ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA CÔNG NGHỆ PHẦN MỀM

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

Chương trình nhận dạng biển số xe ở Việt Nam

GV HƯỚNG DẪN:

Giảng viên Đinh Nguyễn Anh Dũng

SV THỰC HIỆN:

TRẦN VĂN THANH TÂM – 21521411

PHẠM PHƯỚC HUY – 21520932

TP. HỒ CHÍ MINH, 2025

LỜI CẢM ƠN

Chúng em xin gửi lời cảm ơn chân thành và sự tri ân đến Giảng viên Đinh Nguyễn Anh Dũng đã hướng dẫn, đồng hành và tạo điều kiện cho nhóm em hoàn thành đồ án môn Seminar các vấn đề hiện đại của CNPM với đề tài "Chương trình nhận diện biển số xe ở Việt Nam". Trong suốt thời gian thực hiện đồ án, nhóm em đã học hỏi và tích lũy thêm được nhiều kiến thức mới về công nghệ, kĩ năng làm việc nhóm cũng như kinh nghiệm xử lý những vấn đề liên quan lập trình hiện đại. Những nhận xét, góp ý chân tình của thầy chính là cơ sở để nhóm có thể cải tiến và hoàn thiện đề tài này một cách tốt nhất.

Tuy nhiên, trong quá trình thực hiện đề tài, do còn thiếu kinh nghiệm trong việc xây dựng sản phẩm, nhóm có thể không tránh khỏi những thiếu sót. Chính vì vậy, chúng em rất mong nhận được những sự góp ý, hỗ trợ từ phía thầy nhằm hoàn thiện đồ án một cách tốt nhất, và cũng là hành trang để em thực hiện tiếp các đề tài khác trong tương lai.

Một lần nữa, chúng em xin chân thành cảm ơn sâu sắc. Chúng em xin kính chúc thầy luôn dồi dào sức khỏe, thành công trong sự nghiệp và cuộc sống!

Tp. HCM, ngày 05 tháng 02 năm 2025

Sinh viên thực hiện

Trần Văn Thanh Tâm – Phạm Phước Huy

MỤC LỤC

MỤC LỤC HÌNH

TÓM TẮT ĐỒ ÁN

Vấn đề nghiên cứu:

Đồ án này tập trung vào việc xây dựng "Chương trình nhận dạng biển số xe ở Việt Nam". Trong bối cảnh giao thông ngày càng phức tạp tại Việt Nam và chủ trương chuyển đổi số của nhà nước, việc quản lý và giám sát phương tiện trở nên vô cùng quan trọng. Việc ứng dụng công nghệ nhận dạng biển số xe không chỉ giúp cải thiện vấn đề quản lí giao thông mà còn mang lại nhiều lợi ích trong các lĩnh vực khác như an ninh, logistics và quản lý bãi đỗ xe. Chương trình nhận dạng biển số xe hiệu quả tại Việt Nam đòi hỏi sự nghiên cứu và phân tích kỹ lưỡng, yêu cầu hệ thông phải có sự linh hoạt và có khả năng thích ứng với nhiều điều kiện môi trường đa dạng ở Việt Nam. Chương trình hướng tới mục tiêu hoạt động ổn định và chính xác trong mọi điều kiện, đảm bảo tính tin cậy và hiệu quả.

Các hướng tiếp cận và giải quyết vấn đề:

Hướng tiếp cận chính là tập trung vào việc phân tích tìm hiểu các đặc điểm, tính chất đặc trưng của biển số xe Việt Nam. Sau đó, nhóm xác định các các nhiệm vụ cần thiết cần phải thực hiện để nhận diện một biển số xe hiệu quả, đồng thời tìm hiểu các thư viện và công nghệ phù hợp, hỗ trợ cho việc nhận diện vật thể, nhận diện kí tự. Từ đó, nhóm lên kế hoạch chi tiết và tiến hành xây dựng chương trình nhận diện biển số xe ở Việt Nam.

Một số kết quả đạt được:

Kết quả của đồ án bao gồm việc xây dựng một chương trình nhận diện biển số xe, hoạt động được với độ chính xác cao khi áp dụng với các biển số xe ở Việt Nam. Ngoài ra, chương trình phải đơn giản, dễ sử dụng, dễ triển khai trên nhiều nền tảng, thiết bị đồng thời có hiệu suất xử lí nhanh, không tốn nhiều tài nguyên máy tính.

Tóm tắt nội dung báo cáo:

Báo cáo đề tài Xây dựng chương trình nhận diện biến số xe ở Việt Nam bao gồm bốn chương (giới thiệu đề tài, cơ sở lý thuyết, phân tích chương trình, xây dựng ứng dụng) cùng với đó là phần kết luận và tài liệu tham khảo:

- **Chương 1. Giới thiệu đề tài:** Ở chương này, nhóm tập trung vào việc trình bày lý do chọn đề tài, mục tiêu của đề tài, đối tượng nghiên cứu, phạm vi nghiên cứu, và phương pháp thực hiện đề tài.
- **Chương 2. Cơ sở lý thuyết:** Ở chương này, nhóm tập trung vào việc giới thiệu về khái niệm biến số xe, hướng giải quyết bài toàn và các thư viện, công nghệ mà nhóm sử dụng để xây dựng chương trình.
- **Chương 3. Phân tích chương trình:** Ở chương này, nhóm tập trung vào việc trình bày phân tích các kĩ thuật giúp thực hiện chức năng của chương trình.
- **Chương 4. Xây dựng ứng dụng:** Ở chương này, nhóm tập trung vào việc mô tả lại quá trình thực hiện chương trình theo trình tự thời gian.
- **Phần kết luận:** Trình bày các ưu điểm, hạn chế của ứng dụng và hướng phát triển trong thời gian sắp tới của đồ án.
- **Phần tài liệu tham khảo:** Trình bày các tài liệu tham khảo mà nhóm em đã truy cập trong suốt thời gian thực hiện đồ án.

Chương 1. GIỚI THIỆU ĐỀ TÀI

Ở chương này, nhóm tập trung vào việc trình bày lý do chọn đề tài, mục tiêu của đề tài, đối tượng nghiên cứu, phạm vi nghiên cứu, và phương pháp thực hiện đề tài.

1.1. Lý do chọn đề tài

Trong bối cảnh giao thông ngày càng phức tạp tại Việt Nam và chủ trương chuyển đổi số của nhà nước, việc quản lý và giám sát phương tiện trở nên vô cùng quan trọng. Sự gia tăng nhanh chóng của các loại xe, từ ô tô đến xe máy, đặt ra những thách thức không nhỏ cho các cơ quan chức năng. Nhận dạng biển số xe nổi lên như một giải pháp then chốt, giúp tự động hóa các quy trình quản lý, từ đó nâng cao hiệu quả và giảm thiểu sai sót. Việc ứng dụng công nghệ nhận dạng biển số xe không chỉ giúp cải thiện vấn đề quản lí giao thông mà còn mang lại nhiều lợi ích trong các lĩnh vực khác như an ninh, logistics và quản lý bãi đỗ xe.

Việc xây dựng một chương trình nhận dạng biển số xe hiệu quả tại Việt Nam đòi hỏi sự nghiên cứu và phân tích kỹ lưỡng. Biển số xe ở Việt Nam có những đặc điểm riêng về kí tự, font chữ, kích thước và màu sắc, khác biệt so với nhiều quốc gia khác. Điều này đặt ra yêu cầu phải có một hệ thống nhận dạng linh hoạt, có khả năng thích ứng với những biến thể này. Hơn nữa, điều kiện môi trường ở Việt Nam cũng đa dạng, từ ánh sáng ban ngày đến ánh sáng yếu vào ban đêm, từ thời tiết nắng ráo đến mưa bão. Chương trình cần được thiết kế để có thể hoạt động ổn định và chính xác trong mọi điều kiện, đảm bảo tính tin cậy và hiệu quả.

Nhận thấy tiềm năng to lớn của công nghệ nhận dạng biển số xe trong việc giải quyết các vấn đề giao thông và an ninh, nhóm chúng em quyết định chọn đề tài "Chương trình nhận dạng biển số xe ở Việt Nam".

1.2. Mục tiêu

- Xây dựng một chương trình nhận dạng biển số xe hoạt động được với độ chính xác cao khi áp dụng với các biển số xe ở Việt Nam.

- Chương trình dễ sử dụng, hiệu suất tối ưu để phù hợp với nhiều loại cấu hình máy tính.

1.3. Đối tượng nghiên cứu

- Các công nghệ:
  - Ngôn ngữ lập trình: Python.
  - Các thư viện chính: OpenCV, Pytesseract
- Đối tượng sử dụng:
  - Người dùng sử dụng hệ điều hành Windows 11.

1.4. Phạm vi nghiên cứu

- **Phạm vi môi trường:** Triển khai sản phẩm đề tài trên môi trường Windows.
- **Phạm vi chức năng:**
  - Nhận diện và đọc được biển số xe trong hình.

1.5. Phương pháp thực hiện

- Tìm hiểu công nghệ xây dựng chương trình bao gồm: Python, OpenCV, Pytesseract.
- Khảo sát các chương trình, mã nguồn hiện có trên thị trường, đặc biệt là các ứng dụng liên quan đến nhận diện biển số xe, từ đó tiến hành phân tích, xác định hướng đi cụ thể cho đề tài.
- Lập kế hoạch.
- Cài đặt.
- Triển khai và kiểm thử.
- Hoàn thiện sản phẩm.

Chương 2. CƠ SỞ LÝ THUYẾT

Ở chương này, nhóm tập trung vào việc giới thiệu về khái niệm biển số xe, hướng giải quyết bài toàn và các thư viện, công nghệ mà nhóm sử dụng để xây dựng chương trình.

2.1. Khái niệm biển số xe

Biển số xe hay còn gọi là biển kiểm soát xe cơ giới là tấm biển gắn trên mỗi xe cơ giới, được cơ quan nhà nước có thẩm quyền cụ thể là cơ quan công an cấp khi mua xe hoặc chuyển nhượng xe. Biển số xe được làm bằng hợp kim nhôm sắt, có dạng hình chữ nhật hoặc hơi vuông, trên đó có in những con số và chữ mang thông tin của chủ sở hữu. Vùng và địa phương quản lý, các con số cụ thể khi tra trên máy tính còn cho biết danh tính người chủ hay đơn vị đã mua nó, thời gian mua nó,…Đặc biệt trên đó còn có hình quốc huy dập nổi của Việt Nam.

Hình 2.1. Ảnh minh họa biển số xe

Quy định về kích thước: Đối với xe ô tô, biển số xe có hai loại, kích thước như sau: Loại biển hai hàng có chiều cao 165mm, chiều dài 330mm; Loại biển một hàng có chiều cao 110mm, chiều dài 520mm. Đối với xe mô tô, biển số xe là loại biển hai hàng có chiều cao 140mm, chiều dài 190mm.

Cấu trúc ký tự: Với biển số xe máy có trường dữ liệu thứ nhất là ký hiệu địa phương (hai số đầu) và số series. Dữ liệu thứ hai là số thứ tự xe đăng ký gồm 5 số, từ 000.01 đến 999.99. Hiện nay biển số xe máy đang được sử dụng có hai loại. Một là, trước 15/8/2023, biển màu trắng có series là một chữ cái kết hợp với một con số, từ 1 đến 9. Sau ngày 25/8/2023, series biển số xe máy màu trắng không phân biệt phân

khối sẽ có hai chữ cái. Seri biển số xe máy sử dụng lần lượt một trong 11 chữ cái sau đây: A, B, C, D, E, F, G, H, K, L, M.

Với biển số ô tô, hai số đầu là ký hiệu địa phương đăng ký, tiếp theo là series đăng ký và thứ tự xe gồm 5 chữ số từ 000.01 đến 999.99. Series biển số ôtô sử dụng lần lượt một trong 11 chữ cái A, B, C, D, E, F, G, H, K, L, M kết hợp với 1 chữ số tự nhiên từ 1 đến 9.

Thông qua những đặc điểm trên, ta có thể thiết lập nhưng thông số, điều kiện để chọn lọc được những đối tượng phù hợp, phục vụ cho việc tìm và tách biển số xe trong ảnh.

2.2. Hướng giải quyết bài toán

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.

Hình 2.2. Sơ đồ minh họa các bước thực hiện bài toán

2.3. OpenCV

Hình 2.3. Thư viện 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 hay 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 biển số xe đã phát hiện. Ngoài ra, OpenCV cung cấp một số kĩ thuật khác hỗ trợ cho việc nhận diện biển số xe như xoay biển số để làm thẳng biển số giúp việc nhận diện ký tự chính xác hơn, template matching cho phép phát hiện vật thể trong ảnh,…

2.3.1. 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. 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 giúp việc xử lý ảnh hiệu quả hơn khi 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)
```

Hình 2.4. Ảnh gốc

Hình 2.5. 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 được ứng dụng trong nhiều lĩnh vực. Kĩ thuật này được sử dụng để 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)
```

Hình 2.6. Code sử dụng hàm cv2.GaussianBlur()

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.

Hình 2.7. 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)
```

Hình 2.8. Code sử dụng hàm cv2.threshold

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.

Hình 2.9. 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)
```

Hình 2.10. Code thực hiện phép mở

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.

2.3.2. 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. Đường viền là tập hợp các điểm ranh giới xunh quanh của một đối tượng trong ảnh. 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)
```

Hình 2.11. Code sử dụng hàm cv2.findContours

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.

2.3.3. 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.

2.3.4. 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.

2.3.5. Xoay biển số

Sau khi tách được biển số xe ra khỏi ảnh gốc, ta có thể thực hiện kĩ thuật xoay biển số để đảm bảo biển số thẳng giúp bước nhận diện kí tự được chính xác hơn. Với vùng biển số được xác định trong ảnh sau khi áp dụng kĩ thuật phát hiện biên, ta sẽ tìm ra các cạnh của biển số và tạo ra một hình chữ nhật hoặc một đa giác gần đúng có bốn đỉnh. Các đỉnh này sẽ được lưu vào một mảng. Mỗi đỉnh là một tọa độ (x, y).

Kĩ thuật xoay biển số gồm bốn bước chính:

- **Sắp xếp các điểm để tìm 2 điểm có tọa độ y thấp nhất (tạo thành cạnh dưới của biển số):**

```
approx_sorted = sorted(approx[:, 0], key=lambda x: x[1])
```

Hình 2.12. Code thực hiện sắp xếp các điểm của đường viền biển số

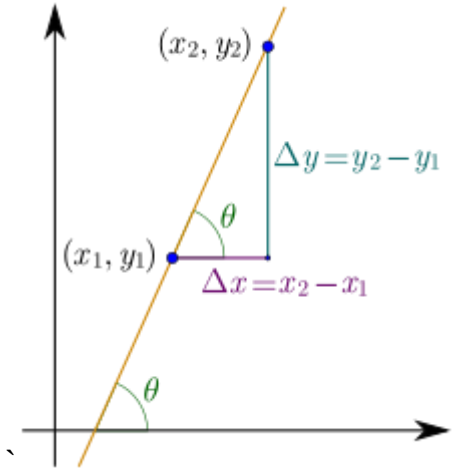
Trong đó:

- approx là mảng 2D chứa các điểm của đường viền biển số, mỗi điểm có dạng (x, y).
- approx[:, 0] để trích xuất tất cả các giá trị x của các điểm trong mảng approx, với : có nghĩa là lấy tất cả các dòng của mảng, 0 có nghĩa là lấy các giá trị ở cột đầu tiên (tọa độ x của các điểm)
- sorted() là hàm sắp xếp của Python, key=lambda x: x[1] có nghĩa là sắp xếp các phần tử trong mảng dựa trên giá trị x[1] (tọa độ y của các điểm).

Kết quả approx_sorted là một mảng chứa các tọa độ x của điểm trong đó được sắp xếp theo tọa độ y tăng dần và chọn ra hai điểm có giá trị y lớn nhất.

- **Tính toán độ dốc giữa hai điểm đã chọn và tính toán góc xoay:**
  - ❖ Tính toán độ dốc:

Khi biển số bị nghiêng, cạnh dưới của nó sẽ không song song với trục ngang (trục x). Việc tính toán độ dốc giúp chúng ta xác định được mức độ nghiêng của cạnh dưới của biển số so với trục ngang. Độ dốc (Slope) hay còn gọi là gradient của một đường thẳng trong không gian 2 chiều là tỉ lệ của sự gia tăng giữa hai điểm trên trục y của đường thẳng chia cho sự gia tăng giữa hai điểm trên trục x của đường thẳng đó:



Hình 2.13. Minh họa cách tính độ dốc trên trục tọa độ Ox, Oy

$$\text{Độ dốc} = \frac{\Delta y}{\Delta x} = \frac{y_2 - y_1}{x_2 - x_1}$$

Hình 2.14. Công thức tính toán độ dốc

Trong đó: (x1, y1) và (x2, y2) là hai điểm của đoạn thẳng mà ta cần tính độ dốc

❖ Code thực hiện tính toán độ dốc trong mã nguồn:

```
dx, dy = pt2[0] - pt1[0], pt2[1] - pt1[1]
```

Hình 2.15. Code thực hiện tính toán độ dốc

Trong đó:

- pt1 và pt2 là hai điểm nằm ở cạnh dưới của biển số có tọa độ y thấp nhất.
- pt1[0] và pt2[0] là các giá trị tọa độ x của hai điểm.
- pt1[1] và pt2[1] là các giá trị tọa độ y của hai điểm.
- Độ dốc = dx / dy.

❖ Code tính toán góc xoay dựa vào độ dốc đã có:

```
angle = math.degrees(math.atan2(dy, dx))
```

Hình 2.16. Code thực hiện tính toán góc xoay biển số

Hàm math.atan2(dy, dx) là một hàm thuộc thư viện math trong Python dùng để tính toán góc giữa trục x và một đoạn thẳng trong không gian 2 chiều. Hàm này trả về giá trị góc theo đơn vị radian từ $-\pi$ đến π. Hàm math.degrees(x) chuyển đổi giá trị góc từ radian sang độ theo công thức 1 radian = 180/π độ.

Ngoài ra, ta cần thực hiện thêm bước điều chỉnh góc để góc xoay luôn nằm trong khoảng -45 đến 45 độ:

```
if -180 < angle < -135:
    angle += 180
elif 135 < angle < 180:
    angle -= 180
```

Hình 2.17. Code thực hiện điều chỉnh góc xoay

- **Tiến hành xoay ảnh dựa trên góc tìm được:**

❖ Tính toán tâm của ảnh: Lấy ra chiều cao h và chiều rộng w của ảnh.

```
(h, w) = image.shape[:2]
```

Hình 2.18. Code thực hiện lấy chiều cao và rộng của ảnh

❖ Tính tâm của ảnh: Tâm của ảnh là điểm trung tâm của nó, có tọa độ là (w // 2, h //2), chính là điểm mà ảnh sẽ xoay quanh.

```
center = (w // 2, h // 2)
```

Hình 2.19. Code thực hiện tính toán tâm ảnh

❖ Tính toán ma trận xoay:

Xoay là một phép biến đổi hình học mà mỗi điểm của đối tượng sẽ di chuyển quanh một điểm cố định (thường là điểm trung tâm). Sau khi thực hiện phép xoay một đối tượng quanh một điểm xoay, thì khoảng cách giữa các điểm trong đối tượng và điểm xoay không bị thay đổi.

Ma trận xoay là một phép biến đổi bảo toàn khoảng cách, có nghĩa là mỗi điểm được dịch chuyển theo một quy tắc toán học mà không làm thay đổi tỷ lệ hoặc độ dài của các đoạn thẳng giữa các điểm trong đối tượng. Phép biến đổi này không thay đổi kích thước hoặc hình dạng, mà chỉ thay đổi hướng của đối tượng.

❖ Ma trận xoay trong 2D:

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}$$

Hình 2.20. Ma trận xoay

Trong đó, cos(θ) và sin(θ) là các giá trị lượng giác của góc xoay θ. Việc áp dụng ma trận này làm thay đổi tọa độ của điểm được xoay mà không làm thay đổi độ dài của vector từ điểm trung tâm (điểm xoay) đến điểm đó. Giả sử nếu ta xoay một điểm (x, y) một góc θ thì tọa độ mới của điểm này như sau:

$$x' = x\sin(|\theta|) + y\cos(|\theta|). \text{ And } y' = x\cos(|\theta|) - y\sin(|\theta|)$$

Hình 2.21. Tọa độ mới của điểm (x, y) sau khi xoay

Viết dưới dạng ma trận:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \sin(|\theta|) & \cos(|\theta|) \\ \cos(|\theta|) & -\sin(|\theta|) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Hình 2.22. Viết dưới dạng ma trận

Sau khi áp dụng phép xoay, khoảng cách từ điểm gốc (hay điểm xoay) đến điểm mới không bị thay đổi hay kích thước của đối tượng được giữ nguyên.

Khi xử lý ảnh biển số xe, việc sử dụng ma trận xoay giúp ta xoay ảnh quanh điểm trung tâm ảnh để làm thẳng hoặc thay đổi hướng mà không làm ảnh bị bị méo mó hoặc mất chi tiết. OpenCV cung cấp hàm cv2.getRotationMatrix2D(center, angle, scale) để tính toán ma trận xoay. Hàm này sẽ tạo ra một ma trận xoay affine 2x3, với center là tọa độ của điểm xoay ảnh, angle là góc xoay (đơn vị độ), scale là tỷ lệ phóng to hoặc thu nhỏ ảnh (thường là 1.0 để không thay đổi kích thước).

```
rotation_matrix = cv2.getRotationMatrix2D(center, angle, 1.0)
```

Hình 2.23. Code sử dụng hàm cv2.getRotationMatrix2D

Ma trận xoay affine là một dạng mở rộng của ma trận xoay. Ma trận xoay affine kết hợp phép xoay với phép dịch chuyển (translation) và thu nhỏ/phóng to (scaling). Ma trận xoay affine là một ma trận 2x3, có khả năng thực hiện phép xoay, dịch chuyển và thay đổi tỷ lệ trong một phép biến đổi duy nhất.

Ma trận xoay affine 2x3 có dạng như sau:

$$\begin{pmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \end{pmatrix}$$

Hình 2.24. Ma trận xoay affine 2x3

Trong đó, t_x = (1 - cos(θ)) * c_x + sin(θ) * c_y và t_y = (1 - cos(θ)) * c_y - sin(θ) * c_x, với c_x và c_y là tọa độ của điểm xoay, là các giá trị dịch chuyển (translation) giúp dịch chuyển ảnh sau khi xoay, đảm bảo trong quá trình xoay ảnh sẽ không bị mất phần nào. Gia sử ta áp dụng ma trận xoay affine cho một điểm có tạo độ (x,), ta có tọa độ của điểm mới (x', y') *như sau*:

$$x' = x \cdot \cos(\theta) - y \cdot \sin(\theta) + t_x$$

$$y' = x \cdot \sin(\theta) + y \cdot \cos(\theta) + t_y$$

Hình 2.25. Tọa độ mới của điểm (x, y) sau khi áp dụng ma trận xoay affine

Sau khi tính toán được ma trận xoay, sử dụng hàm cv2.warpAffine để áp dụng ma trận này lên ảnh. Hàm này sẽ xoay toàn bộ ảnh theo góc và dịch chuyển các điểm ảnh theo ma trận.

```
rotated_image = cv2.warpAffine(image, rotation_matrix, (image.shape[1], image.shape[0]))
```

Hình 2.26. Code sử dụng hàm cv2.warpAffine

Trong đó:

- image: là ảnh gốc cần xoay
- rotation_matrix: là ma trận affine 2x3 được tính toán từ hàm cv2.getRotationMatrix2D
- (image.shape[1], image.shape[0]): là kích thước của ảnh mới sau khi xoay.
- image.shape[1] là chiều rộng ảnh và image.shape[0] là chiều cao ảnh.

2.3.6. Template matching OpenCV

Template Matching là một kỹ thuật trong xử lý ảnh giúp tìm kiếm một phần ảnh (ảnh mẫu) trong ảnh lớn hơn và xác định vị trí của nó. Mục tiêu là xác định vị trí, kích thước, và sự tương đồng giữa một ảnh đầu vào (input image) và các ảnh mẫu (template) đã cho trước. Kĩ thuật này thường được sử dụng trong các bài toán nhận diện đối tượng, nhận diện ký tự và nhận diện biển số xe.

Các bước thực hiện bao gồm:

- Chọn mẫu (Template): là hình ảnh mà ta muốn tìm kiếm trong ảnh đầu vào, là một danh sách các ký tự hoặc đối tượng.
- Chạy Template Matching: Sau khi có danh sách mẫu, ta "quét" mẫu trên toàn bộ ảnh đầu vào và tính toán mức độ tương tự giữa mẫu và từng vùng con của ảnh. Để tính được mức độ tương tự này ta thường sử dụng một số phương pháp như: phép đo độ tương quan (correlation), hàm cv2.TM_CCOEFF_NORMED trong OpenCV,…
- Kết quả: Sau khi hoàn thành việc kiểm tra toàn bộ ảnh, mỗi vùng trong ảnh đầu vào sẽ có một giá trị tương tự với mẫu. Giá trị này cho biết mức độ

khớp. Ta có thể lọc kết quả và lấy các khu vực có độ tương tự cao hơn một ngưỡng nhất định.

- OpenCV cung cấp hàm cv2.matchTemplate() giúp so độ khớp giữa mẫu và ảnh.

```
result = cv2.matchTemplate(image, template, cv2.TM_CCOEFF_NORMED)
```

Hình 2.27. Code sử dụng hàm cv2.matchTemplate()

Trong đó:

- image là một ảnh vùng con của ảnh gốc
- template là mẫu cần so độ khớp với vùng ảnh.
- cv2.TM_CCOEFF_NORMED là phương pháp so độ khớp giữa mẫu và ảnh mà mã nguồn sử dụng. Phương pháp này tính toán sự tương đồng giữa mẫu và vùng ảnh bằng cách chuẩn hóa độ tương quan. OpenCV cung cấp nhiều phương pháp so khớp khác nhau như:
    - cv2.TM_CCOEFF_NORMED: Độ tương quan chuẩn hóa.
    - cv2.TM_CCOEFF: Độ tương quan không chuẩn hóa.
    - cv2.TM_CCORR_NORMED: Độ tương đồng (correlation) chuẩn hóa.
    - cv2.TM_CCORR: Độ tương đồng không chuẩn hóa.
    - cv2.TM_SQDIFF_NORMED: Bình phương sai số chuẩn hóa.
    - cv2.TM_SQDIFF: Bình phương sai số không chuẩn hóa.

Với bài toán nhận diện biển số xe, các dữ liệu ảnh biển số xe có sự thay đổi độ sáng do bị ảnh hưởng từ môi trường bên ngoài khá nhiều nên nhóm em lựa chọn hàm cv2.TM_CCOEFF_NORMED vì là một trong những phương pháp phổ biến nhất khi sử dụng template matching, vì nó chuẩn hóa độ tương quan giữa mẫu và ảnh, nghĩa là giúp loại bỏ ảnh hưởng của độ sáng và độ tương phản của ảnh, làm cho kết quả ít bị tác động bởi các thay đổi về ánh sáng. Giả sử vùng ảnh có độ sáng cao hơn mẫu

hoặc vùng ảnh và mẫu có độ tương phản khác nhau, nếu không chuẩn hóa thì chỉ số tương quan sẽ bị lệch và không phản ánh đúng sự tương đồng giữa mẫu và vùng ảnh.

Kết quả trả về một ma trận bao gồm các giá trị từ -1 đến 1. Giá trị 1 nghĩa là mẫu và vùng ảnh tại vị trí đó hoàn toàn khớp với nhau, ngược lại giá trị -1 nghĩa là mẫu và vùng ảnh không khớp nhau.

Sau khi hàm cv2.matchTemplate() thực hiện so khớp và trả về ma trận điểm số, tiếp theo ta sử dụng cv2.minMaxLoc() để tìm ra vị trí có điểm số tốt nhất.

```
_, score, _, _ = cv2.minMaxLoc(result)
```

Hình 2.28. Code sử dụng hàm cv2.minMaxLoc()

Hàm cv2.minMaxLoc trả về bốn giá trị: Giá trị nhỏ nhất (_), giá trị lớn nhất (score), vị trí của giá trị nhỏ nhất (_), và vị trí của giá trị lớn nhất (_). Ta chỉ lấy giá trị lớn nhất, điểm số càng cao thì sự tương đồng càng lớn.

2.4. 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.

2.5. Pytesseract

Hình 2.29. Tesseract OCR

Pytesseract là một thư viện Python giúp người dùng sử dụng công cụ Tesseract OCR (Optical Character Recognition) 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.

Quá trình nhận dạng văn bản của **Tesseract** OCR có thể được chia thành các bước chính sau:

- ❖ Tiền xử lý ảnh để tối ưu hóa quá trình nhận diện trước khi thực hiện nhận dạng:
  - Chuyển Đổi Sang Grayscale: Tesseract thường chuyển ảnh màu sang ảnh đen trắng (grayscale) để đơn giản hóa quá trình nhận dạng, vì các yếu tố màu sắc không quan trọng đối với OCR.
  - Ngưỡng Hóa (Thresholding): Sau khi chuyển sang grayscale, Tesseract áp dụng phương pháp ngưỡng hóa (Otsu's thresholding) cho ảnh để chuyển thành ảnh nhị phân (đen và trắng).

- Làm Mượt (Smoothing): Một số hình ảnh có thể chứa nhiều nhiễu, do đó Tesseract có thể áp dụng các phương pháp làm mượt (Gaussian blur) để làm giảm sự ảnh hưởng của nhiễu.

❖ Phân Đoạn Văn Bản (Text Segmentation):

Phân đoạn là một bước quan trọng trong quá trình nhận diện văn bản. Tesseract cần phân tách ảnh thành các phần nhỏ như các dòng, từ và ký tự để dễ xử lý hơn. Quá trình phân đoạn của Tesseract có thể được chia thành các bước sau:

- Nhận diện vùng văn bản:

Bước đầu tiên là xác định vùng văn bản trong ảnh, các đối tượng có khả năng chứa văn bản, ví dụ như các đoạn văn bản, các ô hoặc các bảng. Tesseract sử dụng các phương pháp như connected component analysis (phân tích thành phần liên kết) để phát hiện các vùng có khả năng chứa văn bản. Connected Component Analysis (CCA) là quá trình phân tích các thành phần liên kết trong một ảnh nhị phân (ảnh đen trắng). Các thành phần liên kết là các tập hợp pixel có giá trị giống nhau (thường là pixel trắng trong ảnh nhị phân) và có liên kết với nhau theo một hướng (hàng, cột hoặc chéo). CCA thực hiện việc nhóm các pixel có giá trị giống nhau lại thành các khu vực liên kết trong ảnh. Mỗi khu vực liên kết này có thể là một ký tự, một từ hoặc một phần của văn bản.

- Phân đoạn dòng (Line segmentation):

Sau khi các vùng có thể chứa văn bản được xác định, Tesseract tiến hành phân đoạn thành các dòng văn bản. Quá trình phân đoạn này thực hiện bằng cách tìm các đường kẻ (line separators) giữa các dòng, thường là những khoảng trống hoặc sự thay đổi trong khoảng cách giữa các ký tự, các từ hoặc các đoạn văn.

- Phân đoạn từ (Word segmentation):

Sau khi phân đoạn thành các dòng, Tesseract sẽ phân đoạn dòng văn bản thành các từ. Phương pháp phân đoạn từ thường dựa vào khoảng cách giữa các ký tự và khoảng cách giữa các từ trong văn bản.

- Phân đoạn ký tự (Character segmentation):

Cuối cùng là tách các ký tự trong mỗi từ. Mỗi ký tự trong một từ sẽ được tách riêng biệt để thực hiện nhận diện. Tesseract sử dụng thuật toán phân đoạn ký tự để xác định ranh giới ký tự, tính toán hộp bao quanh (bounding boxes) để xác định ranh giới các ký tự trong ảnh và cắt ký tự.

❖ Nhận Diện Ký Tự (Character Recognition)

Tesseract sử dụng các phương pháp học máy để nhận diện các ký tự trong ảnh. Từ phiên bản Tesseract 4.x, Tesseract sử dụng LSTM (Long Short-Term Memory),

là mô hình được huấn luyện với dữ liệu văn bản lớn và có khả năng học các mối quan hệ phức tạp giữa các ký tự trong văn bản. Tesseract sử dụng LSTM để nhận diện ký tự và các từ trong một ngữ cảnh dài và có thể xử lý các văn bản bao gồm các ký tự méo mó hoặc bị mờ.

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'
```

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:

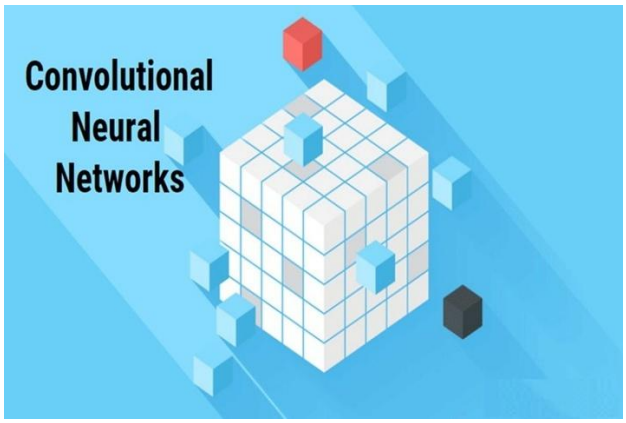
- psm 0 - Orientation and Script Detection (OSD): Phát hiện hướng và loại ngôn ngữ của văn bản trong ảnh.
- psm 1 - Automatic page segmentation with OSD: Tự động phân đoạn trang và nhận diện hướng văn bản.

- psm 6 - Single uniform block of text: Nhận diện một khối văn bản đồng nhất (không chia cột).
- psm 7 - Treat the image as a single text line: Nhận diện một dòng văn bản duy nhất.
- psm 8 - Treat the image as a single word: Nhận diện một từ đơn lẻ.
- psm 9 - Single word in a circle: Nhận diện một từ trong hình tròn.
- psm 10 - Treat the image as a single character, v.v. Nhận diện một ký tự duy nhất.
- 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: Chỉ nhận diện những ký tự được chỉ định trong whitelist giúp giới hạn phạm vi kí tự nhận diện, giúp cái thiện độ chính xác.

Sau khi xử lý, hàm này trả về một chuỗi văn bản được nhận diện từ ảnh đó.

2.6. CNN



Hình 2.30. Thuật toán 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.

2.7. 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 4.2 Quá trình thực hiện.

Chương 3. PHÂN TÍCH CHƯƠNG TRÌNH

Ở chương này, nhóm tập trung vào việc trình bày phân tích các kĩ thuật giúp thực hiện chức năng của chương trình.

3.1. Kĩ thuật tách biển số xe

3.1.1. Sử dụng nhận diện viền vật thể để tách

Thư viện OpenCV2 hỗ trợ các phương thức để vẽ đường viền cho các vật thể trong ảnh. Sử dụng ảnh sau khi được vẽ đường viền để lọc tìm khu vực chứa biển số.

Bước 1: Xử lí ảnh

- Chuyển ảnh sang dạng xám (gray)
- Làm mờ ảnh để khử nhiễu và làm mềm cạnh viền
- Chuyển ảnh sang dạng trắng đen (threshold)
- Chuẩn hóa ảnh (morphological opening)

Hình 3.1. Ảnh minh hoạ sau khi được chuẩn hóa

Bước 2: Vẽ đường viền và lọc khu vực

- Vẽ đường viền trên ảnh
- Lọc khu vực có diện tích nhỏ hơn 10000 (pixel x pixel)

Hình 3.2. Ảnh minh họa sau khi vẽ viền xanh quanh vật thể

Bước 3: Tách biển số

- Chuyển khu vực từ bước 2 sang dạng hình học
- Lọc lấy những hình có 4 cạnh
- Tách khu vực vừa lọc ra thành ảnh mới

Hình 3.3. Ảnh minh họa biển số sau khi tách

3.1.2. Các cải tiến khác

3.1.2.1. Tỉ lệ biển số

Tỉ lệ biển số xe ở Việt Nam (ngang/ dọc) khi được chụp ảnh thường nằm trong khoảng 1.2 đến 1.8 (đối với biển số 2 dòng) hoặc 3 đến 7.5 (đối với biển số 1 dòng).

3.1.2.2. Xoay biển số

Đa phần các ảnh biển số được chụp sẽ không nằm thẳng đứng mà sẽ bị lệch vài độ sang trái hoặc phải. Việc xoay biển số để biển số trở về dạng thẳng đứng sẽ giúp lọc khu vực xuất hiện biển số chính xác hơn.

Bước 1: Sắp xếp các đỉnh của khu vực biển số để tìm 2 điểm có tọa độ y thấp nhất

Bước 2: Tính góc tạo được từ 2 điểm đó và trục Ox

Bước 3: Điều chỉnh để góc xoay luôn nằm trong khoảng -45 đến 45 độ

Bước 4: Tiến hành xoay ảnh dựa trên góc tìm được

3.2. Kĩ thuật đọc kí tự từ biển số

3.2.1. Sử dụng pytesseract để đọc chuỗi kí tự

Thư viện pytesseract hỗ trợ chức năng đọc chuỗi kí tự trong ảnh và cho phép chỉ đọc những kí tự tùy chọn.

Bước 1: Chuyển ảnh sang dạng xám rồi chuyển sang dạng trắng đen threshold

Bước 2: Chọn bộ kí tự để đọc, bao gồm chữ cái từ A đến Z, chữ số từ 0 đến 9. Do mô hình mặc định pytesseract sử dụng là tiếng Anh nên kí tự đặc biệt như Đ (trong biển số xe máy điện MĐ) sẽ không đọc được. Còn nếu sử dụng mô hình bằng tiếng Việt thì độ chính xác tổng thể sẽ giảm xuống.

Bước 3: Tiến hành đọc chuỗi kí tự trong hình bằng phương thức image_to_string mà pytesseract hỗ trợ

3.2.2. Sử dụng pytesseract để đọc từng kí tự

Đôi khi việc đọc một chuỗi kí tự sẽ cho độ chính xác không cao, thay vào đó có thể tách từng kí tự trong biển số ra và dùng pytesseract đọc từng kí tự một.

Bước 1: Chuyển ảnh sang dạng xám rồi chuyển thành dạng trắng đen threshold

Bước 2: Vẽ viền các vật thể trong ảnh

Bước 3: Tiến hành lọc với từng khu vực kí tự tìm được bằng tỉ lệ diện tích (từ 1% đến 9% so với hình), tỉ lệ kích thước (chiều rộng/ chiều cao trong khoảng 0.25 đến 0.7)

Bước 3: Lọc khu vực kí tự bằng cách loại bỏ những khu vực có chiều cao lệch hơn 10% so với chiều cao trung vị

Bước 4: Sắp xếp khu vực kí tự theo dòng, sau đó sắp xếp theo tọa độ x

Bước 5: Dùng pytesseract để đọc kí tự trong khu vực kí tự tìm được

Bước 6: Gộp các kí tự đọc được thành kết quả cuối cùng

Hình 3.4. Ảnh minh họa việc tách các kí tự riêng lẻ

3.2.3. Sử dụng phương pháp template matching để đọc kí tự

OpenCV hỗ trợ chức năng template matching để kiểm tra khu vực trong ảnh khớp với ảnh mẫu (template). Sử dụng template matching để đọc từng kí tự trong biển số xe.

Bước 1: Xử lí ảnh và tách ra từng khu vực kí tự tương tự như bước 1 đến 4 ở phương pháp dùng pytesseract để đọc từng kí tự

Bước 2: Dùng template matching với các khu vực kí tự được tách ở bước 1 để nhận diện kí tự có trong bộ kí tự của biển số xe Việt Nam

Hình 3.5. Ảnh minh hoạ template matching (bên trái là template, bên phải là ảnh chứa khu vực kí tự)

Chương 4. XÂY DỰNG ỨNG DỤNG

Ở chương này, nhóm tập trung vào việc mô tả lại quá trình thực hiện chương trình theo trình tự thời gian.

4.1. Mã nguồn

Link GitHub sản phẩm: https://github.com/BowJaro/Seminar-Modern-Technologies.git

Thành viên trong GitHub:

- Phạm Phước Huy (BowJaro)
- Trần Văn Thanh Tâm (tamxtanh)

4.2. Quá trình thực hiện

4.2.1. Giai đoạn tìm hiểu thông tin

Thời gian thực hiện: 16-09-2024 đến 30-09-2024

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.

4.2.2. Giai đoạn xây dựng chương trình cơ bản

Thời gian thực hiện: 01-10-2024 đến 05-11-2024

a) Chọn công nghệ nhận dạng biển số

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. 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.

b) Chọn công nghệ nhận dạng kí tự

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. 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. 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.

4.2.3. Giai đoạn cải tiến chương trình

Thời gian thực hiện: 01-01-2025 đến 31-01-2025

Chương trình sau khi được viết ở giai đoạn trước thì cơ bản đã có thể đọc được biển số xe tuy nhiên độ chính xác còn rất thấp. Vậy nên nhóm đã tìm hiểu thêm một số hướng để cải thiện độ chính xác. Các phương pháp cải thiện độ chính xác mà nhóm đã thực hiện là xoay biển số, sử dụng template matching của OpenCV, đồng thời kết hợp các phương pháp lại với nhau.

Phương pháp xoay biển số...

Phương pháp template matching...

Tuy nhiên, sau khi thử nghiệm một số giải pháp cải tiến thì trên thực tế độ chính xác lại không cao như trước. Nhóm đưa ra một số giả định như sau:

- Loại ảnh không đồng nhất: Các ảnh mà nhóm chọn nhìn chung được chụp từ camera điện thoại với kích thước, màu sắc chủ đạo, độ sáng, góc chụp... không giống nhau. Điều này dẫn đến việc khó có công thức chung cho việc xác định được chính xác đâu là biển số và đọc kí tự từ biển số đó.
- Chất lượng ảnh không cao: Các ảnh mà nhóm chọn làm mẫu thử thường không có độ phân giải và màu sắc rõ ràng để máy có thể nhận diện dễ dàng.

4.2.4. Giai đoạn tìm hiểu lí thuyết và viết báo cáo

Thời gian thực hiện: 06-11-2024 đến 06-02-2025

Ở giai đoạn này, nhóm tập trung vào việc tìm hiểu lí thuyết của các thư viện, các hàm đã sử dụng trong chương trình. Nhóm chủ yếu tập trung tìm hiểu lí thuyết và các hàm được cung cấp bởi hai thư viện là OpenCV và Pytesseract. Các tìm hiểu thường dừng lại ở mức tìm nguyên lí hoạt động chứ không tập trung sâu vào các thuật toán hay việc huấn luyện mô hình AI. Đồng thời, nhóm bắt đầu viết báo cáo và làm file báo cáo thuyết trình.

KẾT LUẬN

Sau khoảng thời gian một học kì thực hiện đồ án, nhóm em đã hoàn thành chương trình nhận dạng biển số xe ở Việt Nam với những ưu điểm, hạn chế cũng như hướng phát triển như sau:

❖ Ưu điểm

Ứng dụng nhìn chung có các ưu điểm sau đây:

- Hiệu suất xử lí nhanh, không tốn nhiều tài nguyên máy tính.
- Đơn giản, dễ sử dụng, dễ triển khai trên nhiều nền tảng, thiết bị.
- Có thể xử lí ảnh với đuôi ảnh jpg hoặc png.

❖ Hạn chế

Do giới hạn về thời gian và nguồn lực, ứng dụng vì thế vẫn còn một số hạn chế:

- Độ chính xác chưa cao, nhận dạng kí tự còn bị nhầm lẫn.
- Giao diện người dùng chưa thân thiện, vẫn còn hoạt động trên terminal.
- Chưa nghiên cứu sâu về thuật toán của các thư viện.

❖ Hướng phát triển

Trong tương lai, ứng dụng có thể được định hướng phát triển theo nhiều khía cạnh:

- Tối ưu hóa hiệu suất của ứng dụng bằng cách các kĩ thuật tối ưu code Python để có thể xử lí nhiều ảnh hoặc video.
- Cải thiện độ chính xác của việc nhận diện kí tự bằng việc tự huấn luyện mô hình đọc chữ riêng phù hợp với font chữ, kí tự của biển số ở Việt Nam.
- Sử dụng template matching với các ảnh mẫu được lấy từ thực tế ở một góc chụp cố định (ví dụ như bãi giữ xe) để tăng cường độ chính xác của chương trình.
- Sử dụng thêm các biện pháp xử lí ảnh để phù hợp hơn với việc nhận diện biển số xe.

– Cải thiện giao diện người dùng sang dạng cửa sổ ứng dụng, hỗ trợ một số chức năng như đọc biển số từ camera theo thời gian thực, đọc biển số từ video, hình ảnh cho trước.

TÀI LIỆU THAM KHẢO

1. Mrzaizai2k. Vietnamese License Plate Recognition in 3 steps using OpenCV and KNN. Truy cập từ https://www.youtube.com/watch?v=7erlCp6d5w8 (truy cập lần cuối vào 11:25 ngày 06-02-2025)

2. DSwithBappy. Real Time Car Number Plates Extraction in 30 Minutes | OpenCV Python | Computer Vision | EasyOCR. Truy cập từ https://www.youtube.com/watch?v=ltpnWBBT7NI (truy cập lần cuối vào 11:25 ngày 06-02-2025)

3. Nisala Nimesh. (n.d). Automatic Number Plate Detection OpenCV and Py Tesseract Full Project. Truy cập từ https://www.youtube.com/watch?v=o_F0kJqt7EU (truy cập lần cuối vào 11:20 ngày 26-12-2024)

4.Python Geeks. Python OpenCV Detect and Recognize Car License Plate. Truy cập từ https://pythongeeks.org/python-opencv-detect-and-recognize-car-license-plate/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

5. Python Tutorials for Digital Humanities. OCR in Python Tutorials. Truy cập từ https://youtube.com/playlist?list=PL2VXyKi-KpYuTAZz__9KVl1jQz74bDG7i&si=XncQohHeWvu_f5f6 (truy cập lần cuối vào 11:25 ngày 06-02-2025)

6.Python Geeks. Python OpenCV Detect and Recognize Car License Plate. Truy cập từ https://pythongeeks.org/python-opencv-detect-and-recognize-car-license-plate/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

7. Leo Ertuna. How to automatically deskew (straighten) a text image using OpenCV. Truy cập từ https://becominghuman.ai/how-to-automatically-deskew-straighten-a-text-image-using-opencv-a0c30aed83df (truy cập lần cuối vào 11:25 ngày 06-02-2025)

8. mrzaizai2k. License Plate Recognition YOLOv7 and CNN. Truy cập từ https://github.com/mrzaizai2k/License-Plate-Recognition-YOLOv7-and-CNN.git (truy cập lần cuối vào 11:25 ngày 06-02-2025)

9. Stack overflow. Understanding and evaluating template matching methods. https://stackoverflow.com/questions/58158129/understanding-and-evaluating-template-matching-methods (truy cập lần cuối vào 11:25 ngày 06-02-2025)

10. Geeks for Geeks. Template matching using OpenCV in Python. https://www.geeksforgeeks.org/template-matching-using-opencv-in-python/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

11. OpenCV. Template matching. https://docs.opencv.org/3.4/d4/dc6/tutorial_py_template_matching.html (truy cập lần cuối vào 11:25 ngày 06-02-2025)

12. Luatduongia. Biển số xe Việt Nam.

https://luatduonggia.vn/bien-so-xe-la-gi-cac-loai-bien-so-xe-va-bang-tra-cuu-bien-so-xe-cua-cac-tinh-thanh/#google_vignette (truy cập lần cuối vào 11:25 ngày 01-01-2025)

14. Pyimagesearch. OpenCV: Automatic License/Number Plate Recognition (ANPR) with Python.

https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

15. Freecodecamp. Detecting Objects in Images Using the YOLOv8 Neural Network.

https://www.freecodecamp.org/news/how-to-detect-objects-in-images-using-yolov8/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

16. Geeksforgeeks. Convolution Neural Network.

https://www.geeksforgeeks.org/introduction-convolution-neural-network/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

17. Nanonets. Python OCR Tutorial: Tesseract, Pytesseract, and OpenCV.

https://nanonets.com/blog/ocr-with-tesseract/ (truy cập lần cuối vào 11:25 ngày 06-02-2025)

18. Wikipedia. Slope.

https://en.wikipedia.org/wiki/Slope (truy cập lần cuối vào 11:25 ngày 06-02-2025)