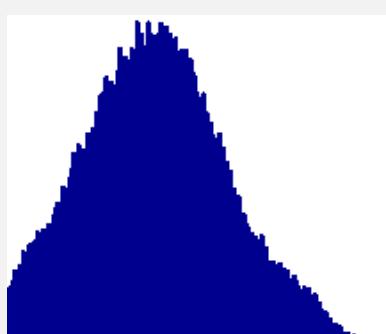
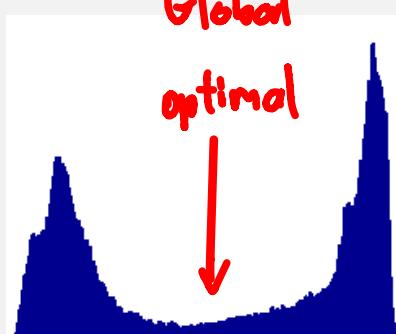
THRESHOLDING

- When does intensity based thresholding work?



moninity Six between State
of Knox And State of Tennessee
Andrew Jackson off the Count
tats. Abols said of the other part
ail Stockley Donelson for A
of the sum of two thousand
hand paid the receipt wher
ath. And by these presents
self alien enoff And Confir
Jackson his heirs And
certain traits or parcels of Land
and acres, one thousand per
and building and

local



moninity Six between State
of Knox And State of Tennessee
Andrew Jackson off the Count
tats. Abols said of the other part
ail Stockley Donelson for A
of the sum of two thousand
hand paid the receipt wher
ath. And by these presents
self alien enoff And Confir
Jackson his heirs And
certain traits or parcels of Land
and acres, one thousand per
and building and

THRESHOLDING

- Local thresholding:

ind County Six between Stockley
of Knox And State of Tennessee
Andrew Jackson off the County
day above said of the other part
and Stockley Donelson for a
of the sum of two thousand
and paid the receipt wheret
hath and by these presents
all alien enfeoff And confirm
Jackson his heirs And a
certain tract or parcels of la
and acres one thousand acre

Indivinity Six between Stockley
of Knox And State of Tennessee
Andrew Jackson of the County
of Knox said of the other party
said Stockley Donelson for
the sum of two thousand
and paid the receipt where
nath And by these presents
by alien enfeoff And Confer
Jackson his heirs And
certain traits or parcels of La
and acres one thousand day
and six months And

Indivinity Six between Stockley
of Knox And State of Tennessee
Andrew Jackson off the County
taty Alor said of the other part
to auf Stockley Donelson for a
of thy Sum of two thousand
and paid thy receipt wherit
rath And by these presents
all alien encoff And Confir
Jackson his heirs And a
certain tracts or parcels of La
and acres one thousand Bayre

Subdivide image into non-overlapping rectangles. These rectangles are chosen small enough so that the illumination of each is approximately uniform. Then determine a global threshold for each subimage.

THRESHOLDING

- Python:
- Simple Threshold:

```
img = cv2.imread("bacteria.png",0)
ret,thresh1 = cv2.threshold(img,100,255,cv2.THRESH_BINARY)
```

To display:

```
plt.subplot(1,2,1),plt.imshow(img,'gray')
plt.title("Original")
plt.subplot(1,2,2),plt.imshow(thresh1,'gray')
plt.title("Original")
plt.show()
```

What is the best thresh?
Todo: try to change them

§ threshold()

```
double cv::threshold ( InputArray src,
                      OutputArray dst,
                      double thresh,
                      double maxval,
                      int type
)
```

Python:

```
retval, dst = cv.threshold( src, thresh, maxval, type[, dst] )
```

Original Image



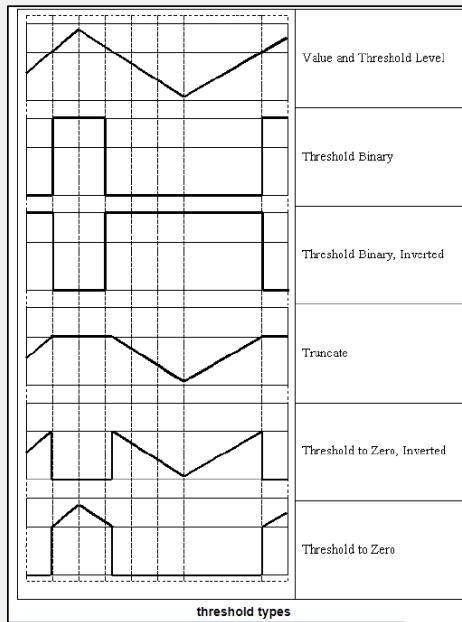
BINARY



THRESHOLDING

- Python:
- Threshold Type

https://docs.opencv.org/3.4.0/d7/d1b/group__imgproc__misc.html#gaa5576



Enumerator	
THRESH_BINARY	Python: cv.THRESH_BINARY
	$dst(x, y) = \begin{cases} maxval & \text{if } src(x, y) > thresh \\ 0 & \text{otherwise} \end{cases}$
THRESH_BINARY_INV	Python: cv.THRESH_BINARY_INV
	$dst(x, y) = \begin{cases} 0 & \text{if } src(x, y) > thresh \\ maxval & \text{otherwise} \end{cases}$
THRESH_TRUNC	Python: cv.THRESH_TRUNC
	$dst(x, y) = \begin{cases} threshold & \text{if } src(x, y) > thresh \\ src(x, y) & \text{otherwise} \end{cases}$
THRESH_TOZERO	Python: cv.THRESH_TOZERO
	$dst(x, y) = \begin{cases} src(x, y) & \text{if } src(x, y) > thresh \\ 0 & \text{otherwise} \end{cases}$
THRESH_TOZERO_INV	Python: cv.THRESH_TOZERO_INV
	$dst(x, y) = \begin{cases} 0 & \text{if } src(x, y) > thresh \\ src(x, y) & \text{otherwise} \end{cases}$
THRESH_MASK	Python: cv.THRESH_MASK
THRESH_OTSU	Python: cv.THRESH_OTSU
	flag, use Otsu algorithm to choose the optimal threshold value
THRESH_TRIANGLE	Python: cv.THRESH_TRIANGLE
	flag, use Triangle algorithm to choose the optimal threshold value

