BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC THĂNG LONG – KHOA CÔNG NGHỆ THÔNG TIN

---o0o---

BÀI TẬP LỚN

MÔN: THỊ GIÁC MÁY TÍNH

CHẤM ĐIỂM THI THPT QUỐC GIA

GIẢNG VIÊN HƯỚNG DẪN: DƯƠNG VĂN LẠC

SINH VIÊN THỰC HIỆN : A42304 NGUYỄN VĂN HIẾU
A42993 HÀ NGỌC TRƯỜNG
A42048 NGUYỄN ĐỨC THÁI

Học kì: Kì 1 nhóm 2 năm 2023-2024

Hà Nội, 10/2023

Lời mở đầu

Sau một học kỳ của môn "Thị giác máy tính" được giảng viên Dương Văn Lạc hướng dẫn. Bài tập lớn là bài tập cuối kỳ đánh dấu sự kết thúc quá trình hoc tập và rèn luyện các kiến thức cơ bản, đồng thời mở ra con đường thực tế áp dụng vào các môn học trong kỳ học tới. Quá trình làm bài tập lớn giúp em thu thập, tổng hợp lại các kiến thức đã học trong suốt 9 tuần qua, qua đó rèn luyện khả năng tính toán và giải quyết các vấn đề thực tế. Trong quá trình làm bài tập lớn này nhóm em đã gặp không ít những khó khăn và trở ngại do vốn kiến thức còn hạn chế. Dù bận rộn rất nhiều công việc nhưng thầy Lạc vẫn giành nhiều thời gian và tâm huyết trong việc hướng dẫn nhóm em. Thầy Lạc luôn quan tâm, chỉ bảo và sửa chữa những vấn đề quan trọng giúp em định hướng và làm việc theo quan điểm đúng đắn, chính sự tận tâm và nhiệt huyết của thầy đã giúp cho nhóm em có được tinh thần, một niềm tin và khối lượng kiến thức phong phú để đến ngày hôm nay bài tập lớn của nhóm em đã được hoàn thành. Với tất cả tấm lòng biết ơn sâu sắc, nhóm em xin chân thành gửi lời cảm ơn đến thầy đã chân tình hướng dẫn - giúp đỡ nhóm em trong suốt quá trình học tập trong môn học.

Trong quá trình làm bài tập lớn, nhóm em xin cam đoan: Những nội dung trong bài tập lớn là do chúng em trực tiếp nghiên cứu, thực hiện dưới sự hướng dẫn của thầy Dương Văn Lạc. Mọi tham khảo trong bài tập đều được trích dẫn rõ ràng về tên tác giả, tên công trình, thời gian và địa điểm.

Chân thành cảm ơn!

Mục Lục

CHƯƠNG 1. PHÂN CÔNG CÔNG VIỆC

1.1. Phân công công việc

XỬ LÝ ẢNH	A42993 HÀ NGỌC TRƯỜNG A42048 NGUYỄN ĐỨC THÁI
GIAO DIỆN NGƯỜI DÙNG	A42304 NGUYỄN VĂN HIẾU
VIẾT BÁO CÁO, SLIDE	A42304 NGUYỄN VĂN HIẾU A42048 NGUYỄN ĐỨC THÁI

1.2. Đánh giá mức tham gia hoàn thành dự án

Dù đang đối mặt với áp lực cuối kỳ học và đòi hỏi nhiều thời gian cho việc ôn tập thi, tôi rất tự hào về sự tích cực và sáng tạo của tất cả các thành viên trong nhóm. Chúng tôi đã cùng nhau đưa ra ý kiến, xây dựng và hoàn thiện bài tập dự án.

Là người đứng đầu nhóm, tôi, Nguyễn Đức Thái, cam kết một cách mạnh mẽ rằng tất cả các thành viên trong nhóm sẽ tham gia đầy đủ và hoàn thành nhiệm vụ của mình một cách xuất sắc. Chúng tôi sẽ làm việc chặt chẽ và hỗ trợ lẫn nhau để đảm bảo chất lượng và thành công của dự án.

CHƯƠNG 2. GIỚI THIỆU CHUNG

2.1. Mô tả khái quát mục tiêu bài toán

Bài thi Trung học phổ thông (THPT) Quốc Gia là một sự kiện quan trọng trong hệ thống giáo dục của một quốc gia. Hàng năm, hàng triệu học sinh tham gia vào bài thi này để đánh giá kiến thức và khả năng của họ. Để đảm bảo tính công bằng và hiệu quả trong việc chấm điểm các bài thi, việc sử dụng thị giác máy tính để tự động hóa quá trình chấm điểm đã trở thành một ưu tiên. Mục tiêu chính của đề tài này là phát triển một hệ thống chấm điểm tự động cho bài thi THPT Quốc Gia bằng sử dụng thị giác máy tính. Hệ thống này sẽ có khả năng chấm điểm các bài thi viết và bài làm trên giấy dựa trên hình ảnh.

