THỰC HÀNH VỀ PHÂN VÙNG ẢNH

1) Phân ngưỡng toàn cục (heuristic iterative threshold)

- **Đề bài:** Tách sản phẩm khỏi nền trên ảnh chụp băng chuyền có ánh sáng tương đối đều.
- **Mục tiêu:** Áp dụng thuật toán ngưỡng toàn cục, hội tụ $T = (m_1+m_2)/2$.
- **Yêu cầu:** Ảnh xám/hoặc RGB (sẽ chuyển xám); báo cáo giá trị ngưỡng và mask nhị phân.
- **Hướng dẫn tóm tắt:**
- Chuyển xám → 2) Khởi tạo T = mean(gray) → 3) Lặp tách G_1/G_2, cập nhật T → 4) Ngưỡng → 5) Hậu xử lý (morphology nếu cần).

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
import cv2, numpy as np, matplotlib.pyplot as plt

def global_threshold(gray, eps=1e-3, max_iter=100):
    T = float(np.mean(gray))
    for _ in range(max_iter):
        G1, G2 = gray[gray >= T], gray[gray < T]
        if len(G1) == 0 or len(G2) == 0: break
        m1, m2 = float(np.mean(G1)), float(np.mean(G2))
        newT = 0.5*(m1+m2)
        if abs(newT - T) <= eps:
            T = newT; break
        T = newT
    _, binimg = cv2.threshold(gray, T, 255, cv2.THRESH_BINARY)
    return binimg, T

img = cv2.imread("data/conveyor.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
binimg, T = global_threshold(gray)
print("T =", round(T,2))
plt.figure(figsize=(8,4)); plt.subplot(1,2,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,2,2); plt.imshow(binimg, cmap='gray'); plt.title("Global threshold"); plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

2) Otsu tối ưu (between-class variance)

Đề bài: Đếm số viên linh kiện trên nền phẳng dưới điều kiện sáng ổn định.

Mục tiêu: Dùng Otsu để tự động tìm ngưỡng tối ưu.

Yêu cầu: Hiển thị histogram và mask; so sánh với ngưỡng thủ công.

Hướng dẫn tóm tắt:

1) Chuyển xám →
2) cv2.threshold(..., THRESH_OTSU)
3)) Vẽ histogram
4) So sánh.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
gray = cv2.cvtColor(cv2.imread("data/parts.jpg"), cv2.COLOR_BGR2GRAY)
T, binimg = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
print("Otsu T =", T)
plt.figure(figsize=(9,4))
plt.subplot(1,3,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,3,2); plt.hist(gray.ravel(), 256); plt.title("Histogram")
plt.subplot(1,3,3); plt.imshow(binimg, cmap='gray'); plt.title("Otsu mask");
plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

3) Phân ngưỡng thích nghi (Mean/Gaussian)

Đề bài: Tách chữ in trên hóa đơn bị bóng/độ sáng không đều.

Mục tiêu: Dùng adaptive threshold (mean/gaussian).

Yêu cầu: Chọn blockSize, C hợp lý; so sánh với Otsu.

Hướng dẫn tóm tắt:

1) Gray
2) cv2.adaptiveThreshold (MEAN_C/Gaussian_C)
3) Điều chỉnh blockSize, C.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
gray = cv2.cvtColor(cv2.imread("data/receipt.jpg"), cv2.COLOR_BGR2GRAY)
bin_mean = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE_THRESH_MEAN_C,
                                 cv2.THRESH_BINARY, 35, 7)
bin_gaus = cv2.adaptiveThreshold(gray,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
                                 cv2.THRESH_BINARY, 35, 7)
plt.figure(figsize=(9,3))
plt.subplot(1,3,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,3,2); plt.imshow(bin_mean, cmap='gray'); plt.title("Adaptive MEAN")
plt.subplot(1,3,3); plt.imshow(bin_gaus, cmap='gray'); plt.title("Adaptive GAUSS")
plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

4) Bayes / Maximum Likelihood Thresholding

Đề bài: Phân tách vùng rỉ sét (đối tượng) trên bề mặt kim loại (nền) khi histogram chồng lấn nhẹ.

Mục tiêu: Dựa vào giả định phân bố Gauss cho H0/H1 và prior P0/P1 để tìm ngưỡng ML/Bayes.

Yêu cầu: Chọn tham số μ,σ gần đúng theo thống kê mẫu nhỏ; hiển thị T và kết quả.

