THỰC HÀNH MÔN THỊ GIÁC MÁY TÍNH
BÀI TẬP THỰC HÀNH SỐ 5
Lab 5. PHÂN ĐOẠN, PHÁT HIỆN CHUYỂN ĐỘNG

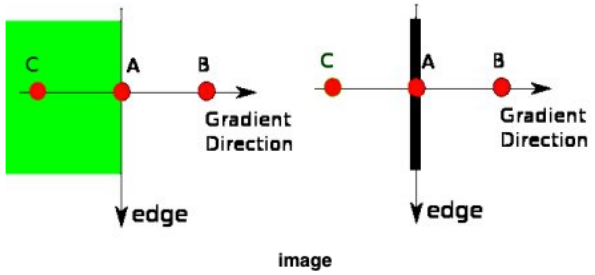
A. Thuật toán Canny dò biên cạnh (Canny for edge detection) gồm 4 bước:

1. *Noise reduction*
   Khử nhiễu với Gaussian filter (5 x 5).
2. *Finding intensity Gradient of image*
   Áp dụng Sobel kernel cho chiều dọc (vertical) G_y và ngang (horizontal) G_x.

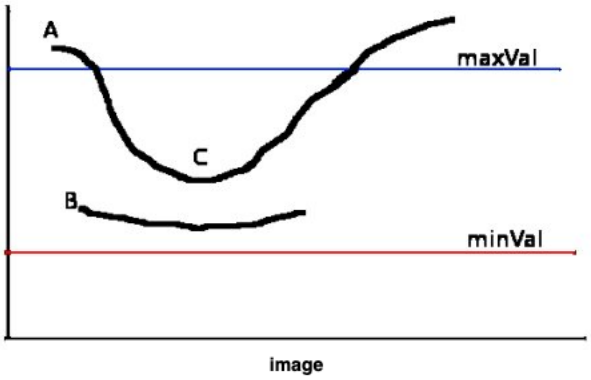
$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

3. *Non-maximum suppresion*



4. *Hysteresis thresholding*
   maxVal: cạnh nào >= maxVal: the classified edges
   minVal: cạnh nào < minVal: the non-edges.



Canny edge detection in OpenCV
cv2.Canny(image, minVal, maxVal, [theSizeOfSobelKernel = 3])

B. Sobel cho dò biên cạnh

```
# Blur the image for better edge detection
img_blur = cv2.GaussianBlur(img, ksize=(5, 5), sigmaX=1, sigmaY=1)
# Sobel Edge Detection
sobelx = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=0, ksize=3) # Sobel Edge Detection on the X axis
sobely = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=0, dy=1, ksize=3) # Sobel Edge Detection on the Y axis
sobelxy = cv2.Sobel(src=img_blur, ddepth=cv2.CV_64F, dx=1, dy=1, ksize=3) # Combined X and Y Sobel Edge Detection
```

C. Phát hiện cạnh bằng RGB

```
blue, green, red = cv2.split(image)
# detect contours using blue channel and without thresholding
contours1, hierarchy1 = cv2.findContours(image=blue, mode=cv2.RETR_TREE, method=cv2.CHAIN_APPROX_NONE)
# draw contours on the original image
image_contour_blue = image.copy()
cv2.drawContours(image=image_contour_blue, contours=contours1, contourIdx=-1, color=(0, 255, 0), thickness=2, lineType=cv2.LINE_AA)

# detect contours using green channel and without thresholding
contours2, hierarchy2 = cv2.findContours(image=green, mode=cv2.RETR_TREE, method=cv2.CHAIN_APPROX_NONE)
# draw contours on the original image
image_contour_green = image.copy()
cv2.drawContours(image=image_contour_green, contours=contours2, contourIdx=-1, color=(0, 255, 0), thickness=2, lineType=cv2.LINE_AA)

# detect contours using red channel and without thresholding
contours3, hierarchy3 = cv2.findContours(image=red, mode=cv2.RETR_TREE, method=cv2.CHAIN_APPROX_NONE)
# draw contours on the original image
image_contour_red = image.copy()
cv2.drawContours(image=image_contour_red, contours=contours3, contourIdx=-1, color=(0, 255, 0), thickness=2, lineType=cv2.LINE_AA)
```

BÀI TẬP BIÊN CẠNH

1. Tải hình ảnh bằng OpenCV.
2. Tiến hành tìm cạnh bằng Canny và Sobel.
3. Dùng phương pháp tính nonzero (NNZ) để đếm số lượng điểm và so sánh kết quả giữa Canny với Sobel.
4. Dùng phương pháp RGB để tìm biên cạnh.
5. Sử dụng hình thái học để làm mịn ảnh, sau đó dùng Canny để tìm biên cạnh. So sánh kết quả NNZ với việc chỉ sử dụng Canny.
6. Xây dựng lại thuật giải Canny.