§ threshold()

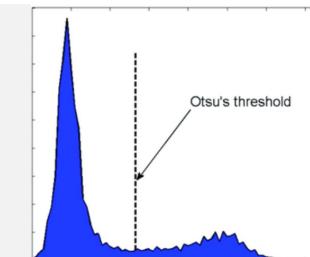
```
double cv::threshold ( InputArray src,
                      OutputArray dst,
                      double thresh,
                      double maxval,
                      int type
)
```

Python:

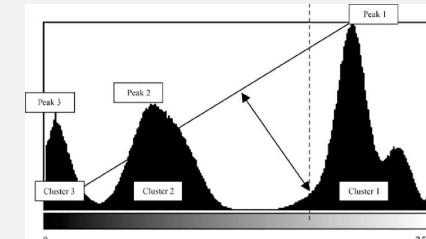
```
retval, dst = cv.threshold( src, thresh, maxval, type[, dst] )
```

Parameters

src input array (multiple-channel, 8-bit or 32-bit floating point).
 dst output array of the same size and type and the same number of channels as src.
 thresh threshold value.
 maxval maximum value to use with the THRESH_BINARY and THRESH_BINARY_INV thresholding types.
 type thresholding type (see the cv::ThresholdTypes).



Segmentation of Rice Seedlings Using the YCrCb Color Space and an Improved Otsu Method, Juan Liao et al. (2018)



Automatic and Accurate Extraction of Road Intersections from Raster Maps, Yao-Yi Chiang et al. (2009)

THRESHOLDING

- Python:
- Simple Threshold:

```
img = cv2.imread("bacteria.png",0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
```

```
plt.subplot(1,2,1),plt.imshow(img,'gray')
plt.title("Original")
plt.subplot(1,2,2),plt.imshow(thresh1,'gray')
plt.title("Original")
plt.show()
```

To display:

What is the best thresh?
Todo: try to change them

§ threshold()

```
double cv::threshold ( InputArray src,
OutputArray dst,
double thresh,
double maxval,
int type
)
```

Python:

```
retval, dst = cv.threshold( src, thresh, maxval, type[, dst] )
```

Original Image



BINARY



THRESHOLDING

- Python:
- Simple Threshold:

```
img = cv2.imread("bacteria.png",0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)
```

```
titles = ['Original
Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]
for i in range(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

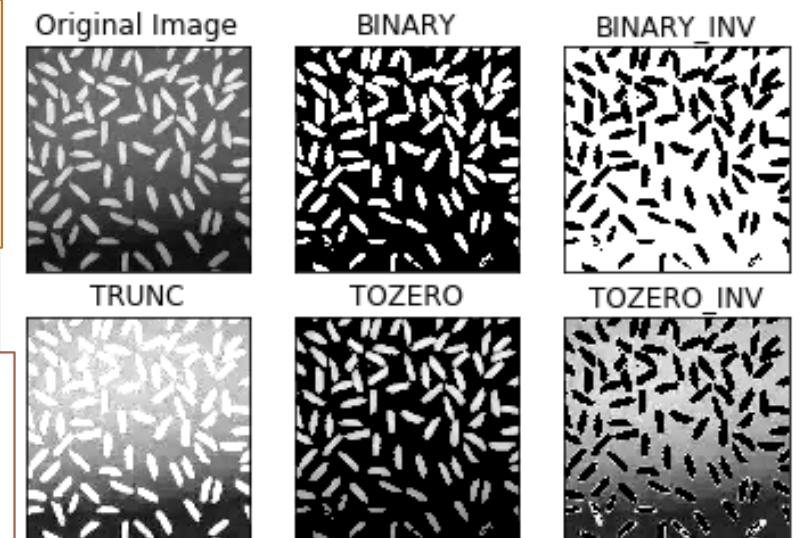
To display:

§ threshold()

```
double cv::threshold ( InputArray src,
                      OutputArray dst,
                      double      thresh,
                      double      maxval,
                      int        type
)
```

Python:

```
retval, dst = cv.threshold( src, thresh, maxval, type[, dst] )
```



THRESHOLDING

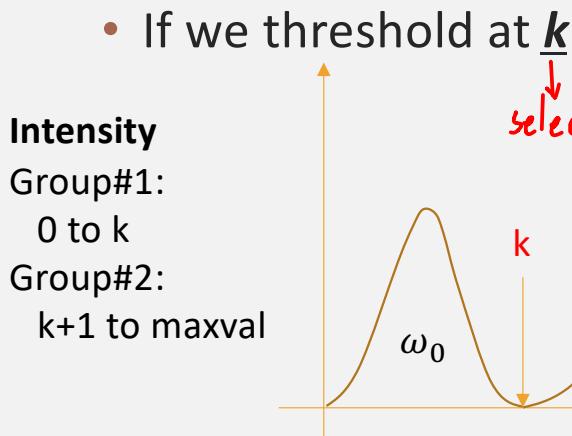
- **Otsu's thresholding:**

- An **automatic** method for choosing a best threshold.
- Image histogram as a probability distribution

$$p_i = \frac{n_i}{N}$$

n_i is the number of pixels with gray level i
 N is the total number of pixels

Normalized Histogram
 p_i probability of a pixel having gray level i



- If we threshold at k

Average mean:

$$\mu_T = \sum_{i=0}^{L-1} ip_i$$

Sum of pdf group #1:

$$\omega_0(k) = \sum_{i=0}^k p_i$$

Group#1 mean:

$$\mu_0(k) = \sum_{i=0}^k ip_i$$

Sum of pdf group #2:

$$\omega_1(k) = \sum_{i=k+1}^{L-1} p_i$$

Group#2mean:

$$\mu_1(k) = \sum_{i=k+1}^{L-1} ip_i$$

THRESHOLDING

- Otsu's thresholding:

- Best k

k which maximize (σ_B^2)

$$\text{Between-class variance } (\sigma_B^2(k)) = \frac{(\mu_T - \omega_0(k) \cdot \mu_0(k))^2}{\omega_0(k)(1-\omega_0(k))}$$

Algorithm [WIKIPEDIA]

1. Compute histogram and probabilities of each intensity level
2. Step through all possible thresholds k =(minimum intensity),... to... (maximum intensity - 1)
 1. Update ω_i and μ_i
 2. Compute σ_B^2
3. Desired threshold corresponds to the maximum $\sigma_B^2(k)$

group 1 = $0 \rightarrow k$
 group 2 = $k+1 \rightarrow \infty$

Ref: Otsu, N., "A Threshold Selection Method from Gray-Level Histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, 1979, pp. 62-66.

THRESHOLDING

- Otsu's thresholding

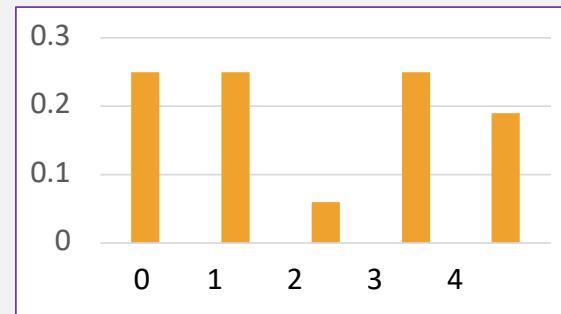
- Intensity level in range [0,4]

0	4	3	1
0	1	0	3
1	0	2	4
3	1	3	4

Find best threshold at k for segmenting this Image

Gray	0	1	2	3	4
n_i					
$p_i = n_i/N$					

$$\sum = 1$$



THRESHOLDING

k $\in [0 - 4]$

- Otsu's thresholding Find best threshold at k for segmenting this Image

Gray	0	1	2	3	4
n_i	4	4	1	4	3
$p_i = n_i/N$	0.25	0.25	0.06	0.25	0.19

Between-class variance $= \frac{(\mu_T \omega_0(k) - \mu_0(k))^2}{\omega_0(k) \omega_1(k)}$

$$\mu_T = 0 * 0.25 + 1 * 0.25 + 2 * 0.06 + 3 * 0.25 + 4 * 0.19 = 1.88$$

$$k = 0 \quad \omega_0(0) = 0.25, \omega_1(0) = 0.75 \quad \mu_0(0) = 0 * 0.25$$

$$\text{bc variance} = (1.88 * 0.25 - 0)^2 / (0.25 * 0.75) = 1.1781$$

$$k = 1 \quad w_0(1) = 0.5, w_1(1) = 0.5, \mu_0(1) = 0 * 0.25 + 1 * 0.25 = 0.25$$

bc variance = 1.9044

$$k = 2 \quad w_0(2) = 0.56, w_1(2) = 0.44, \mu_0(2) = 1 * 0.25 + 2 * 0.06 = 0.37$$

bc = 1.892

$$k = 3 \quad w_0(3) = 0.81, w_1(3) = 0.19, \mu_0(3) = 0.37 + 3 * 0.25 = 1.12$$

bc = 1.0543

THRESHOLDING

k is the best threshold, L is number of gray level .

- $k = 1$
- Group 1: $1, \dots, k \rightarrow$ gray level 0,1
- Group 2: $k+1, \dots, N \rightarrow$ gray level 2,3,4

$$f_{binary}(x, y) = \begin{cases} 0 & \text{if } 1 \leq f(x, y) \leq k \\ 1 & \text{if } k < f(x, y) \leq L \end{cases}$$

$\rightarrow 0$

$\rightarrow 1$

0	4	3	1
0	1	0	3
1	0	2	4
3	1	3	4

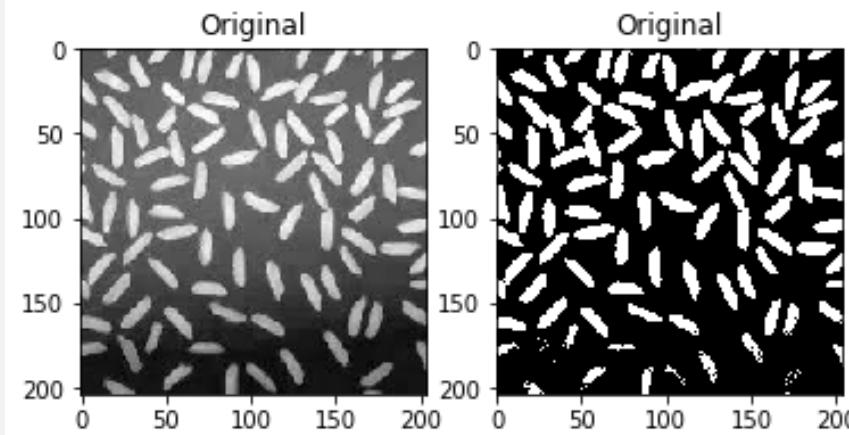


Binary image

THRESHOLDING

- Python:
- Otsu's Threshold:

