Họ và tên: Huỳnh Vĩ Hà

MSSV: 15520175

Bài tập 2: THUẬT TOÁN PHÁT HIỆN ĐƯỜNG THẲNG TRONG ẢNH

Bước 1: Tính ma trận biên cạnh theo trục x

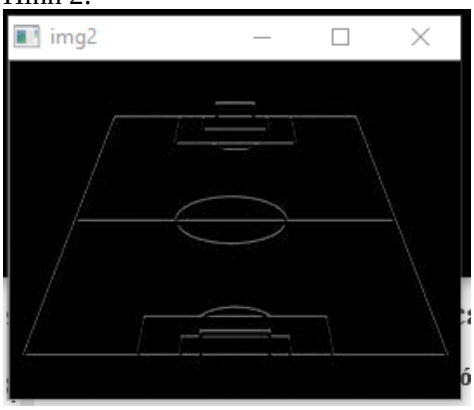
- **Lọc ảnh với kernel như bên dưới**

0	-1	0
0	0	0
0	1	0

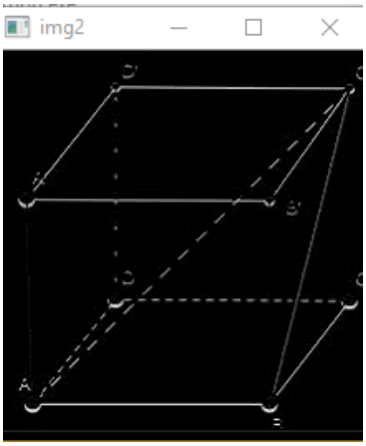
- **Code:**
  kernel2 = np.array([[0, -1, 0], [0, 0, 0], [0, 1, 0]])
  img2 = cv2.filter2D(img, -1, kernel2)

- **Một số kết quả**
  Hình 1:

  Hình 2:



Hình 3:



Bước 2: Tính ma trận biên cạnh theo trục y

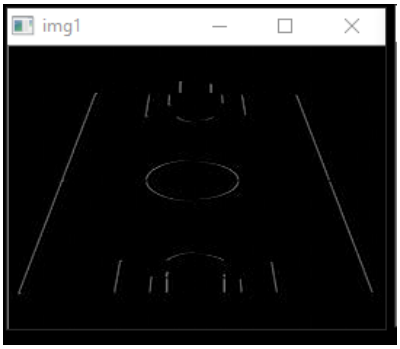
- **Lọc ảnh như kernel bên dưới**

0	0	0
-1	0	1
0	0	0

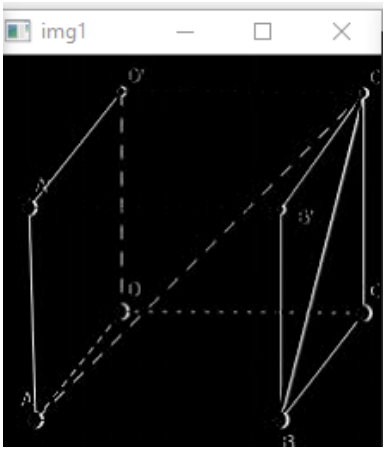
- **Code:**
kernel1 = np.array([[0, 0, 0], [-1, 0, 1], [0, 0, 0]])
img1 = cv2.filter2D(img, -1, kernel1)

- **Một số kết quả:**
Hình 1:

Hình 2:



Hình 3:



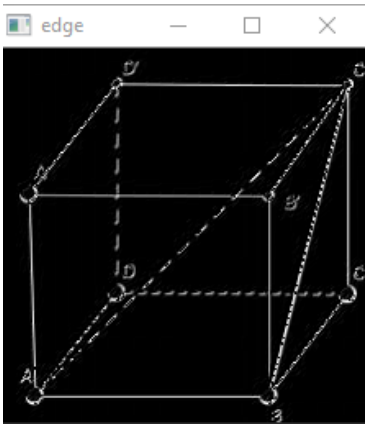
Bước 3: Tính ma trận biên cạnh tổng hợp theo cả 2 trục x và y

- **Code:**
  edge = img1 + img 2

- **Một số kết quả**
  Hình 1:

Hình 2:

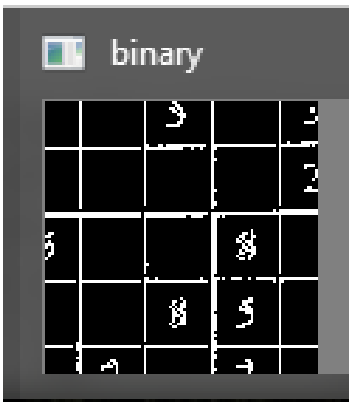
Hình 3:



Bước 4: Nhị phân hóa ma trận biên cạnh (image binarization)

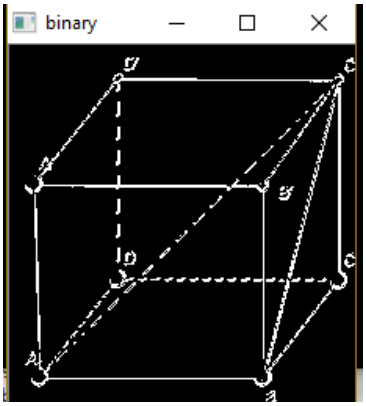
- **Code:**
  ret, binary = cv2.threshold(edge, 50, 255, cv2.THRESH_BINARY)

- **Một số kết quả:**
  Hình 1



Hình 2:

Hình 3:



Bước 5: Cài đặt thuật toán xác định biên cạnh

1. Cài đặt bằng thuật toán bắt cặp ngẫu nhiên

- **Cấu trúc dữ liệu**
  x1, y1: tọa độ điểm thứ nhất
  x2, y2: tọa độ điểm thứ hai
  lines: lưu các cặp điểm phát sinh ngẫu nhiên có tọa độ nằm trên đường thẳng trong ảnh.

- **Chọn hai điểm ngẫu nhiên**

Ta chọn hai điểm ngẫu nhiên có giá trị bằng 255 (màu trắng) trong ảnh đã nhị phân hóa.

- **Code:**

```
while 1:
        x1 = random.randint(1, height - 1)
        y1 = random.randint(1, width- 1)
        if binary[x1][y1] == 255:
                break
while 1:
        x2 = random.randint(1, height - 1)
        y2 = random.randint(1, width - 1)
        if binary[x2][y2] == 255:
                break
```

- **Tìm ra các đường thẳng**

  Duyệt qua ma trận ảnh nhị phân, nếu điểm A có giá trị bằng 255 (màu trắng) và thuộc đường thẳng tạo bởi hai điểm vừa chọn ngẫu nhiên thì tăng biến count lên 1.

  Sau khi duyệt xong, nếu biến count lớn hơn một giá trị ngưỡng nào đó (do ta quy định) thì thêm tọa độ hai điểm ngẫu nhiên vào danh sách lines.

  Tiếp tục phát sinh thêm hai điểm ngẫu nhiên khác và làm như trên.

- **Code:**

```
count = 0
        for x in range(height):
                for y  in range(width):
                        if binary[x][y] == 255:
                                if (x - x1)*(y2 - y1) == (y - y1)*(x2 - x1):
                                        count += 1
        if count > 50:
                lines += [[x1, y1, x2, y2, count]]
```

- **Vẽ đường thẳng đã phát hiện lên ảnh gốc**

  Ta vẽ bằng cách tính toán tọa độ của hai điểm trên đường thẳng (dựa vào tọa độ các điểm phát sinh ngẫu nhiên đã lưu trong lines).

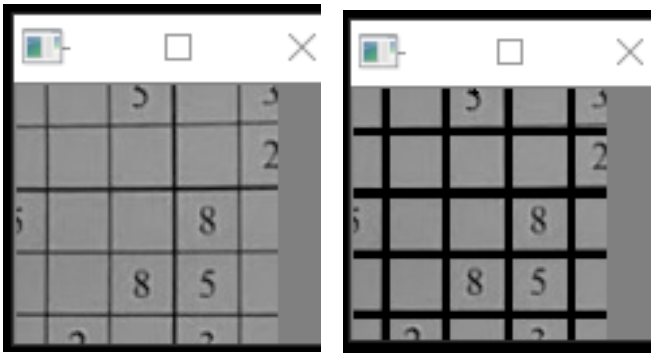
  Sau đó dùng hàm cv2.line để vẽ đường thẳng từ hai điểm vừa rìm được như trên.

- **Code:**

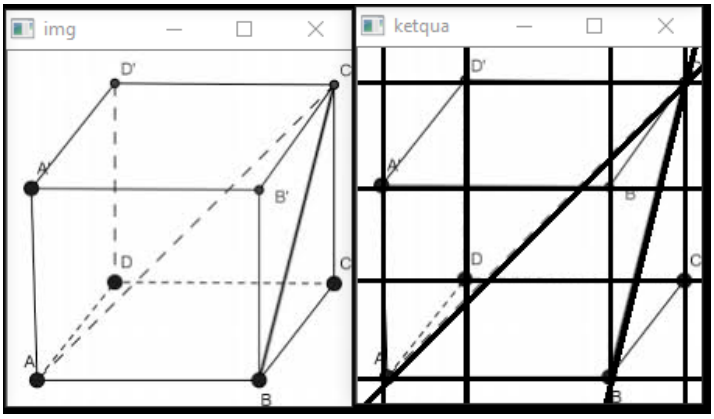
```
for (x1, y1, x2, y2, count) in lines:
        print(x1, y1, x2, y2, count)
        a = (x1 - x2)
        b = (y1 - y2)
        x3 = x1 + 1000*a
        x4 = x2 - 1000*a
        y3 = y1 + 1000*b
        y4 = y2 - 1000*b
        cv2.line(img,(y3, x3) ,(y4, x4),(0,0,255),2)
```

- **Một số kết quả:**

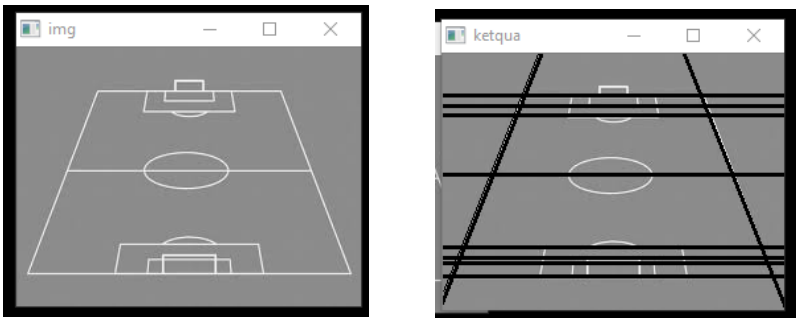
  Hình 1:



Hình 2:



Hình 3:



2. Cài đặt bằng thuật toán Hough Transform for Line Detetion

- **Cấu trúc dữ liệu**
  width: chiều rộng ảnh
  height: chiều cao ảnh
  tmp: chiều dài đường chéo
  M: ma trận (tmp x 181) khởi tạo bằng 0, đại diện cho không gian (r, theta)
  lines: danh sách lưu giá trị (r, theta) của các đường thẳng được tìm thấy.

- **Code (phần cơ sở dữ liệu)**
  width = binary.shape[1]
  height = binary.shape[0]
  *tmp = int(math.sqrt(width**2 + height**2))*
  M = np.zeros((tmp, 91), dtype = np.int)
  lines = list()

- **Chuyển một điểm sang không gian hough**
  Duyệt qua tất cả các điểm ảnh trong ảnh, nếu ảnh đó có giá trị là 255 (màu trắng) thì ra tiến hành chuyển điểm đó sang không gian hough như sau:
  Cho theta chạy từ giá trị 0 đến giá trị pi, ở mỗi giá trị của theta, ta tính giá trị r theo công thức:
  r = x.sin(theta) + y.cos(theta)

Như vậy, cứ một điểm ở không gian (x, y) sẽ được chuyển thành 180 điểm (thành một đường cong) trong không gian (r, theta).
Ta lưu một điểm trong không gian (r, theta) bằng cách cho M[r, theta] tăng thêm 1, nếu M[r1, theta1] = x có nghĩa là có x điểm đi qua vị trí [r1, theta1].

- **Code:**

```
for x in range(height):
    for y in range(width):
        if binary[x][y] == 255:
            for theta in range(181):
                rho = int(x*math.sin(theta*math.pi/180) +
                y*math.cos(theta*math.pi/180))
                M[rho, theta] += 1
```

- **Tìm ra các đường thẳng:**

Nếu giá trị tại M[r, theta] càng lớn thì càng có nhiều đường cong đi qua điểm đó, tất điểm đó đại diện cho một đường thẳng bên không gian (x, y).
Do đó, ta duyệt ma trận M, nếu M[r, theta] có giá trị lớn hơn một ngưỡng nhất nào đó (do ta quy định) thì ta đưa điểm đó vào list lines

- **Code:**

```
for rho in range(-tmp, tmp):
    for theta in range(181):
        if M[rho, theta] > 70:
            lines += [[rho, theta*math.pi/180]]
```

- **Vẽ đường thẳng đã phát hiện lên ảnh gốc**

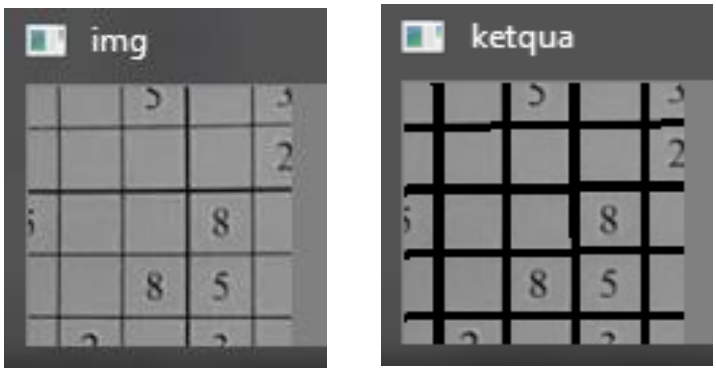
Ta vẽ bằng cách tính toán tọa độ của hai điểm trên đường thẳng (dựa vào r và theta).
Sau đó dùng hàm cv2.line để vẽ đường thẳng từ hai điểm vừa rìm được như trên.

- **Code:**

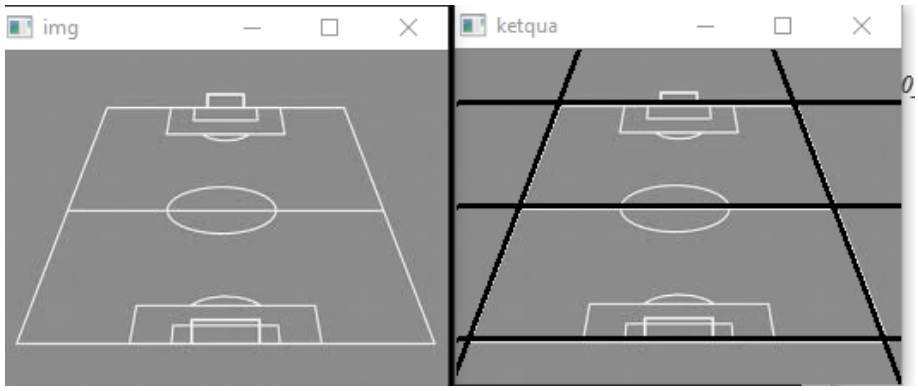
```
for (rho,theta) in lines:
    print(rho, theta)
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))
    cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)
```

- **Một số kết quả**

Hình 1



Hình 2:



Hình 3: