TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN XỬ LÝ ẢNH SỐ**

**RETRIEVING THE TIME FROM THE PHOTO OF A OCLOCK**

*Người hướng dẫn*: **TS PHẠM XXX XXX**

*Người thực hiện*: **PHƯƠNG VY – 5XXHXX**

Lớp **: 19HXXXX**

Khoá  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN CUỐI KÌ MÔN XỬ LÝ ẢNH SỐ**

**RETRIEVING THE TIME FROM THE PHOTO OF A OCLOCK**

*Người hướng dẫn*: **TS PHẠM XXX XXX**

*Người thực hiện*: **PHƯƠNG VY – 5XXHXX**

Lớp **: 19HXXXX**

Khoá  **: 23**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024**

LỜI CẢM ƠN

Em muốn gửi lời cảm ơn sâu sắc tới thầy đã dành thời gian giảng dạy và hướng dẫn cho em trong quá trình hoàn thành đồ án cuối kì môn Xử lý ảnh số với đề tài "Trích xuất thời gian từ ảnh của đồng hồ". Thầy đã giúp em khám phá và tiếp cận với những kiến thức mới và thú vị trong lĩnh vực xử lý ảnh số, đặc biệt là sử dụng thư viện OpenCV.

Em rất cảm kích với sự hỗ trợ và động viên của thầy trong suốt quá trình thực hiện đồ án, giúp em giải quyết những vấn đề và thử nghiệm những ý tưởng mới. Thầy luôn tạo môi trường học tập tích cực và chân thành, giúp cho em cảm thấy yên tâm và tự tin trong việc hoàn thành đồ án.

Em hy vọng sẽ có cơ hội được học hỏi thêm từ thầy trong tương lai và gửi lại lời cảm ơn sâu sắc từ em.

**ĐỒ ÁN ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm đồ án của riêng tôi và được sự hướng dẫn của TS Phạm Văn Huy. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong đồ án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung đồ án của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Phương Vy*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày tháng năm

(kí và ghi họ tên)

TÓM TẮT

Đề tài xử lý ảnh số này để trích xuất thời gian từ ảnh của đồng hồ dựa theo các kim thời gian. Em bắt đầu bằng việc đọc một ảnh đồng hồ và chuyển đổi nó thành ảnh xám. Sau đó, sử dụng phép biến đổi HoughCircle để tìm vùng hình tròn lớn nhất chứa đồng hồ trong ảnh. Tiếp theo, đoạn mã sử dụng phép biến đổi HoughLineP để tìm các đường thẳng trên ảnh, đại diện cho các kim của đồng hồ. Sau đó, nó nhóm các đường thẳng song song lại tính toán để có 3 nhóm các đường thẳng song song đại diện cho 3 kim đồng hồ. Tiếp theo, xác định các loại kim đồng hồ dựa trên chiều dài và độ dày của chúng. Sau đó vẽ hình chữ nhật bao quanh các kim với màu khác nhau. Tiếp theo, tính toán góc của các kim đồng hồ dựa trên vector tâm của đồng hồ. Cuối cùng, nó tính toán thời gian hiện tại của đồng hồ dựa trên góc của các kim, sau đó làm tròn và ghi kết quả lên ảnh.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc387692905)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc387692906)

[TÓM TẮT iv](#_Toc387692907)

[MỤC LỤC 1](#_Toc387692908)

CHƯƠNG 1 – Vẽ hình chữ nhật xung quanh các kim đồng hồ với màu khác nhau một cách tự động