D. PHÁT HIỆN ĐỐI TƯỢNG

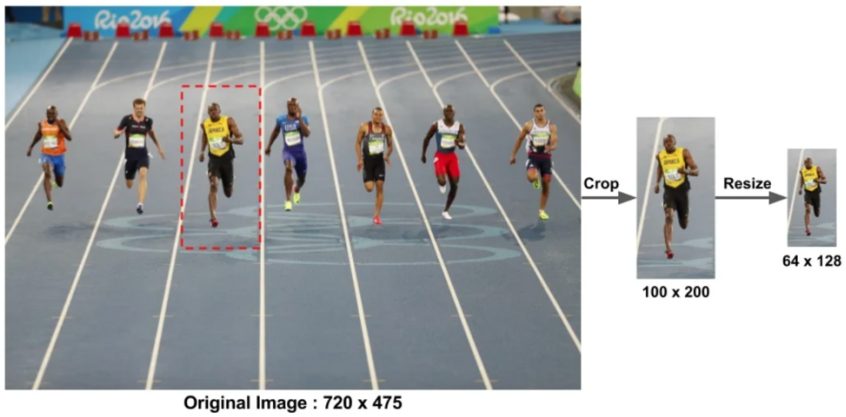
1. Histogram of Oriented Gradients (HoG)

- HoG được sử dụng để trích xuất các tính năng từ dữ liệu hình ảnh.
- Bộ mô tả HOG tập trung vào cấu trúc hoặc hình dạng của một đối tượng.

- HOG sẽ tạo một Biểu đồ cho từng vùng này riêng biệt. Biểu đồ được tạo bằng cách sử dụng độ dốc và hướng của các giá trị pixel có tên là 'Biểu đồ của Gradients định hướng'

Các bước:

- **Xử lý hình ảnh (resized/crop/…)**



- **Tính gradients**

121	10	78	96	125
48	152	68	125	111
145	78	85	89	65
154	214	56	200	66
214	87	45	102	45

- Change in X direction(G_x) = 89 – 78 = 11
- Change in Y direction(G_y) = 68 – 56 = 8

- **Tính magnitude và direction**

Magnitude:

$$\text{Total Gradient Magnitude} = \sqrt{[(G_x)^2+(G_y)^2]}$$

$$\text{Total Gradient Magnitude} = \sqrt{[(11)^2+(8)^2]} = 13.6$$

Direction:

$$\Phi = \text{atan}(Gy / Gx)$$

OpenCV: https://learnopencv.com/histogram-of-oriented-gradients/

2. **Binary Robust Independent Elementary Features (BRIEF)**

Dựa trên nguyên tắc của:

$$Where\ \tau(p;x,y)\ is\ defined\ as:$$
$$\tau(p;x,y) = \begin{cases} 1 & : p(x) < p(y) \\ 0 & : p(x) \geq p(y) \end{cases}$$
$$p(x)\ is\ the\ \text{int}ensity\ value\ at\ pixel\ x.$$

OpenCV: https://docs.opencv.org/3.4/dc/d7d/tutorial_py_brief.html

3. Harris Corner Detector

Dùng hộp trượt để đánh dấu phát hiện góc.

https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html

https://www.geeksforgeeks.org/feature-detection-and-matching-with-opencv-python/

Harris Corner Detector in OpenCV

OpenCV has the function **cv.cornerHarris()** for this purpose. Its arguments are:

- **img** - Input image. It should be grayscale and float32 type.
- **blockSize** - It is the size of neighbourhood considered for corner detection
- **ksize** - Aperture parameter of the Sobel derivative used.
- **k** - Harris detector free parameter in the equation.

BÀI TẬP

1. Tiến hành đọc 1 ảnh màu vào img
2. Lấy hình ảnh vừa đọc, chuyển xám lưu thành img2.
3. Dùng HoG chiết xuất các đỉnh góc của các đối tượng trong img.
4. Dùng BRIEF chiết xuất các đỉnh góc của các đối tượng trong img.
5. Dùng Harris chiết xuất các đỉnh góc của các đối tượng trong img.
6. Lặp lại phương pháp HoG, BRIEF, Harris cho img2 và so sánh kết quả.
7. Dùng HoG để chiết xuất đặc trưng tìm các góc trên từng hình ảnh đã wavelet.

E. PHÁT HIỆN CHUYỂN ĐỘNG

1. Tiến hành đọc vào 1 video và tách thành từng frame.
2. Thực hiện trừ nền ứng với từng frame ảnh với MOG2:

```
cv2.createBackgroundSubtractorMOG2()
result = mog.apply(gray)
```

3. Phát hiện chuyển động (motion detection) theo các bước:

   - Bước 1: tách từng frame
   - Bước 2: chuyển từng frame sang gray

- Bước 3: trừ nền bằng MOG2
- Bước 4: áp dụng Morphological Operations loại bỏ nhiễu và các khu vực trống
- Bước 5: vẽ đường bao quanh (loại bỏ các contour < 100 pixels)
- Bước 6: Lưu kết quả

4. Tiến hành thống kê số hành động đã phát hiện được trong video và phân loại chúng theo kích thước.

=== HẾT ===