Phần 1: Lý thuyết

1. Histogram là gì?

Là một dạng biểu đồ hình cột cho ta thấy được sự biến động và thay đổi của tập hợp các dữ liệu thống kê.

Trong lĩnh vực xử lí ảnh, histogram là biểu đồ tần xuất được dùng để thống kê số lần xuất hiện các mức sáng trong ảnh. Từ đó ta biết được ảnh bị thiếu sáng, thừa sáng, độ tương phản cao, thấp không.

2. Pixcel-wise, path-wise, image-wise transform là gì?

Pixel-wise transformations: These are operations that are performed on each individual pixel in an image. Examples include color conversion, brightness adjustment, contrast adjustment, etc.

Path-wise transformations: These are operations that are performed along a specific path within an image. For example, zooming in or out on a particular object in an image along a defined path.

Image-wise transformations: These are operations that are performed on the entire image. Examples include flipping an image, rotating an image, changing the size of an image, etc.

Câu tl của Minh Huy:

Thao tác trên pixel là những thao tác có sự tính toán và tác động chỉ trên một điểm ảnh. Vd:push – wipe – uncover effect, blending effect, animation, tăng giảm độ sáng.

Thao tác trên cụm pixel là những thao tác có sự tính toán trên một vùng ảnh và chỉ có tác động trong phạm vi vùng ảnh đó. Vd: correlation, convolution, pooling, morphology.

Thao tác trên toàn bộ ảnh là những thao tác có sự tính toán và tác động trên toàn bộ tấm ảnh. Vd: seam carving, chromakey, cân bằng histogram.

3. Nhập môn CV là gì?

Thị giác máy tính là một lĩnh vực trí tuệ nhân tạo (AI), cho phép máy tính và hệ thống lấy thông tin có ý nghĩa từ hình ảnh kỹ thuật số, video và các đầu vào trực quan khác, và thực hiện hành động hoặc đưa ra đề xuất dựa trên thông tin đó. Nếu AI cho phép máy tính suy nghĩ, thì thị giác máy tính cho phép chúng nhìn, quan sát và hiểu. (xác định vật thể, theo dõivật thể, đo đạc, phát hiện vật thể, phân loại vật thể).

Nhập môn thị giác máy tính là môn học thuộc lĩnh vực Thị giác máy tính. Nội dung môn nàyhọc chưa liên quan đến ngữ nghĩa của hình ảnh, chỉ học các thuật thoán ở cấp độ pixel, cụm pixel hay toàn bộ bức ảnh.

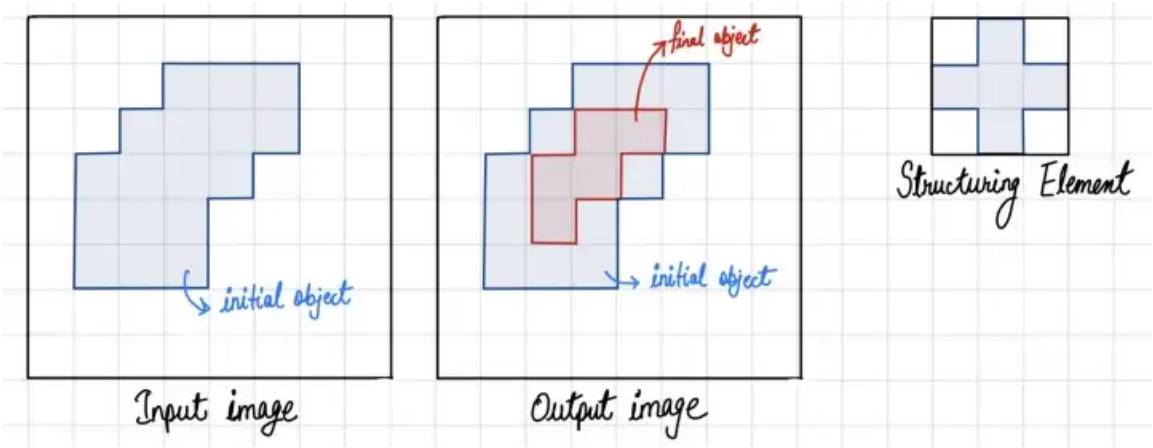
Trái với Nhập môn thị giác máy tính, Thị giác máy tính nâng cao là môn học liên quan đến ngữ nghĩa của hình ảnh. Nội dung môn này bao gồm lý thuyết nâng cao, các chủ đề nghiêncứu hiện tại trong thị giác máy tính với trọng tâm là các nhiệm vụ nhận dạng và học sâu, cácphương pháp tiếp cận thực tế để xây dựng các hệ thống Thị giác máy tính thực sự.

