



**TRƯỜNG ĐẠI HỌC VINH**  
**VINH UNIVERSITY**

*Nơi tạo dựng tương lai cho tuổi trẻ*



## Chương 4. Phân vùng ảnh

ThS. Nguyễn Thị Minh Tâm  
Email: tamntm@vinhuni.edu.vn

Đại học Vinh  
Viện Kỹ thuật Công nghệ

**ĐẠI HỌC VINH - 2022**



## Phân vùng ảnh

- Phân vùng ảnh nhằm phân tích ảnh thành những thành phần có cùng tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông có thể là cùng mức xám, cùng màu,...
- Các điểm trong một vùng ảnh có độ biến thiên giá trị mức xám tương đối đồng đều



## Tìm hiểu về phân vùng ảnh

1. Phân vùng dựa trên phân ngưỡng: threshold image

cv2.threshold

Ngưỡng nhị phân (binary), ngưỡng thích nghi (adaptive threshold), ngưỡng tối ưu (Otsu)

2. Các phép biến đổi hình thái: dilate, erode, opening, closing

3. Contour

4. Phân vùng ảnh bằng thuật toán watershed

Kiểm tra giữa kỳ: lý thuyết: 20h tối thứ 3 (5/4): 30'

Trắc nghiệm online

Thực hành: kiểm tra trên lớp thực hành (làm đúng đề yêu cầu)

Bắt buộc dùng máy bàn



## Phân vùng ảnh

- Giả sử ảnh  $X$  được phân thành  $N$  vùng khác nhau  $R_1, R_2, \dots, R_N$
- Quy tắc phân vùng là một xác nhận logic.
- Phân vùng ảnh chia tập  $X$  thành các tập con (vùng)  $R_i$  ( $i=1, 2, \dots, N$ ), sẽ có các tính chất sau:
- Các vùng  $R_i$  ( $i=1, 2, \dots, N$ ) phải bao phủ toàn bộ ảnh.
- Hai vùng khác nhau phải giao nhau bằng rỗng.  $R_i \cap R_j = \emptyset$  ( $i \neq j$ )
- Với mỗi vùng  $R_i$  ( $i=1, 2, \dots, N$ ), tính đồng đều trên vùng đó được bảo đảm.  $P(R_i) = \text{TRUE}$  ( $i=1, 2, \dots, N$ )
- Hợp của hai vùng ảnh khác nhau là một vùng không đồng đều  $P(R_i \cup R_j) = \text{FALSE}$  ( $i \neq j$ ).



## Các kỹ thuật phân vùng ảnh

- Dựa vào đặc tính của vùng ảnh, kỹ thuật phân vùng ảnh có thể được chia thành ba phương pháp cơ bản sau đây.
  - Phân vùng ảnh dựa vào ngưỡng của độ xám.
  - Phân vùng ảnh dựa vào sự phân chia miền kề.
  - Phân vùng ảnh dựa vào xác định biên.



## Phân vùng ảnh dựa vào ngưỡng của độ xám

- Ngưỡng (threshold) là một số nằm trong đoạn từ 0 đến 255. Giá trị ngưỡng sẽ chia tách giá trị độ xám của ảnh thành 2 miền riêng biệt.
  - Miền thứ nhất là tập hợp các điểm ảnh có giá trị nhỏ hơn giá trị ngưỡng.
  - Miền thứ hai là tập hợp các các điểm ảnh có giá trị lớn hơn hoặc bằng giá trị ngưỡng



## Thuật toán Simple Thresholding

- Phân ngưỡng ảnh đơn giản (Simple Thresholding) thực hiện phân ngưỡng bằng cách:
  - Thay thế giá trị lớn hơn hoặc bằng ngưỡng 1 giá trị mới
  - Thay thế giá trị bé hơn giá trị ngưỡng bằng 1 giá trị mới

```
if src[i] >= T:  
    dest[i] = MAXVAL  
else:  
    dest [i] = 0
```