2.2. Một số nghiên cứu liên quan

1. Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR).

2. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems (NIPS).

3. OpenCV, hay Open Source Computer Vision Library, là một thư viện mã nguồn mở phát triển dành cho thị giác máy tính và xử lý hình ảnh. OpenCV cung cấp một loạt các công cụ và thư viện để xử lý và phân tích hình ảnh số, cũng như thực hiện các tác vụ liên quan đến thị giác máy tính, như nhận diện khuôn mặt, xác định đối tượng, xử lý video, và nhiều ứng dụng khác.

CHƯƠNG 3. CƠ SỞ LÝ THUYẾT

3.1. Các khái niệm cơ bản

- Ảnh: Gồm hai loại ảnh chính
  + Ảnh màu: Mỗi điểm ảnh trong ảnh màu có thông tin về màu sắc, thường được biểu diễn bằng ba kênh màu (Red, Green, Blue - RGB). Mỗi kênh chứa thông tin về màu tại mỗi điểm ảnh.
  + Ảnh xám: Ảnh xám chỉ chứa thông tin về mức xám (độ sáng) tại mỗi điểm ảnh. Nó không chứa thông tin về màu sắc và thường được biểu diễn bằng giá trị từ 0 (đen) đến 255 (trắng), hoặc trong một dải giá trị xám.
- Chuyển đổi Grayscale: nó giúp giảm số lượng thông tin mà bạn phải xử lý và tập trung vào các đặc trưng quan trọng của ảnh. Thông thường, việc chuyển đổi này được thực hiện bằng cách tính trung bình của các kênh màu (đỏ, xanh lá, và lam) tại mỗi điểm ảnh để tạo ra giá trị màu xám tương ứng.
- Làm mịn hình ảnh (Image Smoothing): việc làm mịn hình ảnh bằng cách sử dụng một hạ bậc thấp (low-pass) bộ lọc kernel. Điều này hữu ích để loại bỏ nhiễu. Thực chất, nó loại bỏ nội dung tần số cao (ví dụ: nhiễu, cạnh) khỏi hình ảnh. Vì vậy, các đường biên sẽ bị làm mờ một chút trong quá trình này (có cũng có các kỹ thuật làm mờ không làm mờ các đường biên).
- Xóa viền hình ảnh: được sử dụng để loại bỏ phần viền không mong muốn hoặc không cần thiết từ một hình ảnh. Điều này thường được thực hiện để làm sạch và tập trung vào phần chính của hình ảnh.
- Cắt ảnh: là quá trình loại bỏ một phần của hình ảnh gốc để chỉ giữ lại một phần cụ thể hoặc tạo ra một phần riêng biệt của hình ảnh. Quá trình này thường được thực hiện bằng cách xác định vùng cắt và sau đó trích xuất phần ảnh nằm trong vùng đó. Cắt ảnh là một phần quan trọng trong xử lý hình ảnh để tập trung vào các đối tượng hoặc phần cụ thể của hình ảnh và loại bỏ thông tin không cần thiết.

3.2. Các thư viện được sử dụng

- Numpy: : là một thư viện cho tính toán khoa học và tính toán số học với Python. NumPy cho phép bạn thao tác với các ma trận, mảng và phân tích dữ liệu số học với tốc độ nhanh hơn so với các phương pháp thông thường.
- Opencv: là một thư viện mã nguồn mở chuyên về thị giác máy tính và xử lý ảnh. Nó cung cấp nhiều công cụ và thuật toán mạnh mẽ để thực hiện xử lý ảnh, phân tích hình ảnh, và thực hiện các tác vụ thị giác máy tính. OpenCV được viết bằng C++ nhưng cung cấp các liên kết cho nhiều ngôn ngữ, bao gồm Python.
- Imutils: là một thư viện Python nhằm đơn giản hóa và cải thiện quá trình xử lý và hiển thị hình ảnh khi sử dụng OpenCV. Thư viện này không thay thế OpenCV mà chỉ bổ sung và cung cấp một số tiện ích hữu ích để giúp bạn làm việc với hình ảnh dễ dàng hơn. imutils cung cấp một số tính năng quan trọng sau
- Tkiner: là một thư viện chuẩn để xây dựng giao diện người dùng đồ họa (GUI). Tkinter là một phần của thư viện chuẩn của Python, nên không cần cài đặt thêm bất kỳ phần mềm bên ngoài nào.
- PIL: (Python Imaging Library) là một thư viện Python cho xử lý hình ảnh, cho phép bạn tạo, chỉnh sửa và xử lý hình ảnh với đa dạng định dạng.