```
img = cv2.imread("bacteria.png",0)
ret2,th2 = cv2.threshold(img,0,255,cv2.THRESH_OTSU)
```

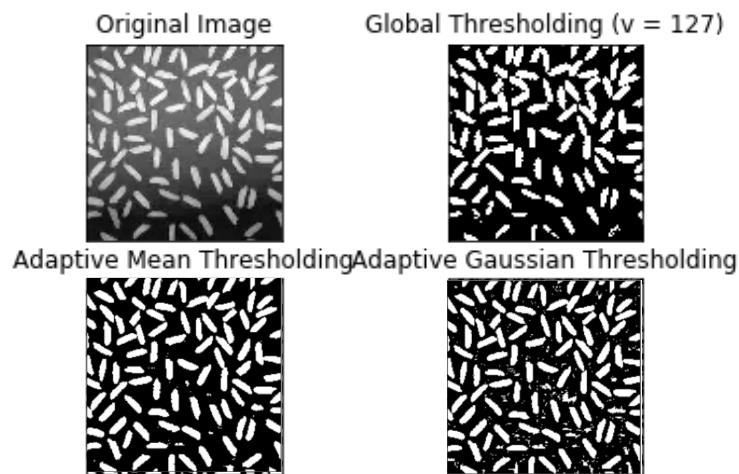


Advantages or disadvantages?

THRESHOLDING

- Python:
local threshold
- Adaptive Threshold:

```
img = cv2.imread("bacteria.png",0)
ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,\n                           cv2.THRESH_BINARY,21,0)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\n                           cv2.THRESH_BINARY,21,0)
```



§ adaptiveThreshold()

```
void cv::adaptiveThreshold ( InputArray src,
                            OutputArray dst,
                            double maxValue,
                            int adaptiveMethod,
                            int thresholdType,
                            int blockSize,
                            double C
                        )
```

Python:

```
dst = cv.adaptiveThreshold( src, maxValue, adaptiveMethod, thresholdType, blockSize, C[, dst] )
```

https://docs.opencv.org/3.4.0/d7/d1b/group_imgproc_misc.html#ga72b913f352e4a1b1b397736707afcde3

Parameters

src	Source 8-bit single-channel image.
dst	Destination image of the same size and the same type as src.
maxValue	Non-zero value assigned to the pixels for which the condition is satisfied
adaptiveMethod	Adaptive thresholding algorithm to use, see cv::AdaptiveThresholdTypes . The BORDER_REPLICATE BORDER_ISOLATED is used to process boundaries.
thresholdType	Thresholding type that must be either THRESH_BINARY or THRESH_BINARY_INV, see cv::ThresholdTypes .
blockSize	Size of a pixel neighborhood that is used to calculate a threshold value for the pixel. 3, 5, 7, and so on.
C	Constant subtracted from the mean or weighted mean (see the details below). Normally, it is positive but may be zero or negative as well.

Enumerator

ADAPTIVE_THRESH_MEAN_C Python: cv.ADAPTIVE_THRESH_MEAN_C	the threshold value $T(x,y)$ is a mean of the $\text{blockSize} \times \text{blockSize}$ neighborhood of (x,y) minus C
ADAPTIVE_THRESH_GAUSSIAN_C Python: cv.ADAPTIVE_THRESH_GAUSSIAN_C	the threshold value $T(x,y)$ is a weighted sum (cross-correlation with a Gaussian window) of the $\text{blockSize} \times \text{blockSize}$ neighborhood of (x,y) minus C. The default sigma (standard deviation) is used for the specified blockSize. See cv::getGaussianKernel

Increase or decrease blockSize?

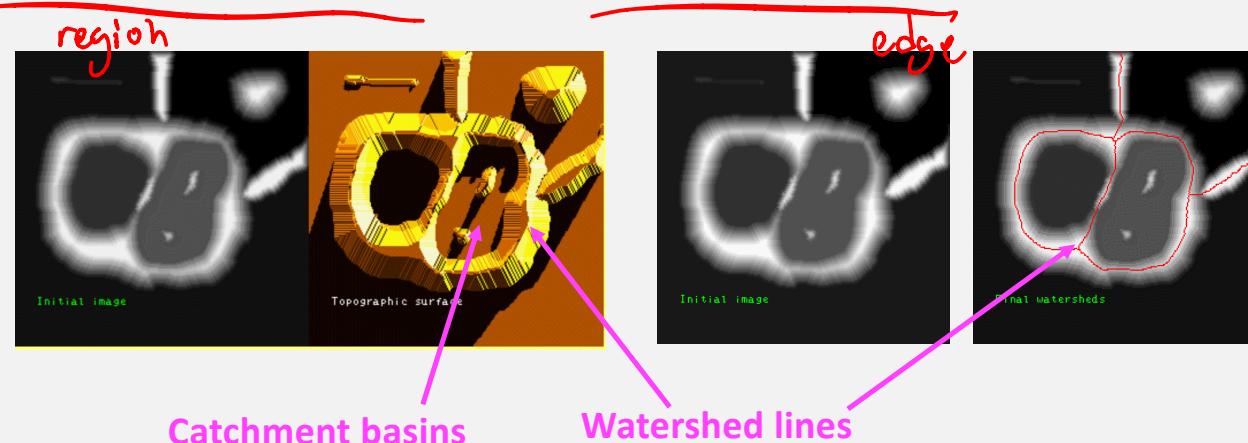
SEGMENTATION PART II

- Region-based segmentation
- K-means clustering
- Deep learning-based Techniques

REGION-BASED SEGMENTATION

REGION GLOWING

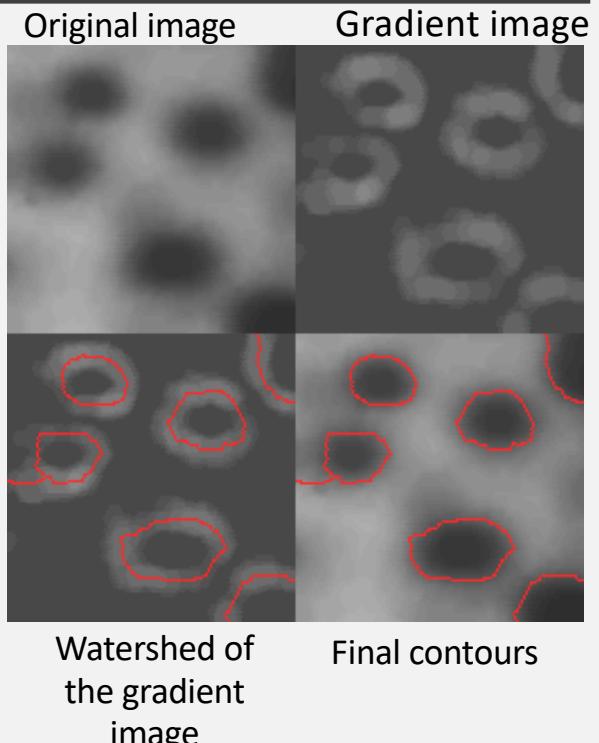
- If you have an image, how to select the most prominent contours?
 - Watershed transform - If we **flood this surface from its minima** and, if we prevent the merging of the waters coming from different sources, we partition the image into two different sets: **the catchment basins** and the **watershed lines**.



References: Serge Beucher, IMAGE SEGMENTATION AND MATHEMATICAL MORPHOLOGY,
