**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**\_\_\_\_\*\*\*\_\_\_\_**

**Logo

Description automatically generated**

**BÁO CÁO BÀI TẬP LỚN**

**HỌC PHẦN : THỊ GIÁC MÁY TÍNH**

**ĐỀ TÀI: XÂY DỰNG CHƯƠNG TRÌNH**

**NHẬN DẠNG BIỂN SỐ XE ô tô**

Giảng viên hướng dẫn: TS. Nguyễn Hữu Tuân

Sinh viên thực hiện:

Hải Phòng, ngày 28 tháng 12 năm 2023

**BÀI TẬP LỚN**

**HỌC PHẦN: THỊ GIÁC MÁY TÍNH**

1. **Tên đề tài**

***XÂY DỰNG CHƯƠNG TRÌNH NHẬN DẠNG BIỂN SỐ XE Ô TÔ***

1. **Mục đích**

Tạo phần mềm nhận diện biển số xe ô tô có giao diện.

1. **Công việc cần thực hiện**

* Khảo sát và tìm hiểu đề tài.
* Các định yêu cầu của đề tài.
* Xây dựng phần mềm nhận diện biển số xe ô tô.
* Bảo vệ bài tập lớn.

1. **Yêu cầu**

* Kết quả làm bài tập lớn: Báo cáo bài tập lớn
* Báo cáo bài tập lớn phải được trình bày theo mẫu quy định (kèm theo), báo cáo có thể kết xuất thành tệp định dạng PDF

***Hải Phòng, 29 tháng 11 năm 2023***

**GIẢNG VIÊN HƯỚNG DẪN**

**Mục lục**

[Phần I. Giới thiệu bài toán nhận dạng biển số xe ô tô 1](#_Toc154648169)

[1.1. Khái quát 1](#_Toc154648170)

[1.2. Nhiệm vụ 1](#_Toc154648171)

[1.3. Mụch đích 1](#_Toc154648172)

[Phần II. Tổng quan bài toán nhận dạng biển số xe ô tô 2](#_Toc154648173)

[2.1. Khái niệm biển số xe 2](#_Toc154648174)

[2.2. Xử lý ảnh và OpenCV 2](#_Toc154648175)

[2.3. Hướng giải quyết 3](#_Toc154648176)

[Phần III. Phương pháp giải quyết bài toán 4](#_Toc154648177)

[nhận dạng biển số xe ô tô 4](#_Toc154648178)

[3.1. Phát hiện và tách biển số xe 4](#_Toc154648179)

[3.1.1 Tổng quát 4](#_Toc154648180)

[3.1.2 Chuyển ảnh xám 6](#_Toc154648181)

[3.1.3 Tăng độ tương phản 6](#_Toc154648182)

[3.1.4 Giảm nhiễu bằng bộ lọc Gaussian 6](#_Toc154648183)

[3.1.5 Nhị phân hóa với ngưỡng động 8](#_Toc154648184)

[3.1.6 Phát hiện cạnh Canny 9](#_Toc154648185)

[3.1.7 Lọc biển số với Contour 10](#_Toc154648186)

[3.2. Phân đoạn ký tự 11](#_Toc154648187)

[3.2.1 Tổng quát 11](#_Toc154648188)

[3.2.2 Xoay biển số 12](#_Toc154648189)

[3.2.3 Tìm vùng đối tượng 12](#_Toc154648190)

[3.2.4 Tìm và tách ký tự 13](#_Toc154648191)

[3.3 Nhận diện ký tự 13](#_Toc154648192)

[3.3.1 AI (Artificial Intelligent) 13](#_Toc154648193)

[3.3.2 Machine Learning 14](#_Toc154648194)

[3.3.3 Thuật toán KNN (K - Nearest Neighbor) 16](#_Toc154648195)

[3.3.4 Hướng làm bài 19](#_Toc154648196)

[Phầm IV. Cài đặt và thử nghiệm 22](#_Toc154648197)

[4.1 IDE : Pycharm 22](#_Toc154648198)

[4.2 Ngôn ngữ lập trình: Python 22](#_Toc154648199)

[4.3 Thư viện 22](#_Toc154648200)

[4.4 Xây dựng chương trình 23](#_Toc154648201)

[Phần V. Kết luận 35](#_Toc154648202)

[5.1 Kiểm thử chương trình : 35](#_Toc154648203)

[5.2 Ưu nhược điểm 37](#_Toc154648204)

[Tài liệu tham khảo 38](#_Toc154648205)

**Danh mục hình ảnh**

[Hình 3.1. 1 Tổng quát phát hiện và tách biển số 4](#_Toc151995381)

[Hình 3.1. 2 Ảnh ban đầu 5](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995382)

[Hình 3.1. 3 Ảnh tương phản 5](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995383)

[Hình 3.1. 4 Ảnh giảm nhiễu 5](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995384)

[Hình 3.1. 5 Ảnh nhị phân 5](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995385)

[Hình 3.1. 6 Phát hiện cạnh Canny 5](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995386)

[Hình 3.1. 7 Nhiễu 7](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995387)

[Hình 3.1. 8 Ma trận lọc Gaussian 7](#_Toc151995388)

[Hình 3.2. 1 Tổng quát phân đoạn ký tự 11](#_Toc151995788)

[Hình 3.2. 2 Biển chưa xoay 12](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995789)

[Hình 3.2. 3 Biển đã xoay 12](file:///D:\ky_7\thi_giac_may_tinh\Project-BSX-1\Báo%20cáo%20-%20Nguyễn%20Thế%20Anh%20-%2082617.docx#_Toc151995790)

[Hình 3.3. 1 Các lớp trong AI 14](#_Toc151995394)

[Hình 3.3. 2 Phân loại Machine Learning 15](#_Toc151995395)

[Hình 3.3. 3 Ví dụ về KNN 17](#_Toc151995396)

[Hình 3.3. 4 Xét khả năng tiêu thụ của xe 18](#_Toc151995397)

[Hình 3.3. 5 Dữ liệu sau chuẩn hóa 18](#_Toc151995398)

[Hình 3.3. 6 Sau khi tính khoảng cách và xếp hạng 18](#_Toc151995399)

[Hình 3.3. 7 Tập dữ liệu huấn luyện 20](#_Toc151995400)

[Hình 3.3. 8 Biển số trước khi nhận diện 21](#_Toc151995401)

[Hình 3.3. 9 Biển số sau khi nhận diện 21](#_Toc151995402)

[Hình 5. 1 Ảnh ban đầu 35](#_Toc154648306)

[Hình 5. 2 Ảnh kết quả đúng 35](#_Toc154648307)

[Hình 5. 3 Ảnh ban đầu 36](#_Toc154648308)

[Hình 5. 4 Ảnh kết quả sai 36](#_Toc154648309)

[Hình 5. 5 Ảnh kết quả sai 36](#_Toc154648310)

**Lời nói đầu**

Hiện nay ở nước ta, số lượng xe ô tô, xe máy tham gia giao thông trên đường là rất lớn dẫn đến tiêu tốn rất nhiều nhân lực, vật lực cho việc quản lý phương tiện cá nhân trong bãi gửi xe. Nếu không có một công cụ thuận tiện thì việc quản lý phương tiện cá nhân rất mất thời gian, dễ gây nhầm lẫn, thiệt hay cho người sử dụng dịch vụ tại các bãi đỗ xe.

Để giảm tải cho các công việc như thu tiền, bảo hiểm xe, tìm xe cộ trong bãi đỗ xe, trên thế giới đã phát triển công nghệ giám sát tự động đối với các phương tiện giao thông, chính nhờ tính cá nhân của biển số xe mà nó đã trở thành đối tượng chính được sử dụng để nghiên cứu, phát triển trong công nghệ này.

# Phần I. Giới thiệu bài toán nhận dạng biển số xe ô tô

## 1.1. Khái quát

Nội dung :

* Tìm hiểu về biển số xe và hệ thống nhận dạng biển số xe
* Phát biểu bài toán và hướng giải quyết
* Nghiên cứu một số thuật toán xử lý ảnh và nhận dạng kí tự ứng dụng trong việc nhận dạng biển số xe

## 1.**2. Nhiệm vụ**

Từ nội dung nêu trên, đề tài của em sẽ bao gồm các nhiệm vụ sau:

* Tìm hiểu khái quát về xử lý ảnh và bài toán nhận dạng biển số xe.
* Tìm hiểu thông tin về biển số xe và phân loại biển số xe của Việt Nam.
* Tìm hiểu các công đoạn chính của bài toán nhận dạng biển số xe gồm 3 khâu chính:
* Phát hiện vị trí và tách biển số xe
* Phân đoạn kí tự trong biển số xe
* Nhận dạng kí tự
* Cài đặt và thử nghiệm chương trình

## 1.**3. Mụch đích**

Xây dựng chương trình có khả năng nhận diện và đọc nội dung biển số xe ô tô cả 2 loại biển: 1 hàng và 2 hàng với cả 2 sự lựa chọn: chọn ảnh có sẵn trong máy và đọc ảnh qua webcam.

# Phần II. Tổng quan bài toán nhận dạng biển số xe ô tô

## 2.1. Khái niệm biển số xe

**Ở Việt Nam, biển kiểm soát xe cơ giới (hay còn gọi tắt là biển kiểm soát, biển số xe) là tấm biển gắn trên mỗi xe cơ giới, được cơ quan công an cấp (đối với xe quân sự do Bộ Quốc phòng cấp) khi mua xe mới 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 số và chữ (biển xe dân sự không dùng các chữ cái I, J, O, Q, W. Chữ R chỉ dùng cho xe rơ-moóc, sơ-mi rơ-moóc) cho biết: 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 phục vụ cho công tác an ninh, đặc biệt trên đó còn có hình Quốc huy Việt Nam dập nổi.**

**Tiêu chuẩn về kích thước: Ở mỗi nước thường có tiêu chuẩn về kích thước nhất định, còn riêng Việt Nam tỉ lệ kích thước giữa các biển số là gần như giống nhau. Biển số xe có 2 loại, kích thước như sau: Loại biển số dài có chiều cao 110 mm, chiều dài 470 mm; loại biển số ngắn có chiều cao 200 mm, chiều dài 280 mm nên ta sẽ giới hạn tỉ lệ cao/rộng là 3.5 ≤ cao/rộng ≤ 6.5 (biển một hàng) và 0.8 ≤ cao/rộng ≤ 1.5 (biển hai hàng).**

**Số lượng kí tự trong biển số xe nằm trong khoảng [7,9]. Chiều cao của chữ và số: 80mm, chiều rộng của chữ và số: 40mm.**

**Từ những đặc điểm trên ta có thể thiết lập nhưng thông số, điều khiển để lọc chọn những đối tương phù hợp mà ta cần**

## 2.2. Xử lý ảnh và OpenCV

Xử lý ảnh là một phân ngành trong xử lý số tín hiệu với tín hiệu xử lý là ảnh. Đây là một phân ngành khoa học mới rất phát triển trong những năm gần đây. Xử lý ảnh gồm 4 lĩnh vực chính: xử lý nâng cao chất lượng ảnh, nhận dạng ảnh, nén ảnh và truy vấn ảnh. Sự phát triển của xử lý ảnh đem lại rất nhiều lợi ích cho cuộc sống của con người. Ngày nay xử lý ảnh đã được áp dụng rất rộng rãi trong đời sống như: photoshop, nén ảnh, nén video, nhận dạng biển số xe, nhận dạng khuôn mặt, nhận dạng chữ viết, xử lý ảnh thiên văn, ảnh y tế,....

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. Opencv có các interface cho C/C++, Python Java vì vậy hỗ trợ được cho Window, Linux, MacOs lẫn Android, iOS OpenCV có cộng đồng hơn 47 nghìn người dùng và số lượng download vượt quá 6 triệu lần. Opencv có rất nhiều ứng dụng như:

* Nhận dạng ảnh
* Xử lý hình ảnh
* Phục hồi hình ảnh/video
* Thực tế ảo
* Các ứng dụng khác

## 2.3. 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 đồ án này em sẽ giải quyết vấn đề theo 3 bước chính:

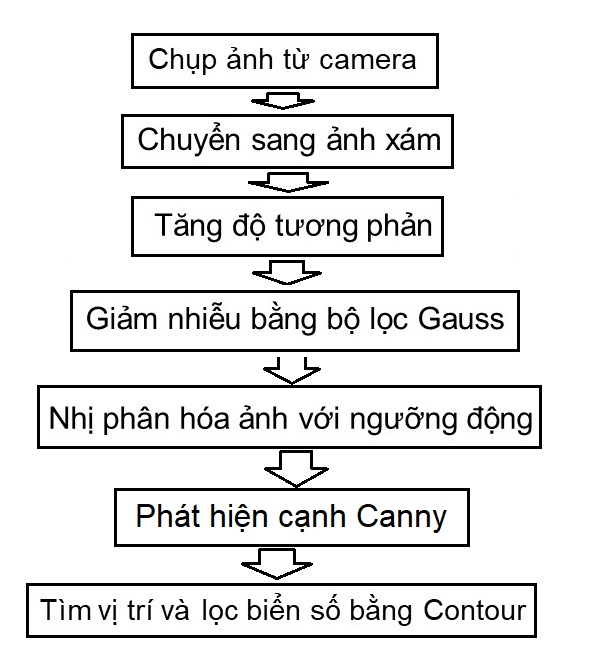
* 1. Phát hiện vị trí và tách biển số xe từ một hình ảnh có sẵn từ đầu vào là camera
  2. Phân đoạn các kí tự có trong biển số xe
  3. Nhận diện các kí tự đó rồi đưa về mã ASCII

# Phần III. Phương pháp giải quyết bài toán

# nhận dạng biển số xe ô tô

## 3.1. Phát hiện và tách biển số xe

### 3.1.1 Tổng quát



Hình 3.1. Tổng quát phát hiện và tách biển số

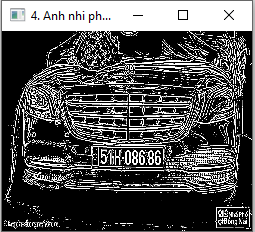
1. Từ video thu được qua webcam, ta sẽ cắt từng frame ảnh ra từ clip đầu vào để xử lý, tách biển số.
2. Chuyển sang ảnh xám để diện được biển số từ sự thay đổi đột ngột về cường độ ánh sáng giữa biển số và môi trường xung quanh
3. Tăng độ tương phản với hai phép toán hình thái học Top Hat và Black Hat để làm nổi bật thêm biển số giữa phông nền, hỗ trợ cho việc xử lý nhị phân sau này
4. 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
5. Lấy ngưỡng động (Adaptive Threshold) sẽ giúp ta tách được thông tin biển số và thông tin nền
6. Sử dụng thuật toán phát hiện cạnh Canny để trích xuất những chi tiết cạnh của biển số
7. Tìm và lọc biển số bằng các tỉ lệ cao/rộng hay diện tích của biển số để loại bỏ những chi tiết nhiễu số sẽ giúp xác định được đúng biển số

Quy trình xử lý ảnh :



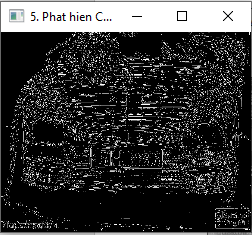
Hình 3.1. 2 Ảnh ban đầu

Hình 3.1. 3 Ảnh tương phản

Hình 3.1. 4 Ảnh giảm nhiễu

Hình 3.1. 5 Ảnh nhị phân



Hình 3.1. 6 Phát hiện cạnh Canny

### 3.1.2 Chuyển ảnh xám

Ảnh xám (Gray Scale) đơn giản là một hình ảnh trong đó các màu là các sắc thái của màu xám với 256 cấp độ xám biến thiên từ màu đen đến màu trắng, nằm trong giải giá trị từ 0 đến 255, nghĩa là cần 8 bits hay 1 byte để biểu diễn mỗi điểm ảnh này. Lý do cần phải phân biệt giữa ảnh xám và các ảnh khác nằm ở việc ảnh xám cung cấp ít thông tin hơn cho mỗi pixel. Với ảnh thông thường thì mỗi pixel thường được cung cấp 3 trường thông tin trong khi với ảnh xám chỉ có 1 trường thông tin, việc giảm khối lượng thông tin giúp tăng tốc độ xử lý, đơn giản hóa giải thuật nhưng vẫn đảm bảo các tác vụ cần thiết

Ở bài này 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.

### 3.***1.3 Tăng độ tương phản***

Để làm tăng độ tương phản của biển số, 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.

Phép Top Hat là kết quả của phép trừ ảnh của ảnh ban đầu với ảnh sau khi thực hiện phép mở, dùng để làm nổi bật nhưng chi tiết trắng trong nền tối

Phép Black Hat là kết quả của phép trừ ảnh của ảnh sau khi thực hiện phép đóng với ảnh ban đầu. Dùng làm nổi bật chi tiết tối trong nền trắng.

### 3.1.4 Giảm nhiễu bằng bộ lọc Gaussian

#### a. Nhiễu

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***.***

Trên thực tế có nhiều loại nhiễu, nhưng người ta thường chia làm ba loại: nhiễu cộng, nhiễu nhân và nhiễu xung. Bản chất của nhiễu thường tương ứng với tần số cao và cơ sở lý thuyết của bộ lọc là chỉ cho những tín hiệu có tần số nhất định đi qua, nên người ta thường sử dụng bộ lọc thông thấp hay trung bình.

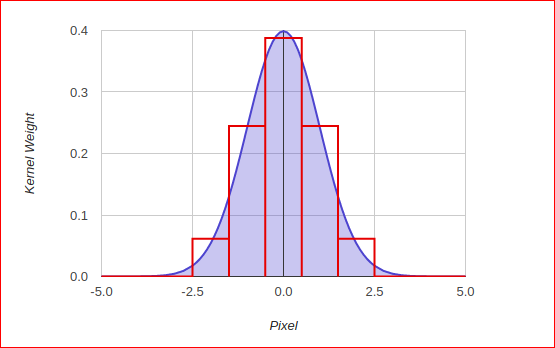


Hình 3.1. 7 Nhiễu

#### b. Bộ lọc Gaussian

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

Ý tưởng chung là giá trị mỗi điểm ảnh sẽ phụ thuộc nhiều vào các điểm ảnh ở gần hơn là các điểm ảnh ở xa. Trọng số của sự phụ thuộc được lấy theo hàm Gauss (cũng được sử dụng trong quy luật phân phối chuẩn).



Hình 3.1. Ma trận lọc Gaussian

### 3.1.5 Nhị phân hóa với ngưỡng động

Ảnh nhị phân Là ảnh mà giá trị của các điểm ảnh chỉ được biểu diễn bằng hai giá trị là 0 (Đen) và 255 (Trắng).

#### a. Nhị phân hóa

Là quá trình biến đổi một ảnh xám thành ảnh nhị phân.

* Gọi giá trị cường độ sáng tại một điểm ảnh là I(x,y) .
* INP(x,y) là cường độ sáng của điểm ảnh trên ảnh nhị phân .
* (Với 0 < x < image.width) và (0 < y < image.height).

Để biến đổi ảnh xám thành ảnh nhị  phân. Ta so sánh giá trị cường độ sáng của điểm ảnh với một ngưỡng nhị phân **T**.

* Nếu I(x,y) > **T** thì INP(x, y) = 0.
* Nếu I(x,y) > **T** thì INP(x, y) = 255.

#### b. Nhị phân hóa với ngưỡng động

Việc nhị phân hóa ảnh với ngưỡng toàn cục như thông thường sẽ rất khó khăn khi phải tự tính toán và chọn mức ngưỡng phù hợp cho từng ảnh khác nhau. Nhị phân hóa ảnh ngưỡng động sẽ giúp tính toán ngưỡng cho phù hợp với từng ảnh.

Gồm 3 bước cơ bản :

* Chia tấm ảnh thành nhiều khu vực, cửa sổ (Region) khác nhau.
* Dùng một thuật toán để tìm một giá trị **T** phù hợp với từng cửa sổ.
* Áp dụng phương pháp nhị phân hóa cho từng khu vực, cửa sổ với ngưỡng **T** phù hợp.

Có rất nhiều phương pháp để tìm **T**, ở đây em sử dụng một kiểu thuật toán mà thư viện OpenCV hỗ trợ là ADAPTIVE\_THRESH\_GAUSSIAN\_C tức lấy trung bình các giá trị xung quanh điểm ngưỡng động đang xét **T**(x,y) với phân phối Gauss rồi trừ đi hằng số C.

### 3.1.6 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 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. Phương pháp này đi theo 4 bước chính:

* Giảm nhiễu (Noise reduction)
* Tính toán Gradient (Gradient calculation)
* Loại bỏ những điểm không phải là cực đại (Non-maximum suppression)
* Lọc ngưỡng (Double threshold)

1. Giảm nhiễu

Làm mờ ảnh, giảm nhiễu dùng bộ lọc Gauss kích thước 5x5. Kích thước 5x5 thường hoạt động tốt cho giải thuật Canny

1. Tính toán Gradient

Ta dùng 2 bộ lọc Sobel X và Sobel Y (3x3) để tính đạo hàm Gx và Gy

Tìm gradient và hướng được làm tròn về 4 hướng: hướng ngang (0 độ), hướng chéo bên phải (45 độ), hướng dọc (90 độ) và hướng chéo trái (135 độ).

1. Loại bỏ những điểm không phải là cực đại

Ở bước này, dùng một filter 3x3 lần lượt chạy qua các pixel trên ảnh gradient. Trong quá trình lọc, xem xét xem độ lớn gradient của pixel trung tâm có phải là cực đại so với các gradient ở các pixel xung quanh. Nếu là cực đại, ta sẽ ghi nhận sẽ giữ pixel đó lại. Còn nếu pixel tại đó không phải là cực đại lân cận, ta sẽ set độ lớn gradient của nó về zero. Ta chỉ so sánh pixel trung tâm với 2 pixel lân cận theo hướng gradient

1. Lọc ngưỡng

Lọc ngưỡng: ta sẽ xét các pixel dương trên mặt nạ nhị phân kết quả của bước trước. Nếu giá trị gradient vượt ngưỡng max\_val thì pixel đó chắc chắn là cạnh. Các pixel có độ lớn gradient nhỏ hơn ngưỡng min\_val sẽ bị loại bỏ. Còn các pixel nằm trong khoảng 2 ngưỡng trên sẽ được xem xét rằng nó có nằm liền kề với những pixel được cho là "chắc chắn là cạnh" hay không. Nếu liền kề thì ta giữ, còn không liền kề bất cứ pixel cạnh nào thì ta loại

### 3.1.7 Lọc biển số với 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. Có 4 thuật toán Contour Tracing chung nhất. Hai trong số đó có tên là: Square Tracing algorithm và Moore – Neighbor Tracing là dễ để thực hiện và thường xuyên được dùng để dò tìm contour của một mẫu. Với thư viện OpenCV người ta áp dụng thuật toán Suzuki’s Contour tracing

1. Thuật toán Square Tracing

Duyệt từ pixel ngoài cùng bên trái phía dưới, đi lên cho tới khi gặp pixel có giá trị bằng 255 (pixel này sẽ được gọi là pixel start) thì bắt đầu di chuyển theo quy tắc sau:

* Nếu gặp Pixel có giá trị bằng 255 thì rẽ trái.
* Nếu gặp Pixel có giá trị bằng 0 thì rẽ phải.
* Di chuyển cho tới khi quay lại pixel start thì dừng lại

1. Thuật toán Moore – Neighbor

Thuật toán này có chút khác biệt so với thuật toán Square Tracking, cụ thể: khi gặp pixel có giá trị bằng 255 đầu tiên (pixel start) thì ta sẽ quay lại pixel trước đó, sau đó đi vòng qua các pixel thuộc 8-connected theo chiều kim đồng hồ cho tới khi gặp pixel khác có giá trị bằng 255. Và điệu kiện kết thúc cũng giống như thuật toán Square Tracking

1. Thuật toán Suzuki’s Tracing

Đây là thuật toán được thư viện OpenCV sử dụng, ngoài khả năng xác định được biên của vật thể như hai phương pháp trên. Phương pháp Suzuki’s Tracing còn có khả năng phân biệt được đường biên ngoài (Outer) và đường biên trong (Hole) của vật thể

1. Lọc biển số

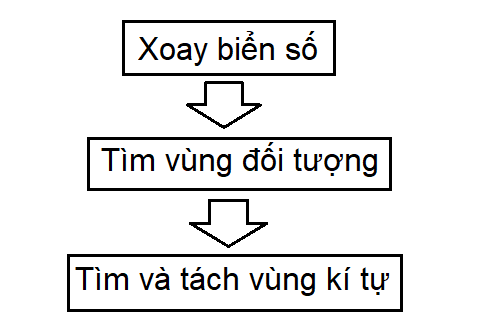
Đầu tiên ta làm xấp xỉ contour thành một hình đa giác và chỉ lấy những đa giác nào chỉ có 4 cạnh. Nghĩa là lúc xấp xỉ contour bộ nhớ chỉ ghi nhớ vị trí các đỉnh của đa giác đó thành một mảng. Số cạnh của đa giác sẽ bằng số đỉnh và bằng chiều dài của mảng đó

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.

## 3.2. Phân đoạn ký tự

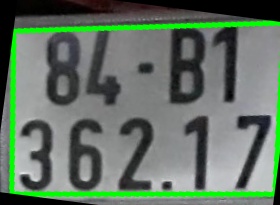
### 3.2.1 Tổng quát



Hình 3.2. Tổng quát phân đoạn ký tự

1. Xoay biển số để tăng khả năng nhận diện
2. Tìm tất cả các vùng kín cho là kí tự và lọc ra những kí tự đúng
3. Tách hình ảnh nhưng kí tự đó ra và đưa vào bộ nhận diện

Quy trình xử lý :

Hình 3.2. 2 Biển chưa xoay

Hình 3.2. 3 Biển đã xoay

### 3.2.2 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,...

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
4. Xoay ảnh theo góc quay đã tính. Nếu ngược lại điểm A nằm cao hơn điểm B ta cho góc quay âm

### 3.2.3 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 .

Trong ảnh 4.3 - 2 những đường màu vàng là đường contour và nếu so sánh với ảnh nhị phân 4.3 -1 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.

### 3.2.4 Tìm và tách ký 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.

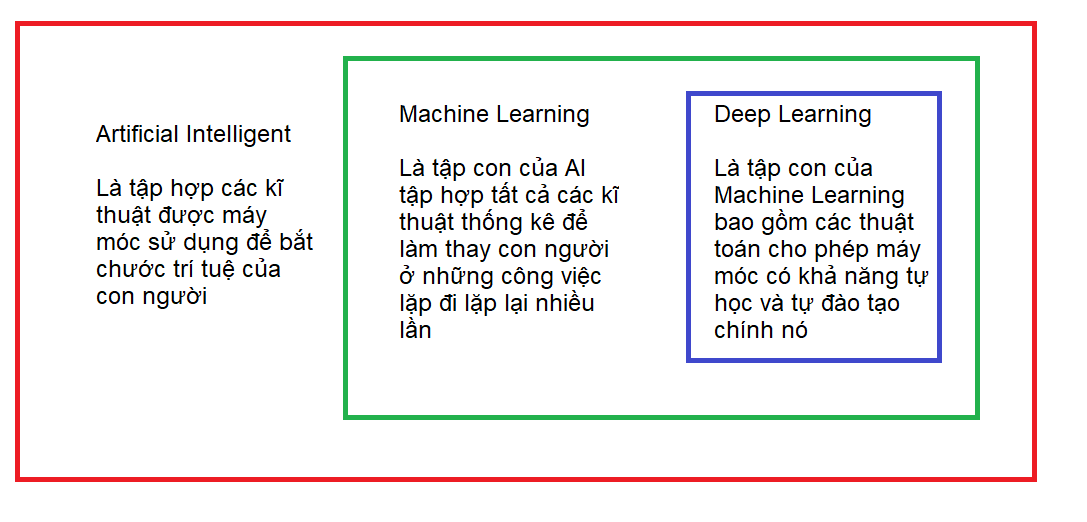
## 3.3 Nhận diện ký tự

Thực chất quá trình nhận diện kí tự chính là quá trình chuyển đổi từ một hình ảnh là ma trận giá trị các điểm ảnh về một dạng thông tin khác như trong đề tài này là mã ASCII để có thể giao tiếp với người dùng. Để hiểu hơn về nhận diện, chúng ta cần đi ngược lại về ngành khoa học trí tuệ nhân tạo (Artificial Intelligent) hay còn gọi là AI.

### 3.3.1 AI (Artificial Intelligent)

Trí tuệ nhân tạo là dạng trí tuệ được biểu hiện ở máy móc, không giống như những dạng trí tuệ tự nhiên ở con người hay động vật. Thông thường cụm từ “trí tuệ nhân tạo” sẽ được hiểu như máy móc có khả năng bắt chước hành vi và nhận thức của con người như việc học tập hay xử lý vấn đề. Trong vài năm trở lại đây, AI thật sự bùng nổ, đặc biệt là từ năm 2015. Phần lớn trong số đó liên quan đến những tiện ích sẵn có của GPU khiến cho việc xử lý song song nhanh, rẻ và mạnh mẽ hơn.

Bên trong AI bao gồm hai khối nhỏ là Máy học (Machine learning) và Học sâu (Deep learning).



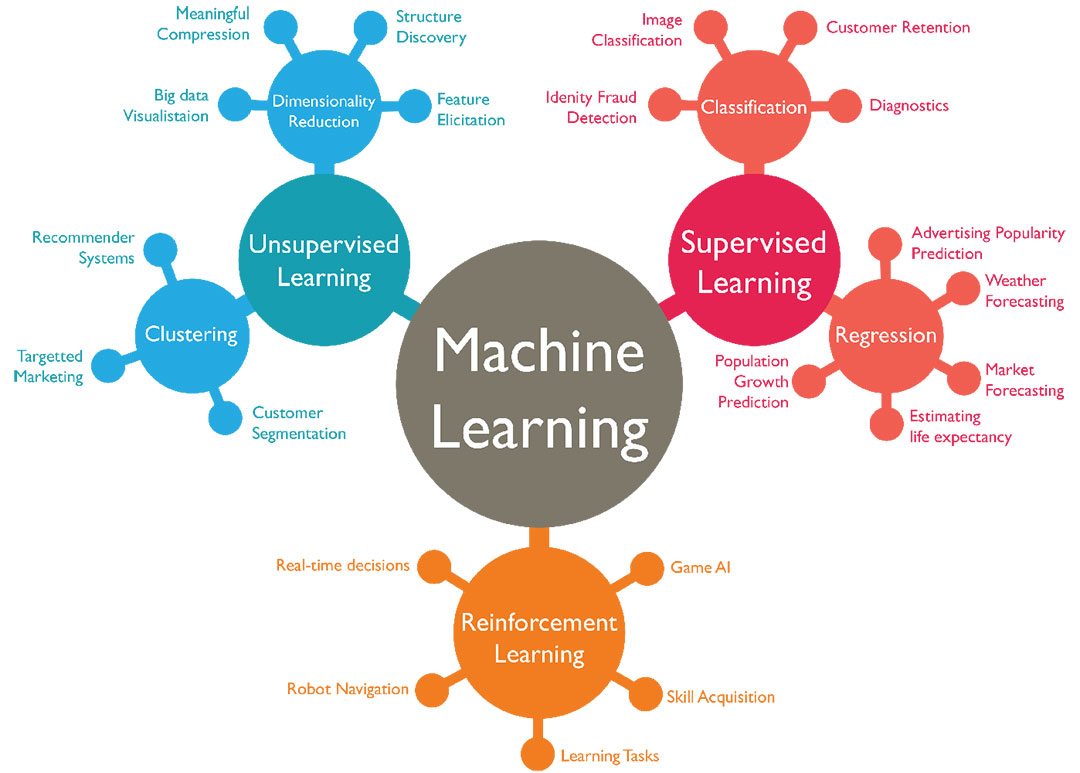
Hình 3.3. Các lớp trong AI

### 3.3.2 Machine Learning

 Machine learning là một lĩnh vực sử dụng các thuật toán cho phép máy tính có thể học từ dữ liệu để thực hiện các công việc thay vì được lập trình một cách rõ ràng. Một số ứng dụng có thể kể đến như xử lý hình ảnh (gắn thẻ, nhận diện kí tự, ô tô tự lái), phân tích văn bản (lọc mail spam, phân tích ngữ nghĩa và khai thác thông tin),...

Về cách hoạt động của Machine learning nói nôm na ta cần một tập dữ liệu huấn luyện bao gồm nhiều mẫu huấn luyện. Mỗi mẫu huấn luyện sẽ là một thể hiện của bài toán (có đầu vào và lời giải). Machine learning sẽ học từ các thể hiện đó để tìm ra lời giải phù hợp với từng đầu vào mới. Nó giống như khi dạy một đứa trẻ cách tâng cầu, ta sẽ tâng cầu vài lần cho đứa trẻ quan sát. Sau đó đứa trẻ sẽ bắt đầu học để tự tâng cầu. Trong đề tài đồ án này, em sẽ đưa vào tập dữ liệu mẫu là hình ảnh các chữ số, kí tự của biển số Việt Nam đã được gắn nhãn là một mã ASCII tương ứng để máy tự học.

Machine Learning có nhiều thuật toán khác nhau nhưng nổi bật nhất là hai thuật toán **Học có giám sát** (Supervised learning) và **Học không giám sát** (Unsupervised learning).



Hình 3.3. Phân loại Machine Learning

1. Học có giám sát.

Là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là (data, label), tức (dữ liệu, nhãn). Ví dụ như thuật toán dò các khuôn mặt trong một bức ảnh đã được phát triển từ rất lâu. Thời gian đầu, Facebook sử dụng thuật toán này để chỉ ra các khuôn mặt trong một bức ảnh và yêu cầu người dùng tag friends- tức gán nhãn cho mỗi khuôn mặt. Số lượng cặp dữ liệu (khuôn mặt, tên người) càng lớn, độ chính xác ở những lần tự động tag tiếp theo sẽ càng lớn.

Thuật toán học có giám sát lại tiếp tục được chia ra hai phần nhỏ gồm bài toán phân loại (Classification) khi các nhãn của dữ liệu đầu vào chia thành hữu hạn các nhóm và bài toán hổi quy (Regression) khi nhãn không được chia thành các nhóm mà là một giá trị thực cụ thể. Ví dụ: một cô gái x tuổi, cao y m, thu nhập hàng tháng z triệu đồng thì sẽ kết hôn khi nào?

1. Học không giám sát

Trong thuật toán này, chúng ta không biết được đầu ra hay nhãn mà chỉ có dữ liệu đầu vào. Thuật toán học không giám sát sẽ dựa vào cấu trúc của dữ liệu để thực hiện một công việc nào đó, ví dụ như phân nhóm (clustering) hoặc giảm số chiều của dữ liệu (dimension reduction) để thuận tiện trong việc lưu trữ và tính toán.

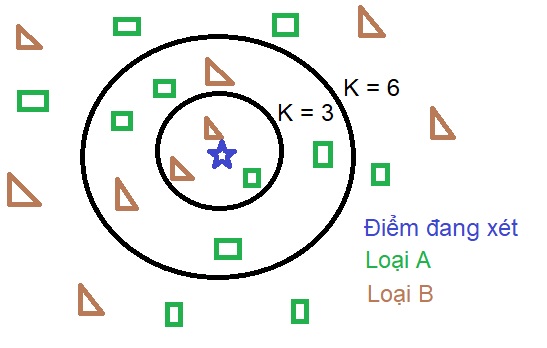
Bao gồm hai loại chính: Phân nhóm (clustering) toàn bộ dữ liệu đầu vào thành các nhóm nhỏ dựa trên sự liên quan giữa các dữ liệu trong mỗi nhóm ví dự như chia nhóm khách hàng dựa trên hành vi mua hàng, độ tuổi, giới tính của họ. Còn lại là bài toán kết hợp (Association) được dùng khi ta muốn khám phá ra một quy luật nào đó dựa trên tập dữ liệu cho trước như những khách hàng mua laptop thường có xu hướng mua thêm con chuột hay chó thường trung thành với chủ hơn mèo.

### 3.3.3 Thuật toán KNN (K - Nearest Neighbor)

KNN là một trong những thuật toán học có giám sát đơn giản nhất trong Machine Learning, có thể sử dụng cho cả bài toán phân loại và hồi quy. Về ý tưởng là gán kết quả với dữ liệu training gần giống với mẫu nhất. Ví dụ như khi đi câu cá, ta không biết cá câu lên là cá rô hay cá chép, nhưng khi so sánh những đặc điểm về mắt, mang, vây,... từ những con cá rô, cá chép đã thấy thì cuối cùng có thể quyết định xem con cá mình câu được thuộc nhóm cá nào.

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



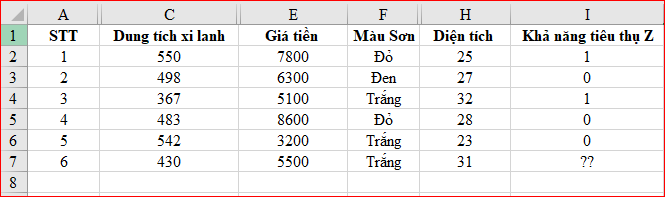
Hình 3.3. Ví dụ về KNN

Việc điểm đang xét thuộc loại nào còn phụ thuộc vào hệ số K hay trọng số khoảng cách... mà người dùng đặt sao cho phù hợp với bài toán đang xét. Chẳng hạn ở hình trên nếu ta xét K = 3 thì điểm đang xét sẽ thuộc loại B, ngược lại nếu K = 6 thì nó thuộc loại A. Ngoài ra người ta có thể để trọng số cao hơn cho những điểm gần hơn hay ít khi sử dụng K = 1 để đảm bảo kết quả đầu ra được tối ưu.

Thông thường việc tính khoảng cách đến các điểm sẽ theo công thức Euclid:

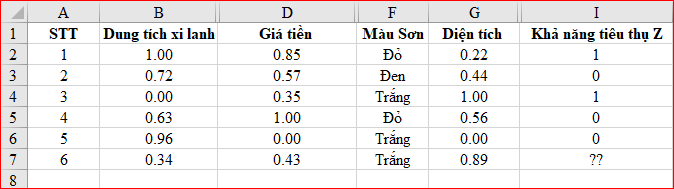
Khi thực hiện so sánh có thể bỏ qua dấu căn bậc 2. Ngoài ra nếu như khoảng cách giữa các biến quá lớn như biến x lớn hơn xấp xỉ 1000000 lần thì ta cũng cần chuẩn hóa lại dữ liệu theo công thức:

Để dễ hiểu hơn em xin lấy một ví dụ về một công ty muốn đưa dòng xe ô tô mới của mình vào thị trường và muốn biết xe sẽ bán chạy hay ế. Họ tổng hợp các dữ liệu về dung tích xi lanh, giá tiền, màu sơn, diện tích xe của các dòng xe đã xuất ra thị trường. Đặt Z là khả năng tiêu thụ cụ các dòng xe. Z = 0 nghĩa là xe ế, Z = 1 nghĩa là xe sẽ bán chạy



