TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP GIỮA KÌ MÔN NHẬP MÔN XỬ LÝ ẢNH SỐ**

**NUMBER BOUNDING BOXES DETECTION**

*Người hướng dẫn*: **TS NGUYỄN VĂN HUY**

*Người thực hiện*: **HOÀNG KIẾN THIẾT– 51702187**

Lớp **: 17050202**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP GIỮA KÌ MÔN NHẬP MÔN XỬ LÝ ẢNH SỐ**

**NUMBER BOUNDING BOXES DETECTION**

Người hướng dẫn: **TS NGUYỄN VĂN HUY**

Người thực hiện: **HOÀNG KIẾN THIẾT**

Lớp **: 17050202**

Khoá  **: 21**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2020**

LỜI CẢM ƠN

Em xin chân thành gửi lời cảm ơn này đến thầy Nguyễn Văn Huy và cô Võ Hoàng Anh, hai giảng viên phụ trách giảng dạy bộ môn Nhập môn xử lý ảnh số. Nhờ có sự tận tình giảng dạy, truyền đạt kiến thức của quý thầy cô mà em mới đủ kiến thức để hoàn thành bài tập này.

Song song với đó, em cũng xin gửi lời cảm ơn đến Khoa Công Nghệ Thông Tin, trường Đại học Tôn Đức Thắng vì đã tạo điều kiện cho em học tập, nghiên cứu trong suốt quá trình học tập môn học này nói riêng và cả quá trình học Đại học nói chung. Một lần nữa em xin gửi lời cảm ơn chân thành đến mọi người và chúc tất cả thật nhiều sức khỏe.

**BÀI TẬP GIỮA KỲ ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là sản phẩm bài tập giữa kỳ của riêng tôi và được sự hướng dẫn của TS Nguyễn Văn Huy. Các nội dung nghiên cứu, kết quả trong bài tập giữa kỳ 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 bài tập giữa kỳ 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 bài tập giữa kỳ 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)*

*Hoàng Kiến Thiết*

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

Trong bài báo cáo giữa kỳ này em sẽ trình bày toàn bộ quá trình hoàn thành bài tập giữa kỳ cũng như kết quả cuối cùng mà em đạt được. Bài báo cáo sẽ gồm 3 phần xuyên suốt quá trình xây dựng và phát triển chương trình gồm phần giới thiệu, phần phân tích thuật toán và cuối cùng là phương hướng phát triển cho chương trình. Trong đó, trọng tâm nhất là phần trình bày thuật toán của chương trình.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc59179585)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc59179586)

[TÓM TẮT iv](#_Toc59179587)

[MỤC LỤC 1](#_Toc59179588)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 3](#_Toc59179589)

[CHƯƠNG 1 – GIỚI THIỆU 4](#_Toc59179590)

[1.1 Phân tích yêu cầu của đề bài 4](#_Toc59179591)

[1.1 Input của chương trình 4](#_Toc59179592)

[1.2 Output của chương trình 5](#_Toc59179593)

[1.2 Hướng dẫn sử dụng chương trình 5](#_Toc59179594)

[CHƯƠNG 2 – TRÌNH BÀY THUẬT TOÁN CỦA CHƯƠNG TRÌNH 6](#_Toc59179595)

[2.1 Tổng quan chương trình 6](#_Toc59179596)

[2.2 Xử lý ảnh xám và làm mượt 9](#_Toc59179597)

[2.2 Lấy ngưỡng cục bộ và tìm các thành phần liên thông 10](#_Toc59179598)

[2.3 Xử lý hình thái 12](#_Toc59179599)

[2.4 Phát hiện cạnh 14](#_Toc59179600)

[2.5 Vẽ hộp bao quanh các con số 16](#_Toc59179601)

[CHƯƠNG 3 – TỔNG KẾT 20](#_Toc59179602)

[3.1 Nhận xét 20](#_Toc59179603)

[3.1.1 Ưu điểm 20](#_Toc59179604)

[3.1.2 Khuyết điểm 20](#_Toc59179605)

[3.2 Phương hướng phát triển 21](#_Toc59179606)

**DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1. Ảnh input của chương trình 4](#_Toc59179607)

[Hình 2. Ảnh output của chương trình 5](#_Toc59179608)

[Hình 3. Sơ đồ khối của chương trình 9](#_Toc59179609)

[Hình 4. Kết quả của việc làm mờ ảnh xám 9](#_Toc59179610)

[Hình 5. Ảnh sau khi lấy ngưỡng 11](#_Toc59179611)