Phần 2: Thực hành

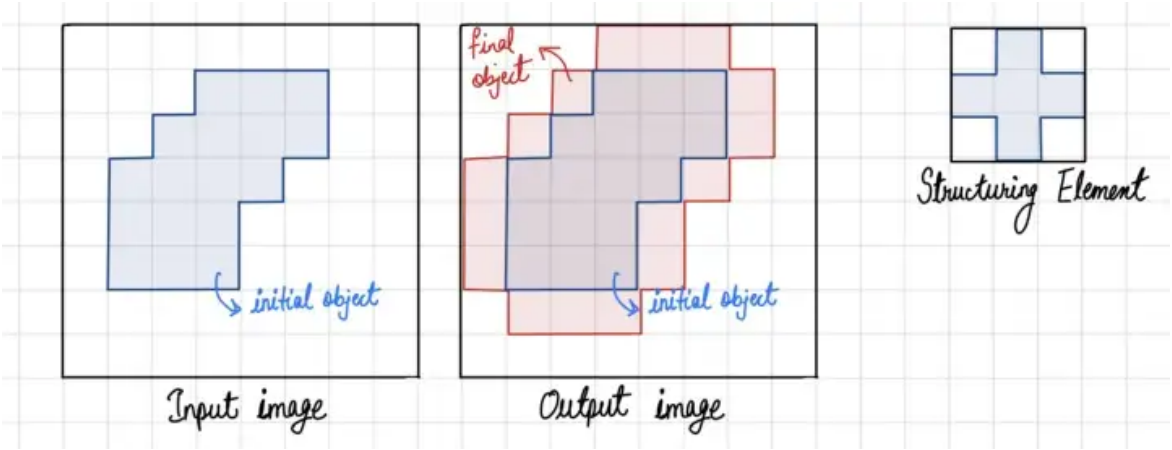
1. Bài toán đếm tế bào máu

Các phép biến đổi Morphological:

- Erosion: Xối mòn → làm cho các thành phần chính nhỏ đi. Cách thực hiện: nó sẽ cho kernal chạy từ đầu đến cuối ảnh, nếu tất cả giá trị 1 của kernal tương ứng với giá trị 1 của ảnh thì điểm gốc có giá trị là 1, nêu không có giá trị là 0.



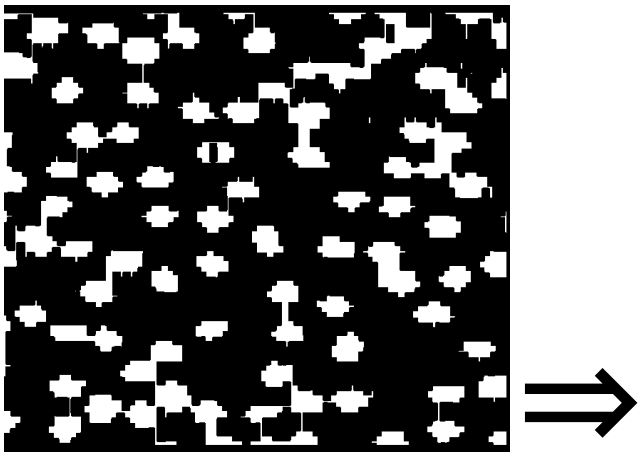
- Dilation: làm đầy → làm cho các thành phần chính to ra. Tương tự như erosion, mà đây là phép or.

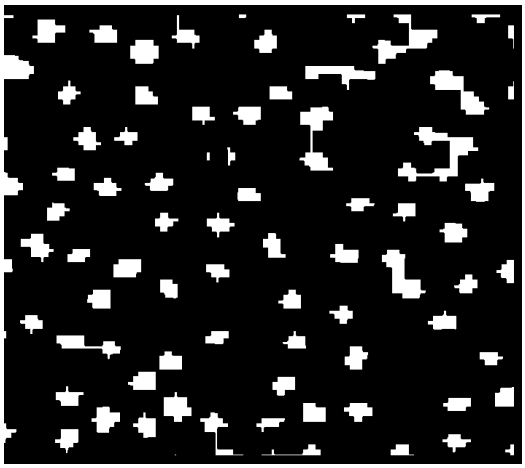


- Opening: (Co trước, nở sau) loại bỏ nhiễu → những nhiễu chấm tròn có thể bị loại bỏ. Loại bỏ các phần lồi lõm và làm cho đường bao các đối tượng trong ảnh trở nên mượt mà hơn.
- Closing: (nở trước, co sau) làm đầy những khoảng trống nhỏ bên trong vật thể. Làm trơn đường bao các đối tượng, lấp đầy các khoảng trống biên và loại bỏ những hố nhỏ.
- Gradient: là phần diện tích dilation - erosion.