CHƯƠNG 4. MÔ TẢ DỮ LIỆU

4.1. Dữ liệu ban đầu (Ảnh)

Dữ liệu đầu vào là một ảnh bài thi của bất kì một học sinh nào thi THPT quốc gia.

Ảnh gồm các bài thi:

+ Toán: 50 câu
+ Anh: 40 câu
+ Các môn tổ hợp 120 câu.

4.2. Xử lý Ảnh

4.2.1. Lọc các đối tượng

Trước tiên chúng tôi viết một hàm getContours được sử dụng để xác định và lọc các đối tượng trong một hình ảnh. Dưới đây là mô tả chi tiết cách hoạt động của hàm:

+ Đầu tiên hình ảnh đước truyền vào sẽ được chuyển thành màu xám.
+ Sau đó nó sẽ được làm mịn bằng cách làm mờ bằng gaussian để giảm nhiễu.
+ Sau đó hình ảnh làm mịn được biến đổi thành hình ảnh Canny để tìm biên của các đối tượng.
+ Hàm cv2.dilate và cv2.erode được sử dụng để làm sáng và loại bỏ nhiễu trong hình ảnh Canny.
+ Hàm cv2.findContours tìm các contours trong hình ảnh đã được xử lý và lưu chúng trong biến contours.
+ Tiếp theo, hàm xác định các contours hợp lệ dựa trên diện tích và số cạnh.
+ Danh sách contours hợp lệ được sắp xếp theo diện tích giảm dần.
+ Hàm trả về hình ảnh gốc và danh sách contours hợp lệ.

Chức năng này có thể được sử dụng để xác định và lọc các đối tượng trong hình ảnh dựa trên diện tích và số cạnh của chúng.

Dưới đây là đoạn mã của hàm getContours:

```python
def getContours(img,cThread=[100,100],minArea=1000, filter=4):
    imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    imgBlur = cv2.GaussianBlur(imgGray, (5, 5), 1)
    imgCanny = cv2.Canny(imgBlur,cThread[0],cThread[1])
    kernel = np.ones((5,5))
    imgDilation = cv2.dilate(imgCanny, kernel, iterations = 3)
    imgThre = cv2.erode(imgDilation, kernel, iterations = 2)

    contours, _ = cv2.findContours(imgThre, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    final_countours = []
    for contour in contours:
        area = cv2.contourArea(contour)
        if area > minArea:
            peri = cv2.arcLength(contour,True)
            approx = cv2.approxPolyDP(contour,0.02*peri,True)
            bbox = cv2.boundingRect(approx)
            if filter > 0:
                if len(approx) == filter:
                    final_countours.append([len(approx),area,approx,bbox,contour])
            else:
                final_countours.append([len(approx),area,approx,bbox,contour])
    final_countours = sorted(final_countours, key = lambda x:x[1], reverse=True)

    return img, final_countours
```

Hình 4.1. Hàm getContours

4.2.2. Cắt hình ảnh

Chúng tôi viết hàm ‘wrapImage’ để thực hiện phép biến đổi (warp) trên hình ảnh đầu vào img dựa trên các điểm (contours) trong danh sách points và kích thước đích (widthImg và heightImg). Hàm này thường được sử dụng để thực hiện việc cắt (crop) và biến đổi hình ảnh để chỉ còn lại một phần cụ thể của hình ảnh gốc, và loại bỏ lề (padding) từ hình ảnh đích nếu được chỉ định. Hàm cụ thể như sau:

+ `img`: Đây là hình ảnh đầu vào mà bạn muốn biến đổi.
+ `points`: Đây là danh sách (list) chứa các điểm (contour) sử dụng để xác định phép biến đổi. Mã yêu cầu rằng danh sách này phải chứa chính xác 4 điểm.
+ `widthImg` và `heightImg`: Đây là kích thước mục tiêu (width và height) cho hình ảnh đầu ra sau biến đổi. Hình ảnh sẽ được biến đổi để có kích thước này.
+ `pad`: Đây là một tham số tùy chọn để xác định phạm vi của lề (padding) được loại bỏ khỏi hình ảnh đầu ra. Giá trị mặc định là 0, tức là không có lề.

+ Trước hết, hàm gọi `get_4_contour(points)` để đảm bảo rằng danh sách `points` chứa chính xác 4 điểm. Điều này đảm bảo rằng phép biến đổi sẽ được thực hiện với 4 điểm duy nhất.