[Hình 6. Trước và sau khi lấy ngưỡng cho ảnh 12](#_Toc59179612)

[Hình 7. Ma trận kernel 12](#_Toc59179613)

[Hình 8. Kết quả của phép Morphology Opening 13](#_Toc59179614)

[Hình 9. Kết quả của phép Morphology closing 14](#_Toc59179615)

[Hình 10. Kết quả của Canny edged detection 16](#_Toc59179616)

[Hình 11. Ảnh sau khi vẽ các hình hộp chữ nhật 18](#_Toc59179617)

[Hình 12. Kết quả cuối cùng của chương trình 19](#_Toc59179618)

[Hình 13. Nội dung file output.txt 20](#_Toc59179619)

CHƯƠNG 1 – GIỚI THIỆU

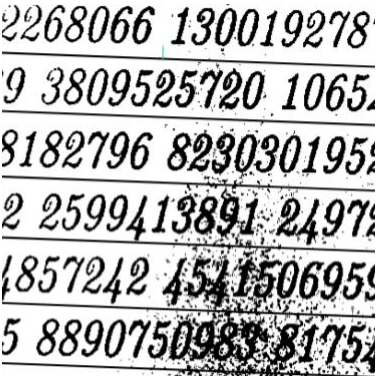
1.1 Phân tích yêu cầu của đề bài

Đề bài cụ thể được mô tả trong file MidTerm.pdf, có thể tóm tắt yêu cầu của đề bài như sau:

1.1 Input của chương trình

Cho hình ảnh các con số có màu đen trên nền trắng. Hình ảnh đầu vào có thể bị ảnh hưởng bởi sự làm nhiễu ví dụ như các thành phần bị ngắt kết nối của mỗi số, nền của ảnh lộn xộn…

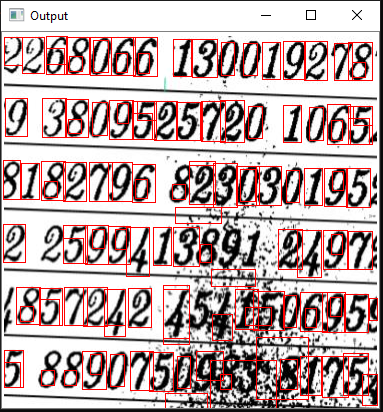
Đề bài yêu cầu viết một chương trình sử dụng Python để tìm và vẽ các hộp giới hạn của mỗi số xuất hiện trong hình ảnh đã cho. Hộp giới hạn của một số là một hình chữ nhật bao quanh hoàn toàn số đó. Cụ thể hình ảnh đầu vào của chương trình như sau:



Hình . Ảnh input của chương trình

1.2 Output của chương trình

Output của chương trình theo yêu cầu của đề bài bao gồm ảnh gốc với các hộp giới hạn số được hiển thị bằng màu đỏ (output.jpg)



Hình . Ảnh output của chương trình

Ngoài ra, output của chương trình còn bao gồm một tệp văn bản trong đó mỗi dòng bao gồm mỗi thông tin về hộp giới hạn (tọa độ trên cùng bên trái, chiều rộng và chiều cao của mỗi hộp giới hạn được phát hiện được phân tách bằng ký tự khoảng trắng (output.txt).

1.2 Hướng dẫn sử dụng chương trình

Chương trình được viết bằng ngôn ngữ Python3, sử dụng được trên hầu hết các hệ điều hành có cài đặt Python3. Đầu tiên, người dùng cần giải nén file HoangKienThiet\_51702187\_midterm.zip, trong đó bao gồm file 51702187.py chứa toàn bộ source code của chương trình; đi kèm với đó là file input.png - ảnh input của chương trình là; trong khi đó các file output.png và output.txt là hai file output của chương trình và cuối cùng là file HoangKienThiet\_51702187\_report.docx chứa nội dung của bài báo cáo này.

Để sử dụng được chương trình, máy người dùng cần phải cài đặt thư viện OpenCV và thư viện numpy. Có thể cài đặt hai thư viện trên thông qua hai câu lệnh chạy trên terminal:

pip install opencv-python3

pip install numpy

Sau khi đã cài đặt các thư viện trên, truy cập đường dẫn trực tiếp đến thư mục như file 51702187.py trên terminal. Sau đó gõ lệnh python 51702187.py và nhấn enter để triển khai chương trình. Kết quả của chương trình là output như đã trình bày ở trên.

CHƯƠNG 2 – TRÌNH BÀY THUẬT TOÁN CỦA CHƯƠNG TRÌNH

2.1 Tổng quan chương trình

Chương trình của em là file 51702187.py sử dụng Python3 dùng để xử lý ảnh input theo yêu cầu cuối cùng của đề bài. Trong chương trình có sử dụng một số hàm tiêu biểu trong xử lý ảnh của openCV như hàm đọc ảnh, làm mượt, lấy ngưỡng ảnh, chuyển đổi ảnh nhị phân, xử lý hình thái, hàm phát hiện cạnh, các hàm vẽ v.v…

Các hàm được sử dụng cụ thể sẽ được trình bày qua các bước theo mã giã dưới đây:

Input: input.png

- Bước 1: Xử lý ảnh xám và làm mượt

gray ← cvtColor(img)

blur ← GaussianBlur(gray, (5, 5), 0)

- Bước 2: Lấy ngưỡng và thành phần liên thông

thresh←adaptiveThreshold(blur)

(ret, markers) ←connectedComponents(thresh)

- Bước 3: Xử lý hình thái

convertBinary ←bitwise\_not(thresh)

kernel←getStructuringElement(cv2.MORPH\_CROSS, (3,3))

opening←morphologyEx(convertBinary,cv2.MORPH\_OPEN, kernel)

closing ← morphologyEx(opening, cv2.MORPH\_CLOSE, kernel)

- Bước 4: Phát hiện cạnh

revertBinary← bitwise\_not(closing)

edged← Canny(revertBinary)

(cnts,hier)←findContours(edged)

sorted(cnts)

- Bước 5: Vẽ boundary box

**for** c cnts **do**

**if** contourArea(c) > 100 and cv2.contourArea(c) < 500 **then**

[X, Y, W, H] ← boundingRect(c)

cv2.rectangle(img, (X, Y), (X + W, Y + H), (0,0,255), 1)

fOutput.write([X, Y, W, H])

**endif**

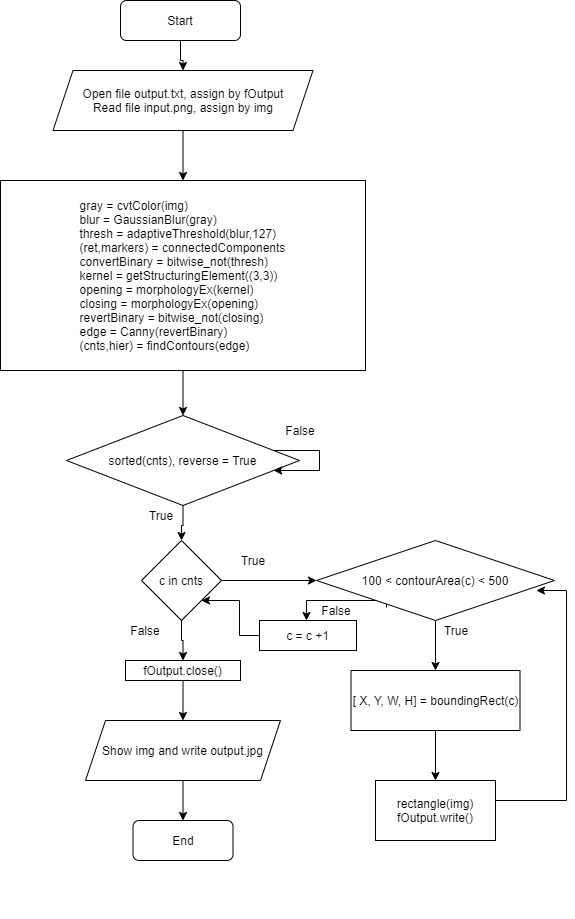
**endfor**

Output : imshow('Output',img)

imwrite('output.jpg',img)

output.txt

Từ mã giã được viết ở trên, em trình bày thuật toán của chương trình bằng sơ đồ khối bên dưới:



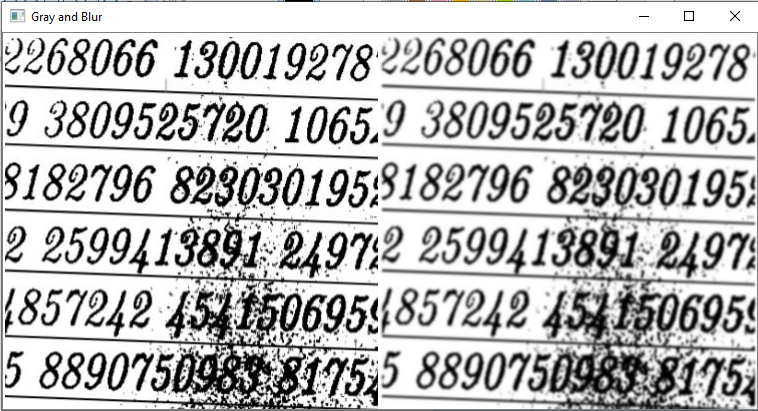
Hình . Sơ đồ khối của chương trình

2.2 Xử lý ảnh xám và làm mượt

Ảnh input của chương trình là file input.png. Đi kèm với đó là một file output.txt được tạo sẵn cho việc ghi lại tọa độ các ô vẽ bao quanh con số trong ảnh input theo yêu cầu của đề bài.

Để đọc ảnh input, em sử dụng hàm imread() trong openCV và gán kết quả trả về vào biến img. Cùng với đó em khởi tạo đối tượng fOutput để lưu kết quả mở file output.txt sử dụng hàm open() trong thư viện sys của Python dưới phân quyền ‘w’, tức là tạo một file mới để ghi, nếu file đã tồn tại thì sẽ bị ghi mới.

Sau khi đã đọc được ảnh đầu vào, em tiến hành lấy ảnh xám bằng cách sử dụng hàm cv2.cvtColor() và làm mượt ảnh sử dụng bộ lọc cv2.GausianBlur()



Hình . Kết quả của việc làm mờ ảnh xám

Mục đích của việc sử dụng hàm cv2.GaussionBlur() là để làm mượt ảnh, giảm bớt đi độ nhiễu của ảnh. Trong chương trình của mình, em chọn sử dụng bộ lọc Gausion Blur bởi đây là phương pháp làm mờ ảnh phổ biến và được sử dụng rộng rãi, hiệu quả trong quá trình tiền xử lý ảnh. Các đối tượng đầu vào của hàm này bao gồm gray – đối tượng ảnh xám của ảnh đầu vào và kernel là ma trận 5x5.

2.2 Lấy ngưỡng cục bộ và tìm các thành phần liên thông

Sau khi đã giảm nhiễu được ảnh input, em tiến hành lấy ngưỡng cho ảnh đã giãm nhiễu. Nhận xét thấy phần nền của ảnh có độ sáng tối phân bố không đồng đều. Cụ thể là góc trên bên trái của ảnh thì khá mờ, một vài số bị đứt nét, trong khi đó góc dưới bên phải của ảnh là một vùng ảnh bị nhiễu bởi các chấm đen. Ngoài ra độ đậm, nhạt của các con số cũng khác nhau nữa. Từ những phân tích trên, em chọn sử dụng hàm cv2.adaptiveThreshold lấy ngưỡng cục bộ cho ảnh để đạt hiệu quả tốt hơn sơ với việc sử dụng ngưỡng chung cho toàn bộ ảnh.

Thuật toán của cv2.adaptiveThreshold là xác định ngưỡng cho một pixel dựa trên một vùng nhỏ xung quanh nó và kết quả nhận được là các ngưỡng khác nhau cho các vùng khác nhau của cùng một hình ảnh.

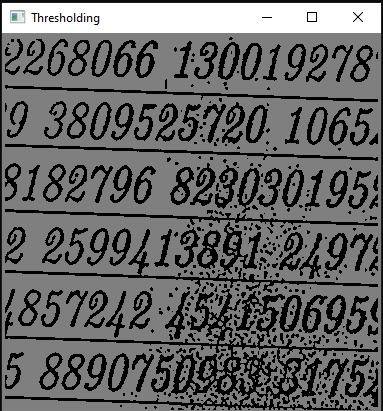
Ngoài các tham số cơ bản như đối tượng ảnh cần lấy ngưỡng, giá trị ngưỡng, phương thức cv2.adaptiveThreshold nhận ba tham số đầu vào:

- cv2.ADAPTIVE\_THRESH\_MEAN\_C: Giá trị ngưỡng là giá trị trung bình của vùng lân cận trừ đi hằng số C. Hoặc cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C: Giá trị ngưỡng là tổng trọng số gaussian của các giá trị lân cận trừ đi hằng số C. Trong chương trình em sử dụng giá trị ngưỡng này.

- BlockSize xác định kích thước của vùng lân cận và C là một hằng số được trừ đi từ tổng trung bình hoặc tổng trọng số của các pixel lân cận. Trong chương trình của mình, em chọn BlockSize là 3.

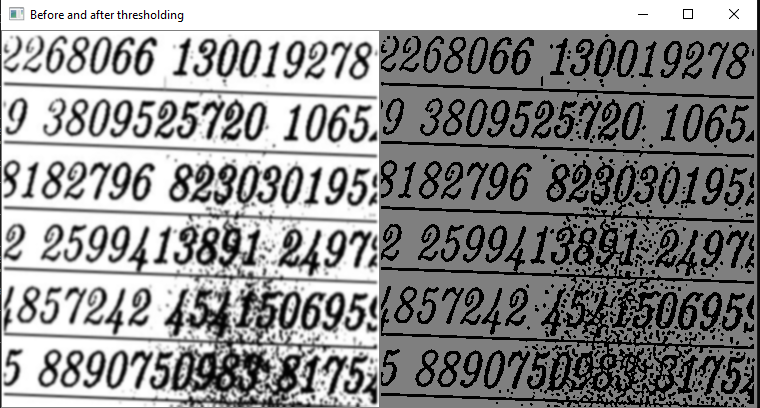
- Hằng số C được đề cập ở trên, trong chương trình của mình em chọn hằng số C bằng 2.

Sau khi lấy ngưỡng, em tiến hành tìm các thành phần liên thông. Kết quả thu được là hình ảnh bên dưới:



Hình . Ảnh sau khi lấy ngưỡng

Trước khi lấy ngưỡng và tìm các thành phần liên thông, ảnh chỉ mới được làm giảm nhiễu, khi đó các phần chấm đen vẫn còn ảnh hưởng nhiều đến các con số và các đường kẻ ngang ảnh còn đậm nét. Để cải thiện điều này thì việc lấy ngưỡng ảnh thích hợp là điều rất cần thiết. Ta có thể quan sát so sánh sau để thấy rõ hơn việc trước khi lấy ngưỡng và sau khi lấy ngưỡng thích hợp cho ảnh:

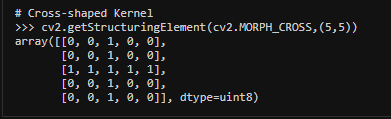


Hình . Trước và sau khi lấy ngưỡng cho ảnh

2.3 Xử lý hình thái

Để vẽ được các hình hộp bao quanh các con số được chuẩn xác nhất, giai đoạn xử lý hình thái góp phần rất quan trọng.

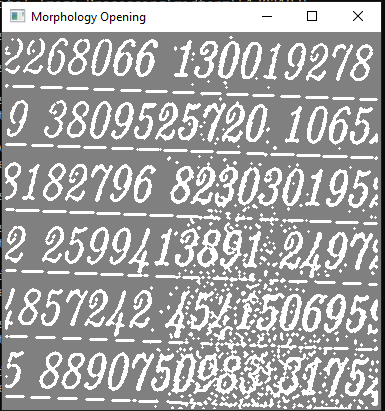
Trước khi tiến hành xử lý hình thái cho ảnh, ta cần chuẩn bị một ảnh nhị phân bằng cách sử dụng hàm cv2.bitwise\_not() và truyền vào ảnh ta đã lấy ngưỡng. Cùng với đó là một ma trận kernel 3x3 được khởi tạo bằng hàm cv2.getStructuringElement() và tham số đặc biệt là cv2.MORPH\_CROSS. Ma trận kernel có dạng như sau:



Hình . Ma trận kernel

Trên đây là ví dụ cho ma trận kernel 5x5, nhưng nhận thấy ma trận kernel 3x3 cho kết quả tốt hơn đối với ảnh input của đề bài nên em đã sử dụng ma trận kernel 3x3 để sử dụng cho việc xử lý hình thái.

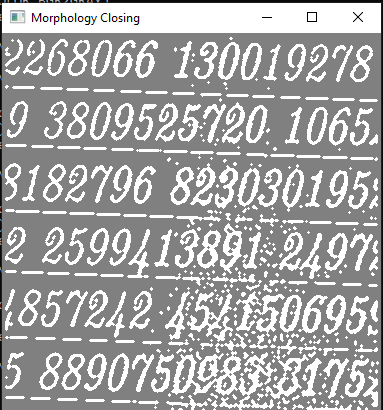
Em nhận xét thấy ảnh đầu vào có các chấm đen xung quanh các con số, do đó em sử dụng phương pháp xử lý hình thái Openning bằng cách sử dụng hàm cv2.morphologyEx() truyền vào các tham số là ảnh nhị phân, phương thức cv2.MORPH\_OPEN, và ma trận kernel đã chuẩn bị ở trên. Kết quả thu được như sau:



Hình . Kết quả của phép Morphology Opening

Sau bước xử lý hình thái này, các chấm đen gần với các con số đã được giảm bớt, còn lại các chấm đen xa các con số hơn vẫn chưa xử lý được. Cùng với đó các đường kẻ ngang đậm nét ban đầu cũng đã được làm nhỏ lại và đứt khúc hơn.

Đến đây em nhận thấy các con số vẫn còn các chấm đen chồng lên nó, từ đó em sử dụng phương pháp xử lý hình thái Closing bằng cách gọi hàm cv2.morphologyEx() truyền vào hai tham số ảnh và kernel như ở trên, chỉ khác là gọi phương thức cv2.MORPH\_CLOSE thay vì phương thức cv2.MORPH\_OPEN như ở trên. Kết quả thu được như sau:



Hình . Kết quả của phép Morphology closing

2.4 Phát hiện cạnh

Sau khi tiến hành xử lý hình thái cho ảnh, em tiến hành thuật toán tìm các đường bao quanh đối tượng phục vụ cho việc vẽ các hình hộp chữ nhật bao xung quanh các con số. Cụ thể em sử dụng hàm Canny được xây dựng sẵn trong openCV để thực hiện giai đoạn này.

Tham số đầu tiên là hình ảnh đầu vào mà ta cần xử lý, ở đây là ảnh sau khi đã thực hiện các giai đoạn tiền xử lý, lấy ngưỡng và xử lý hình thái. Vì đây là ảnh nhị phân đã được xử lý từ trước, các đối tượng cần phát hiện cạnh là số là màu trắng, còn ảnh nền là màu đen. Tuy nhiên hàm Canny mà em dùng lại thực hiện việc phát hiện cạnh đối với các đối tượng màu đen và nền ảnh trắng. Do đó trước khi truyền ảnh này vào, ta cần đảm bảo các con số có màu đen và nền ảnh là màu trắng bằng cách sử dụng hàm cv2.bitwise\_not một lần nữa để chuyển đổi màu trắng – đen cho hình ảnh.

Đối số thứ hai và thứ ba lần lượt là threshold1 và threshold2, đây là hai ngưỡng cần thiết phát hiện các cạnh bao xung quanh một đối tượng trên ảnh. Đối số tiếp theo là apertureSize. Nó là kích thước của nhân Sobel, được sử dụng để tìm độ dốc (sự thay đổi các pixel) hình ảnh. Theo mặc định thì apertureSize bằng 3 và em cũng không thay đổi con số này trong khi sử dụng hàm Canny.

Đối số cuối cùng là L2gradient, nó dùng để chỉ định phương trình tìm độ lớn gradient. Theo mặc định thì L2gradient là False, khi đó phương trình mà nó chỉ định là phương trình:

Edge\_Gradient (G) = | Gx | + | Gy |

Tuy nhiên theo kết quả nghiên cứu của em, nếu ta gán L2gradient là True thì độ chính xác sẽ tốt hơn do đó đối với chương trình của mình thì em gán L2gradient là True. Khi đó phương trình mà nó chỉ định sẽ là:

Edge\_Gradient (G) =

Angle(Ө) = ()

Sau khi sử dụng phương pháp Canny, đây là kết quả thu được:



Hình . Kết quả của Canny edged detection

2.5 Vẽ hộp bao quanh các con số

Sau khi phát hiện được các cạnh của các con số, em tiến hành vẽ các hộp chữ nhật bao quanh nó dựa trên cạnh được phát hiện trước đó.

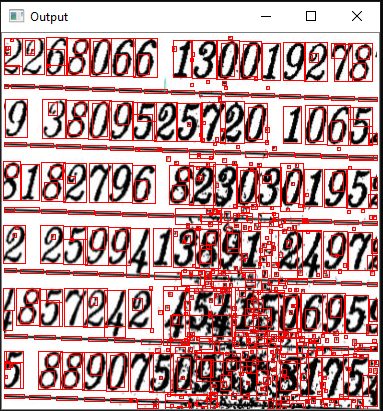
Đầu tiên đễ vẽ các hộp chữ nhật, ta cần biết được các contour của các đối tượng. Contour có thể hiểu là tập hợp các điểm liên tục tạo thành một đường cong (boundary) và không có khoảng hở trong đường cong đó.

Trong openCV có hỗ trợ chúng ta hàm để tìm contour là hàm findContours. Hàm này trả về hai đối tượng là contours - danh sách các contour có trong bức ảnh nhị phân, mỗi một contour được lưu trữ dưới dạng vector các điểm và hierarchy - danh sách các vector, chứa mối quan hệ giữa các contour. Tham số đầu vào của hàm là ảnh nhị phân, typeofContour và methodofContour. Tham số đầu tiên em sử dụng là ảnh nhị phân edged, tức ảnh sau khi trải qua các giai đoạn tiền xử lý, lấy ngưỡng, xử lý hình thái, phát hiện cạnh như đã trình bày ở trên. Hai đối số tiếp theo tương ứng là cv2.RETR\_LIST và cv2.CHAIN\_APPROX\_SIMPLE. Trong đó, cv2.RETR\_LIST là chế độ truy xuất đường viền và cv2.CHAIN\_APPROX\_SIMPLE là phương pháp tìm xấp xỉ đường viền.

Sau khi có được kết quả từ hàm findContours là cnts (contours) và hier (hierarchy), em sử dụng hàm sorted để sắp xếp theo thứ tự giảm dần của danh sách các contour mình tìm được phục vụ cho việc vẽ các hình hộp chữ nhật bao xung quanh các con số.

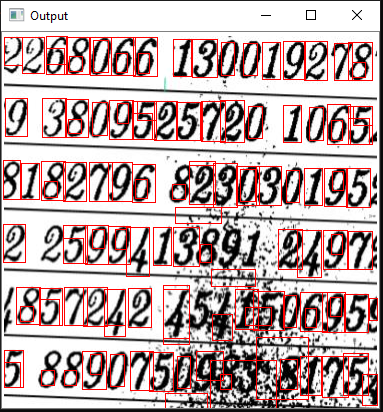
Để vẽ các hình hộp chữ nhật theo yêu cầu, em duyệt qua danh sách các contour đã sắp xếp ở trên, sau đó sử dụng hàm boundingRect của openCV cho từng contour mình đang xét. Hàm boundingRect trả về danh sách 4 đối tượng, ở đây em dùng lần lượt là X, Y, W, và H. Trong đó X, Y là tọa độ pixel trên cùng bên trái của mỗi hình hộp chữ nhật, trong khi đó W là độ rộng (width) và H là chiều cao (height) của hình hộp chữ nhật đó.

Sau khi có tọa độ X,Y, độ rộng W và chiều cao H của mỗi hình hộp chữ nhật, em dùng hàm rectangle của openCV, truyền vào ảnh input và tọa độ cần thiết để vẽ cho mỗi hình hộp chữ nhật. Kết quả thu được là ảnh input với các hình hộp chữ nhật bao quanh các đối tượng.



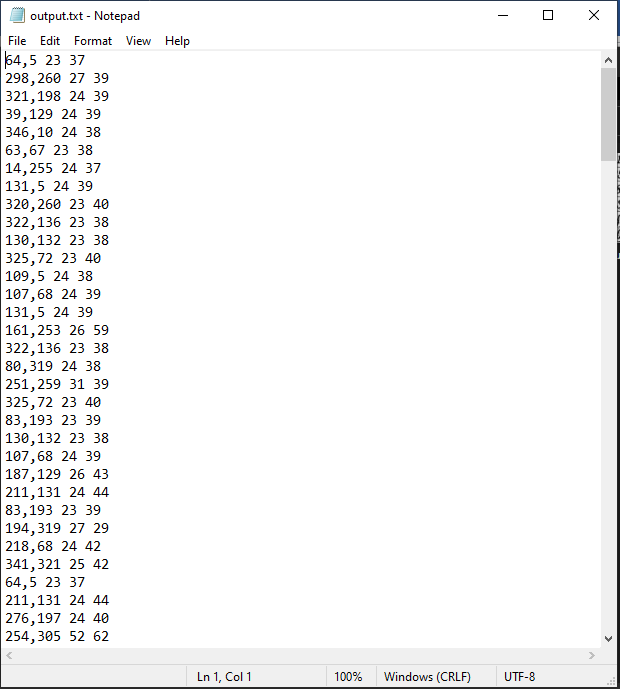
Hình . Ảnh sau khi vẽ các hình hộp chữ nhật

Từ kết quả trong hình 2.7 trên, em nhận thấy các đối tượng không cần thiết như các đường kẻ ngang, các chấm nhỏ, v.v… vẫn chưa được xử lý hết và được vẽ hình hộp chữ nhật bao quanh nó. Để giảm thiểu các đối tượng vẽ không mong muốn, chương trình của em đưa ra điều kiện kiểm tra kích thước của ô chữ nhật, nếu to quá hoặc nhỏ quá thì không vẽ ô chữ nhật đó. Cụ thể em chỉ xét các ô chữ nhật có kích thước dựa trên kết quả trả về của hàm contourArea lớn hơn 100 và nhỏ hơn 500. Khi đó kết quả sẽ giảm bớt được các ô hình chữ nhật không mong muốn như sau:



Hình . Kết quả cuối cùng của chương trình

Ảnh này cũng sẽ là kết quả của chương trình, file ảnh là output.png. Đi kèm với file ảnh output này là file text có tên output.txt ghi lại tọa độ các hình hộp chữ nhật được vẽ. Cụ thể mỗi dòng trong file output.txt là tọa độ của mỗi hình hộp chữ nhật được ghi lại dưới dạng X,Y W H. Trong đó X,Y là tọa độ pixel trên cùng bên trái của mỗi hình hộp chữ nhật, trong khi đó W là độ rộng (width) và H là chiều cao (height) của hình hộp chữ nhật đó.



Hình . Nội dung file output.txt

CHƯƠNG 3 – TỔNG KẾT

3.1 Nhận xét

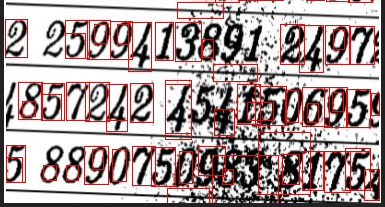
3.1.1 Ưu điểm

So với yêu cầu của đề bài là vẽ các hình hộp bao quanh các chữ số thì chương trình của em đã vẽ được gần như đầy đủ các con số có trong hình (có khoảng 5 số chưa phát hiện được). Cụ thể chương trình của em đã xử lý được các chữ số với độ đậm nhạt, sáng tối và độ nhiễu khác nhau, từ đó vẽ được các hình hộp bao quanh tương ứng cho mỗi số nằm trên toàn bộ ảnh. Cùng với đó là các chữ số dính liền nhau cũng được xử lý để cho kết quả thích hợp và các chữ số bị mất nét ở hai bên biên của ảnh vẫn có thể xử lý và cho kết quả theo yêu cầu của đề bài.

3.1.2 Khuyết điểm

Vẫn còn tồn tại một vài con số chưa thể phát hiện được, cụ thể ở dòng số 4 của ảnh input là số 2, 9 và 1; ở dòng thứ 5 là số 4, 5. Các con số khác tuy đã phát hiện được nhưng của chỉ có thể vẽ được đường bao cho một phần chữ số.

Ngoài ra chương trình còn hạn chế ở việc vẽ một vài ô hình chữ nhật không mong muốn và các ô chữ nhật bị trùng lắp với nhau.



Hình 3.1: Hạn chế của chương trình

3.2 Phương hướng phát triển

Từ những ưu điểm, hạn chế ở trên, chương trình định ra mục tiêu phát triển là cải thiện việc vẽ các ô chữ nhật xung quanh chữ số, hạn chế nhất có thể sự trùng lắp các ô chữ nhật. Ngoài ra, các ô chữ nhật bao xung quanh hình phải được vẽ bao xung quanh chữ số chứ không phải chỉ bao được một phần.

Định hướng cho việc cải thiện việc vẽ các ô chữ nhật bị trùng lắp của em là sử dụng thuật toán Non-maximum suppression. Ý tưởng của thuật toán này là truyền vào tập danh sách các proposals box ký hiệu là B với B ={b1,b2,…,bn}, với bi là proposal thứ i; Tập điểm của mỗi proposal box ký hiệu là S với S={s1,s2,…,sn}, si là điểm confidence của box bi; Giá trị ngưỡng overlap threshold N.. Cả hai giá trị bi và si đều là output của mạng neural network. Đầu ra là một tập các proposals box D tức là tập các proposals đã loại bỏ dư thừa tương ứng với từng object trong hình. Độ chuẩn xác của thuật toán này phụ thuộc vào giá trị ngưỡng overlap, do chưa thể tìm được giá trị ngưỡng thích hợp để sử dụng đối với bài toán này nên việc sử dụng thuật toán Non-maximum suppression để giải quyết bài toán đặt ra chỉ dừng ở mức độ nghiên cứu và định hướng phát triển chứ không áp dụng và trình bày cụ thể.

**TÀI LIỆU THAM KHẢO**

1. Alexander Mordvintsev & Abid K. Revision. OpenCV-Python Tutorials. [online] Available at: <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html>. [Accessed 5 September 2020]
2. Doxygen. Open Source Computer Vision. [online] Available at: <https://docs.opencv.org/3.4/da/d0c/tutorial_bounding_rects_circles.html>. [Accessed 5 September 2020].
3. Doxygen. Open Source Computer Vision. [online] Available at: <https://docs.opencv.org/3.4/da/d0c/tutorial_bounding_rects_circles.html>. [Accessed 5 September 2020].