Hướng dẫn tóm tắt:

1) Lấy mẫu pixel hai lớp
2) Ước lượng μ,σ,P
3) Tính T
4) Ngưỡng toàn ảnh.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
def calc_threshold(mu0,s0,mu1,s1,P0,P1):
    # nghiệm xấp xỉ cho trường hợp Gauss khác phương sai
    return (mu0*s1**2 - mu1*s0**2 + s0*s1*np.sqrt((mu1-mu0)**2 + 2*(s1**2 -
s0**2)*np.log((s1*P0)/(s0*P1))))/(s1**2 - s0**2)

img = cv2.imread("data/steel_rust.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# ví dụ: các giá trị ước lượng (thay bằng thống kê mẫu thực tế)
mu0,s0,mu1,s1,P0,P1 = 120, 12, 165, 15, 0.7, 0.3
T = calc_threshold(mu0,s0,mu1,s1,P0,P1)
_, binimg = cv2.threshold(gray, T, 255, cv2.THRESH_BINARY)
print("Bayes-ML T =", round(T,2))
plt.subplot(1,2,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,2,2); plt.imshow(binimg, cmap='gray'); plt.title("Bayes-ML mask");
plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

5) Dò biên + Liên kết biên + Hough (biên/đường thẳng)

Đề bài: Xác định vạch kẻ đường/lằn cắt trên tấm vật liệu công nghiệp.

Mục tiêu: Dùng Canny → HoughLinesP để trích rìa và đường thẳng; tạo mặt nạ phân vùng.

Yêu cầu: Tối ưu tham số Canny, Hough.

Hướng dẫn tóm tắt:

1) Làm mượt nhẹ
2) Canny
3) HoughLinesP
4) Vẽ/ghép mask khu vực quan tâm

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
img = cv2.imread("data/lanes.jpg")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray, 80, 160)
lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=80, minLineLength=60,
maxLineGap=10)
out = img.copy()
if lines is not None:
    for x1,y1,x2,y2 in lines[:,0]:
        cv2.line(out, (x1,y1), (x2,y2), (0,255,0), 2)
plt.subplot(1,3,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,3,2); plt.imshow(edges, cmap='gray'); plt.title("Canny")
plt.subplot(1,3,3); plt.imshow(cv2.cvtColor(out, cv2.COLOR_BGR2RGB));
plt.title("Hough lines"); plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

6) Region Growing (lan tỏa vùng) từ hạt giống

Đề bài: Tách tổn thương trên ảnh siêu âm/CT vùng có mức xám tương đồng quanh "hạt giống" do bác sĩ chỉ định.

Mục tiêu: Cài đặt region growing 8-lân cận với ngưỡng sai khác cục bộ $|I(p)-I(seed)| < \tau$.

Yêu cầu: Hỗ trợ nhiều seed; dừng khi không thể lan.

Hướng dẫn tóm tắt:

1) Chọn seed(s)
2) BFS/stack 8-neigh
3) Kiểm tra điều kiện tương đồng
4) Đánh dấu vùng.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
from collections import deque

def region_growing(gray, seeds, tau=5):
    H,W = gray.shape; visited = np.zeros_like(gray, np.uint8); out = np.zeros_like(gray, np.uint8)
    dirs = [(-1,-1),(-1,0),(-1,1),(0,-1),(0,1),(1,-1),(1,0),(1,1)]
    q = deque()
    for sy,sx in seeds:
        q.append((sy,sx)); visited[sy,sx]=1; out[sy,sx]=255
    while q:
        y,x = q.popleft()
        for dy,dx in dirs:
            ny,nx = y+dy, x+dx
            if 0<=ny<H and 0<=nx<W and not visited[ny,nx]:
                if abs(int(gray[ny,nx]) - int(gray[y,x])) <= tau:
                    visited[ny,nx]=1; out[ny,nx]=255; q.append((ny,nx))
    return out

gray = cv2.cvtColor(cv2.imread("data/ultrasound.png"), cv2.COLOR_BGR2GRAY)
mask = region_growing(gray, seeds=[(120,180)], tau=6)
plt.subplot(1,2,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,2,2); plt.imshow(mask, cmap='gray'); plt.title("Region growing"); plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

7) Split–Merge (tách & hợp vùng tứ phân)

Đề bài: Phân đoạn nền trời/biển/đất trong ảnh phong cảnh nhiễu nhẹ.

Mục tiêu: Cài đặt split–merge theo cây tứ phân với tiêu chuẩn đồng nhất σ khu vực.

Yêu cầu: Tham số: ngưỡng độ lệch chuẩn σ_max và diện tối thiểu.

Hướng dẫn tóm tắt:

1) Split theo σ>σ_max
2) Merge các vùng kề nếu union thỏa tiêu chuẩn
3) Gán nhãn.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
from skimage.segmentation import felzenszwalb
# Để gọn, dùng felzenszwalb như 1 baseline region-based; hoặc tự cài split-merge
nếu muốn.
img = cv2.cvtColor(cv2.imread("data/landscape.jpg"), cv2.COLOR_BGR2RGB)
seg = felzenszwalb(img, scale=100, sigma=0.8, min_size=150)
plt.subplot(1,2,1); plt.imshow(img); plt.title("Input")
plt.subplot(1,2,2); plt.imshow(seg, cmap='tab20'); plt.title("Region-based
seg"); plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

8) K-means (gom cụm theo màu/đặc trưng)

Đề bài: Tách vùng vườn cây, sông, nhà trong ảnh vệ tinh nhỏ nhờ màu/HSV.