# Thuật toán Simple Thresholding

```
import cv2
import numpy as np
from matplotlib import pyplot as plt

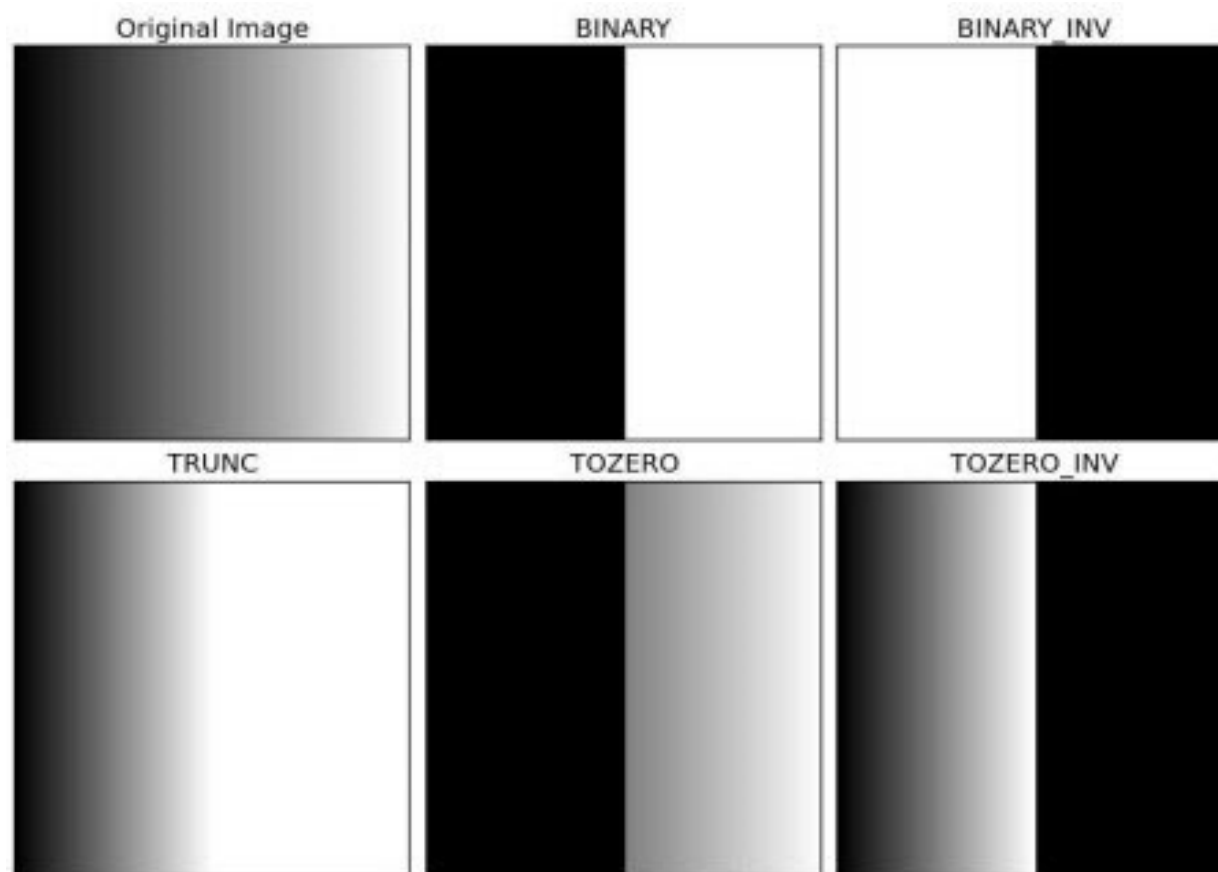
img = cv2.imread('gradient.png',0)
ret,thresh1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
ret,thresh2 = cv2.threshold(img,127,255,cv2.THRESH_BINARY_INV)
ret,thresh3 = cv2.threshold(img,127,255,cv2.THRESH_TRUNC)
ret,thresh4 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO)
ret,thresh5 = cv2.threshold(img,127,255,cv2.THRESH_TOZERO_INV)

titles = ['Original Image','BINARY','BINARY_INV','TRUNC','TOZERO','TOZERO_INV']
images = [img, thresh1, thresh2, thresh3, thresh4, thresh5]

for i in xrange(6):
    plt.subplot(2,3,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])

plt.show()
```







## Thuật toán Simple Thresholding

THRESH\_BINARY: Nếu giá trị pixel lớn hơn ngưỡng thì gán bằng maxval  
Ngược lại gán bằng 0

THRESH\_BINARY\_INV: Nếu giá trị pixel lớn hơn ngưỡng thì gán bằng 0  
Ngược lại gán bằng maxval

THRESH\_TRUNC: Nếu giá trị pixel lớn hơn ngưỡng thì gán giá trị bằng ngưỡng  
Ngược lại giữ nguyên giá trị

THRESH\_TOZERO: Nếu giá trị pixel lớn hơn ngưỡng thì giữ nguyên giá trị  
Ngược lại gán bằng 0

THRESH\_TOZERO\_INV: Nếu giá trị pixel lớn hơn ngưỡng thì gán giá trị bằng 0  
Ngược lại giữ nguyên



## Thuật toán Adaptive Thresholding

- Phương pháp phân ngưỡng đơn giản không phù hợp cho nhiều trường hợp, như là ánh sáng không đồng đều trên ảnh. Trong trường hợp đó chúng ta dùng hàm `adaptiveThreshold()`.
- Thuật toán Adaptive Thresholding phân ngưỡng thích nghi: chia nhỏ bức ảnh thành những vùng nhỏ (region), và đặt giá trị ngưỡng trên những vùng nhỏ đó.

Phương pháp này tính giá trị trung bình của các  $n$  điểm xung quanh pixel đó rồi trừ cho  $C$  chứ không lấy ngưỡng cố định ( $n$  thường là số lẻ, còn  $C$  là số nguyên bất kỳ).



# Thuật toán Adaptive Thresholding

**`cv2.adaptiveThreshold(src, maxValue, adaptiveMethod, thresholdType, blockSize, C)`**

- **max\_value:** giá trị mức xám lớn nhất (255)
- **adaptiveMethod:**
  - `cv2.ADAPTIVE_THRESH_MEAN_C`: giá trị ngưỡng  $T(x,y)$  là trung bình của vùng các điểm lân cận của  $(x,y)$   $\text{blockSize} \times \text{blockSize}$  trừ  $C$ .
  - `cv2.ADAPTIVE_THRESH_GAUSSIAN_C`: giá trị ngưỡng  $T(x, y)$  là tổng có trọng số (tương quan chéo với cửa sổ Gaussian) của vùng lân cận  $\text{blockSize} \times \text{blockSize}$  của  $(x, y)$  trừ đi  $C$ .
- **thresholdType:** Kiểu phân ngưỡng phải là 1 trong 2 kiểu `THRESH_BINARY` hoặc `THRESH_BINARY_INV`
- **block\_size:** kích thước cửa sổ cho việc tính toán ngưỡng cửa sổ này có kích thước là 3, 5, 7, ...
- **C:** là tham số bù trừ trong trường hợp cần điều chỉnh đối với những bức ảnh có độ tương phản quá lớn. Khi đó, ngưỡng động thực sự là giá trị trung bình trong cửa sổ trừ đi  $C$ .



# Thuật toán Adaptive Thresholding

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('sudoku.png',0)
img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C, cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C, cv.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)',
          'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]
for i in range(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))
plt.show()
```



## Hàm phân ngưỡng Adaptive Thresholding

- `cv2.adaptiveThreshold`
- `blockSize`: Kích thước của vùng, bắt buộc phải là một số lẻ lớn hơn 0.
- `C`: hằng số, giá trị từ -255 đến 255. Có thể gán `C` bằng 0 để đỡ rối.
- `adaptiveMethod` nhận vào một trong hai giá trị là `cv.ADAPTIVE_THRESH_MEAN_C` và `cv.ADAPTIVE_THRESH_GAUSSIAN_C`, đó là các phương pháp tính ngưỡng.
- `ADAPTIVE_THRESH_MEAN_C`: Tính trung bình các láng giềng xung quanh điểm cần xét trong vùng `blockSize * blockSize` trừ đi giá trị hằng số `C`.
- `ADAPTIVE_THRESH_GAUSSIAN_C`: Nhân giá trị xung quanh điểm cần xét với trọng số gauss rồi tính trung bình của nó, sau đó trừ đi giá trị hằng số `C`.
- `thresholdType`: Tương tự như Simple Thresholding đã trình bày ở trên.



## Ví dụ Adaptive Thresholding