<https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html>

REGION GLOWING

- If you have an image, how to select the most prominent contours?
 - Watershed transform - If we apply this transformation to the image gradient, the catchment basins should theoretically correspond to the homogeneous grey level regions of this image.
 - Marker-controlled watershed - A major enhancement of the watershed transformation consists in flooding the topographic surface from a previously defined set of markers.



References: Serge Beucher, IMAGE SEGMENTATION AND MATHEMATICAL MORPHOLOGY,
<https://people.cmm.minesparis.psl.eu/users/beucher/wtshed.html>

REGION GROWING

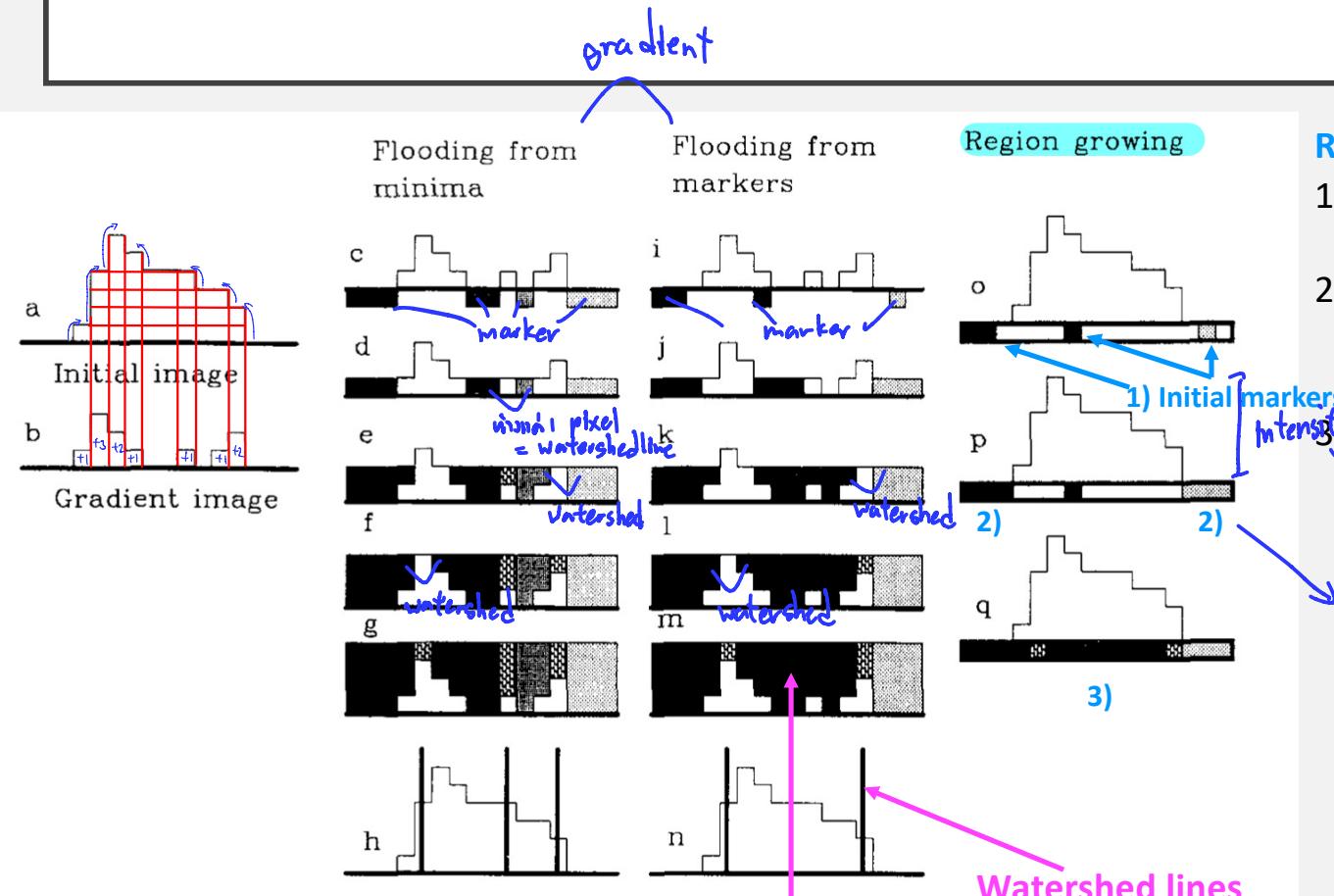


Fig.2 : Comparison between various region growing algorithms
Catchment basins

Reference: Fernand Meyer. Color image segmentation. In *Image Processing and its Applications*, 1992

REGION GLOWING

queue watershed line

g#1, #2 ~~0.04~~ Region growing step

Region growing step for color image:
1) RGB from Fig. 2a, a marker are drawn.

- 1) RGB from Fig. 3a-c, markers are defined in Fig. 3d
 - 2) Similarity between the point already in the markers and its neighbors. If (r_1, g_1, b_1) and (r_2, g_2, b_2) are RGB for pixel #1 and pixel #2, respectively, The color difference defined by
Max($|r_1 - r_2|, |g_1 - g_2|, |b_1 - b_2|$)
 - 3) An ordered queue with four levels is created, e.g., h and i (in the labelled region) is difference by 1, h will be put in the priority 1.
 - 4) First point to be extracted from the ordered queue is b which will get the same label with c (label = 3), a which is a neighbour of b is put into highest priority.
 - 5) Next to consider is l will have the same label as k (label = 4), and a has a label = 3.
 - 6) Now highest priority is empty and then suppressed, consider the next priority. h has label = 4, g, f are put in label = 4.
 - 7) e already in the queue and has neighbours of 2 distinct labelled regions, e is a frontier point.

Label = 1 is always assigned with a point in the queue, label = 2 is the frontier point (watershed line)

SEGMENTATION OF A COLOR IMAGE

Input Color Image

Red

Green