Mục tiêu: Gom cụm K=3–5 trên không gian màu (RGB/HSV) để phân mảnh đất.

Yêu cầu: Chuẩn hóa dữ liệu, reshape N×3; hiển thị kết quả theo label.

Hướng dẫn tóm tắt:

1) Đổi không gian màu (tùy chọn)
2) Reshape
3) cv2.kmeans
4) Gộp lại mask.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
img = cv2.cvtColor(cv2.imread("data/satellite.jpg"), cv2.COLOR_BGR2RGB)
Z = img.reshape((-1,3)).astype(np.float32)
K=4; criteria=(cv2.TERM_CRITERIA_EPS+cv2.TERM_CRITERIA_MAX_ITER, 20, 1.0)
_,labels,centers = cv2.kmeans(Z, K, None, criteria, 5, cv2.KMEANS_PP_CENTERS)
seg = centers[labels.flatten()].reshape(img.shape).astype(np.uint8)
plt.subplot(1,2,1); plt.imshow(img); plt.title("Input")
plt.subplot(1,2,2); plt.imshow(seg); plt.title(f"K-means K={K}"); plt.show()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

9) Phân vùng chuyển động (video) – Frame differencing & MOG2

Đề bài: Đếm người/xe đi qua cổng bằng camera đứng yên.

Mục tiêu: So sánh phân đoạn chuyển động bằng sai khác khung và cv2.createBackgroundSubtractorMOG2.

Yêu cầu: Làm mượt, ngưỡng, mở/đóng hình thái học; vẽ bounding boxes.

Hướng dẫn tóm tắt:

1) VideoCapture
2) diff | MOG2
3) threshold + morphology
4) findContours
5) Vẽ BB.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
cap = cv2.VideoCapture("data/gate.mp4")
bg = cv2.createBackgroundSubtractorMOG2(history=300, varThreshold=25, detectShadows=True)
prev = None
while True:
    ret, frame = cap.read()
    if not ret: break
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    gray = cv2.GaussianBlur(gray,(5,5),0)

    # MOG2
    fg = bg.apply(frame)
    _, fgth = cv2.threshold(fg, 200, 255, cv2.THRESH_BINARY)
    fgth = cv2.morphologyEx(fgth, cv2.MORPH_OPEN, np.ones((3,3),np.uint8), iterations=2)

    # diff
    if prev is None: prev = gray; continue
    diff = cv2.absdiff(gray, prev); prev = gray
    _, diffth = cv2.threshold(diff, 25, 255, cv2.THRESH_BINARY)

    # hiển thị
    cv2.imshow("MOG2", fgth); cv2.imshow("FrameDiff", diffth)
    if cv2.waitKey(1)==27: break
cap.release(); cv2.destroyAllWindows()
```

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

10) Watershed (theo miền dựa trên địa hình) cho vật thể chạm nhau

Đề bài: Đếm đồng xu/hạt bi dính nhau trên nền tương đối đồng đều.

Mục tiêu: Dùng watershed trên ảnh khoảng cách đã tách “marker” để tách các vật thể chạm.

Yêu cầu: Pipeline: lọc → nhị phân → distanceTransform → tìm peak → watershed.

Hướng dẫn tóm tắt:

1) Tiền xử lý
2) Otsu
3) Khoảng cách & peak
4) Marker
5) Watershed.

THỰC HÀNH VỀ PHÂN VÙNG ẢNH

```
from scipy import ndimage as ndi
from skimage.feature import peak_local_max
from skimage.segmentation import watershed

img = cv2.imread("data/coins.png")
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
_, bw = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
bw = cv2.morphologyEx(bw, cv2.MORPH_OPEN, np.ones((3,3),np.uint8), iterations=1)

dist = cv2.distanceTransform(bw, cv2.DIST_L2, 5)
coords = peak_local_max(dist, footprint=np.ones((3,3)), labels=bw)
mask = np.zeros(dist.shape, dtype=bool); mask[tuple(coords.T)] = True
markers, _ = ndi.label(mask)
labels = watershed(-dist, markers, mask=bw.astype(bool))

plt.figure(figsize=(9,3))
plt.subplot(1,3,1); plt.imshow(gray, cmap='gray'); plt.title("Gray")
plt.subplot(1,3,2); plt.imshow(bw, cmap='gray'); plt.title("Otsu")
plt.subplot(1,3,3); plt.imshow(labels, cmap='nipy_spectral'); plt.title("Watershed"); plt.show()
```

THỰC HÀNH VỀ CẢI THIỆN VÀ KHÔI PHỤC ẢNH

Đây là nội dung bài học biến đổi fourier cho ảnh số, bạn hãy dựa vào nội dung bài học này xây dựng cho tôi 5 bài thực hành, mỗi bài đều có phần đề bài, mục tiêu, yêu cầu và hướng dẫn. Chú ý sử dụng ngôn ngữ lập trình python và các đề bài mang tính thực tiễn cao nhé