```
import cv2 as cv
import numpy as np
from matplotlib import pyplot as plt
img = cv.imread('sudoku.png',0)
img = cv.medianBlur(img,5)
ret,th1 = cv.threshold(img,127,255,cv.THRESH_BINARY)
th2 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_MEAN_C,\
    cv.THRESH_BINARY,11,2)
th3 = cv.adaptiveThreshold(img,255,cv.ADAPTIVE_THRESH_GAUSSIAN_C,\
    cv.THRESH_BINARY,11,2)
titles = ['Original Image', 'Global Thresholding (v = 127)',
    'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]
for i in xrange(4):
    plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    plt.title(titles[i])
    plt.xticks([],plt.yticks([]))
plt.show()
```

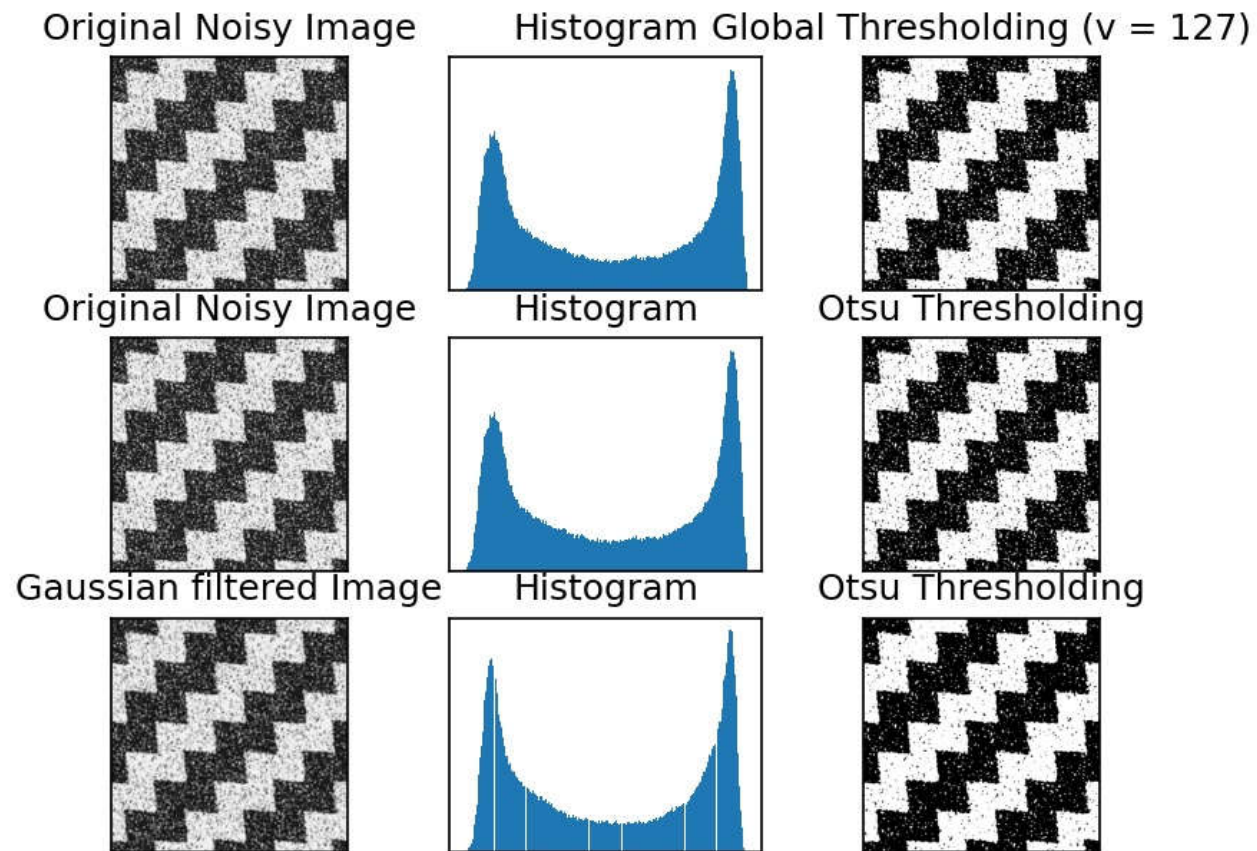


## Phân ngưỡng với ngưỡng tối ưu (Otsu)

- Phân ngưỡng Otsu tự động tính toán một giá trị ngưỡng từ histogram của ảnh và cho kết quả tốt trong trường hợp histogram của ảnh có hai đỉnh (nên sử dụng bộ lọc Gauss trước khi phân ngưỡng Otsu)
- Hàm `cv2.threshold()` được sử dụng, nhưng truyền vào bổ sung giá trị `cv2.THRESH_OTSU`. Đối với giá trị ngưỡng, chỉ cần truyền giá trị 0. Thuật toán sẽ tìm giá trị ngưỡng tối ưu và trả về dưới dạng đầu ra thứ hai, `retVal`.
- `ret2,th2 = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)`



# Phân ngưỡng với ngưỡng tối ưu (Otsu)





## Đường bao đối tượng (Contour)

- Contour là “tập các điểm-liên-tục tạo thành một đường cong (curve) (boundary), và không có khoảng hở trong đường cong đó.
- Trong một contour, các điểm có cùng /gần xấp xỉ một giá trị màu, hoặc cùng mật độ.
- Contour là một công cụ hữu ích được dùng để phân tích hình dạng đối tượng, phát hiện đối tượng và nhận dạng đối tượng”.



## Contour của ảnh

- Hàm tìm Contour của ảnh:

**`contours, hierarchy = cv2.findContours(image, mode, method)`**

Trong đó:

- **contours**: Danh sách các contour có trong bức ảnh nhị phân. Mỗi một contour được lưu trữ dưới dạng vector tọa độ các điểm
- **hierarchy**: Danh sách các vector, chứa mối quan hệ giữa các contour.
- **image**: ảnh nhị phân 8 bit (thu được bằng cách sử dụng các hàm `threshold`, `adaptiveThreshold`, `Canny` trên ảnh gốc)
- **mode**: các kiểu trích xuất contour, có các dạng sau: **RETR\_TREE**, **RETR\_EXTERNAL**, **RETR\_LIST**, **RETR\_CCOMP**, **RETR\_FLOODFILL**.
- **method**: các phương pháp xấp xỉ contour. Có các phương thức sau: **CHAIN\_APPROX\_SIMPLE**, **CHAIN\_APPROX\_NONE**, **CHAIN\_APPROX\_TC89\_L1**, **CHAIN\_APPROX\_TC89\_KCOS**.



## Contour của ảnh

- Hàm vẽ contour:
- `drawContours(image, contours, contourIdx, color, thickness)`
- Trong đó:
  - `image`: ảnh, có thể là ảnh grayscale hoặc ảnh màu.
  - `contours`: danh sách các contour, nếu để số âm: vẽ tất cả
  - `contourIndex`: chỉ định contour được vẽ, thông thường chúng ta để -1
  - `color`: Giá trị màu của contour chúng ta muốn vẽ, ở dạng BGR, vd vẽ contour màu xanh lá cây thì set là `(0,255,0)`.
  - `thickness` : độ dày của đường contour cần vẽ, giá trị `thickness` là số âm thì sẽ vẽ hình được tô màu



## Vẽ Contour

- Đọc ảnh
- Đổi sang ảnh xám
- Phân ngưỡng
- Tìm contour (`findContours`)
- Vẽ contour



## Ví dụ tìm Contour