Blue

label

Marker Image Ordered queue

abiflat

allflat

inflow

watershed

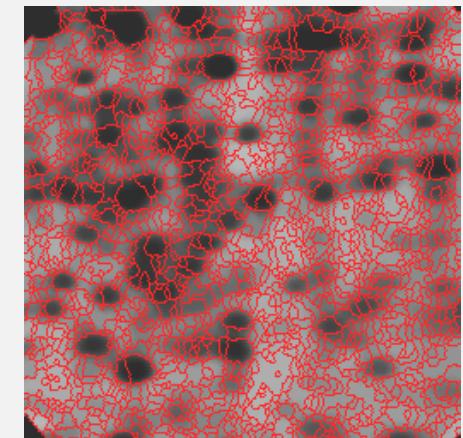
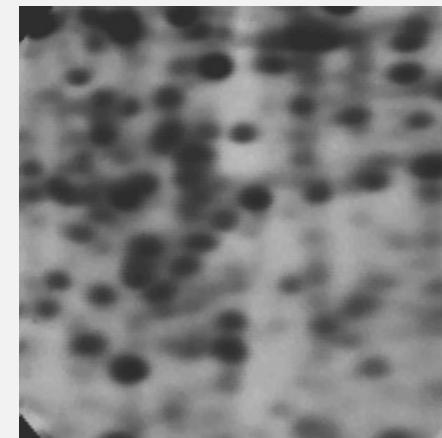
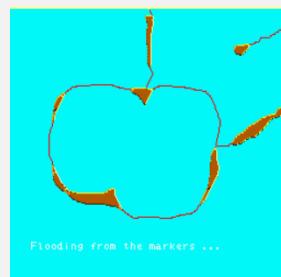
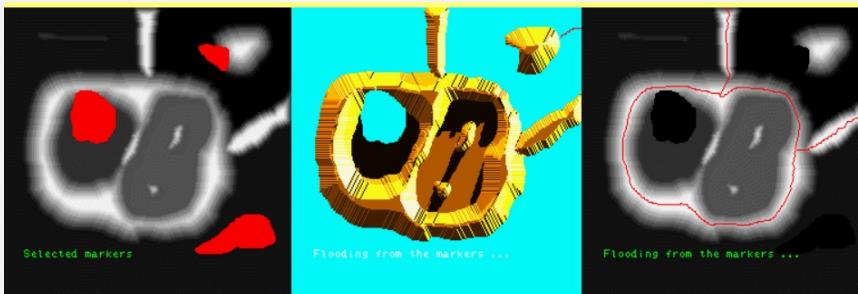
outflow

Fig. 3

Fernand Meyer. Color image segmentation. In *Image Processing and its Applications*, 1992

MARKER-BASED WATERSHED

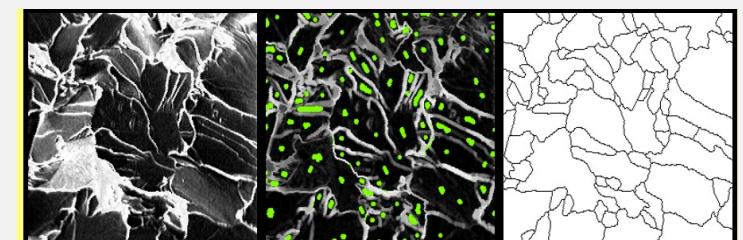
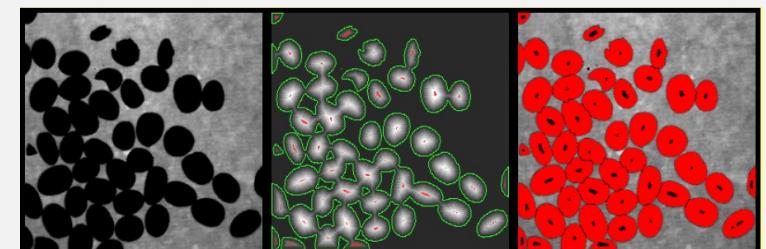
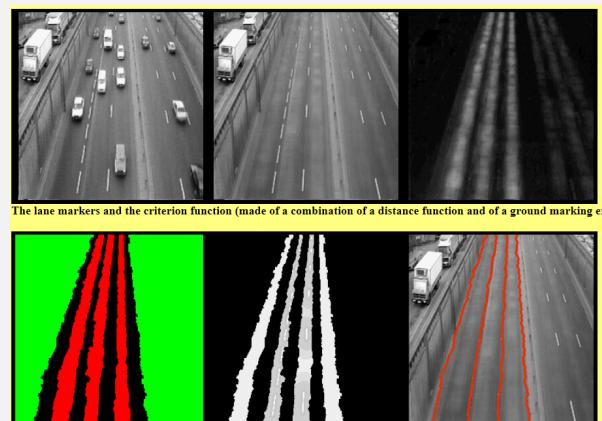
- Apply transformation on image gradient (can be over-segmentation)
- *Marker-controlled watershed*
- **we prevent any over-segmentation.**



<http://cmm.ensmp.fr/~beucher/wtshed.html>

MARKER-BASED WATERSHED

- Applications:
 - Road segmentation
 - Coffee beans separation
 - Traffic monitoring
 - Cleavage fracture in steel



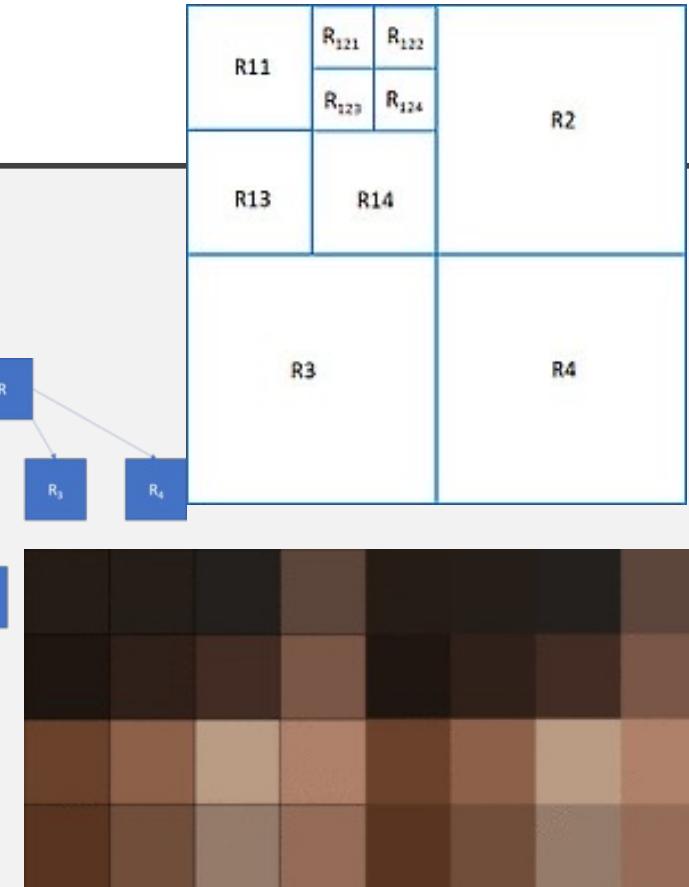
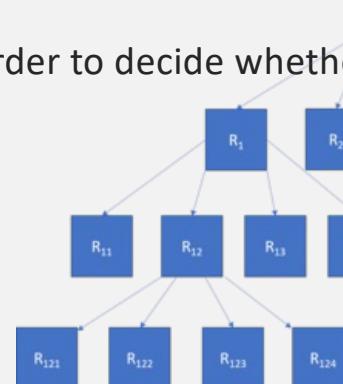
<http://cmm.ensmp.fr/~beucher/wtshed.html>

REGION SPLITTING AND MERGING

- In **Region splitting**, the whole image is first taken as a single region.
- If the region does not follow the predefined rules, then it is further divided into multiple regions (usually 4 quadrants).
- the predefined rules are carried out on those regions in order to decide whether to further subdivide or to classify that as a region.
- An example of a predefined rule:

$$|Z_{max} - Z_{min}| \leq T$$

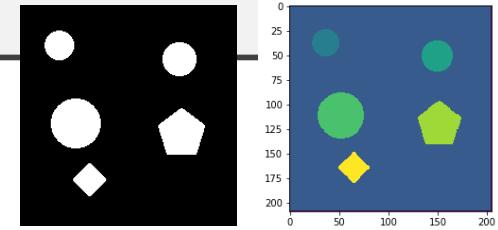
where Z_{max} and Z_{min} are the maximum and minimum pixel intensity values, respectively. T is a threshold.



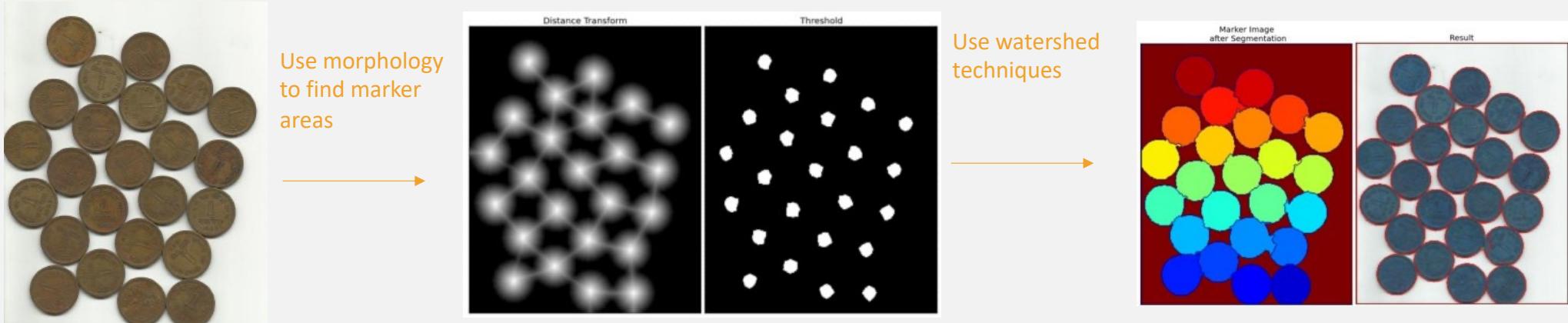
References:

https://en.wikipedia.org/wiki/Split_and_merge_segmentation
<https://towardsdatascience.com/image-segmentation-part-2-8959b609d268>

DEMO: REGION GROWING (MARKER-BASED WATERSHED)



- A topographic surface where high intensity denotes peaks and hills while low intensity denotes valleys
- To partition the image into two different sets:
the catchment basins and the watershed lines.



https://docs.opencv.org/3.1.0/d3/db4/tutorial_py_watershed.html Fernand Meye, The watershed concept and its use in segmentation (2012)

◆ watershed()

