Seminar các vấn đề hiện đại của CNPM - SE400.P11

Thành viên:

21520932 – Phạm Phước Huy

21521411 – Trần Văn Thanh Tâm

BÁO CÁO GIỮA KÌ

I. Cơ sở lí thuyết

Với bài toán nhận dạng biển số xe, nhóm xác định có các nhiệm vụ cần thực hiện như sau: Đầu tiên, chúng ta cần xác định vị trí biển số xe từ ảnh dữ liệu đầu vào, sau đó tìm cách tách vùng ảnh biển số xe và cuối cùng là nhận diện các kí tự từ vùng ảnh này. Với những yêu cầu trên, nhóm đã tìm hiểu được một số công nghệ có thể giải quyết bài toán nhận diện biển số xe bao gồm: Phát hiện vật thể trong ảnh bằng OpenCV hoặc YOLO, nhận dạng ký tự trong hình ảnh bằng pytesseract hoặc CNN.

1. OpenCV

OpenCV (tên viết tắt của Open Source Computer Vision Library) là thư viện mã nguồn mở chuyên dùng trong xử lý ảnh và thị giác máy tính, cung cấp các chức năng giúp xác định vị trí biển số xe trong ảnh, hỗ trợ phân tích và xử lý ảnh biển số xe trước khi tiến hành bước nhận diện từng ký tự.

Với yêu cầu phát hiện biển số xe, OpenCV cung cấp một số công cụ như CascadeClassifier, là một thư viện OpenCV sử dụng thuật toán Haar-cascasde để phát hiện các đối tượng cụ thể trong ảnh như biển số xe hoặc khuôn mặt. Đây là một mô hình machine learning học từ hàng ngàn ảnh biển số xe mẫu, tìm ra các đặc điểm nổi bật của biển số xe như có hình chữ nhật, phạm vi các loại ký tự xác định, thường có độ tương phản giữa các ký tự và nền của biển số, v.v. Haar-cascade dựa trên các đặc điểm này để phát hiện các khu vực có thể là biển số xe trong ảnh.

Ngoài phát hiện vị trí biển số xe bằng các thư viện, mô hình như CascadeClassifier, OpenCV cung cấp một số hàm, công cụ khác để xác định và tách vùng ảnh biển số xe. Có nhiều cách

làm khác nhau với những kĩ thuật khác nhau, và sau đây là một cách trích xuất ảnh biển số xe mà nhóm đã tìm hiểu được: Đầu tiên, ta cần xử lý ảnh gốc thông qua các kĩ thuật: chuyển đổi ảnh màu sáng ảnh xám, làm mờ ảnh bằng bộ lọc Gaussian, phân ngưỡng Otsu, mở rộng hình thái học (Morphological Opening). Tiếp theo, ta tiến hành bước phát hiện đường viền của các đối tượng. Sau đó, ta tiến hành bước xử lý và lọc các đường viền nhằm tìm ra vùng ảnh có kích thước và tỷ lệ hình dạng phù hợp với biển số xe nhất. Cuối cùng, nếu tìm được vùng ảnh đạt tiêu chí, ta tiến hành cắt vùng ảnh biến số xe đã phát hiện.

1.1. Tiền xử lý ảnh

- Chuyển đổi ảnh màu sang ảnh xám:

Ảnh xám hay còn gọi là ảnh đơn sắc (monochromatic) là ảnh mà tại mỗi điểm ảnh có một giá trị mức xám. Không như ảnh màu mỗi pixel có ba giá trị đại diện cho màu đỏ (Red), xanh lá (Green) và xanh dương (Blue), còn ảnh xám với mỗi pixel chỉ có một giá trị đại diện cho độ sáng (intensity). Mỗi pixel của ảnh xám thường có giá trị nằm trong khoảng 0 đến 255: giá trị 0 tương ứng với màu đen, giá trị 255 tương ứng với màu trắng và các giá trị từ 1 đến 254 tương ứng với các mức xám. Trong bài toán nhận diện biển số xe, việc chuyển đổi ảnh màu sang ảnh xám giúp nhận biết các đặc điểm đường nét, cấu trúc, đối tượng của ảnh một cách dễ dàng hơn. Lý do đầu tiên là giảm độ phức tạp và khối lượng dữ liệu của ảnh, chuyển đổi ảnh màu sang ảnh xám giúp giảm dữ liệu từ ba giá trị cho mỗi pixel xuống một giá trị cho mỗi pixel, từ đó giảm số lượng công việc tính toán, tăng tốc độ xử lý, tập trung vào thông tin vật thể ảnh và không bị ảnh hưởng bởi màu sắc. Lý do thứ hai là nâng cao khả năng nhận diện độ tương phản và đường viền ảnh, biển số xe thường có màu nền tối và màu kí tự sáng hoặc ngược lại màu nền sáng và màu kí tự tối, việc chuyển sang ảnh xám giúp làm nổi bật sự khác biệt về độ sáng, độ tương phản giữa các vùng ảnh nền và các kí tự biển số, từ đó giúp việc nhận diện các đường viền của biển số xe hiệu quả hơn. Lý do thứ ba là các ký tự và nền của biển số được thiết kế tuân theo các quy định của nhà nước, không thay đổi về màu sắc. Màu sắc có thể là yếu tố gây nhiễu cho quá trình nhận diện biển số xe, làm việc nhận diện trở nên khó khăn hơn. Ngoài ra, việc chuyển ảnh xám là điều rất cần thiết, đảm bảo việc xử lý ảnh hiệu quả nếu ta sử dụng các thuật toán xử lý ảnh như phân ngưỡng (Thresholding), phát hiện cạnh (Edge detection), phân đoạn (Segmentation), v.v.

Trong OpenCV, ta thực hiện chuyển ảnh xám với hàm cv2. cvtColor() với tham số cv2.COLOR_BGR2GRAY giúp chuyển đổi từ ảnh màu sang ảnh xám. Ví dụ:

```
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Ảnh gốc

Ảnh xám

- Làm mở ảnh bằng bộ lọc Gaussian:

Gaussian Blur (làm mờ/ mịn Gaussian) là một phương pháp làm mờ hình ảnh bởi một hàm phân phối chuẩn (Gaussian Distribution), là một phân phối xác suất cực kì quan trọng trong nhiều lĩnh vực. Kĩ thuật này được sử dụng nhằm mục đích làm giảm nhiễu ảnh mà không ảnh hưởng hay làm mất các đặc điểm quan trọng của đối tượng trong ảnh, hiệu ứng hình ảnh của kĩ thuật này tương tự như việc chúng ta nhìn một hình ảnh qua màn hình mờ. Gaussian Blur áp dụng bộ lọc Gaussian để làm mờ từng pixel của ảnh, bằng cách thay đổi giá trị của mỗi pixel thành giá trị trung bình có trọng số của các pixel lân cận nó, dựa trên hàm Gaussian.

Trong bài toán nhận diện biển số xe, ảnh biển số xe thường bị nhiễu (Noise) bởi các yếu tố như ánh sáng, bụi bẩn, độ tương phản, độ sắc nét, v.v. với việc sử dụng Gaussian Blur giúp giảm nhiễu (Noise reduction), làm mờ các điểm nhiễu hay các chi tiết không quan trọng có thể ảnh hưởng tới việc nhận diện các kí tự, từ đó giúp hệ thống tập trung vào đối tượng chính trong ảnh là biển số xe, phát hiện cạnh của biển số hiểu quả hơn và nhận diện kí tự rõ ràng hơn.

Trong OpenCV, ta có sử dụng Gaussian Blur với hàm cv2.GaussianBlur(). Ví dụ:

```
gray_blurred = cv2.GaussianBlur(gray, (5, 5), 0)
```

Trong đó:

- Tham số thứ nhất: gray là ảnh đã chuyển màu thành xám.
- Tham số thứ hai: (5,5) là kích thước bộ lọc Gaussian. có nghĩa là giá trị mỗi pixel ảnh được tính toán dựa trên một ma trận 5x5 lân cận nó.
- Tham số thứ ba: 0 là đại diện cho sigma, sigma càng lớn thì ảnh càng mờ. Ở đây khi chúng ta nên cài đặt sigma = 0, OpenCV sẽ tự động tính toán giá trị sigma phù hợp để tối ưu hiệu quả dựa trên kích thước bộ lọc.

Ảnh đã làm mờ bằng bộ lọc Gaussian

- Phân ngưỡng Otsu:

Phân ngưỡng (Thresholding) là một kĩ thuật nhị phân hóa bằng thuật toán Otsu trong xử lý ảnh, đây là kĩ thuật chuyển đổi từ ảnh xám (với mỗi pixel ảnh có giá trị từ 0 đến 255) sang ảnh nhị phân (ảnh đen trắng, với mỗi pixel ảnh chỉ có một trong hai giá trị là 0 hoặc 255). Phân ngưỡng tức là phân chia ảnh dựa trên một giá trị ngưỡng xác định (ngưỡng cường độ sáng). Ngưỡng là một giá trị được chọn để phân chia các pixel ảnh thành hai nhóm, một nhóm gồm các pixel có giá trị cường độ sáng dưới ngưỡng và một nhóm khác gồm các pixel có giá trị cường độ sáng trên ngưỡng. Sau khi phân chia nhóm, nhóm dưới ngưỡng hay các pixel có cường độ sáng thấp được chuyển thành màu đen và nhóm trên ngưỡng hay các pixel có cường độ sáng cao được chuyển thành màu trắng, từ đó giúp làm nổi bật các đặc điểm quan trọng của ảnh. Cụ thể với biển số xe, phương pháp này thường sẽ chuyển các kí tự thành màu trắng và nền thành màu đen.

Với các hình ảnh có sự phân bố độ sáng phức tạp, một ngưỡng cố định có thể không hiệu quả hoặc thiếu chính xác. Vì vậy, nhóm chọn phương pháp phân ngưỡng Otsu là phương pháp tự động chọn giá trị ngưỡng giúp tối ưu độ chính xác của việc phân ngưỡng.

Trong OpenCV, chúng ta có thể áp dụng phân ngưỡng Otsu với hàm cv2.threshold:

```
cv2.threshold(gray_blurred, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
```

Trong đó:

- gray_blurred là ảnh đã chuyển sang ảnh xám.
- 0 là giá trị ngưỡng ban đầu và được tự động tính toán lại cho phù hợp bởi phân ngưỡng Otsu
- 255 là giá trị cường độ cho các pixel vượt ngưỡng (mức trắng).
- cv2.THRESH_BINARY + cv2.THRESH_OTSU: Sử dụng kết hợp phân ngưỡng nhị phân và phương pháp Otsu.

Ảnh đã qua xử lý phân ngưỡng

- Mở rộng hình thái học (Morphological Opening):

Hình thái học trong xử lý ảnh là một lý thuyết, kĩ thuật của toán học và khoa học máy tính dùng để nghiên cứu, xử lý các hình dạng, cấu trúc của các đối tượng trong ảnh. Phép toán hình thái học đã được phát triển để áp dụng cho hình ảnh nhị phân và sau đó được mở rộng cho ảnh xám. Đây là một trong những kĩ thuật được áp dụng trong giai đoạn tiền xử lý, nhằm chỉnh sửa và xử lý các hình dạng trong ảnh, ví dụ như loại bỏ các nhiễu, tách biệt các đối tượng, làm sắc nét các cạnh, v.v. Hình thái học bao gồm một số phép toán như:

- Erosion (Phép co): Làm nhỏ đối tượng trong ảnh

- Dilation (Phép giãn nở): Làm lớn đối tượng trong ảnh.
- Opening (Phép mở): Kết hợp giữa phép co và và phép giãn nở để loại bỏ nhiễu nhỏ và giữ lại đối tượng chính.
- Closing (Phép đóng): Kết hợp giữa phép giãn nở và phép co, dùng để loại bỏ các lỗ hổng nhỏ trong đối tượng.

Các phép toán này được áp dụng dựa trên một cấu trúc phần tử (kernel), là một ma trận hình học dùng để di chuyển qua các pixel ảnh để thực hiện các phép toán co, giãn nở, mở, đóng.

Với bài toán nhận diện biển số xe, nhóm em sử dụng phép mở (Opening) để xử lý ảnh nhị phân. Phép toán mở thực hiện phép co trước tiên, loại bỏ các chi tiết nhiễu nhỏ và thực hiện phép giãn nở sau đó, giúp khôi phục lại hình dáng ban đầu của các đối tượng. Từ đó làm sạch các chi tiết gây nhiễu mà không làm thay đổi hình dạng của đối tượng biển số xe trong ảnh.

Trong OpenCV, phép mở được thực hiện bằng cách, tạo một cấu trúc phần tử (kernel) có kích thước 3x3 hoặc 5x5 dùng để quét qua từng điểm ảnh và thực hiện phép mở dựa trên cấu trúc phần tử này:

```
kernel = cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3))

morph_opening = cv2.morphologyEx(threshold, cv2.MORPH_OPEN, kernel)
```

Trong đó:

- cv2.getStructuringElement(cv2.MORPH_RECT, (3, 3)): Tạo một kernel 3x3. Tùy vào nhu cầu xử lý ảnh, ta có thể thay đổi kích thước cho phù hợp, kích thước nhỏ thường phù hợp để loại bỏ các nhiễu nhỏ.
- cv2.morphologyEx(threshold, cv2.MORPH_OPEN, kernel): Áp dụng phép mở với kernel đã định nghĩa trước đó trên ảnh nhị phân đã qua phân ngưỡng.

Ảnh đã áp dụng phép mở hình thái học

1.2. Phát hiện biên (Contours)

Trong OpenCV cung cấp hàm cv2.findContours, là một hàm dùng để tìm và phân tích các đường viền hay đường biên (contours) của các đối tượng trong ảnh nhị phân. Định nghĩa đường viền là tập hợp các điểm ranh giới xunh quanh một đối tượng trong ảnh, giúp xác định hình dạng, kích thước và vị trí của đối tượng. Cách hàm cv2.findContours hoạt động như sau:

Đầu tiên, hàm duyệt qua từng pixel trong ảnh nhị phân, hàm sẽ kiểm tra sự thay đổi về độ sáng giữa các vùng khác nhau. Nếu có sự thay đổi từ vùng tối (màu đen) sang vùng sáng (màu trắng) hoặc ngược lại, nó sẽ đánh dấu điểm đó là một phần của đường viền. Sau khi xác định được điểm đầu tiên của đường viền, hàm sẽ tiếp tục tìm các điểm liên tiếp xung quanh điểm đó. Các điểm được tìm thấy sẽ được liên kết lại với nhau để tạo thành một đường viền khép kín hoặc gần khép kín bao quanh đối tượng. Quá trình này sẽ lặp lại cho đến khi toàn bộ biên của đối tượng được xác định. Khi đã hoàn tất việc tìm đường viền của đối tượng, hàm sẽ lưu nó như một tập hợp các điểm liên tiếp (gọi là contour).

```
contours, hierarchy = cv2.findContours(image, mode, method)
```

Trong đó:

- image là ảnh nhị phân cần xác định contours
- mode: Chế độ truy xuất contours, xác định loại đường viề nào sẽ được tìm thấy. Một số chế độ phổ biến như: cv2.RETR_EXTERNAL (Chỉ lấy các biên ngoài cùng của

đối tượng, bỏ qua các biên bên trong), cv2.RETR_LIST (Lấy tất cả các biên trong ảnh nhưng không tổ chức theo cấu trúc phân cấp), cv2.RETR_CCOMP (Lấy tất cả các biên và tổ chức chúng thành hai cấp độ phân cấp: các đối tượng ngoài cùng và các lỗ bên trong), v.v.

- method: xác định cách lưu trữ các điểm của đường viên. Một số method như: cv2.CHAIN_APPROX_NONE (Lưu toàn bộ các điểm của đường viền, giúp giữ lại độ chi tiết cao), cv2.CHAIN_APPROX_SIMPLE (Lưu các điểm quan trọng để mô tả đường viền, giảm dung lượng bộ nhớ và tăng hiệu suất)

Hàm trả về hai giá trị:

- contours: Một danh sách các đường viền.
- hierarchy: Thông tin về mối quan hệ phân cấp giữa các contours (như contours cha, con hoặc lân cận). Nếu không cần sử dụng các thông tin liên quan đến phân cấp giữa các contours, ta có thể bỏ qua giá trị hierarchy.

1.3. Lọc và phân tích các biên tìm được

Sau khi tìm thấy tất cả các biên bằng hàm cv2.findContours, ta cần lọc lại danh sách biên này vì không phải biên nào cũng cần thiết, có thể tồn tại một số biên nhiễu hoặc không phải là biển số. Đầu tiên, ta sẽ dùng hàm cv2.contourArea để tính diện tích của mỗi biên để loại bỏ các vùng nhỏ không cần thiết. Ở đây, chỉ giữ lại các vùng có diện tích lớn hơn 10000, vì vùng biển số xe thường có diện tích khá đáng kể. Tiếp theo ta sử dụng hàm cv2.approxPolyDP để xấp xỉ các đường biên thành một đa giác với số lượng đỉnh ít hơn, giúp ta loại bỏ các chi tiết nhỏ, các điểm lồi lõm do nhiễu ảnh hoặc các chi tiết không không quan trọng. Hàm cv2.approxPolyDP dựa trên thuật toán Douglas-Peucker, giúp làm giảm số lượng điểm trong một đường viền hoặc đường cong để tạo ra một phiên bản đơn giản hơn. Thuật toán này chọn ra các điểm chính dọc theo đường viền và loại bỏ các điểm không cần thiết, giúp đường viền trở nên đơn giản hơn. Cú pháp:

cv2.approxPolyDP(contour, epsilon, closed)

Trong đó:

- Contour là đường biên cần xấp xỉ.

- Epsilon là giá trị xác định mức độ xấp xỉ, tỷ lệ với chu vi (peri). Nếu epsilon càng lớn, đường biên xấp xỉ càng đơn giản. Ta cần thử nghiệm giá trị này với nhiều dữ liệu biển số xe để tìm ra giá trị epsilon phù hợp.
- Closed có giá trị true hoặc false để xác định đường viền có khép kín hay không.

1.4. Cắt vùng biển số xe

Sau khi đã lọc và phân tích các biên, ta kiểm tra nếu đường biên có 4 đỉnh và tỷ lệ khung hình (chiều rộng / chiều cao) nằm trong một khoảng phù hợp, thì đó có khả năng là biển số xe. Sau đó, ta sử dụng hàm cv2.boundingRect để cắt vùng ảnh chứa biển số xe.

2. YOLO

YOLO là một mô hình mạng CNN dùng để phát hiện, nhận dạng ảnh với tốc độ xử lý nhanh, phù hợp cho các ứng dụng yêu cầu thời gian thực. Mô hình này hoạt động bằng cách chia dữ liệu ảnh đầu vào thành các ô lưới, sau đó kiểm tra xem có đối tượng nào trong mỗi ô hay không, khi YOLO phát hiện được đối tượng trong một ô, nó sẽ bao quanh đối tượng đó bằng một khung gọi là bounding box. Tiếp theo, YOLO sẽ tính toán và sử dụng một số kỹ thuật để tìm ra danh sách các bounding box có độ tin cậy cao nhất. Cuối cùng, YOLO sử dụng tọa độ của các bounding box này để xác định vùng ảnh chỉ chứa biển số xe.
Cách sử dụng YOLO trong việc nhận diện biển số xe:

Đầu tiên, chúng ta cần huấn luyện mô hình YOLO, chuẩn bị dữ liệu đầu vào để huấn luyện mô hình bao gồm các ảnh biển số xe, lưu ý các ảnh nào cần được gán nhãn để tạo bounding box xung quanh biển số (có thể sử dụng công cụ gán nhãn như LabelImg). Sau khi huấn luyện YOLO với bộ dữ liệu đã gán nhãn thành công, sử dụng mô hình này với các ảnh đầu vào thực tế và kiểm tra độ chính xác khi mô hình xử lý ảnh và trả về vùng chỉ chứa biển số xe. Nếu độ chính xác chưa đạt yêu cầu, chúng ta có thể chỉnh sửa hoặc thêm dữ liệu huấn luyện đầu vào.

3. Pytesseract

Pytesseract là một thư viện Python giúp người dùng sử dụng công cụ Tesseract OCR của Google một cách dễ dàng. Tesseract OCR là công cụ mã nguồn mở, hỗ trợ nhận diện ký tự quang học, giúp đọc và chuyển đổi các ký tự trong ảnh thành văn bản. Thư viện này có độ chính xác cao trong nhận diện kí tự từ ảnh, hỗ trợ nhiều ngôn ngữ khác nhau, cho phép tùy chỉnh, giới hạn phạm vi kí tự nhận diện. Cách sử dụng Pytesseract khá đơn giản, đầu tiên cần chuẩn bị phần ảnh biển số xe được tách ra từ ảnh gốc, sau đó xử lý ảnh bằng OpenCV như:

chuyển ảnh về dạng trắng đen, tăng độ sáng, độ tương phản, làm mờ, v.v. nhằm mục đích cải thiển khả năng nhận diện ký tự, tăng độ chính xác của Pytesseract. Sau khi đã qua bước xử lý ảnh, việc còn lại là sử dụng hàm pytesseract.image_to_string() do Pytesseract cung cấp để nhận diện các ký tự từ ảnh biển số xe. Cú pháp:

pytesseract.image_to_string(image, lang='eng', config='...')

Trong đó:

- image là ảnh đã qua bước tiền xử lý .
- lang='eng': Thiết lập ngôn ngữ cho OCR.
- Với lang='eng', Tesseract sẽ nhận diện các ký tự tiếng Anh, biển số xe thường chứa các ký tự chữ cái A-Z và số 0-9.
- config='...': dùng để tùy chỉnh các chế độ phân đoạn và giới hạn phạm vi các ký tự.

Ví dụ:

```
config='--psm 10 -c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789'
```

Trong đó:

- psm (Page Segmentation Mode) có nghĩa là "Chế độ Phân đoạn Trang.", đây là tham số chỉ định Tesseract OCR cách xử lý và phân tích văn bản trong ảnh. Có nhiều loại psm khác nhau như: psm 0 - Orientation and Script Detection (OSD), psm 1 - Automatic page segmentation with OSD, psm 6 - Single uniform block of text, psm 7 - Treat the image as a single text line, psm 8 - Treat the image as a single word, psm 9 - Single word in a circle, psm 10 - Treat the image as a single character, v.v. Với bạn toán nhận diện biển số xe, ta có thể sử dụng các chế độ psm 6, psm 7, psm 10.
- -c tessedit_char_whitelist=ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789: Giới hạn phạm vi kí tự nhận diện, giúp cái thiện độ chính xác.

4. CNN

Thuật toán CNN (Convolutional Neural Network) là một loại mô hình deep learning để nhận dạng và phân loại hình ảnh có độ chính xác cao. Mô hình này được ứng dụng rộng rãi trong nhận diện biển số xe, ảnh, xử lý video, nhận diện chữ viết tay, v.v.

Cách sử dụng CNN trong việc nhận diện biển số xe:

Đầu tiên, chúng ta cần tạo một bộ dữ liệu để huấn luyện mô hình bao gồm các ảnh ký tự được tách ra từ nhiều biển số xe khác nhau, mỗi ảnh cần được gán nhãn tương ứng với ký tự mà nó đại diện. Tiếp theo ta cần thực hiện một số bước xử lý ảnh để thu được kết quả huấn luyện tốt hơn như: đưa tất cả ảnh về cùng kích thước ví dụ như 28x28 hoặc 32x32 pixel để thống nhất dữ liệu ảnh đầu vào cho mô hình, chuyển ảnh sang dạng đen trắng (grayscale), chuẩn hóa giá trị pixel v.v.

Sau khi đã có bộ dữ liệu đầu vào, bước tiếp theo là xây dựng cấu trúc cho mô hình CNN với một số lớp chính như Convolutional Layer, Pooling Layer, Fully Connected Layer, Output Layer. Tiếp theo, chia dữ liệu thành 2 phần training set dùng để huấn luyện mô hình và test set để đánh giá độ chính xác của mô hình sau khi huấn luyện. Cuối cùng, kiểm tra về độ chính xác của mô hình với những dữ liệu ảnh thực tế, nếu kết quả chưa đạt như mong muốn, chúng ta có thể điều chỉnh lại cấu trúc của mô hình hoặc thêm dữ liệu đầu vào.

5. Tổng kết

Sau quá trình tìm hiểu, thử nghiệm và so sánh kết quả giữa các công nghệ trên, nhóm quyết định chọn công nghệ phát hiện vật thể trong ảnh bằng OpenCV và nhận dạng ký tự trong hình ảnh bằng pytesseract để giải quyết bài toán nhận diện biển số xe. Chi tiết về lý do chọn hai công nghệ này sẽ được trình bày ở Mục II. Quá trình thực hiện.

II. Quá trình thực hiện

1. Xác định bài toán

Thời gian thực hiện: Từ tuần 3 đến tuần 4 (đúng tiến độ).

Bài toán nhận diện biển số xe là một bài toán có ứng dụng phổ biến, được áp dụng ở nhiều lĩnh vực. Nhóm dự định sẽ thực hiện nhận diện biển số bằng cách dùng camera máy tính để chụp ảnh và xử lí dữ liệu trên hình ảnh đó.

Sau khi tìm hiểu tổng quan về cách thực hiện, nhóm nhận thấy để giải được bài toán này cần giải 2 bài toán nhỏ hơn đó là *nhận dạng biển số trong hình* và *đọc các kí tự trên biển số*. Mỗi bài toán đều có sẵn các thư viện viết bằng Python hỗ trợ, không cần phải tự train các mô hình AI riêng. Do vậy, nhóm quyết định sẽ tiến hành viết chương trình bằng Python và sử dụng các thư viện có sẵn để thực hiện.

2. Chọn công nghệ nhận dạng biển số

Dự định ban đầu là tìm hiểu và chọn công nghệ trước rồi mới tiến hành code. Tuy nhiên, trong quá trình làm thì nhóm đã thay đổi cách làm đó là tiến hành viết code để chạy chương trình rồi mới đưa ra kết luận chọn công nghệ phù hợp. Do đó, so với bản kế hoạch ban đầu thì giai đoạn này tương đương với 2 giai đoạn chọn công nghệ và viết chương trình.

Thời gian thực hiện: Từ tuần 4 đến tuần 7 (đúng tiến độ), giai đoạn này không chỉ được thực hiện một lần mà còn được cải thiện dần kể cả sau khi thực hiện giai đoạn 3 (Nhận dạng kí tự từ hình ảnh).

Ban đầu, nhóm tìm hiểu được có 2 thư viện hỗ trợ cho việc nhận dạng vật thể trong hình ảnh đó là OpenCV và YOLO. Để tối ưu thời gian thực hiện, nhóm quyết định chia việc cho 2 thành viên để mỗi người tìm hiểu theo một thư viện khác nhau. Phạm Phước Huy tìm hiểu về OpenCV, Trần Văn Thanh Tâm tìm hiểu về YOLO. Theo kế hoạch, sau khoảng 2 tuần, kết quả của thư viện nào khả thi hơn thì sẽ chọn thư viện đó. Kết quả là sau 2 tuần, chỉ có thư viện OpenCV cho ra kết quả, còn phương án còn lại không nhận diện được biển số xe. Do vậy, nhóm quyết định chọn thư viện OpenCV để giải quyết bài toán nhận diện biển số xe. Tuy đến lúc này, phần code nhận diện biển số vẫn chưa cho ra kết quả chính xác cao, nhưng nhóm quyết định tiếp tục chuyển sang giai đoạn chọn công nghệ nhận dạng kí tự từ hình ảnh, rồi sau đó mới quay lại cải thiện phần code của nhận diện biển số.

Sau khi chọn công nghệ nhận dạng kí tự từ ảnh và quay trở lại cải thiện độ chính xác của thuật toán nhận dạng biển số trong ảnh, nhóm tập trung vào việc tiền xử lí ảnh và lọc đối tượng. Dưới đây là một số chính để bước thực hiện việc nhận dạng biển số trong ảnh:

- Tiền xử lí ảnh: Bước này có nhiều cách xử lí khác nhau, kết quả cuối cùng luôn là ảnh dạng trắng đen. Quá trình xử lí ảnh có thể dùng kết hợp nhiều bước như chuyển thành dạng ảnh xám, chỉnh độ sánh, độ tương phản, Gaussian blur, threshold... Tính tới thời điểm hiện tại, nhóm đã thử qua một số phương pháp và thấy kết quả khả quan nhất là dùng bộ các bước xử lí: chuyển ảnh sang màu xám, Gaussian blur và threshold.
- Chọn vật thể (biển số): Vẽ viền vật thể dựa trên ảnh đã qua xử lí, sau đó dùng thuật toán để vẽ đa giác xấp xỉ cho vật thể, chỉ chọn những vật thể là hình tứ giác có diện tích lớn hơn 10000 (pixel x pixel), có tỉ lệ ngang/ dọc trong khoảng 1.2 đến 1.8 hoặc 3.0 đến 7.5. Những thông số cụ thể này có được là do quá trình thử nghiệm, kiểm tra và chọn ra bộ số phù hợp.

3. Chọn công nghệ nhận dạng kí tự từ hình ảnh

Thời gian thực hiện: từ tuần 6 đến tuần 7 (đúng tiến độ)

Có nhiều thư viện hỗ trợ việc nhận dạng kí tự trong hình ảnh, trong đó có 2 thư viện phổ biến nhất là pytesseract và CNN. Nhóm quyết định chia 2 người tìm hiểu 2 thư viện độc lập rồi so sánh. Phạm Phước Huy tìm hiểu về CNN, Trần Văn Thanh Tâm tìm hiểu thư viện pytesseract. Kết quả là thư viện pytesseract dễ thực hiện hơn, cho ra kết quả trước nên được chọn để triển khai trong chương trình.

Việc sử dụng thư viện pytesseract tương đối đơn giản, nhưng để cho ra kết quả chính xác thì cần phải xử lí ảnh. Ở bước trên, sau khi xác định biển số thì phần biển số sẽ được tách và lưu ra một ảnh riêng, ảnh mới này sẽ được sử dụng để đọc kí tự bằng pytesseract. Tính tới hiện tại, nhóm triển khai đồng thời 2 hướng để nhận diện kí tự:

- Dùng thuật toán nhận diện vật thể bằng công nghệ OpenCV tương tự như bước tách vật thể như trên để lọc ra từng kí tự. Sau đó dùng pytesseract để nhận diện từng kí tự đó. Hướng này theo lí thuyết sẽ cho độ chính xác cao, tuy nhiên hiện tại việc tách từng kí tự ra độc lập lại khiến pytesseract đọc nhầm một số kí tự tương đối rõ nét (ví dụ số 9 thường bị nhận diện nhầm thành số 3).
- Dùng hàm của pytesseract để đọc trực tiếp tất cả các kí tự trong ảnh. Hướng này có cách triển khai đơn giản. Theo lí thuyết thì cách này không có độ chính xác cao bằng cách trên. Do vậy có một số kí tự đọc không được kết quả chính xác như cách trên. Nhưng ngược lại, việc đọc cả chuỗi kí tự cùng lúc lại không gây ra tình trạng đọc nhầm một số kí tự tương đối rõ nét như 9 thành 3. Do đó hướng thực hiện này vẫn chưa bị loại bỏ hoàn toàn trong chương trình mà vẫn giữ lại để nghiên cứu và so sánh kết quả nhằm cải thiện độ chính xác cho phương pháp thứ nhất.

4. Các vấn đề khác

Thời gian thực hiện: Tuần 7. Theo kế hoạch ban đầu thì việc hoàn thành chương trình sẽ được thực hiện trước ngày 01/11/2024. Tiến độ hiện tại là đang chậm hơn so với dự định, chương trình vẫn chưa hoàn thành do độ chính xác còn rất thấp, cần cải thiện nhiều. Do vậy nhóm vẫn sẽ tiếp tục cải thiện chương trình song song với giai đoạn tìm hiểu lí thuyết trong kế hoạch.

Tính tới thời điểm hiện tại, độ chính xác ở cả 2 bước xác định biển số và nhận diện kí tự vẫn chưa cao. So sánh độ ưu tiên thì việc cải thiện phần nhận diện kí tự trong biển số là cần thiết hơn, do đó hiện tại nhóm ưu tiên tập trung cải thiện phần này.

Phương hướng hiện tại của nhóm là bước tiền xử lí ảnh, lựa chọn các thông số xử lí ảnh để cho ra độ chính xác cao nhất trong bước nhận diện kí tự. Song song với đó, nhóm chọn giải pháp tạm thời là chạy cả 2 phương pháp nhận diện kí tự rồi so sánh kết quả với khuôn mẫu biển số xe ở Việt Nam.

Các vấn đề khác:

- Kí tự "Đ" chỉ có trong tiếng Việt có thể xuất hiện trong biển số, việc sử dụng mô hình được train bằng tiếng Anh sẽ không đọc được kí tự này.
- Các biển số có màu nền khác màu trắng sẽ không đọc được kí tự.
- Biển số trong hình gốc ban đầu nếu có độ nghiêng lớn sẽ không nhận diện được.

Trong các vấn đề trên, vấn đề thứ nhất vẫn chưa có phương hướng xử lí. Việc biển số có màu nền không phải trắng có thể xử lí bằng cách tiền xử lí ảnh tách ra 2 màu riêng biệt ra thành trắng đen. Việc biển số bị nghiêng có thể sử dụng phương pháp xoay hình để xử lí dựa vào góc tính được từ 2 trong 4 đỉnh của biển số.

III. Kết quả

Dưới đây là link video demo chương trình nhận diện biển số mà nhóm đã trình bày tới thời điểm làm báo cáo giữa kì:

https://youtu.be/IZtAsdJgRks

Dưới đây là link GitHub của nhóm:

https://github.com/BowJaro/Seminar-Modern-Technologies.git

Trong source code, nhóm sử dụng mẫu gồm 12 ảnh với 16 biển số được lưu trong thư mục data. Kết quả hiện tại nhận diện đúng 5 trên 16 ảnh.

TÀI LIỆU THAM KHẢO

1. Python OpenCV Detect and Recognize Car License Plate - Python Geeks (truy cập lần cuối 01-11-2024)
2. License Plate Recognition YOLOv7 and CNN - GitHub (truy cập lần cuối 01-11-2024)
3. How to automatically deskew (straighten) a text image using OpenCV (truy cập lần cuối 01-11-2024)
4. License Plate Recognition - GitHub (truy cập lần cuối 01-11-2024)
5. Nhận diện biển số xe Việt Nam (truy cập lần cuối 01-11-2024)
6. Vietnamese license plate OpenCV KNN - GitHub (truy cập lần cuối 01-11-2024)