```
import cv2

img = cv2.imread('G:/Codes/colorfull_ballon.jpg')
imggray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
ret, thresh = cv2.threshold(imggray, 127, 255, 0)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)
print("Number of contours = " + str(len(contours)))
print(contours[0])

cv2.drawContours(img, contours, -1, (0, 255, 0), 3)
cv2.drawContours(imggray, contours, -1, (0, 255, 0), 3)

cv2.imshow('Image', img)
cv2.imshow('Image GRAY', imggray)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



## Ví dụ Contour của ảnh

```
import cv2
import numpy as np
import random as rng #Để tạo số ngẫu nhiên, lấy màu vẽ contour
img = cv2.imread('colorfull_ballon.jpg')

img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
img_gray = cv2.blur(img_gray, (3,3))

lower_threshold = 90
# Phát hiện biên, đổi sang ảnh nhị phân
canny_output = cv2.Canny(img_gray, lower_threshold, lower_threshold * 2)
# Find contours
contours, _ = cv2.findContours(canny_output, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
print(contours)
# Draw contours
img_out = np.zeros((canny_output.shape[0], canny_output.shape[1], 3), dtype=np.uint8) #Tạo ảnh đen để vẽ contour
for i in range(len(contours)):
    color = (rng.randint(0,256), rng.randint(0,256), rng.randint(0,256))
    cv2.drawContours(img_out, contours, i, color, 2)

cv2.imshow('Original',img)
cv2.imshow('Contours', img_out)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



- Buổi tối báo cáo:

- ☐ Các phép biến đổi hình thái (morphology): erode, dilate, open, close
- ☐ Thuật toán phân vùng watershed

- **Làm bài tập nộp:**

- Đọc barcode
- Đọc qrcode
- Phát hiện mặt người (face detection)





- **Làm bài tập nộp:**
- Đọc barcode
- Đọc qrcode
- Phát hiện mặt người (face detection)
- Nhận diện



## Xử lý hình thái học – Image Morphology

- Xử lý các phép toán liên quan đến hình thái học, chủ yếu dùng với ảnh xám và ảnh nhị phân nhằm:
- Làm mỏng đối tượng, loại bỏ các đối tượng nhỏ do nhiễu sinh ra
- Làm dày đối tượng, nối liền các phần rời rạc của đối tượng
- Làm đầy chỗ trống bên trong đối tượng
- Các phép xử lý hình thái học thường được dùng trong tiền xử lý cho các bài toán nhận dạng mã vạch (barcode), biển số xe (license plates),...







## Xử lý hình thái học – Image Morphology

- Các phép toán cơ bản:
- Dilate: Làm dày đối tượng, nối liền các đối tượng
- Erode: Làm mỏng đối tượng, khử nhiễu, loại bỏ các đối tượng nhỏ
- Opening transform: lần lượt sử dụng 2 phép toán Erode và Dilate để loại bỏ nhiễu và làm dày đối tượng trở về hình dạng ban đầu
- Closing transform: lần lượt sử dụng 2 phép toán dilate và erode để loại bỏ nhiễu và làm dày đối tượng, nối liền các phần của đối tượng, sau đó làm mỏng đối tượng đưa đối tượng trở về hình dạng ban đầu



## Xử lý hình thái học

Xói mòn: Erosion	Giãn nở: Dilation	Mở: Opening Erode và dilate	Đóng: Closing Dilate và Erode
			
<pre>img = cv.imread('j.png',0) kernel = np.ones((5,5),np.uint8) erosion = cv.erode(img,kernel,iterations = 1) dilation = cv.dilate(img,kernel,iterations = 1) opening = cv.morphologyEx(img, cv.MORPH_OPEN, kernel) closing = cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)</pre>			