```
void cv::watershed ( InputArray image,
                     InputOutputArray markers
                   )
```

Python:

```
cv.watershed( image, markers ) -> markers
```

```
#include <opencv2/imgproc.hpp>
```

Performs a marker-based image segmentation using the watershed algorithm.

The function implements one of the variants of watershed, non-parametric marker-based segmentation algorithm, described in [169] .

Before passing the image to the function, you have to roughly outline the desired regions in the image markers with positive (>0) indices. So, every region is represented as one or more connected components with the pixel values 1, 2, 3, and so on. Such markers can be retrieved from a binary mask using **findContours** and **drawContours** (see the [watershed.cpp demo](#)). The markers are "seeds" of the future image regions. All the other pixels in markers , whose relation to the outlined regions is not known and should be defined by the algorithm, should be set to 0's. In the function output, each pixel in markers is set to a value of the "seed" components or to -1 at boundaries between the regions.

Note

Any two neighbor connected components are not necessarily separated by a watershed boundary (-1's pixels); for example, they can touch each other in the initial marker image passed to the function.

Parameters

image Input 8-bit 3-channel image.

markers Input/output 32-bit single-channel image (map) of markers. It should have the same size as image .

See also

[findContours](#)

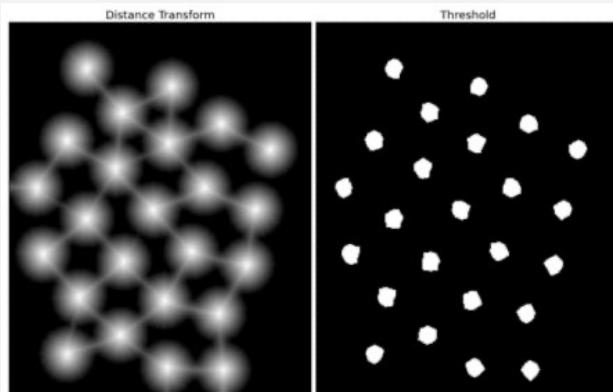
Examples:

[samples/cpp/watershed.cpp](#)

https://docs.opencv.org/4.5.3/d3/d47/group__imgproc__segmentation.html#ga3267243e4d3f95165d55a618c65ac6e1

DEMO: REGION GROWING (MARKER-BASED WATERSHED)

- Distance Transform:

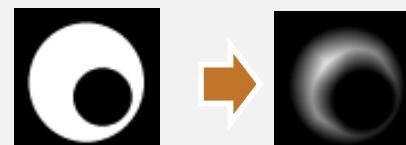


- The distance transform is an operator normally only applied to binary images.
- The result of the transform is a grey level image that looks similar to the input image, except that the grey level intensities of points inside foreground regions are changed to show **the distance to the closest boundary from each point**.

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

→

0	0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	1	0
0	1	2	2	2	2	2	1	0
0	1	2	3	3	2	1	0	
0	1	2	2	2	2	2	1	0
0	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0



References: Pedro Felzenszwalb and Daniel Huttenlocher. [Distance transforms of sampled functions](#). Technical report, Cornell University, 2004. <https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm>