Chương 1 NGHIÊN CỨU TỔNG QUAN

Ứng dụng của project trong thực tế :

- Hệ thống nhận diện khuôn mặt thông qua các camera được gắn tại sân bay và các tòa nhà
- Các trợ lý ảo (như Siri, Google Assistant, Alexa) có khả năng nghe, hiểu, trả lời và làm việc cho mình
- Những ứng dụng trong y sinh, dùng AI để chẩn đoán bệnh dựa trên phim chụp X-quang, X-ray và MRI
- Quen thuộc hơn thì có những dòng smart TV, áp dụng công nghệ AI để cải tiến chất lượng hình ảnh hoặc nhận diện giọng nói…

1.1 Tổng quan bài toán nhận dạng biển số xe

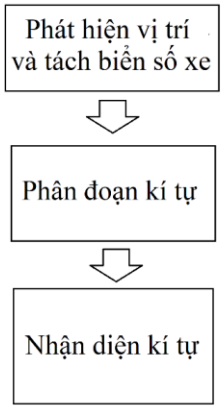
1.1.1 Xử lý ảnh và Open CV

OpenCV (Open Computer Vision) là một thư viện mã nguồn mở hàng đầu cho xử lý về thị giác máy tính, machine learning, xử lý ảnh. OpenCV đươc viết bằng C/C++, vì vậy có tốc độ tính toán rất nhanh, có thể sử dụng với các ứng dụng liên quan đến thời gian thực.

1.1.2 Hướng giải quyết

Hiện nay trên thế giới đã có rất nhiều cách tiếp cận khác nhau với việc nhận dạng biển số xe, tuy nhiên trong phạm vi project này nhóm em sẽ giải quyết vấn đề theo 3 bước chính:

1. Phát hiện vị trí và tách biển số xe từ một hình ảnh có sẵn từ đầu vào
2. Phân đoạn các kí tự có trong biển số xe
3. Nhận diện các kí tự đó rồi đưa về mã ASCII



Hình 2.1 Các bước chính trong nhận dạng biển số xe

Chương 2 PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1 Phân tích yêu cầu

2.1.1 Yêu cầu chức năng

- Có thể đọc dữ liệu ảnh đầu vào
- Tách được biển số từ video và ảnh đầu vào
- Đọc biển số đã tách
- Hiển thị biển số nhận dạng được ra màn hình
- Có giao diện dễ sử dụng

2.2 Biểu đồ ca sử dụng

2.2.1 Chức năng đưa dữ liệu vào

- Các tác nhân: Người dùng
- Điều kiện trước: Người dùng phải chọn ảnh đầu vào

- Điều kiện sau: Hệ thống nhận được ảnh
- Mô tả: Ca sử dụng này người dùng thực hiện nạp dữ liệu đầu vào và hệ thống đọc được dữ liệu ảnh

2.2.2 Chức năng đưa ra ảnh đã threshold

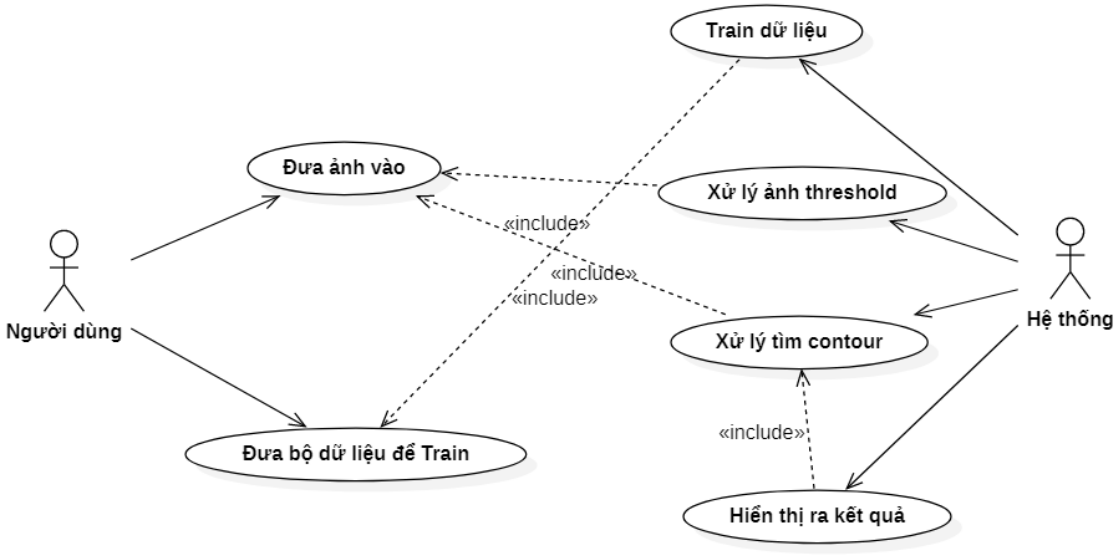
- Các tác nhân: hệ thống
- Điều kiện trước: Có ảnh đầu vào
- Điều kiện sau: Hệ thống đưa ra bức ảnh đã threshold
- Mô tả: Ca sử dụng này nhằm đưa ra ảnh sau khi threshold để dễ dàng nhận dạng biển số xe

2.2.3 Chức năng đưa ra các contour tìm được

- Các tác nhân: Hệ thống
- Điều kiện trước: Ảnh đã qua xử lý và được tách biển số ra
- Điều kiện sau: Hiển thị các contour tìm được
- Mô tả: Ca sử dụng này nhằm đưa ra các contour tìm được để nhận dạng dễ dàng

2.2.4 Chức năng nhận dạng và hiển thị biển số xe

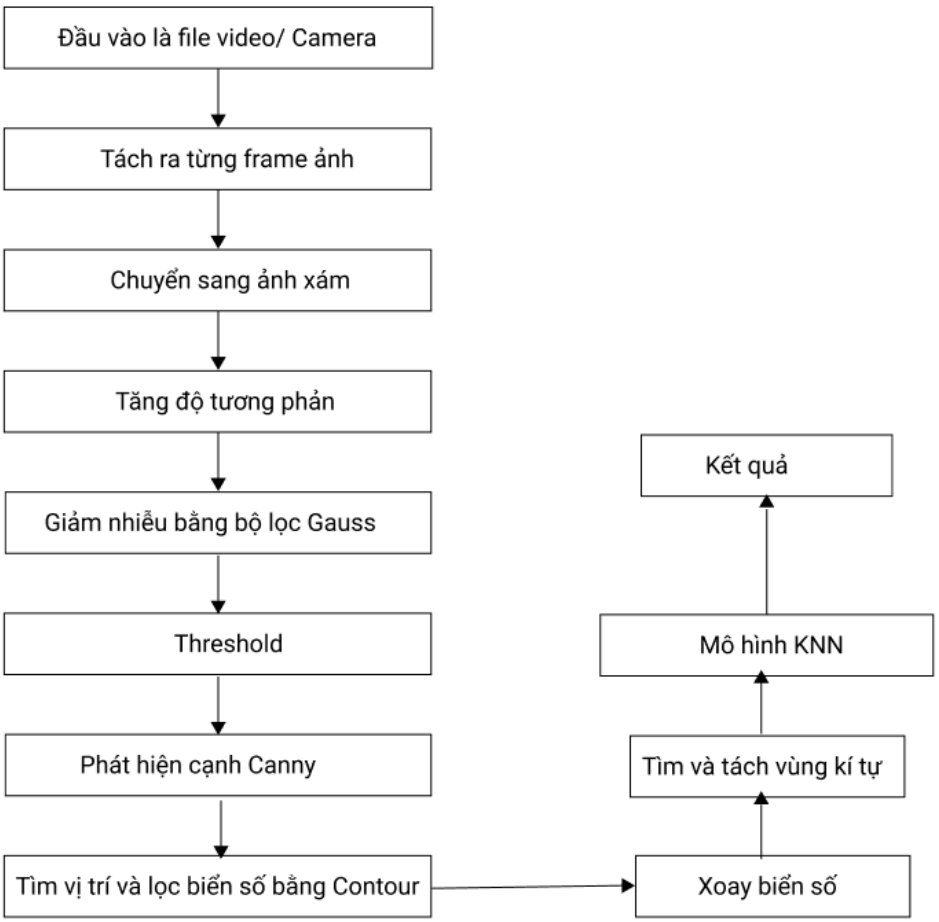
- Các tác nhân: Hệ thống
- Điều kiện trước: Ảnh đã qua xử lý
- Điều kiện sau: Hệ thống đưa ra kết quả sau khi nhận dạng và in lên ảnh gốc
- Mô tả: Đưa ra kết quả nhận dạng được



Hình 3.1 Biểu đồ use-case

2.3 Phát hiện vị trí và tách biển số xe

2.3.1 Hướng giải quyết



Hình 3.2 Lưu đồ nhận dạng biển số xe

Đầu tiên từ ảnh đưa vào ta sẽ cắt ra được từng frame ảnh để xử lý tách biển số. Sau đó ta sẽ chuyển ảnh sang màu xám. Tiếp theo ta tăng độ tương phản với hai phép toán hình thái học Top Hat và Black Hat để làm nổi bật thêm biển số giữa phông nền, hỗ trợ cho việc xử lý nhị phân sau này. Sau đó, ta giảm nhiễu bằng bộ lọc Gauss để loại bỏ những chi tiết nhiễu có thể gây ảnh hưởng đến quá trình nhận diện, đồng thời làm tăng tốc độ xử lý.

Việc lấy ngưỡng sẽ giúp ta tách được thông tin biển số và thông tin nền, ở đây nhóm em chọn lấy ngưỡng động (Adaptive Threshold). Tiếp đó ta sử dụng thuật toán phát hiện cạnh Canny để trích xuất những chi tiết cạnh của biển số. Trong quá trình xử lý máy tính có thể nhầm lẫn biển số với những chi tiết nhiễu, việc lọc lần cuối bằng các tỉ lệ cao/rộng hay diện tích của biển số sẽ giúp xác

định được đúng biển số. Cuối cùng, ta sẽ xác định vị trí của biển số trong ảnh bằng cách vẽ Contour bao quanh.

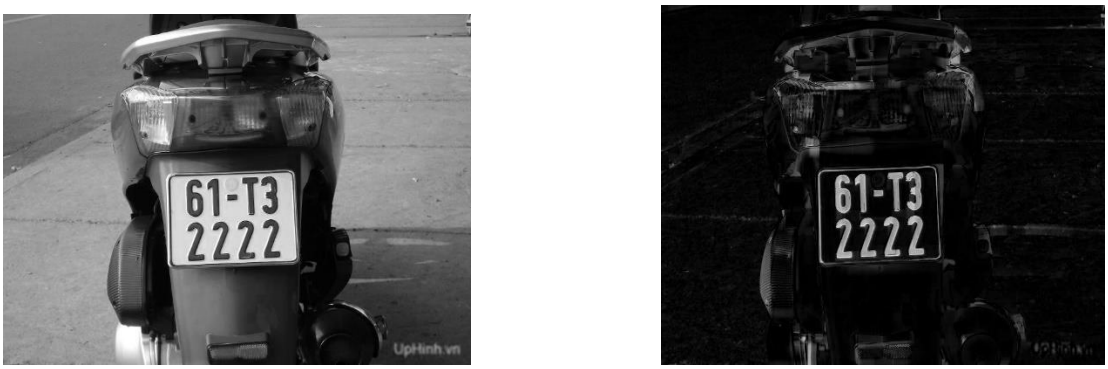
2.3.2 Chuyển sang ảnh xám

Ở bài này nhóm em sẽ chuyển ảnh xám từ hệ màu HSV thay vì RGB vì với không gian màu HSV ta có ba giá trị chính là: Vùng màu (Hue), độ bão hòa (Saturation), cường độ sáng (Value). Vì lý do đó không gian màu HSV thích nghi tốt hơn đối với sự thay đổi ánh sáng từ môi trường ngoài. Khi chuyển đổi, ảnh xám ta cần là ma trận các giá trị cường độ sáng tách ra từ hệ màu HSV.

2.3.3 Tăng độ tương phản

2.3.3.1 Black Hat

Black Hat là kết quả của phép trừ ảnh đóng với ảnh gốc. Điều này được sử dụng để trích xuất các chi tiết nhỏ từ các cấu trúc trong ảnh gốc, đặc biệt tăng cường các đối tượng sáng trong nền tối.



Hình 3.3 Black hat

2.3.3.2 Top Hat

Top Hat là kế quả của phép trừ ảnh đã mở với ảnh gốc. Điều này được sử dụng để trích xuất các chi tiết nhỏ từ các cấu trúc trong ảnh gốc, đặc biệt tăng cường đối tượng tối trong nền sáng.

Hình 2.4 Top Hat

Để làm tăng độ tương phản của biển số, nhóm em sử dụng chủ yếu hai phép Top Hat và Black Hat. Ý tưởng chung là ảnh đầu ra sẽ là ảnh gốc cộng thêm ảnh qua phép Top Hat và trừ đi ảnh qua phép Black Hat. Những chi tiết đã sáng sẽ sáng hơn và những chi tiết tối lại càng tối hơn, từ đó sẽ làm tăng độ tương phản cho biển số.

Hình 2.5 Ảnh sau khi tăng tương phản

2.3.4 Giảm nhiễu bằng bộ lọc Gaussian

Noise được hiểu cơ bản là các dạng chấm hạt nhỏ phân bố trên hình ảnh. Noise có thể làm biến dạng các chi tiết trong ảnh khiến cho chất lượng ảnh thấp.

Bộ lọc Gaussian được cho là bộ lọc hữu ích nhất, được thực hiện bằng cách nhân chập ảnh đầu vào với một ma trận lọc Gaussian , sau đó cộng chúng lại để tạo thành ảnh đầu ra.

Hình 2.6 Ảnh sau khi lọc Gaussian

2.3.5 Phát hiện cạnh Canny

Trong hình ảnh, thường tồn tại các thành phần như: vùng trơn, góc/cạnh và nhiễu. Cạnh trong ảnh mang đặc trưng quan trọng, thường là thuộc đối tượng trong ảnh. Do đó, để phát hiện cạnh trong ảnh, có nhiều giải thuật khác nhau như toán tử Sobel, toán tử Prewitt, Zero crossing .... nhưng ở đây nhóm em chọn giải thuật Canny vì hương pháp này hơn hẳn các phương pháp khác do ít bị tác động của nhiễu và cho khả năng phát hiện các biên yếu.

Sau khi sử dụng phát hiện biên canny, dù đã trích xuất được những chi tiết cạnh của biển số, tuy nhiên vẫn còn quá nhiều chi tiết thừa trong hình ảnh, từ đây chúng ta sẽ vẽ contour, áp dụng nhưng đặc điểm của biển số để lọc lấy ra biển số chính xác.

Hình 2.9 Ảnh sau khi phát hiện biên canny

2.3.6 Vẽ contour

Có thể hiểu Contour là tập hợp các điểm tạo thành đường cong kín bao quanh một đối tượng nào đó. Thường dùng để xác định vị trí, đặc điểm của đối tượng.

Hàm trong OpenCV được biểu diễn như sau:

findContours(InputOutputArray image, OutputArratOfArrays contours, OutputArray hierarchy, int mode, int method, Point offset = Point())

Các tham số:

image : hình ảnh cần tìm biên, là ảnh nhị phân.

contours : lưu trữ các đường biên tìm được, mỗi đường biên được lưu trữ dưới dạng một vector của các điểm.

hierarchy : chứa thông tin về hình ảnh như số đường viền, xếp hạng các đường viền theo kích thước, trong ngoài, ..

mode :

CV_RETR_EXTERNAL : khi sử dựng cờ này nó chỉ lấy ra những đường biên bên ngoài, nhưng biên bên trong của vật thể bị loại bỏ.

CV_RETR_LIST : Khi sử dụng cờ này nó lấy ra tất cả các đường viền tìm được.

CV_RETR_CCOMP : khi sử dụng cờ này nó lấy tất cả những đường biên và chia nó làm 2 level, những đường biên bên ngoài đối tượng, và những đường biên bên trong đối tượng.

CV_RETR_TREE : khi sử dụng cờ này nó lấy tất cả các đường biên và tạo ra một hệ thống phân cấp đầy đủ của những đường lồng nhau.

method :

CV_CHAIN_APPROX_NONE : sử dụng cờ này sẽ lưu trữ tất cả các điểm của đường viền .

CV_CHAIN_APPROX_SIMPLE : Ví dụ : một hình chữ nhật sẽ được mã hoá bằng toạ độ của 4 đỉnh.

CV_CHAIN_APPROX_TC89_L1 or CV_CHAIN_APPROX_TC89_KCOS : Áp dụng thuật toán xấp xỉ Tech-Chin.

Hình 2.10 Vẽ Contour với OpenCV

Vì có quá nhiều đường bao quanh các vật không phải là biển số nên chúng ta sẽ áp dụng những đặc trưng riêng về tỉ lệ cao/ rộng, diện tích trong khung hình cố định để lọc ra đúng biển số.

Đầu tiên ta làm xấp xỉ contour thành một hình đa giác và chỉ lấy những đa giác nào chỉ có 4 cạnh. Nghĩa là lúc xấp xỉ contour bộ nhớ chỉ ghi nhớ vị trí các đỉnh của đa giác đó thành một mảng. Số cạnh của đa giác sẽ bằng số đỉnh và bằng chiều dài của mảng đó.

Tiếp theo ta tính toán tỉ lệ cao/rộng và diện tích của biển số phù hợp, sau đó ta lưu tất cả những biển số có trong hình dưới dạng tọa độ các đỉnh

Từ đây, ta cắt hình ảnh biển số từ các tọa độ vị trí đã biết để phục vụ cho mục đích tiếp theo "Tách các kí tự trong biển số". Lưu ý ở đây ta cắt từ ảnh nhị phân luôn để máy tính xử lý nhanh hơn, tốn ít thời gian hơn.

2.4 Phân đoạn kí tự

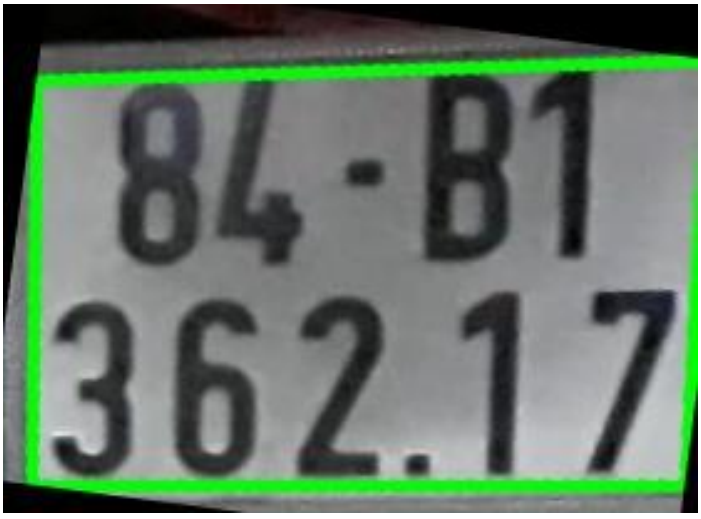
2.4.1 Xoay biển số

Khi chụp ảnh đầu vào, không phải lúc nào biển số cũng ở chính diện, có thể bị méo sang trái, sang phải, nghiêng góc dẫn đến nếu cứ sử dụng ảnh biển số đã cắt mà không điều chỉnh góc độ dẫn đến ảnh kí tự được cắt ra đưa vào bộ nhận diện rất dễ bị sai. Ví dụ giữa số 1 và số 7, số 2 và chữ Z, chữ B và số 8,...

Hình 2.11 Ảnh biển số chưa xoay

Phương pháp xoay ảnh em sử dụng ở đây là:

1. Lọc ra tọa độ 2 đỉnh A,B nằm dưới cùng của biển số
2. Từ 2 đỉnh có tọa độ lần lượt là A(x1, y1) và B(x2,y2) ta có thể tính được cạnh đối và cạnh kề của tam giác ABC
3. Ta tính được góc quay $\widehat{BAC} = \tan^{-1}\left(\frac{cạnh\ đối}{cạnh\ kề}\right) = \tan^{-1}\left(\frac{BC}{AC}\right)$
4. Xoay ảnh theo góc quay đã tính. Nếu ngược lại điểm A nằm cao hơn điểm B ta cho góc quay âm

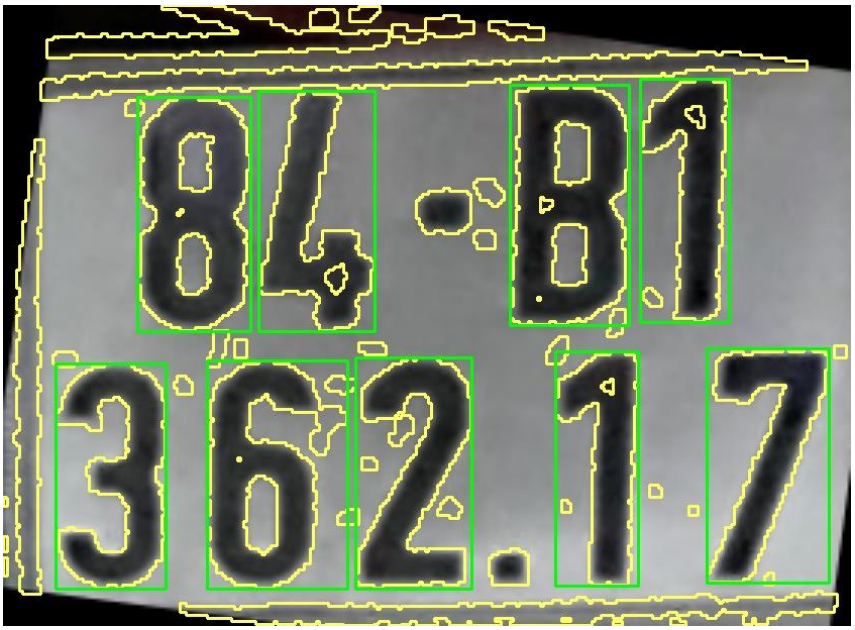


Hình 2.12 Ảnh biển số đã xoay

2.4.2 Tìm vùng đối tượng

Từ ảnh nhị phân, ta lại tìm contour cho các kí tự (phần màu trắng). Sau đó vẽ những hình chữ nhật bao quanh các kí tự đó. Tuy nhiên việc tìm contour này

cũng bị nhiễu dẫn đến việc máy xử lý sai mà tìm ra những hình ảnh không phải kí tự. Ta sẽ áp dụng các đặc điểm về tỉ lệ chiều cao/rộng của kí tự, diện tích của kí tự so với biển số, những đường màu vàng là đường contour và nếu so sánh với ảnh nhị phân thì có rất nhiều đường nhiễu như đường viền biển số, dấu gạch, dấu chấm... Sau khi đã áp dụng các điều kiện thì sẽ vẽ ra những hình chữ nhật màu xanh bao quanh các kí tự.



Hình 2.13 Tìm vùng đối tượng

2.4.3 Tìm và tách kí tự

Sau khi đã nhận dạng từng kí tự bằng hình chữ nhật và cũng đã có tọa độ vị trí 4 đỉnh của hình đó, ta lúc này có thể cắt hình ảnh kí tự đó ra phục vụ cho giai đoạn sau "Nhận diện kí tự". Lưu ý ở đây ta cắt ảnh nhị phân chứ không cắt từ ảnh gốc.

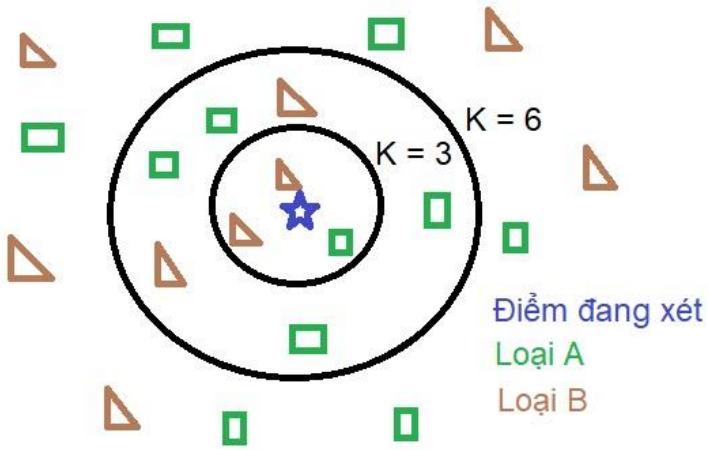
2.5 Nhận diện kí tự

2.5.1 Thuật toán KNN

KNN hoạt động theo quy trình gồm 4 bước chính:

1. Xác định tham số K (số láng giềng gần nhất).

2. Tính khoảng cách từ điểm đang xét đến tất cả các điểm trong tập dữ liệu cho trước
3. Sắp xếp các khoảng cách đó theo thứ tự tăng dần
4. Xét trong tập K điểm gần nhất với điểm đang xét, nếu số lượng điểm của loại nào cao hơn thì coi như điểm đang xét thuộc loại đó



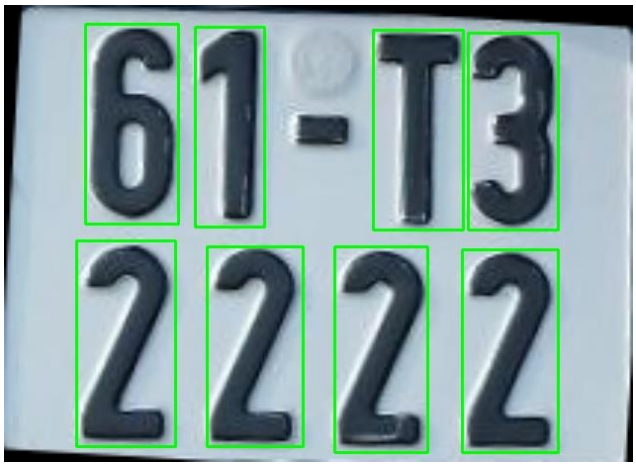
Hình 2.14 Ví dụ về KNN

$$X^* = \frac{X - \min(X)}{\max(X) - \min(X)}$$

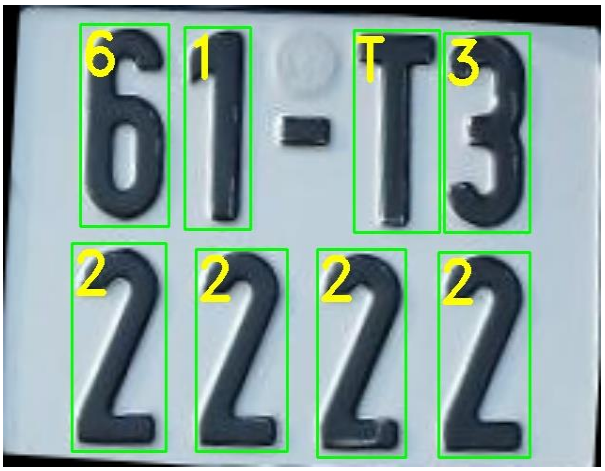
2.5.2 Hướng giải quyết

Ở giai đoạn cuối này được thực hiện theo những bước sau:

1. Tạo tập dữ liệu để huấn luyện
2. Huấn luyện mô hình KNN
3. Đưa hình ảnh từ bước "Phân đoạn kí tự" vào mô hình KNN đã tạo để đưa ra kết quả
4. In ra kết quả biển số



Hình 2.16 Biển số trước khi nhận diện



Hình 2.17 Biển số sau khi nhận diện

Hình 2.18 Biển số xe được in lên ảnh gốc

Chương 3 CHƯƠNG TRÌNH DEMO

3.1 Hình ảnh của chương trình

Sau khi nhấn nút chọn ảnh từ thư mục tùy ý, ảnh sẽ hiện lên ở khung label góc trên bên phải. Tiếp theo ta nhấn nút nhận dạng để tiến hành nhận dạng biển số và in lên label chính giữa. Nút thông tin dùng để hiển thị các thông tin của biển số xe như : tên, tỉnh,…

Hình 3.1 Ảnh demo nhận dạng biển số xe từ ảnh

Chương 4 KẾT LUẬN

4.1 Kết quả đạt được:

- Xây dựng được ứng dụng Desktop nhận dạng biển số
- Sử dụng được python và PyQt5 để chương xây dựng trình
- Biết cách sử dụng công cụ QT Designer để thiết kế giao diện

4.2 Hạn chế:

- Còn một số chức năng chưa hoàn thành
- Tốc độ khởi chạy chương trình còn chậm, chưa tối ưu
- Nhận dạng có thể bị sai, tỉ lệ chưa được cao
- Ảnh đầu vào phải rõ nếu không sẽ không nhận dạng được
- Nhận dạng được tất cả các hình ảnh dù không phải là biển số

4.3 Hướng phát triển:

- Khắc phục các hạn chế, lập trình và kiểm thử các chức năng còn thiếu.
- Tối ưu mã nguồn tốc độ chạy của chương trình
- Thiết kế giao diện dễ nhìn, đẹp mắt hơn