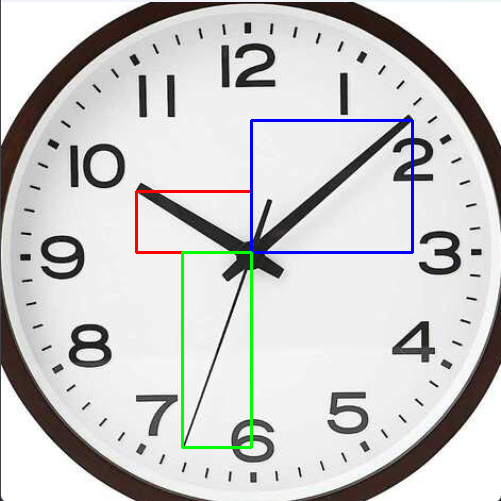
1.1 Thuật toán

1.1.1 Yêu cầu

Đầu vào: Ảnh đồng hồ



Đầu ra: Ảnh đồng hồ với các hình chữ nhật được vẽ tự động bao quanh các kim đồng hồ với màu khác nhau (kim giây – xanh lá, kim phút – xanh dương, kim giờ - đỏ)



1.1.2 Thuật toán sử dụng

**Bước 1:** Xử lý ảnh để dễ nhận biết các đối tượng

* Đổi ảnh từ BGR sang HSV và đảo màu
* Áp dụng CLAHE tăng cường độ tương phản
* Áp dụng THRESH\_BINARY và THRESH\_OTSU để tìm ngưỡng tốt nhất để đổi sang ảnh nhị phân
* Làm mờ ảnh bằng GaussianBlur

**Bước 2:** Tìm hình tròn lớn nhất, cắt ảnh theo hình tròn lớn nhất và resize về kích cỡ chung 500x500

* Áp dụng HoughCircles tìm hình tròn có độ dày lớn nhất
* Cắt hình ảnh theo hình tròn vừa tìm được
* Resize về kích thước 500x500 theo tâm hình tròn

**Bước 3:** Tìm 3 kim đồng hồ

* Áp dụng phương pháp Canny Edge Detection để phát hiện các cạnh trong ảnh
* Tìm các đường thẳng sử dụng Hough Line Transform
* Xác định tâm của đồng hồ là (250,250) vì ảnh luôn được cắt theo 500x500
* Tìm các đường thẳng gần tâm bằng cách tính toán góc giữa các đoạn thẳng và tâm đồng hồ
* Nhóm các đường thẳng gần song song lại thành các nhóm riêng
  + Xoá các đoạn thẳng quá xa tâm đồng hồ với **np.linalg.norm** để tính khoảng cách Euclidean giữa các điểm của đoạn thẳng và tâm
  + Tính góc của các đoạn thẳng bằng np.arctan2 sau đó đổi sang độ bằng np.degrees
  + Nhóm các đoạn thẳng có góc chênh lệch dưới 12 độ vào 1 nhóm
* Loại bỏ các đường thẳng không nằm trong tâm, quá xa so với nhóm, giữ lại các nhóm có đoạn thẳng dài nhất (loại bỏ đuôi của kim đồng hồ)
  + Tính góc giữa vector của đoạn thẳng và vector từ tâm hình tròn tới tâm đường thẳng, sau đó nhóm các đường thẳng theo góc (chênh lệch dưới 30 độ). Dùng np.linalg.norm, sau đó tính cos\_theta
  + Sắp xếp các nhóm tạm thời theo độ dài trung bình của các đoạn thẳng trong nhóm. Tính toán khoảng cách Euclid giữa chúng bằng hàm np.linalg.norm()
  + Giữ lại nhóm có độ dài trung bình lớn nhất trong temp. Ta có được 3 nhóm đại diện cho 3 kim đồng hồ
* Tìm ra hình dạng của 3 kim đồng hồ dựa vào 3 nhóm đường thẳng trên gồm chiều dài và độ dày
  + Tìm chiều dài bằng khoảng cách giữa điểm xa tâm hình tròn nhất tới tâm hình tròn trong danh sách các đường thẳng
  + Tìm độ dày bằng khoảng cách lớn nhất giữa các đường thẳng song song có trong 1 nhóm (độ dài của vector vuông góc giữa hai đường thẳng)
* Xác định chính xác 3 kim đồng hồ
  + Kim giây: Đường thẳng có độ dày nhỏ nhất
  + Kim phút: Đường thẳng dài hơn trong 2 đường thẳng còn lại
  + Kim giờ: Đường thẳng còn lại
* Vẽ hình chữ nhật xung quanh 3 kim đồng hồ với mỗi màu riêng bằng cv2.rectangle()