+ Tiếp theo, hàm tạo hai mảng numpy `pts1` và `pts2`. `pts1` chứa tọa độ của 4 điểm trong `points`, và `pts2` chứa 4 điểm với các tọa độ được xác định bởi `[[0, 0], [widthImg, 0], [0, heightImg], [widthImg, heightImg]]`. Các điểm trong `pts1` và `pts2` sẽ được sử dụng để xác định phép biến đổi. Chức năng chính của hàm này là thực hiện phép biến đổi hình ảnh dựa trên các điểm xác định, và sau đó cắt bỏ lề (padding) nếu được chỉ định.

+ Hàm `cv2.getPerspectiveTransform(pts1, pts2)` sử dụng để tính toán ma trận biến đổi (perspective transformation matrix) dựa trên tọa độ của các điểm trong `pts1` và `pts2`. Ma trận này sẽ được sử dụng để biến đổi hình ảnh.

+ Hàm `cv2.warpPerspective` được sử dụng để thực hiện phép biến đổi của hình ảnh đầu vào `img` bằng cách sử dụng ma trận biến đổi `matrix`. Kết quả là hình ảnh đã biến đổi với kích thước mới.

+ Sau đó, dòng mã `wrap = wrap[pad:wrap.shape[0]-pad, pad:wrap.shape[1]-pad]` được sử dụng để loại bỏ lề (padding) từ hình ảnh đầu ra, nếu `pad` lớn hơn 0.

Với hàm get_4_contour là hàm nhận vào một danh sách points chứa các điểm (contours), và mục đích của hàm là tìm và trả về một mảng chứa 4 điểm đại diện cho 4 điểm góc cụ thể của hình ảnh hoặc vùng cần quan tâm

Dưới đây là hình ảnh của hàm wrapImage và get_4_contour:

```
def wrapImage(img, points, widthImg, heightImg, pad = 0):
    points= get_4_contour(points)
    pts1 = np.float32(points)
    pts2 = np.float32([[0, 0],[widthImg, 0], [0, heightImg],[widthImg, heightImg]]) # PREPARE POINTS FOR WARP
    matrix = cv2.getPerspectiveTransform(pts1, pts2)
    wrap = cv2.warpPerspective(img, matrix, (widthImg, heightImg))
    wrap = wrap[pad:wrap.shape[0]-pad,pad:wrap.shape[1]-pad]
    return wrap
```

Hình 4.2. Hàm wrapImage

```python
def get_4_contour(points):
    center = np.mean(points, axis=0).astype(int)

    points_above_center = np.array([point.squeeze() for point in points if point.squeeze()[1] < center[0][1]])
    points_below_center = np.array([point.squeeze() for point in points if point.squeeze()[1] >= center[0][1]])

    top_left = points_above_center[np.argmin(points_above_center[:, 0])]
    top_right = points_above_center[np.argmax(points_above_center[:, 0])]
    botton_left = points_below_center[np.argmin(points_below_center[:, 0])]
    botton_right = points_below_center[np.argmax(points_below_center[:, 0])]
    return np.array([[top_left], [top_right], [botton_left],[botton_right]])
```

Hình 4.3. Hàm get_4_contour

CHƯƠNG 5. MÔ HÌNH HÓA BÀI TOÁN

5.1. Mô tả bài toán

Bài toán là một đề tài ứng dụng thị giác máy tính để giải quyết việc chấm điểm bài thi THPT Quốc gia một các công bằng cho hàng triệu thí sinh. Trong đó đầu vào là ảnh của một bài thi đã được các thí sinh tích vào các ô đáp án và đầu ra là kết quả làm bài của thí sinh đó đạt được.

Để giải quyết bài toán này ta sử dụng các phương pháp xử lý ảnh và ứng dụng thư viện OpenCv để nhận dạng các đáp án, số báo danh và mã đề thi của thí sinh đã tô. Sau đó so sánh với đáp án đã được nhập trước đó rồi trả ra điểm số của thí sinh, số báo danh và mã đề thi của thí sinh đó.

5.2. Xây dựng mô hình

Sau khi đã xây dựng các hàm liên quan đến việc xử lý ảnh thì chúng tôi bắt đầu xây dựng mô hình giải quyết bài toán chấm điểm thi THPTQG.

5.2.1. Cắt vùng phiếu

- findFullAnswerSheet(pathImage, width=1830, height=2560): Hàm này nhận một đường dẫn đến một hình ảnh (pathImage) và tạo ra một ảnh mới bằng cách cắt và biến đổi hình ảnh ban đầu để tạo ra ảnh vùng phiếu trả lời đầy đủ. Hàm sử dụng các hàm getContours và get_4_contour để xác định và sắp xếp lại các điểm góc của vùng phiếu trả lời. Kết quả là ảnh vùng phiếu trả lời đầy đủ với kích thước mục tiêu là width và height.
- get_test_code_image(image): Hàm này nhận một ảnh vùng phiếu trả lời (image) và trả về một ảnh con chứa phần vùng chứa mã đề thi. Hàm này cắt ảnh gốc theo tỷ lệ cố định để lấy phần mã đề thi.
- get_student_code_image(image): Hàm này tương tự như get_test_code_image, nhưng nó trả về ảnh con chứa phần vùng chứa mã số học sinh.
- get_sheet_ans_image(image): Hàm này trích xuất và trả về phần vùng chứa các câu trả lời trong hình ảnh vùng phiếu trả lời. Nó cắt và trả về phần chứa các câu trả lời.

– get_part_sheet_ans_image(image, dis=10): Hàm này nhận một ảnh và thực hiện phân chia hình ảnh thành 4 phần tương ứng với các góc của vùng câu trả lời. Tham số dis được sử dụng để xác định khoảng cách giữa các phần.

Các hàm này được sử dụng để xác định và cắt các phần khác nhau của vùng phiếu trả lời từ một hình ảnh đầu vào, giúp tiện lợi trong việc trích xuất thông tin từ phiếu trả lời trong ứng dụng xử lý ảnh.

Dưới đây là hình ảnh mô tả các hàm cho việc xử lý cắt các vùng ảnh

```python
#cắt vùng phiếu
def findFullAnswerSheet(pathImage, width =1830, height =2560):
    img = cv2.imread(pathImage)
    _, countours = sb.getContours(img,minArea=300000)
    points_test_paper = sb.get_4_contour(countours[0][2])
    return sb.wrapImage(img, points_test_paper, width, height)

def get_test_code_image(image):
    height, width, channels = image.shape
    per_width = [0.876, 0.94]
    per_height = (0.0975, 0.298)
    return image[int(per_height[0]*height):int(per_height[1]*height),\
                 int(per_width[0]*width):int(per_width[1]*width)]
def get_student_code_image(image):
    height, width, channels = image.shape
    per_width = (0.727, 0.845)
    per_height = (0.0975, 0.298)
    return image[int(per_height[0]*height):int(per_height[1]*height),\
                 int(per_width[0]*width):int(per_width[1]*width)]
def get_sheet_ans_image(image):
    height, width, channels = image.shape
    per_height = (0.326, 0.945)
    per_width = (0.055, 0.925)
    full_sheet = image[int(per_height[0]*height):int(per_height[1]*height),\
                 int(per_width[0]*width):int(per_width[1]*width)]
    return full_sheet
def get_part_sheet_ans_image(image,dis=10):
    height, width, channels = image.shape
    dis = 0.03562 * width
    A, B = image[:,:int(round((width+dis)/2 - dis,0))], image[:,int(round((width+dis)/2,0)):]
    A1,A2 = A[:,:int(round((A.shape[1]+dis)/2 - dis,0))], A[:,int(round((A.shape[1]+dis)/2,0)):]
    B1,B2 = B[:,:int(round((B.shape[1]+dis)/2 - dis,0))], B[:,int(round((B.shape[1]+dis)/2,0)):]
    return [A1,A2,B1,B2]
```

Hình 5.1. Các hàm xử lý việc cắt ảnh

5.2.2. Lấy mã sinh viên

Hàm get_student_code(img_sbd, thresh_value = 150):

– img_sbd: Đây là hình ảnh chứa số báo danh của học sinh.

- thresh_value: Ngưỡng (threshold) để chuyển hình ảnh thành ảnh nhị phân. Giá trị mặc định là 150.
- Hàm này có các bước chính:
  + Chuyển hình ảnh thành hình ảnh xám.
  + Áp dụng Gaussian Blur để làm mịn hình ảnh.
  + Xoá viền của hình ảnh bằng cách sử dụng phép dilation và erosion.
  + Chuyển hình ảnh thành ảnh nhị phân, trong đó giá trị dưới ngưỡng thresh_value được đặt thành 0 (đen), và ngược lại được đặt thành 255 (trắng).
  + Sau đó, hình ảnh được chia thành 6 cột bằng cách lấy tổng số cột và chia cho 6. Các vị trí của ô tròn trong cột được xác định bằng cách lấy tổng số dòng và chia cho 10, tạo ra một danh sách bubble_positions chứa vị trí dòng của các ô tròn.

Hàm sau đó duyệt qua từng cột của hình ảnh, và trong mỗi cột, tìm dòng (số từ 0 đến 9) mà có giá trị trung bình (mean) của ô tròn nhỏ nhất (tức là ô tròn đó chứa nhiều màu đen nhất). Giá trị này được thêm vào chuỗi student_ID làm mã số học sinh. Sau khi duyệt qua tất cả các cột, hàm trả về mã số học sinh

Dưới đây là hình ảnh đoạn mã của hàm lấy mã sinh viên của từng thí sinh

```python
def get_student_code(img_sbd, thresh_value = 150):
    # Xử lý ảnh
    gray = cv2.cvtColor(img_sbd, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    # Xoá viền
    kernel = np.ones((5,5))
    imgDilation = cv2.dilate(blur, kernel, iterations = 2)
    # Phóng to phần tô màu
    erosion = cv2.erode(imgDilation, kernel, iterations = 2)
    # Chuyển thành giá trị nhị phân, giá trị nào dưới 100 về 0, ngược lại 255 (đen và trắng)
    _, thresh = cv2.threshold(erosion, thresh_value, 255, cv2.THRESH_BINARY)

    # cv2_imshow(thresh)
    column_width = img_sbd.shape[1] // 6
    bubble_positions = [int(i*(img_sbd.shape[0]//10)) for i in range(11)]

    student_ID = ""
    for i in range(6): # Có 6 cột số báo danh
        column = thresh[:,i*column_width:(i+1)*column_width]
        selected_number = None
        min_mean = float('inf')

        for pos in range (10): # Có 10 dòng từ 0-9
            start, end = bubble_positions[pos], bubble_positions[pos+1]
            mean_value = np.mean(column[start:end, :])
            # Tìm giá trị trung bình nhỏ nhất
            # Nhỏ nhất là chứa nhiều phần tử màu đen nhất, tức chọn ô này
            if mean_value < min_mean:
                min_mean = mean_value
                selected_number = pos
        student_ID += str(selected_number)
    return student_ID
```

Hình 5.2. Hàm lấy mã sinh viên

5.2.3. Lấy mã đề thi

Hàm này hoàn toàn tương tự với get_student_code, nhưng được sử dụng để nhận dạng và trích xuất mã đề thi từ hình ảnh. Và nó được chia thành 3 cột thay vì 6.

Chức năng của cả hai hàm là trích xuất thông tin số học sinh và mã đề thi từ hình ảnh dựa trên sự tương quan giữa các ô tròn và nền của hình ảnh.

Dưới đây là hình ảnh hàm lấy mã đề thi của thí sinh

```python
def get_test_code(img, thresh_value = 150):
    # Xử lý ảnh
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    # Xoá viền
    kernel = np.ones((5,5))
    imgDilation = cv2.dilate(blur, kernel, iterations = 2)
    # Phóng to phần tô màu
    erosion = cv2.erode(imgDilation, kernel, iterations = 2)
    # Chuyển thành giá trị nhị phân, giá trị nào dưới 100 về 0, ngược lại 255 (đen và trắng
    _, thresh = cv2.threshold(erosion, thresh_value, 255, cv2.THRESH_BINARY)
    # cv2_imshow(thresh)
    column_width = img.shape[1] // 3
    bubble_positions = [int(i*(img.shape[0]//10)) for i in range(11)]

    test_ID = ""
    for i in range(3): # Có 6 cột số báo danh
        column = thresh[:,i*column_width:(i+1)*column_width]
        selected_number = None
        min_mean = float('inf')

        for pos in range (10): # Có 10 dòng từ 0-9
            start, end = bubble_positions[pos], bubble_positions[pos+1]
            mean_value = np.mean(column[start:end, :])
            # Tìm giá trị trung bình nhỏ nhất
            # Nhỏ nhất là chứa nhiều phần tử màu đen nhất, tức chọn ô này
            if mean_value < min_mean:
                min_mean = mean_value
                selected_number = pos
        test_ID += str(selected_number)
    return test_ID
```

Hình 5.3. Hàm lấy mã đề thi của từng thí sinh

5.2.4. Lấy đáp án của thí sinh

Đoạn mã bạn cung cấp là một hàm có tên get_my_ans, được sử dụng để xác định và trích xuất các đáp án từ một hình ảnh đáp án trong bài thi. Dưới đây là giải thích từng phần của mã:

- image_ans: Đây là hình ảnh chứa các đáp án trong bài thi.
- ANSWER_KEY: Đây là một chuỗi đại diện cho đáp án đúng của bài thi. Chúng ta so sánh kết quả trích xuất với ANSWER_KEY để xác định đáp án đúng và đáp án sai.
- thresh_value: Ngưỡng (threshold) để chuyển hình ảnh thành hình ảnh nhị phân. Giá trị mặc định là 150.

- limit_value: Giá trị ngưỡng để xác định nếu một ô tròn trống (không được chọn đáp án) hoặc đáp án đúng là "N". Giá trị mặc định là 240.
- start: Vị trí bắt đầu xác định các đáp án trong hình ảnh.
- Hàm get_my_ans có các bước chính:
    + Hình ảnh đáp án được chia thành các cột và được xử lý để làm mịn, xoá viền và chuyển thành hình ảnh nhị phân, tương tự như các bước xử lý đã được trình bày trong mã trước.
    + Hình ảnh được chia thành 6 phần (theo số phần của đề thi), và trong mỗi phần, các ô tròn đáp án được xác định dựa trên giá trị trung bình của các ô tròn. Đáp án được xác định bằng cách tìm ô tròn có giá trị trung bình thấp nhất. Nếu giá trị trung bình nhỏ hơn limit_value, và nhỏ hơn giá trị trung bình hiện tại, thì ô tròn đó được xem là đáp án, và kết quả được thêm vào danh sách my_answers. Nếu không có đáp án nào được tìm thấy, kết quả sẽ là "N".
    + Hàm sau đó so sánh kết quả my_answers với ANSWER_KEY để xác định đáp án đúng và đáp án sai. Nếu đáp án trùng khớp với ANSWER_KEY, thì kết quả đó được đánh dấu màu xanh lá cây trong hình ảnh.

Dưới đây là hình ảnh hàm lấy đáp án của thí sinh:

```python
def get_my_ans(image_ans, ANSWER_KEY, thresh_value = 150, limit_value = 240, start = 55):
    translate = {"A": 0, "B": 1, "C": 2, "D": 3}
    revert_translate = {0: "A", 1: "B", 2: "C", 3: "D", -1: "N"}
    img = image_ans[:,start:]

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)

    # Xoá viền
    kernel = np.ones((5,5))
    imgDilation = cv2.dilate(blur, kernel, iterations = 2)
    # Phóng to phần tô màu
    erosion = cv2.erode(imgDilation, kernel, iterations = 2)
    # Chuyển thành giá trị nhị phân, giá trị nào dưới 100 về 0, ngược lại 255 (đen và trắng)
    _, thresh = cv2.threshold(erosion, thresh_value, 255, cv2.THRESH_BINARY)

    cnts, _ = cv2.findContours(thresh.copy(), cv2.RETR_LIST,
                               cv2.CHAIN_APPROX_SIMPLE)
    cnts = contours.sort_contours(cnts,
                                  method="top-to-bottom")[0]
    cv2.drawContours(img, cnts, -1, (0, 0, 255), 5)
    ans_char = ['A','B','C','D']
    height_sub = img.shape[0] // 6
    n_part, n_questions = 6,5
    pad = 5
    my_answers = []
    count = 0
    for i_part in range(n_part):
        sub_img = thresh[height_sub*i_part+pad*3:height_sub*(i_part+1)-pad*2,pad:-pad]

        bubble_positions = [int(idx*(sub_img.shape[0]//n_questions)) for idx in range(n_questions+1)]
        for j in range(n_questions):
            start, end = bubble_positions[j], bubble_positions[j+1]
            row_1_question = sub_img[start:end,:]
            min_mean = float('inf')
            selected_ans = None
            width_sub = row_1_question.shape[1] // 4
            for i_ans in range(4):
                mean_value = np.mean(row_1_question[:,i_ans*width_sub:(i_ans+1)*width_sub])
                # print(mean_value)
                # Nếu không chọn đáp án, sẽ có limit_value chặn
                if mean_value < limit_value and mean_value < min_mean:
                    min_mean = mean_value
                    selected_ans = ans_char[i_ans]
            if selected_ans is not None:
                my_answers.append(selected_ans)
                if(my_answers[i_part*5+j]==ANSWER_KEY[i_part*5+j]):
                    cv2.drawContours(img, cnts[count+1], -1, (0, 255, 0), 5)
                count = count + 1
            else:
                my_answers.append("-")

    return my_answers
```

Hình 5.4. Hàm lấy đáp án của thí sinh

CHƯƠNG 6. GIAO DIỆN NGƯỜI DÙNG

Sau khi xây dựng xong các hàm xử lý và tính điểm cho từng thí sinh chúng tôi bắt đầu vào việc xây dựng một giao diện đơn giản tiện lợi dễ dàng sử dụng. Giao diện được xây dựng bằng tkiner một thư viện đồ họa chuẩn trong python. Giao diện gồm các chức năng:

- Chức năng Upload hình ảnh:
  - Người dùng có khả năng chọn một tệp hình ảnh từ máy tính của họ thông qua nút "Select File".
  - Hình ảnh sau khi được chọn sẽ được hiển thị trong cửa sổ ứng dụng.
- Chức năng Chấm Điểm Hình Ảnh (Chức năng "Predict"):
  - Có Ba phần "Predict_nhap" và "Predict_chup" và nhập số đáp án của đề thi tương ứng với việc dự đoán từ hình ảnh đã được chọn và dự đoán từ hình ảnh đã chụp và chọn số đáp án của đề thi để tự động chấm điểm.
  - Khi người dùng nhấn vào nút "Predict_nhap" hoặc "Predict_chup," ứng dụng thực hiện các bước sau:
  - Xử lý hình ảnh đầu vào (hình ảnh đã chọn hoặc đã chụp) để trích xuất thông tin bài thi.
  - Tính điểm dựa trên thông tin đã trích xuất và so sánh với đáp án được cung cấp (ANS_KEY).
  - Hiển thị điểm số, mã số sinh viên, và mã đề thi lên hình ảnh.
  - Hiển thị hình ảnh đã được chấm điểm trong ứng dụng giao diện.
- Chức năng số đáp án tối đa:
  - Cho phép người dùng nhập vào số đáp án của bài thi đề thi.
- Lưu Kết Quả:
  - Kết quả của quá trình chấm điểm (điểm số, mã số sinh viên, mã đề thi) được lưu vào một tệp tin văn bản (score.txt hoặc tệp văn bản khác) với thông tin của các bài thi trước đó.

– Hiển Thị Kết Quả:

+ Kết quả bao gồm điểm số, mã số sinh viên, và mã đề thi được hiển thị trong cửa sổ ứng dụng để người dùng có thể xem.

Dưới đây là hình ảnh demo giao diện người dùng

Hình 6.1. HÌnh ảnh giao diện người dùng

CHƯƠNG 7. KẾT LUẬN

Trên đây là toàn bộ quá trình chúng tôi xây dựng bài toán và hướng giai quyết cho vấn đề chấm điểm tự động cho bài thi THPT quốc gia. Dưới đây đánh giá về kết quả sau khi bài toán được giải quyết:

- Ưu điểm
  - + Mô hình cho kết quả tốt, đúng
  - + Có thể chấm điểm được cho các hình ảnh bị nghiêng, mờ
- Nhược điểm
  - + Khi chụp ảnh cần chụp đủ các góc của ảnh

CHƯƠNG 8. CÁC HƯỚNG PHÁT TRIỂN TIẾP THEO

Sau khi hoàn thành dự án hiện tại, chúng ta nhận thấy có thể mở rộng ứng dụng này để cho phép người dùng chấm điểm cho bất kỳ bài thi nào, chẳng hạn như bài thi TOEIC hoặc bất kỳ bài trắc nghiệm nào khác. Điều này đòi hỏi việc thay đổi và bổ sung chức năng để hỗ trợ việc chấm điểm cho các loại bài thi khác nhau.

- Các tính năng và cải tiến có thể được thêm vào ứng dụng bao gồm:
  + Cấu hình Đáp án: Cho phép người dùng cấu hình bộ đáp án cho bài thi cụ thể. Điều này có thể bao gồm chọn loại đáp án (A, B, C, D, hoặc khác), số lượng câu hỏi, và định dạng của đáp án.
  + Giao diện Người dùng Tùy chỉnh: Cải thiện giao diện người dùng để cho phép người dùng dễ dàng tải lên hình ảnh bài thi và cấu hình cài đặt**.**
  + Hỗ trợ nhiều Định dạng Hình ảnh: Mở rộng ứng dụng để hỗ trợ nhiều định dạng hình ảnh đầu vào, chẳng hạn như JPG, JPEG, PNG và PDF.
  + Xác định Vị trí Đáp án: Tự động xác định vị trí của các ô đáp án trên hình ảnh bài thi, dựa trên các thông số người dùng cung cấp.
  + Giao diện Trực quan: Cung cấp giao diện người dùng trực quan để xác nhận và điều chỉnh vị trí của các ô đáp án trước khi chấm điểm.
  + Thống kê và Lưu trữ Kết quả: Hiển thị kết quả chấm điểm và lưu trữ lịch sử kết quả của các bài thi trước đó.
  + Bảo mật và Quản lý Người dùng: Đảm bảo tính riêng tư và bảo mật thông tin bài thi của người dùng và cung cấp khả năng quản lý tài khoản.
  + Hỗ trợ Nhiều Loại Đáp án: Hỗ trợ nhiều loại đáp án (A, B, C, D, E, F, G...) dựa trên loại bài thi cụ thể.
  + Chia sẻ Kết quả: Cho phép người dùng chia sẻ kết quả chấm điểm qua email hoặc mạng xã hội.
  + Tích hợp AI và Máy học: Nếu có thể, cải tiến bằng cách sử dụng các thuật toán AI và máy học để cải thiện độ chính xác của chấm điểm.

+ Tài liệu Hướng dẫn Sử dụng: Tạo tài liệu hướng dẫn sử dụng chi tiết để người dùng có thể dễ dàng sử dụng ứng dụng.

Các cải tiến này sẽ giúp ứng dụng trở thành một công cụ mạnh mẽ để chấm điểm cho nhiều loại bài thi và làm cho quá trình chấm điểm trở nên dễ dàng và hiệu quả hơn.