THỰC HÀNH VỀ TÁCH BIÊN

1) So khớp bộ dò biên trên ảnh thật & ảnh có nhiễu

- **Đề bài.** Cho 1 ảnh đời thực (đường phố, tòa nhà, mặt người, sản phẩm…), hãy so sánh kết quả tách biên giữa các toán tử: **Roberts, Prewitt, Sobel, Scharr** (kèm tuỳ chọn làm trơn Gaussian trước khi lấy gradient).

Mục tiêu.

- Hiểu vai trò làm trơn trước khi lấy đạo hàm; so sánh độ nhạy nhiễu và "độ dày" biên giữa các toán tử.
- Thực hành chuẩn hoá biên độ gradient và đặt **ngưỡng** theo phần trăm cực đại.

Hướng dẫn.

- Đọc ảnh màu → xám → float32.
- (Tuỳ chọn) Làm trơn Gaussian với sigma ∈ {0, 1, 2}.
- Tính gradient với 4 toán tử, chuẩn hoá magnitude về 0..255, nhị phân hoá theo ngưỡng tỉ lệ (ví dụ 0.2–0.4).
- Lặp lại trên ảnh đã thêm nhiễu; so sánh trực quan và bằng chỉ số I/OU biên nhị phân (tùy chọn).

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np
from scipy.ndimage import convolve, gaussian_filter
import matplotlib.pyplot as plt

def grad2d(f, scheme='sobel', sigma=None, bc='reflect'):
    g = f.copy()
    if sigma is not None:
        g = gaussian_filter(g, sigma=sigma, mode=bc)
    if scheme == 'roberts':
        gx = g[:-1,:-1] - g[1:,1:]; gy = g[1:,:-1] - g[:-1,1:]
        mag = np.zeros_like(g); mag[:-1,:-1] = np.hypot(gx, gy); return mag
    if scheme == 'prewitt':
        kx = np.array([[-1,0,1],[-1,0,1],[-1,0,1]],np.float32)
        ky = np.array([[ 1,1,1],[ 0,0,0],[-1,-1,-1]],np.float32)
    elif scheme == 'sobel':
        kx = np.array([[-1,0,1],[-2,0,2],[-1,0,1]],np.float32)
        ky = np.array([[ 1,2,1],[ 0,0,0],[-1,-2,-1]],np.float32)
    elif scheme == 'scharr':
        kx = np.array([[3,0,-3],[10,0,-10],[3,0,-3]],np.float32)/32.0
        ky = kx.T
    Gx = convolve(g, kx, mode=bc); Gy = convolve(g, ky, mode=bc)
    return np.hypot(Gx, Gy)

def binarize(mag, thr_ratio=0.25):
    mmax = mag.max() if mag.size else 0
    return (mag >= thr_ratio*mmax).astype(np.uint8)*255

img = cv2.imread('input.jpg', cv2.IMREAD_GRAYSCALE).astype(np.float32)
noisy = (img + np.random.normal(0, 10, img.shape)).clip(0,255).astype(np.float32)

for sigma in [None, 1.0]:
    for scheme in ['roberts','prewitt','sobel','scharr']:
        mag = grad2d(img, scheme=scheme, sigma=sigma)
        mask = binarize(mag, 0.25)
        cv2.imwrite(f'edges_{scheme}_sigma{sigma}.png', mask)

        magN = grad2d(noisy, scheme=scheme, sigma=sigma)
        maskN = binarize(magN, 0.25)
        cv2.imwrite(f'edges_{scheme}_sigma{sigma}_noisy.png', maskN)
```

THỰC HÀNH VỀ TÁCH BIÊN

2) Tách biên & tự động "scan phẳng" tài liệu

- **Đề bài.** Chụp một tờ giấy A4 bằng điện thoại (góc nghiêng). Hãy tách biên tờ giấy, tìm tứ giác lớn nhất và **biến đổi phối cảnh** để "scan phẳng" tài liệu.

Mục tiêu.

- Dùng **Canny**/Sobel để nhấn mạnh cạnh.
- Tìm contour lớn nhất, xấp xỉ đa giác, sắp xếp 4 đỉnh, cv2.getPerspectiveTransform.

Yêu cầu.

- Ảnh đầu vào chứa biên giấy trên nền tương phản.

Hướng dẫn.

- Làm mờ Gaussian nhẹ ($\sigma \approx 1.2$), tính Sobel hoặc dùng Canny (thử ngưỡng auto bằng Otsu).
- Sử dụng các toan tử hình thái mở rộng/đóng (morphology) để liền mảnh cạnh.
- Tìm contour lớn nhất có 4 đỉnh; sắp xếp thứ tự (TL, TR, BR, BL).
- Warp về kích thước A4 (tỉ lệ ~ $\sqrt{2}$, ví dụ 1240×1754 px).
- So sánh trước/sau bằng PSNR/SSIM (tuỳ chọn).

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np

def order_pts(pts):
    s = pts.sum(axis=1); diff = np.diff(pts, axis=1)
    tl = pts[np.argmin(s)]; br = pts[np.argmax(s)]
    tr = pts[np.argmin(diff)]; bl = pts[np.argmax(diff)]
    return np.array([tl,tr,br,bl], dtype=np.float32)

img = cv2.imread('doc.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray, (5,5), 1.2)
edges = cv2.Canny(blur, 0, 0)  # dùng auto Otsu
# Auto ngưỡng bằng Otsu
_,th = cv2.threshold(blur, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
edges = cv2.Canny(blur, th*0.5, th*1.5)

cnts,_ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnt = max(cnts, key=cv2.contourArea)
peri = cv2.arcLength(cnt, True)
approx = cv2.approxPolyDP(cnt, 0.02*peri, True)
pts = approx.reshape(-1,2).astype(np.float32)
if len(pts) != 4: raise ValueError("Không tìm được tứ giác tài liệu")
src = order_pts(pts)
w,h = 1240,1754
dst = np.array([[0,0],[w-1,0],[w-1,h-1],[0,h-1]], np.float32)
M = cv2.getPerspectiveTransform(src, dst)
scan = cv2.warpPerspective(img, M, (w,h))
cv2.imwrite('doc_scanned.jpg', scan)
```

THỰC HÀNH VỀ TÁCH BIÊN

3) Dò làn đường cơ bản cho ảnh camera hành trình

- **Đề bài.** Với ảnh (hoặc frame video) từ camera trước xe trên đường cao tốc, hãy phát hiện **hai biên làn đường** cơ bản.

Mục tiêu.

- Tiền xử lý ROI (vùng nón phía trước).
- Dùng **Sobel hướng x/y** để khuếch đại biên dọc, rồi **Hough Lines** để tìm đường thẳng.

Yêu cầu.

- Ảnh điều kiện ngày, trời nắng; có thể lấy frame từ video dashcam mẫu.

Hướng dẫn.

- Cắt ROI hình thang (mask) ở nửa dưới ảnh.
- Làm mờ, Sobel theo x (hoặc magnitude), ngưỡng để được edges.
- Dùng cv2.HoughLinesP tìm đoạn thẳng; tách trái/phải theo hệ số góc.
- Vẽ đường trung bình/ngoại suy.

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np

img = cv2.imread('road.jpg')
h,w,_ = img.shape
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
roi = np.array([[(w*0.1,h), (w*0.45,h*0.6), (w*0.55,h*0.6), (w*0.9,h)]], np.int32)
mask = np.zeros_like(gray); cv2.fillPoly(mask, roi, 255)
gray = cv2.bitwise_and(gray, mask)

blur = cv2.GaussianBlur(gray,(5,5),1.2)
gx = cv2.Sobel(blur, cv2.CV_32F, 1, 0, ksize=3)
mag = np.abs(gx); thr = 0.3*mag.max()
edges = (mag >= thr).astype(np.uint8)*255

lines = cv2.HoughLinesP(edges, 1, np.pi/180, threshold=60, minLineLength=50, maxLineGap=150)
left, right = [], []
if lines is not None:
    for x1,y1,x2,y2 in lines[:,0]:
        if x2==x1: continue
        slope = (y2-y1)/(x2-x1+1e-6)
        if slope < -0.5: left.append((x1,y1,x2,y2))
        elif slope > 0.5: right.append((x1,y1,x2,y2))
for segs,color in [(left,(0,255,0)),(right,(0,0,255))]:
    for x1,y1,x2,y2 in segs:
        cv2.line(img,(x1,y1),(x2,y2),color,3)
cv2.imwrite('lanes_overlay.jpg', img)
```

THỰC HÀNH VỀ TÁCH BIÊN

4) Kiểm tra lỗi bề mặt (QC) sản phẩm kim loại/nhựa bằng Laplace + hình thái học

- **Đề bài.** Với ảnh cận cảnh bề mặt sản phẩm (tấm kim loại sơn tĩnh điện, vỏ nhựa), phát hiện **xước/lỗi bề mặt** dạng rãnh mảnh.

Mục tiêu.

- Vận dụng **toán tử Laplace (đạo hàm bậc 2)** để nhấn mạnh điểm đổi cong/biên mảnh.
- Chuỗi xử lý: làm trơn → Laplace → |response| → ngưỡng thích nghi → đóng/mở để loại nhiễu.

Yêu cầu.

- Ảnh macro ánh sáng đều; có vài vệt xước mảnh.

Hướng dẫn.

- GaussianBlur(σ≈1–2).
- Lọc Laplacian ksize=3 (hoặc chập với mặt nạ 3×3/“đồng hồ cát” như trong slide).
- Lấy trị tuyệt đối, chuẩn hoá, ngưỡng (Otsu/percentile).
- Dùng morphologyEx (open→remove hạt, close→nối nét).
- (Tùy chọn) Loại bỏ vùng quá nhỏ bằng cv2.connectedComponentsWithStats.

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np

img = cv2.imread('surface.jpg', cv2.IMREAD_GRAYSCALE)
blur = cv2.GaussianBlur(img,(5,5),1.5)
lap = cv2.Laplacian(blur, cv2.CV_32F, ksize=3)
resp = np.abs(lap)
resp = (resp/resp.max()*255).astype(np.uint8)
_, mask = cv2.threshold(resp, 0, 255, cv2.THRESH_BINARY+cv2.THRESH_OTSU)
mask = cv2.morphologyEx(mask, cv2.MORPH_OPEN, np.ones((3,3),np.uint8), iterations=1)
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, np.ones((5,5),np.uint8), iterations=1)

n, lbl, stats, _ = cv2.connectedComponentsWithStats(mask, 8)
min_area = 30
keep = np.zeros_like(mask)
for i in range(1,n):
    if stats[i,cv2.CC_STAT_AREA] >= min_area:
        keep[lbl==i] = 255

cv2.imwrite('defect_mask.png', keep)
cv2.imwrite('defect_overlay.png', cv2.cvtColor(img, cv2.COLOR_GRAY2BGR))
```

THỰC HÀNH VỀ TÁCH BIÊN

5) Đếm vật tròn (đồng xu/bi-bóng bi) bằng biên Canny + HoughCircles

- **Đề bài.** Chụp ảnh bàn có nhiều đồng xu/bi tròn. Hãy phát hiện và **đếm số lượng** đối tượng tròn, kèm bounding/viền.

Mục tiêu.

- Tiền xử lý ánh sáng, cân bằng histogram.
- Dùng **Canny** (hoặc Sobel → Canny) và **Hough hình tròn**.
- Thực tế: lọc theo bán kính, theo độ tròn (cấp phối hợp lệ).

Yêu cầu.

- Ảnh từ trên xuống; ít chồng lấn mạnh.

Hướng dẫn.

- CLAHE hoặc cv2.equalizeHist để cân bằng sáng.
- GaussianBlur, Canny(ngưỡng đôi), sau đó cv2.HoughCircles.
- Vẽ vòng tròn, in tổng số.

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np

img = cv2.imread('coins.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8)).apply(gray)
blur = cv2.GaussianBlur(clahe, (7,7), 1.2)
edges = cv2.Canny(blur, 80, 160)
circles = cv2.HoughCircles(blur, cv2.HOUGH_GRADIENT, dp=1.2, minDist=30,
                           param1=120, param2=25, minRadius=10, maxRadius=80)
out = img.copy()
count = 0
if circles is not None:
    circles = np.round(circles[0]).astype(int)
    for (x,y,r) in circles:
        cv2.circle(out, (x,y), r, (0,255,0), 2)
        cv2.circle(out, (x,y), 2, (0,0,255), 3)
    count = len(circles)
cv2.imwrite('coins_detected.jpg', out)
print("Số lượng vật tròn:", count)
```

THỰC HÀNH VỀ TÁCH BIÊN

6) Cắt nền (auto-crop) ảnh sản phẩm bằng biên + contour

- **Đề bài.** Cho ảnh sản phẩm chụp trên nền đơn giản (trắng/xám nhạt). Hãy phát hiện biên vật thể chính và **cắt gọn** (tight crop) + xuất PNG nền trong suốt.

Mục tiêu.

- Dùng Canny/Sobel để lấy biên, đóng/ mở để liền khối.
- Chọn contour lớn nhất, tạo mask, **alpha-matte** PNG.

Yêu cầu.

- Ảnh product.jpg. Không có vật thể thứ hai lớn

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np

img  = cv2.imread('product.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(5,5),1.0)
edges = cv2.Canny(blur, 50, 150)

kernel = np.ones((5,5),np.uint8)
mask = cv2.morphologyEx(edges, cv2.MORPH_CLOSE, kernel, iterations=2)
mask = cv2.dilate(mask, kernel, 2)

cnts,_ = cv2.findContours(mask, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnt = max(cnts, key=cv2.contourArea)
mask_full = np.zeros(gray.shape, np.uint8)
cv2.drawContours(mask_full, [cnt], -1, 255, thickness=-1)

# Tight crop
x,y,w,h = cv2.boundingRect(cnt)
crop_img  = img[y:y+h, x:x+w]
crop_mask = mask_full[y:y+h, x:x+w]

# Xuất PNG với alpha
bgra = cv2.cvtColor(crop_img, cv2.COLOR_BGR2BGRA)
bgra[:,:,3] = crop_mask
cv2.imwrite('product_cropped.png', bgra)
```

THỰC HÀNH VỀ TÁCH BIÊN

7) Phát hiện vết nứt bê tông bằng LoG đa tỉ lệ + skeleton

- **Đề bài.** Ảnh bề mặt bê tông/asphalt có vết nứt mảnh. Hãy phát hiện và **làm mảnh** (skeleton) mạng nứt.

Mục tiêu.

- Kết hợp **Laplacian of Gaussian** nhiều σ để bắt vết nứt nhiều bề rộng.
- Ngưỡng thích nghi; **skeletonize** để còn 1-pixel.

Yêu cầu. surface_crack.jpg.

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np
from skimage.morphology import skeletonize

img  = cv2.imread('surface_crack.jpg', cv2.IMREAD_GRAYSCALE)
imgf = img.astype(np.float32)/255.0

def LoG(f, sigma):
    k = int(6*sigma+1)|1
    g  = cv2.GaussianBlur(f,(k,k),sigma)
    lap = cv2.Laplacian(g, cv2.CV_32F, ksize=3)
    return np.abs(lap)

resp = sum(LoG(imgf,s) for s in [0.8,1.2,1.8,2.4])
resp = (resp/resp.max()*255).astype(np.uint8)

thr = cv2.adaptiveThreshold(resp, 255, cv2.ADAPTIVE_THRESH_MEAN_C,
                            cv2.THRESH_BINARY, 35, -5)
thr = cv2.morphologyEx(thr, cv2.MORPH_OPEN, np.ones((3,3),np.uint8), 1)

skel = skeletonize((thr>0).astype(np.uint8)).astype(np.uint8)*255
cv2.imwrite('crack_mask.png', thr)
cv2.imwrite('crack_skeleton.png', skel)
```

THỰC HÀNH VỀ TÁCH BIÊN

8) Tách biên lá cây và đo chu vi/diện tích + "độ răng cưa"

- **Đề bài.** Ảnh một chiếc lá trên nền tương phản. Tách **đường biên** lá, tính **diện tích, chu vi**, và chỉ số "răng cưa" (perimeter²/(4π·area)).

Mục tiêu.

- Edge → contour → polygon approx.
- Ứng dụng hình học ảnh.

Yêu cầu. leaf.jpg. Biết DPI hoặc thước chuẩn (tùy chọn đổi sang mm).

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np
import math

img  = cv2.imread('leaf.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(7,7),1.5)
edges = cv2.Canny(blur, 60, 150)

cnts,_ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnt = max(cnts, key=cv2.contourArea)

area = cv2.contourArea(cnt)
peri = cv2.arcLength(cnt, True)
serration = (peri**2)/(4*math.pi*area + 1e-6)

out = img.copy()
cv2.drawContours(out,[cnt],-1,(0,255,0),2)
cv2.putText(out, f"Area={area:.1f} px^2, Peri={peri:.1f}, S={serration:.3f}",
            (20,30), cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0,0,255), 2)
cv2.imwrite('leaf_metrics.jpg', out)
```

THỰC HÀNH VỀ TÁCH BIÊN

9) Đo kích thước vật thể bằng chuẩn kích thước (coin/A4) + biên

- **Đề bài.** Trong ảnh có **một đồng xu/A4** làm chuẩn. Hãy đo **chiều dài/chiều rộng** của một vật thể (ví dụ con ốc), sử dụng biên + contour và **quy đổi mm**.

Mục tiêu.

- Tách biên → contour lớn nhất của **vật thể** cần đo.
- Tìm **đường kính** đồng xu (HoughCircles) hoặc cạnh A4 (từ bài 2) → suy ra **tỉ lệ px/mm**.

Yêu cầu.

opencv-python, numpy. Ảnh đặt camera gần vuông góc.

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np, math

img  = cv2.imread('measure.jpg')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
blur = cv2.GaussianBlur(gray,(7,7),1.2)

# 1) Tìm đồng xu làm chuẩn
cir = cv2.HoughCircles(blur, cv2.HOUGH_GRADIENT, dp=1.2, minDist=50,
                       param1=120, param2=25, minRadius=20, maxRadius=120)
if cir is None: raise ValueError("Không thấy đồng xu chuẩn")
x,y,r = np.round(cir[0,0]).astype(int)
coin_d_mm = 23.6  # ví dụ đồng 1 Euro; thay bằng chuẩn của bạn
px_per_mm = (2*r) / coin_d_mm

# 2) Biên & contour vật thể
edges = cv2.Canny(blur, 80, 160)
cnts,_ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
cnt = max(cnts, key=cv2.contourArea)
rect = cv2.minAreaRect(cnt)            # (cx,cy),(w,h),angle
(w,h) = rect[1]
w_mm, h_mm = w/px_per_mm, h/px_per_mm

box = np.int0(cv2.boxPoints(rect))
out = img.copy()
cv2.drawContours(out, [box], 0, (0,255,0), 2)
cv2.putText(out, f"{w_mm:.2f} x {h_mm:.2f} mm", (box[0][0], box[0][1]-10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,0,0), 2)
cv2.imwrite('measure_out.jpg', out)
```

THỰC HÀNH VỀ TÁCH BIÊN

10) Tự động "deskew" hoá đơn/biên bản bằng góc biên ưu thế

- **Đề bài.** Nhận ảnh scan hoá đơn bị **nghiêng**. Hãy ước lượng **góc nghiêng** dựa trên các **đường biên ưu thế** (dòng chữ/biên giấy) rồi xoay ảnh về đúng phương.

Mục tiêu.

- Phát hiện biên (Sobel/Canny), dùng **Hough Lines** để lấy góc chính.
- Tính **median angle** của các line, xoay bù.

Yêu cầu.

opencv-python, numpy. Ảnh: receipt.jpg.

THỰC HÀNH VỀ TÁCH BIÊN

```
import cv2, numpy as np, math

img  = cv2.imread('receipt.jpg')
g    = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
g    = cv2.GaussianBlur(g,(5,5),1.2)
ed   = cv2.Canny(g, 60, 180)

lines = cv2.HoughLines(ed, 1, np.pi/180, 150)
angles = []
if lines is not None:
    for rho,theta in lines[:,0]:
        ang = (theta*180/np.pi) - 90  # gần trục ngang
        if -45 < ang < 45:
            angles.append(ang)
skew = np.median(angles) if angles else 0.0

# xoay bù
(h,w) = img.shape[:2]
M = cv2.getRotationMatrix2D((w/2,h/2), skew, 1.0)
deskew = cv2.warpAffine(img, M, (w,h), flags=cv2.INTER_LINEAR,
                        borderMode=cv2.BORDER_REPLICATE)
cv2.imwrite('receipt_deskew.jpg', deskew)
```