Chú ý:
Nếu hiểu rõ bản chất của erosion và dilation, ta có thể chủ động chọn kernal cho hợp lý để tách bỏ các đường nối giữa các tế bào.

Ví dụ: Trong ảnh dưới, ta thấy các đường dọc theo trục oy khá mảnh, nhưng lại nối các tế bào với nhau, làm cho việc đếm sai, do đó để cắt các đường nhỏ này, ta dùng erosion với kernal là HCN có chiều cao nhỏ hơn chiều ngang: `kernel = np.ones((5, 14),np.uint8)`





Thấy còn vài đường mảnh chưa được cắt, sau đó ta có apply tiếp vài phép erosion hay dilation khác, với kernal nhỏ hơn cho phù hợp.

https://towardsdatascience.com/understanding-morphological-image-processing-and-its-operations-7bcf1ed11756

Các bước thực hiện:

B1: Đọc ảnh grayscale

```
image = cv2.imread('im.png')
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

B2: Chuyển thành ảnh binary

```
_, bw = cv2.threshold(gray, 127, 255, cv2.THRESH_BINARY)
img = 255 - bw
```

B3: Áp dụng các phép dilation, erosion, opening, closing

```
kernel = np.ones((20, 5),np.uint8)
img = cv2.erode(img, kernel, iterations = 1)
kernel = np.ones((5, 5),np.uint8)
img = cv2.erode(img, kernel, iterations = 1)
kernel = np.ones((5, 5),np.uint8)
img = cv2.erode(img, kernel, iterations = 1)

kernel = np.ones((5, 14),np.uint8)
img = cv2.erode(img, kernel, iterations = 1)

# cv2.dilate
# cv.morphologyEx(img, cv.MORPH_OPEN, kernel)
# cv.morphologyEx(img, cv.MORPH_CLOSE, kernel)
```

B4: Áp dụng Canny

```
edged = cv2.Canny(img, 30, 200)
```

B5: Tìm contours

```
contours, hierarchy = cv2.findContours(edged, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)
```

B6: Đếm vật thể

2. Bài toán cân bằng histogram

Cân bằng histogram (histogram equalization) là sự điều chỉnh histogram về trạng thái cân bằng, làm cho phân bố (distribution) giá trị pixel không bị co cụm tại một khoảng hẹp mà được "kéo dãn" ra. Trong thực tế, camera thường chịu tác động từ điều kiện sáng. Điều đó khiến cho nhiều ảnh bị tối hoặc quá sáng. Cân bằng histogram là một phương pháp tiền/hậu xử lí ảnh rất mạnh mẽ.

Bài toán cân bằng histogram cần thực hiện 4 bước (Ảnh gray):

B1: Tính histogram image

B2: Tính tổng tích lũy CDF

B3: Cân bằng tổng tích lũy (i - min)/(max - min)*255

B4: Apply vào ảnh input

```python
# b1: Tinh histogram
def histogram(img):
    flat_img = img.ravel() #flatten image
    hist = np.zeros(256, dtype=np.int)
    np.add.at(hist, flat_img, 1) #add 1 in the histogram
    return hist

# b2: Tinh tong tich luy
def tong_tich_luy(img):
    hist = histogram(img)
    ma_tran_tich_luy = [hist[0]]
    for i in hist[1:]:
        ma_tran_tich_luy.append(i + ma_tran_tich_luy[-1])
    return ma_tran_tich_luy

# b3: can bang
def equalization_histogram(img):
    ma_tran_tich_luy = tong_tich_luy(img)
    ma_tran_tich_luy = [(i - ma_tran_tich_luy[0])/(ma_tran_tich_luy[-1] - ma_tran_tich_luy[0]) * 255 for i in ma_tran_tich_luy]
    ma_tran_tich_luy = list(map(int, ma_tran_tich_luy))
    return ma_tran_tich_luy

# B4: ap vao anh
def apply_CDF_function(img):
    ma_tran_tich_luy = equalization_histogram(img)
    img_equalized = np.interp(img, np.arange(0, 256), ma_tran_tich_luy).astype(np.uint8)
    return img_equalized

result = apply_CDF_function(img)
cv2.imshow('result', result)
```

Đối với ảnh RGB có 2 hướng tiếp cận:

Hướng 1: Cân bằng sáng trên 3 channnel

Hướng 2: Chuyển sang HSV, sau đó cân bằng trên V channel

https://viblo.asia/p/xu-li-anh-thuat-toan-can-bang-histogram-anh-GrLZDOogKk0

3. **Bài toán blending image**

Chỉ cần nhớ là ảnh muốn show ra được bằng opencv.imshow thì phải chuyển về kiểu dữ liệu sau: blending = blending.astype(np.uint8).

Các bước:

B1: Resize

B2: Blending

B3: Chuyển type

B4: Show result.

```python
import numpy as np
import cv2


img1 = cv2.imread('im2.jpg')
img2 = cv2.imread('im2.png')

img1 = cv2.resize(img1, (100, 100))
img2 = cv2.resize(img2, (100, 100))

def alphaBlending(img1, img2, alpha=0.2):
    blending = img1 * alpha + img2 * (1- alpha)
    blending = blending.astype(np.uint8)
    return blending

blending = alphaBlending(img1, img2, alpha=0.5)


cv2.imshow('blending', blending)
cv2.waitKey(0)
```

4. **Bài toán Seam carving**

Các bước thực hiện:

Bước 1: Tính energy

```python
def cal_energy(arr):
    energy_x = np.concatenate((arr[:, 1:], np.zeros((arr.shape[0], 1))), axis=1) - arr
    energy_y = np.concatenate((arr[1:, :], np.zeros((1, arr.shape[1]))), axis=0) - arr
    energy = sqrt(energy_x **2 + energy_y **2)
    return energy
```

Bước 2: Tính tổng: energy[i, j] += min(energy[i-1, j-1:j+2])

Bước 3: Tìm min từ dưới lên và xóa

```python
def seam_carving(arr):
    energy = cal_energy(arr)
    for i in range(1, energy.shape[0]):
        for j in range(0, energy.shape[1]):
            if j == 0:
                energy[i, j] += min(energy[i -1, :2])
            elif j == energy.shape[1]:
                energy[i, j] += min(energy[i -1, -2:])
            else:
                energy[i, j] += min(energy[i -1, j-1:j+2])

    # khoi tao
    minimum_seam_index = []
    minimum_seam_index.append(np.argmin(energy[-1, :]))

    # 2. Tim vi tri min cac hang phia tren (vi tri i-1:i+1)
    for i in range(energy.shape[0] - 2, -1, -1):
        if minimum_seam_index[-1] == 0:
            minimum_seam_index.append(minimum_seam_index[-1] + np.argmin(energy[i, :2]))
        elif minimum_seam_index[-1] == arr.shape[1] - 1:
            minimum_seam_index.append(minimum_seam_index[-1] + np.argmin(energy[i, -2:]) -1)
        else:
            minimum_seam_index.append(minimum_seam_index[-1] + np.argmin(energy[i, minimum_seam_index[-1] -1: minimum_seam_index[-1] + 2

    # 3. Dieu chinh lai vi tri minimum_seam_index phu hop voi anh goc
    minimum_seam_index.reverse()

    # 4. xoa cac gia tri tren duong seam tim duoc
    m,n = arr.shape
    arr = arr[np.arange(n) != np.array(minimum_seam_index)[:,None]].reshape(m,-1)

    return arr
```

5. **Bài toán remove the background**

   Có 2 cách tiếp cận:

   a. **Dùng cách thống kê**

   Bước 1: Load 1 ảnh chroma key không có người hoặc có người cũng được.

   Bước 2: Dùng chuột để khoanh vùng có nền xanh (đọt chuối)

   Bước 3: Trong vùng đã khoanh, thống kê: giá trị min, max trên mỗi channel. Sau đó cộng trừ thêm bias khoảng 50.

   Bước 4: Load một tấm ảnh chroma key bất kỳ

   Bước 5: Loại bỏ những điểm ảnh (r, g, b) có giá trị màu không nằm ngoài giá trị thống kê bên trên

   Lưu ý:

   khi cộng trừ thêm bias, chú ý xem giá trị có có bị vòng ngược lại không. 242 + 60 = 46=> khi này là sai, cần chuyển sang giá trị int.

```python
import numpy as np
import cv2


img = cv2.imread('test_img.jpg')
background = cv2.imread('background.png')

min_value = np.array([background[ ... , i].min() for i in range(3)]).astype(int) - 100
max_value = np.array([background[ ... , i].max() for i in range(3)]).astype(int) + 100


mask = cv2.inRange(img, min_value, max_value)

img = cv2.bitwise_and(img, img, mask=~mask)

cv2.imshow('results', img)
cv2.waitKey(0)
```

b. Dùng ML classifier

B1: tạo dữ liệu cho 2 class:

- ảnh background cho class backgroud.
- ảnh bất kì có màu khác background cho forground

B2: Train classifier model

B3: Predict

B4: Tạo mask

B5: Replace pixel

6. Bài toán template matching

https://phamdinhkhanh.github.io/2020/01/06/ImagePreprocessing.html

Tương tự correlation, chỉ đổi dòng code:
arr = img[i : i+filter.shape[0] : -1, j : j+filter.shape[1] : -1, :]

Bài toán này dùng kĩ thuật corration + findcontours
Các bước thực hiện:

Bước 1: Đọc ảnh gray scale và normalize

```python
# 1. Read images
img = cv2.imread('9-ro.jpeg', 0)
kernel = cv2.imread('template.png', 0)


# normalize image
img = img / 255.
kernel = kernel / 255.
```

Bước 2: Tính correlation

```
def correlation(img, kernel):
    xOutput=int((img.shape[0]-kernel.shape[0])+1)
    yOutput=int((img.shape[1]-kernel.shape[1])+1)

    output=np.zeros((xOutput,yOutput))

    for i in range(xOutput):
        for j in range(yOutput):
            output[i,j] = np.sum(np.multiply(kernel,img[i:i+kernel.shape[0],j:j+kernel.shape[1]]))

    return output
```

Bước 3: Chuyển ảnh thành binary

```
img = img[kernel.shape[0]//2: kernel.shape[0]//2 + cor.shape[0], kernel.shape[1]//2: kernel.shape[1]//2 + cor.shape[1]]

th, thresh = cv2.threshold(cor, 127, 255, cv2.THRESH_BINARY|cv2.THRESH_OTSU)
```

Bước 4: Find contour

```
contours, _ = cv2.findContours(thresh, cv2.RETR_LIST, cv2.CHAIN_APPROX_SIMPLE)
```

Bước 5: Find boxes

```
boxes = []
for c in contours:
    rect = cv2.boundingRect(c)
    if rect[2] < 70 or rect[2] > 100 or rect[3] > 100 or rect[3] < 70: continue
    x,y,w,h = rect
    boxes.append((x, y, w, h))

boxes = np.array(boxes)
scores = np.array([0.8] * len(boxes))
```

Bước 6: Dùng MNS

```
idxs = cv2.dnn.NMSBoxes(boxes, scores, score_threshold=0.4, nms_threshold=0.1)
for id in idxs:
    x, y, w, h = boxes[id]
    cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)
    cv2.putText(img, "ID: " + str(id), (x, y - 5), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)

cv2.imshow('img', img)
cv2.waitKey(0)
```

7. Bài toán đếm hàng hóa trên hệ thống băng chuyền

B1: Đọc video: cap = cv2.VideoCapture(video_path)

Cách đọc mỗi frame:

while(True):

ret, frame = cap.read()

if not ret:

break

B2: Tách nền cho các đối tượng đang di chuyển (gói hàng): abs(curr_frame – prev_frame) ta sẽ lấy được cạnh của đối tượng có sự di chuyển:

diff = cv2.absdiff(prev_frame, curr_frame)

Bước này cần trả về danh sách diff giữa các frame → diffs

B3: Trên mỗi diff, áp dụng template matching với pattern là ảnh gói hàng, nơi nào đạt ngưỡng thì ghi nhận tọa độ (gọi là package_cor).

B4: Lần lượt đánh index cho các gói hàng trên diff[0].
B5: Từ diff[1]: Tính ngưỡng IoU đối với mỗi package_cor của cur_diff bắt cặp với tất cả package_cor của prev_diff:

gói hàng nào của cur_diff tồn tại bắt cặp có giá trị IoU ≥ threshold → đánh index giống với gói hàng mà nó được so.

gói hàng nào của cur_diff tất cả các bắt cặp đều < threshold → đánh index mới

B6: Tổng số gói hàng đã chạy trên băng ghi hình bằng index lớn nhất.

8. Bài toán hough transform

```
def deg2grad(deg):
    return deg*3.141592654/180


# read image
img = cv2.imread('test.png', 0)
edges = cv2.Canny(img, 50, 150, apertureSize=3)
lines = cv2.HoughLines(edges, rho=1, theta=np.pi/180, threshold=100)

# Step 0.5: initialize hough table
theta_range = np.arange(-3.14, 3.14, 0.01)
H = np.zeros((500, len(theta_range)), dtype=np.uint8)

# Step 1: accumulate hough space
def accumulate(point):
    #for theta in range(360):
    for theta in theta_range:
        pro = point[0]*np.cos(theta) + point[1]*np.sin(theta)
        # If pro in range of Hough space
        if pro >= 0 and pro < 500:
            # map theta to Hough space
            H[int(pro), int((theta+3.14)/0.01)] += 1


h, w = edges.shape[:2]
for i in range(h):
    for j in range(w):
        if edges[i, j] != 0:
            accumulate([i, j])
lineK = np.where(H > 50)
```

```
for line in lines:
    rho, theta = line[0]
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))
    cv2.line(img,(x1,y1),(x2,y2),(0,0,0),1)
cv2.imshow('a',img)


for i in range(len(lineK[0])):
    pro = lineK[0][i]
    theta = lineK[1][i] * 0.01 - 3.14
    a = np.cos(theta)
    b = np.sin(theta)
    if a ==0:
        x1, x2 = 0, 1000
        y1, y2 = int(pro / b), y1
    elif b == 0:
        x1, x2 = int(pro / a), x1
        y1, y2 = 0, 1000
    else:
        x1, x2 = 0, 1000
        y1, y2 = int(pro/np.sin(theta)),int((pro - x2 * a)/b)
    cv2.line(img,(y1,x1),(y2,x2),(0,0,255),1)
cv2.imshow('result', img)
cv2.waitKey(0)
```

Phần 3: Các hàm và thư viện

CV2: import cv2

cv2.cvtColor(img, cv2.COLOR_BRG2GRAY)

cv2.erode(img, kernel, iter=1) → 1 img (mặc định anchor là tâm kernel)

cv2.dilate(img, kernel, iter=1) → 1 img (mặc định anchor là tâm kernel)

cv2.findContours(img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE) → contours, hierarchy (contours là tập hợp các bộ tọa độ (x,y) là đường bao cho mỗi TPLT, số lượng TPLT = len(contours))

cv2.drawContours(img, contours, -1,(0, 255, 0),3) → vẽ trực tiếp lên img

cv2.inRange(img,upper_bound, lower_bound) → 1 img, ngưỡng là [R, G, B] (trả về 255 nếu pixel nằm trong ngưỡng, 0 nếu ngược lại).

cv2.filter2d(img, -1, kernel) → 1 img

cv2.fillConvexPoly(mask, npar, 255), vẽ trực tiếp lên mask, npar là np.arr các điểm

cv2.Canny(img, low_thres, high_thres) → 1 gray img (phát hiện cạnh)

Tạo video:

fourcc = cv2.VideoWriter_fourcc(*'MP42’)

video = cv2.VideoWriter('video.avi', fourcc, float(24), (width, height))

video.write(frame)

video.release()

NUMPY

Phép toán ma trận: *, +, -, np.abs(arr), np.sqrt(arr), → 1 ma trận

Phân phối: np.var(arr), np.mean(arr), np.median(arr), np.sum(arr), np.min(arr), np.max(arr), np.argmin(arr), np.argmax(arr) → 1 số nguyên

Làm phẳng: arr.flatten()

Gán trên vùng: arr[mask>threshold] = 255 // hạn chế: gt gán cố định

Gán trên vùng: result = np.where((arr>threshold), 255, arr2)

Chuyển về mảng: arr.tolist() // để ghép mảng lại (np ko làm đc)

Trả về kết quả phép & trên tất cả phần tử: np.all()

Trả về kết quả phép || trên tất cả phần tử: np.any()

IMAGEIO: import imageio

Convert từ đối tượng np sang imageio: imageio.core.util.Array(img)

Tạo gif: imageio.mimsave(path, frames)

IMUTILS: import imutils

Xoay ảnh: imutils.rotate_bound(img, angle=90)