## Distance transform

- Tính khoảng cách từ mỗi điểm ảnh đến điểm ảnh khác 0 gần nhất
- Ví dụ

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

Ảnh nhị phân

0.00	0.00	1.00	2.00	3.00
0.00	0.00	1.00	2.00	3.00
1.00	1.00	1.41	2.00	2.24
1.41	1.00	1.00	1.00	1.41
1.00	0.00	0.00	0.00	1.00

Ảnh áp dụng  
distance transform

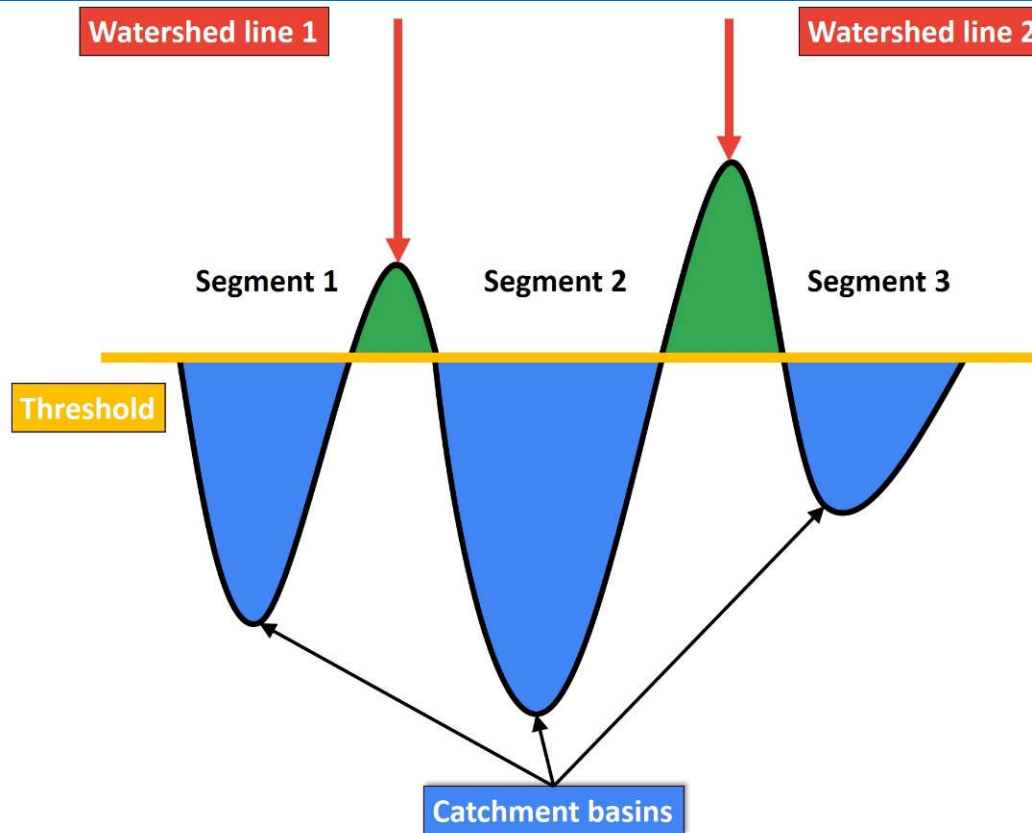


## Thuật toán phân vùng ảnh Watershed

- Thuật toán Watershed cho phép người dùng đánh dấu những vùng hình ảnh chứa object, background và cả những vùng không chắc chắn. Sau khi thuật toán kết thúc, những vùng không chắc chắn sẽ được gán nhãn foreground hoặc background, các đường ranh giới giữa các vùng cũng sẽ được phát hiện
- OpenCV cung cấp hàm **watershed** để thực hiện Image Segmentation.
- **cv2.watershed(inputImage, markers)**
- Trong đó:
  - inputImage - ảnh input 8-bit, 3 channels;
  - markers - ảnh grayscale 32-bit có kích thước của input image chứa các markers của các vùng ảnh foreground, background và “unknown zone” (marker = 0)



# Thuật toán phân vùng ảnh Watershed





## Các bước phân vùng ảnh bằng watershed

- B1: Đọc ảnh
- B2: Chuyển sang ảnh xám, Phân ngưỡng  
`ret, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)`
- B3: lọc nhiễu (dilate và erode hoặc opening)
- **dilate và erode:**
  - `thresh = cv2.dilate(thresh, None, iterations=3)`
  - `thresh = cv2.erode(thresh, None, iterations=3)`
- **opening:**
  - `kernel = np.ones((3,3),np.uint8)`
  - `opening = cv2.morphologyEx(thresh, cv2.MORPH_OPEN, kernel, iterations=2)`
- B4: Tách đối tượng (sử dụng distance transform): mỗi điểm ảnh đối tượng sẽ được thay thế bằng khoảng cách của nó tới điểm ảnh nền gần nhất. Sau khi sử dụng thuật toán, các điểm ảnh ở gần tâm đối tượng (xa biên) sẽ có giá trị cao hơn, còn các điểm ảnh càng ở gần biên sẽ có giá trị nhỏ hơn, giá trị này nằm trong khoảng `[0, 1]`  
`distanceTransform (bw, dist, CV_DIST_L2, 3)`
- B5: Phân ngưỡng ảnh ở B4 để tách đối tượng thành các vùng rời rạc

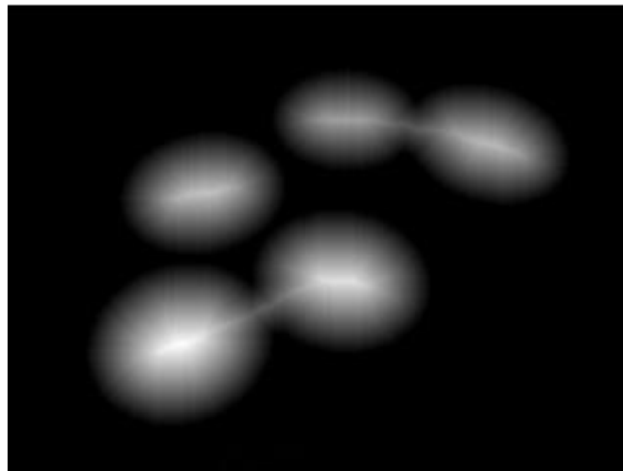




## Kết quả sau khi sử dụng hàm biến đổi khoảng cách



Ảnh ban đầu



Ảnh sau khi áp dụng  
distanceTransform



Ảnh phân ngưỡng để  
tách đối tượng



## Các bước phân vùng ảnh bằng watershed

- B6: Tìm biên của các đối tượng: sử dụng hàm findContours.

**findContours (src, contours, CV\_RET\_EXTRNAL, CV\_CHAIN\_APPROX\_SIMPLE)**

Trong đó:

- CV\_RET\_EXTRNAL: chỉ lấy những đường biên bên ngoài, những đường biên bên trong của đối tượng bị loại bỏ.
- CV\_CHAIN\_APPROX\_SIMPLE: nén đường viền trước khi lưu trữ, nén phân đoạn theo chiều ngang, chiều dọc và chéo.



## Các bước phân vùng ảnh bằng watershed

- B7: Tạo ảnh marker: ảnh cho thuật toán watershed biết đâu là đối tượng đang xét và đâu là đối tượng nền (tách foreground và background)  
`markers = np.zeros (src.size(), CV_32SC1)`
- Vẽ đường viền đối tượng:
- `for i in range(len(contours):`  
    `drawContours (marker, coutours, i, (0,255,0), -1,2)`  
    `cv2.circle (markers, Point (5, 5) , 3, CV_RGB (255, 255, 255) , -1)`
- B7: Thực hiện thuật toán watershed để xác định ranh giới giữa các đối tượng

`watershed (src, markers)`



## Face detection

```
import cv2

faceCascade=
cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
img = cv2.imread('lena.png')
imgGray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

faces = faceCascade.detectMultiScale(imgGray,1.1,4)

for (x,y,w,h) in faces:
    cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

cv2.imshow("Result", img)
cv2.waitKey(0)
```



- Trích chọn đặc trưng (features extraction)



## Đọc barcode

```
import cv2
import numpy as np
from pyzbar.pyzbar import decode

img = cv2.imread('nbarcode.jpg')
img1 = cv2.imread('nbarcode.jpg')

for barcode in decode(img):
    myData = barcode.data.decode('utf-8')
    print(myData)
    pts = np.array([barcode.polygon], np.int32)
    pts = pts.reshape((-1, 1, 2))
    cv2.polylines(img, [pts], True, (255, 0, 255), 5)
    pts2 = barcode.rect
    cv2.putText(img, myData, (pts2[0], pts2[1]), cv2.FONT_HERSHEY_SIMPLEX,
                0.9, (255, 0, 255), 2)
cv2.imshow('Original', img1)
cv2.imshow('Result', img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```



## Bài tập

- Đọc barcode, qrcode
- Phát hiện khuôn mặt (face detection)
- Nhận dạng ký tự quang học ( OCR - Optical Character Recognition): sử dụng thư viện tesseract
  - Cài thư viện pytesseract: `pip install pytesseract`
  - Download gói data: <https://github.com/tesseract-ocr/tessdata>
  - Download gói ngôn ngữ:
    - <https://github.com/tesseract-ocr/langdata> -> Code -> Download Zip
    - Gói tiếng việt: vie
- Nộp vào mục bài tập trong Teams: cả slide và chương trình: 1 file ppt, mỗi chương trình 1 file, đặt tên file là tên sv (không nén file)



- Tìm hiểu về cách dùng hàm trong python. Viết chương trình tạo menu tương ứng với các phép xử lý ảnh đã học.
- Ví dụ:
  - 1. Biến đổi ảnh xám
  - 2. Xoay ảnh
  - 3. Cắt ảnh
  - .....
- Làm dưới dạng giao diện: GUI





```
import cv2
import numpy as np

cap = cv2.VideoCapture('vtest.avi')

ret, frame1 = cap.read()
ret, frame2 = cap.read()

while cap.isOpened():
    diff = cv2.absdiff(frame1, frame2)
    gray = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    _, thresh = cv2.threshold(blur, 20, 255, cv2.THRESH_BINARY)
    dilated = cv2.dilate(thresh, None, iterations=3)
    contours, _ = cv2.findContours(dilated, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)

    for contour in contours:
        (x, y, w, h) = cv2.boundingRect(contour)

        if cv2.contourArea(contour) < 700:
            continue
        cv2.rectangle(frame1, (x, y), (x+w, y+h), (0, 255, 0), 2)
        cv2.putText(frame1, "Status: {}".format('Movement'), (10, 20), cv2.FONT_HERSHEY_SIMPLEX,
                    1, (0, 0, 255), 3)
    #cv2.drawContours(frame1, contours, -1, (0, 255, 0), 2)
```

*Thank you!*



**TRƯỜNG ĐẠI HỌC VINH**  
**VINH UNIVERSITY**

*Nơi tạo dựng tương lai cho tuổi trẻ*