**1.2 Phương pháp và hiện thực**

**Bước 1:** Xử lý ảnh để dễ nhận biết các đối tượng

raw\_image = cv2.imread(image\_path)

clock\_image = cv2.imread(image\_path)

def getBlur(img):

    # Chuyển ảnh màu sang ảnh HSV

    hsv = img.copy()

    hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

    hsv = cv2.bitwise\_not(hsv)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8, 8))

    hsv[:, :, 2] = clahe.apply(hsv[:, :, 2])

    \_, thresh = cv2.threshold(hsv[:, :, 2], 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)

    blur = cv2.GaussianBlur(thresh, (5, 5), 0)

    return blur

blur = getBlur(raw\_image)

**Bước 2:** Tìm hình tròn lớn nhất, cắt ảnh theo hình tròn lớn nhất và resize về kích cỡ chung 500x500

# Tìm vùng hình tròn (đồng hồ)

# Tạo maxRadius bằng 1.5 lần max(width, height) của ảnh

max\_radius = int(max(raw\_image.shape) // 1.5)

circles = cv2.HoughCircles(blur, cv2.HOUGH\_GRADIENT, dp=1, minDist=400, param1=50, param2=30, minRadius=50, maxRadius=max\_radius)

# Lấy hình tròn lớn nhất

clock\_circle = None

center\_x, center\_y = None, None

if circles is not None:

    circles = np.uint16(np.around(circles))

    # Sắp xếp các hình tròn theo thứ tự giảm dần của bán kính

    circles = sorted(circles[0], key=lambda x: x[2], reverse=True)

    max\_circle = circles[0]

    # Vẽ hình tròn lớn nhất = màu đỏ

    clock\_circle = max\_circle

    center\_x, center\_y = max\_circle[0], max\_circle[1]

    copy\_image = raw\_image.copy()

    cv2.circle(copy\_image, (max\_circle[0], max\_circle[1]), max\_circle[2], (0, 0, 255), 2)

    # Hiển thị ảnh

    cv2.imshow('Clock Image', copy\_image)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

else:

    print("Không tìm thấy hình tròn trong ảnh.")

    exit()

# Tạo ảnh mới bằng cách cắt ảnh gốc theo hình tròn lớn nhất

cut\_image = None

image\_show = None

lines = []

if clock\_circle is not None:

    # cắt ảnh theo hình tròn lớn nhất

    x, y, r = clock\_circle

    # bán kính tràn ra ngoài ảnh thì lấy min(x, y)

    if(r > x or r > y):

        r = min(x, y)

    clock\_image = raw\_image[y - r:y + r, x - r:x + r]

    # resize ảnh về kích thước 500x500

    height, width = clock\_image.shape[:2]

    scale = 500 / max(height, width)

    clock\_image = cv2.resize(clock\_image, None, fx=scale, fy=scale)

    cut\_image = clock\_image.copy()

    # Chuyển ảnh màu sang ảnh xám

    clock\_image = getBlur(clock\_image)

**Bước 3:** Tìm 3 kim đồng hồ

* Áp dụng phương pháp Canny Edge Detection để phát hiện các cạnh trong ảnh
* Tìm các đường thẳng sử dụng Hough Line Transform

    # Tìm cạnh của ảnh

    edges = cv2.Canny(clock\_image, 50, 170, apertureSize=3)

    # Sử dụng Hough Line Transform để tìm đường thẳng

    # Tạo maxLineGap từ bán kính của ảnh

    lines = cv2.HoughLinesP(edges, 1, np.pi / 180, threshold=80, minLineLength=10, maxLineGap=3)

* Xác định tâm của đồng hồ là (250,250) vì ảnh luôn được cắt theo 500x500
* Tìm các đường thẳng gần tâm bằng cách tính toán góc giữa các đoạn thẳng và tâm đồng hồ

# Tìm các đường gần tâm đường tròn gộp chúng lại

# Tính tâm của đường tròn theo width và height ban đầu là 500

center = (250, 250)

# Tìm các đường có 1 hướng vector chỉ về tâm của đường tròn

image\_show = cut\_image.copy()

focus\_center\_lines = []

for line in lines:

    x1, y1, x2, y2 = line[0]

    # Tính tâm của đoạn thẳng

    line\_center = ((x1 + x2) // 2, (y1 + y2) // 2)

    # Tính vector từ tâm của đường tròn tới tâm của đoạn thẳng

    vector = (line\_center[0] - center[0], line\_center[1] - center[1])

    # Tính vector của đoạn thẳng

    line\_vector = (x2 - x1, y2 - y1)

    # Tính góc giữa 2 vector

    dot = vector[0] \* line\_vector[0] + vector[1] \* line\_vector[1]

    length\_vector = np.linalg.norm(vector)

    length\_line\_vector = np.linalg.norm(line\_vector)

    cos\_theta = dot / (length\_vector \* length\_line\_vector)

    theta = np.arccos(cos\_theta)

    theta = np.degrees(theta)

    # Nếu góc giữa 2 vector nhỏ hơn 10 độ thì đường thẳng này gần tâm

    if theta < 20 or theta > 160:

        focus\_center\_lines.append(line)

        cv2.line(image\_show, (x1, y1), (x2, y2), (255, 0, 0), 2)

* Nhóm các đường thẳng gần song song lại thành các nhóm riêng

# Nhóm các đường thẳng gần song song lại

groups = []

for line in focus\_center\_lines:

    x1, y1, x2, y2 = line[0]

    # Kiểm tra xem đường thẳng có nằm trong bán kính không

    if np.linalg.norm(np.array([x1, y1]) - np.array(center)) > 250 or np.linalg.norm(np.array([x2, y2]) - np.array(center)) > 250:

        continue

    # Tính góc của đường thẳng

    angle = np.arctan2(y2 - y1, x2 - x1)

    # Tính góc của đường thẳng so tâm của đường tròn

    angle = np.degrees(angle)

    found = False

    for group in groups:

        if abs(angle - group[0]) < 12 or abs(angle - group[0] - 180) < 12 or abs(angle - group[0] + 180) < 12:

            group.append(line)

            found = True

            break

    if not found:

        groups.append([angle, line])

# Sắp xếp các nhóm theo số lượng đường thẳng giảm dần

groups = sorted(groups, key=lambda x: len(x), reverse=True)

# Lấy 3 nhóm có nhiều đường thẳng nhất

groups = groups[:3]

if len(groups) != 3:

    print("Không tìm được 3 kim đồng hồ.")

    exit()

* Loại bỏ các đường thẳng không nằm trong tâm, quá xa so với nhóm, giữ lại các nhóm có đoạn thẳng dài nhất (loại bỏ đuôi của kim đồng hồ)

# Loại bỏ các đường thẳng quá xa so với các đường thẳng cùng nhóm

for group in groups:

    # Lấy tâm đồng hồ (250, 250) làm điểm trung tâm, nhóm các đường thẳng theo 2 nhóm trên và dưới tâm, loại bỏ nhóm ít đường thẳng

    # Tính khoảng cách và góc của các đường thẳng so với tâm đồng hồ

    distances = []

    angles = []

    for line in group[1:]:

        x1, y1, x2, y2 = line[0]

        # Tính tâm của đoạn thẳng

        line\_center = ((x1 + x2) // 2, (y1 + y2) // 2)

        # Tính vector từ tâm của đường tròn tới tâm của đoạn thẳng

        vector = (line\_center[0] - center[0], line\_center[1] - center[1])

        # Tính vector của đoạn thẳng (Xuất phát từ tâm đồng hồ)

        if abs(250 - x1 - y1) < abs( 250 - x2 - y2):

            line\_vector = (x2 - x1, y2 - y1)

        else:

            line\_vector = (x1 - x2, y1 - y2)

        # Tính góc giữa 2 vector

        dot = vector[0] \* line\_vector[0] + vector[1] \* line\_vector[1]

        length\_vector = np.linalg.norm(vector)

        length\_line\_vector = np.linalg.norm(line\_vector)

        cos\_theta = dot / (length\_vector \* length\_line\_vector)

        theta = np.arccos(cos\_theta)

        theta = np.degrees(theta)

        distances.append(length\_vector)

        angles.append(theta)

    # Nhóm các đường thẳng theo góc

    groups\_temp = []

    for i in range(len(group[1:])):

        found = False

        for group\_temp in groups\_temp:

            if abs(angles[i] - group\_temp[0]) < 30 :

                group\_temp.append(group[1:][i])

                found = True

                break

        if not found:

            groups\_temp.append([angles[i], group[1:][i]])

    # Sắp xếp các nhóm theo độ dài trung bình của các đường thẳng trong nhóm

    print(f"groups\_temp: {groups\_temp}")

    for i in range(len(groups\_temp)):

        group\_temp = groups\_temp[i]

        distances\_temp = [np.linalg.norm(np.array(line[0][:2]) - np.array(line[0][2:])) for line in group\_temp[1:]]

        # Thêm vào đầu mảng

        groups\_temp[i].insert(0, np.mean(distances\_temp))

    groups\_temp = sorted(groups\_temp, key=lambda x: x[0], reverse=True)

    # Bỏ dinstance temp

    for i in range(len(groups\_temp)):

        groups\_temp[i] = groups\_temp[i][1:]

    # Lấy nhóm có độ dài trung bình lớn nhất

    group[1:] = groups\_temp[0][1:]

* Tìm ra hình dạng của 3 kim đồng hồ dựa vào 3 nhóm đường thẳng trên gồm chiều dài và độ dày

# Tìm 3 kim đồng hồ

def distance\_between\_parallel\_lines(line1, line2):

    # Get the coordinates of two points on each line

    x1\_1, y1\_1, x2\_1, y2\_1 = line1[0]

    x1\_2, y1\_2, x2\_2, y2\_2 = line2[0]

    # Create two direction vectors of two straight lines

    vector1 = np.array([x2\_1 - x1\_1, y2\_1 - y1\_1])

    vector2 = np.array([x2\_2 - x1\_2, y2\_2 - y1\_2])

    #Creates a vector connecting a point on one line to a point on the other line

    vector\_between\_lines = np.array([x1\_2 - x1\_1, y1\_2 - y1\_1])

    #Calculates the perpendicular distance between the two lines.

    distance = np.abs(np.cross(vector1, vector\_between\_lines)) / np.linalg.norm(vector1)

    return distance

clock\_hands = []

for group in groups:

    hand = None

    # Tìm điểm gần tâm nhất và xa tâm nhất trong các điểm của các đường thẳng

    min\_point = None

    max\_point = None

    # Tìm độ dày của kim đồng hồ (tính bằng khoảng cách giữa tâm của 2 đường thẳng) < max\_length

    points = []

    for line in group[1:]:

        x1, y1, x2, y2 = line[0]

        if min\_point is None:

            min\_point = (x1, y1)

        if max\_point is None:

            max\_point = (x1, y1)

        if np.linalg.norm(np.array([x1, y1]) - np.array(center)) < np.linalg.norm(np.array(min\_point) - np.array(center)):

            min\_point = (x1, y1)

        if np.linalg.norm(np.array([x2, y2]) - np.array(center)) < np.linalg.norm(np.array(min\_point) - np.array(center)):

            min\_point = (x2, y2)

        if np.linalg.norm(np.array([x2, y2]) - np.array(center)) > np.linalg.norm(np.array(max\_point) - np.array(center)):

            max\_point = (x2, y2)

        if np.linalg.norm(np.array([x1, y1]) - np.array(center)) > np.linalg.norm(np.array(max\_point) - np.array(center)):

            max\_point = (x1, y1)

    # Tính độ dày của kim đồng hồ = độ dài lớn nhất giữa 2 điểm

    thickness = 0

    for i in range(len(group[1:])):

        for j in range(i + 1, len(group[1:])):

            distance = distance\_between\_parallel\_lines(group[1:][i], group[1:][j])

            if distance > thickness:

                thickness = distance

    # Tạo đường thẳng từ min\_point đến max\_point

    hand = (250, 250, max\_point[0], max\_point[1])

    clock\_hands.append((hand, thickness))

* Xác định chính xác 3 kim đồng hồ

# Tìm ra kim giờ, kim phút và kim giây

hour\_hand = None

minute\_hand = None

second\_hand = None

# Kim giây là kim có thickness nhỏ nhất

clock\_hands = sorted(clock\_hands, key=lambda x: x[1])

second\_hand = clock\_hands[0][0]

clock\_hands = clock\_hands[1:]

# Kim giờ là ngắn nhất trong 2 kim còn lại

# Sắp xếp các kim còn lại theo chiều dài tăng dần

clock\_hands = sorted(clock\_hands, key=lambda x: np.linalg.norm(np.array(x[0][:2]) - np.array(x[0][2:])))

hour\_hand = clock\_hands[0][0]

minute\_hand = clock\_hands[1][0]

* Vẽ hình chữ nhật xung quanh 3 kim đồng hồ với mỗi màu riêng bằng cv2.rectangle()

# Vẽ kim giờ màu đỏ, kim phút xanh dương và kim giây xanh lá

clock\_image = cut\_image.copy()

x1, y1, x2, y2 = hour\_hand

cv2.line(clock\_image, (x1, y1), (x2, y2), (0, 0, 255), 2)

x1, y1, x2, y2 = minute\_hand

cv2.line(clock\_image, (x1, y1), (x2, y2), (255, 0, 0), 2)

x1, y1, x2, y2 = second\_hand

cv2.line(clock\_image, (x1, y1), (x2, y2), (0, 255, 0), 2)

**1.3 Kết quả**

**Bước 1:** Xử lý ảnh để dễ nhận biết các đối tượng



**Bước 2:** Tìm hình tròn lớn nhất, cắt ảnh theo hình tròn lớn nhất và resize về kích cỡ chung 500x500



**Bước 3:** Tìm 3 kim đồng hồ

* Áp dụng phương pháp Canny Edge Detection để phát hiện các cạnh trong ảnh
* Tìm các đường thẳng sử dụng Hough Line Transform



* Xác định tâm của đồng hồ là (250,250) vì ảnh luôn được cắt theo 500x500
* Tìm các đường thẳng gần tâm bằng cách tính toán góc giữa các đoạn thẳng và tâm đồng hồ



* Nhóm các đường thẳng gần song song lại thành các nhóm riêng



* Loại bỏ các đường thẳng không nằm trong tâm, quá xa so với nhóm, giữ lại các nhóm có đoạn thẳng dài nhất (loại bỏ đuôi của kim đồng hồ)



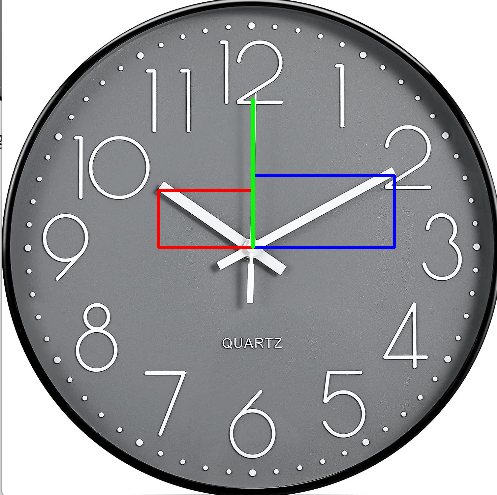
* Tìm ra hình dạng của 3 kim đồng hồ dựa vào 3 nhóm đường thẳng trên gồm chiều dài và độ dày



* Xác định chính xác 3 kim đồng hồ



* Vẽ hình chữ nhật xung quanh 3 kim đồng hồ với mỗi màu riêng bằng cv2.rectangle()

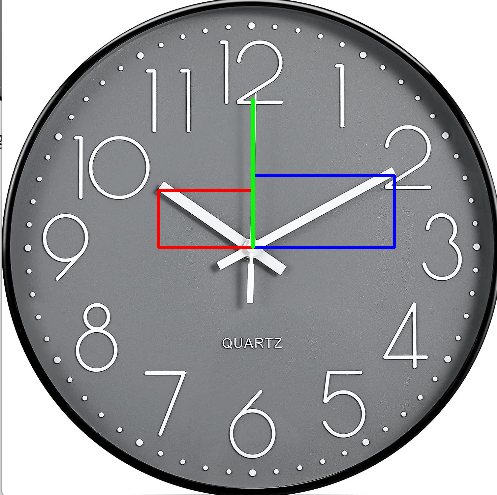
****

CHƯƠNG 2 – Tính thời gian hiện tại của đồng hồ và vẽ lên ảnh

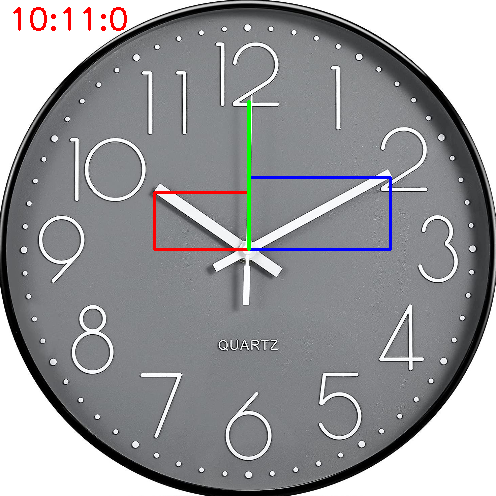
2.1 Thuật toán

2.1.1 Yêu cầu

Đầu vào: Ảnh đồng hồ với các hình chữ nhật được vẽ quanh các kim đồng hồ

****

Đầu ra: Ảnh đồng hồ với số giờ được ghi trên đồng hồ hours:minutes:seconds



2.1.2 Thuật toán sử dụng

**Bước 1:** Tính góc của kim đồng hồ

* Xác định vector của kim đồng hồ và vector có phương thẳng đứng
* Tính cos của góc bằng tích vô hướng giữa 2 vector chia cho tích độ dài 2 vector
* Dùng arccosine tính góc từ giá trị cos. Sau đó đổi từ radian sang độ
* Xác định hướng của góc bằng tích có hướng giữa 2 vector
  + Nếu tích có hướng > 0 thì lấy góc bù
  + Ngược lại lấy góc đã tính được

**Bước 2:** Tính thời gian của kim đồng hồ

* Tính thời gian của từng kim đồng hồ dựa theo góc
  + Với kim giờ: Một vòng đồng hồ được chia thành 12 phần, do đó mỗi góc đơn vị của kim giờ là 360/12 = 30 độ. Vì vậy, để tính thời gian, góc của kim giờ được chia cho 30.
  + Với kim phút và giây: Một vòng đồng hồ được chia thành 60 phút hoặc giây, do đó mỗi góc đơn vị của kim phút hoặc giây là 360/60 = 6 độ. Vậy nên, góc của các kim phút và giây được chia cho 6 để tính thời gian.
* Làm tròn thời gian bằng hàm round()
* Xử lý trường hợp đặc biệt
  + Nếu giây hoặc phút sau khi làm tròn bằng 60, tức là đã đạt đến 60 phút hoặc 60 giây, thì chúng được làm tròn về 0 vì thời gian không thể vượt quá 59 phút hoặc 59 giây

**Bước 3:** Ghi thời gian lên ảnh và lưu ảnh

* Ghi thời gian lên ảnh với hàm cv2.putText()
* Lưu ảnh lại vào thư mục

**2.2 Phương pháp và hiện thực**

**Bước 1:** Tính góc của kim đồng hồ

# Tính góc cho các kim đồng hồ dựa theo tâm của đồng hồ

# Tính góc của kim so với vector chỉ hướng lên trên

def get\_angle(hand):

    # u là vector của kim đồng hồ

    x1, y1, x2, y2 = hand

    # Sắp xếp lại để vector đi từ tâm đồng hồ (250, 250)

    # Tìm xem đầu nào gần tâm đồng hồ nhất

    if np.linalg.norm(np.array([x1, y1]) - np.array([250, 250])) < np.linalg.norm(np.array([x2, y2]) - np.array([250, 250])):

        x1, y1, x2, y2 = x2, y2, x1, y1

    u = [x2 - x1, y2 - y1]

    # v là vector chỉ hướng lên trên

    v = [0, 100]

    # Tính tích vô hướng của 2 vector

    dot\_uv = u[0] \* v[0] + u[1] \* v[1]

    # Tính độ dài của 2 vector

    length\_u = math.sqrt(u[0]\*\*2 + u[1]\*\*2)

    length\_v = math.sqrt(v[0]\*\*2 + v[1]\*\*2)

    # Tính cos của góc giữa 2 vector u.v / (|u| \* |v|)

    cos\_theta = dot\_uv / (length\_u \* length\_v)

    # Đảm bảo cos\_theta nằm trong khoảng [-1, 1]

    cos\_theta = max(min(cos\_theta, 1.0), -1.0)

    # Tính góc giữa 2 vector

    theta = math.acos(cos\_theta)

    # Chuyển radian sang độ

    theta\_degrees = math.degrees(theta)

    # Tính tích có hướng của 2 vector

    cross\_uv = u[0] \* v[1] - u[1] \* v[0]

    # Nếu tích có hướng > 0 thì góc là góc bù của theta

    if cross\_uv > 0:

        # Trả về góc bù của theta

        return 360 - theta\_degrees

    else:

        return theta\_degrees

# Tính góc cho các kim đồng hồ

hour\_angle = get\_angle(hour\_hand)

minute\_angle = get\_angle(minute\_hand)

second\_angle = get\_angle(second\_hand)

print(f"hour\_angle: {hour\_angle}")

print(f"minute\_angle: {minute\_angle}")

print(f"second\_angle: {second\_angle}")

**Bước 2:** Tính thời gian của kim đồng hồ

# Tính thời gian của đồng hồ

hour\_time = hour\_angle / 30

minute\_time = minute\_angle / 6

second\_time = second\_angle / 6

rounded\_hour = round(hour\_time)

rounded\_minute = round(minute\_time)

rounded\_second = round(second\_time)

# nếu phút hoặc giây bằng 60 thì làm tròn thành 0

if rounded\_minute == 60:

    rounded\_minute = 0

if rounded\_second == 60:

    rounded\_second = 0

print(f"Thời gian: {rounded\_hour}:{rounded\_minute}:{rounded\_second}")

**Bước 3:** Ghi thời gian lên ảnh và lưu ảnh

* Ghi thời gian lên ảnh với hàm cv2.putText()
* Lưu ảnh lại vào thư mục

# Viết thời gian lên ảnh

    font = cv2.FONT\_HERSHEY\_SIMPLEX

    cv2.putText(clock\_image, f"{rounded\_hour}:{rounded\_minute}:{rounded\_second}", (10, 30), font, 1, (0, 0, 255), 2)

# Lưu ảnh với thời gian vào thư mục output

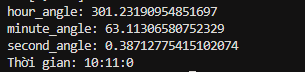
output\_filename = os.path.basename(img\_path)

output\_path = os.path.join(outputpath, output\_filename)

cv2.imwrite(output\_path, clock\_image)

**2.3 Kết quả**

1. Log giá trị của các góc kim đồng hồ tính được và thời gian tương ứng



2. Ảnh sau khi được ghi thời gian lên



**TÀI LIỆU THAM KHẢO**

1. <https://github.com/trietcs/clock-time-extraction-opencv>