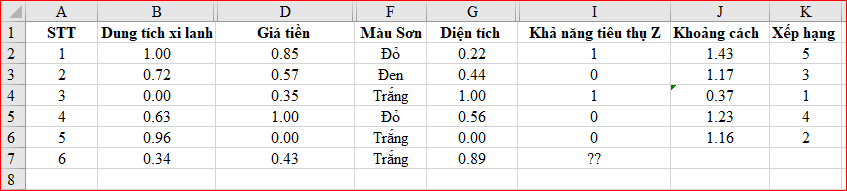
Hình 3.3. Xét khả năng tiêu thụ của xe

Sau đó ta chuẩn hóa dữ liệu:



Hình 3.3. Dữ liệu sau chuẩn hóa

Sau đó ta tính khoảng cách giữa điểm đang xét đến các dữ liệu cho trước. Lưu ý ở các biến có giá trị định tính nếu khác nhau khoảng cách sẽ bằng 1, giống nhau sẽ bằng 0. Ví dụ ta tính khoảng các giữa dòng xe 6 và 1:



Hình 3.3. Sau khi tính khoảng cách và xếp hạng

Nhìn hình trên nếu chọn K=1 thì kết quả Z=1, nếu K=3 thì Z=0, nếu K=5 thì Z=0.

Tuy nhiên phương pháp này có một số ưu và nhược điểm như sau:

* Ưu điểm:
  + Dễ sử dụng và cài đặt
  + Độ phức tạp tính toán nhỏ
  + Việc dự đoán kết quả rất đơn giản
* Nhược điểm:
  + Với K nhỏ, khi gặp nhiễu dễ đưa ra kết quả không chính xác
  + Cần nhiều thời gian lưu trainning set và nếu test tăng lên sẽ tốn rất nhiều thời gian

### 3.3.4 Hướng làm bài

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

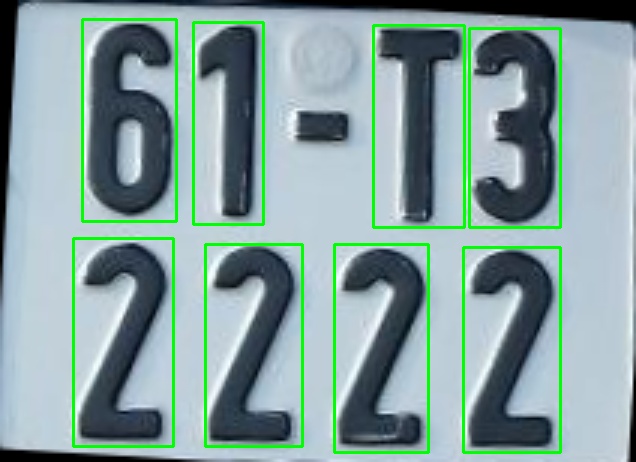
Bước 1 và 2 ta sẽ tạo ra mô hình KNN riêng biệt với code chính. Để khi cần nhận diện kí tự ta không cần phải làm lại các bước từ đầu. Đầu tiên em tạo tập dữ liệu (tập hình ảnh của các chữ số và kí tự) để train từ phần mềm paint. Trong phần mềm Paint ta viết các chữ số và kí tự (trừ kí tự O, I, J) với phông chữ “Biển số xe Việt Nam”, có thể xoay các kí tự này lần lượt với các góc . Kết quả có dạng như sau:



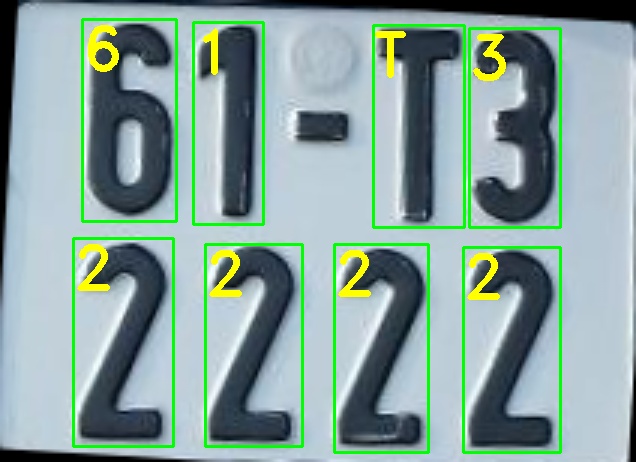
Hình 3.3. Tập dữ liệu huấn luyện

Tiếp theo ta lấy ngưỡng, vẽ contour và cắt từng kí tự. Vì mỗi kí tự có kích thước khác nhau xử lý phức tạp nên cần chuẩn hóa hình ảnh lại với kích thước cao:rộng là 30:20 pixels. Thay vì mỗi kí tự đưa vào mô hình để máy nhận diện thì những kí tự này sẽ được ta gắn nhãn bằng những phím bấm trên máy tính. Sau khi gắn nhãn hết các kí tự ta sẽ lưu hai file .txt là classifications.txt và flattened\_images.txt. File classifications.txt có nhiệm vụ lưu các mã ASCII của các kí tự đó và file flattened\_images.txt sẽ lưu giá trị các điểm ảnh có trong hình ảnh kí tự (hình 20x30 pixel có tổng cộng 600 điểm ảnh có giá trị 0 hoặc 255)

Bước 3 và 4. Ta thực hiện đưa ảnh đang xét vào và tính khoảng cách đến tất cả các điểm trong mẫu, kết quả sẽ là mã ASCII đại điện cho hình ảnh đó. Cuối cùng ta in biển số xe ra hình. Tuy nhiên ở Việt Nam có hai loại biển số là biển một hàng và biển hai hàng. Về ý tưởng chung để phân biệt hai hàng này ta dựa vào vị trí của hình ảnh kí tự, nếu vị trí nằm thấp 1/3 chiều cao của biển số thì kí tự sẽ được xếp vào hàng một. Ngược lại sẽ được xếp vào hàng hai.



Hình 3.3. Biển số trước khi nhận diện



Hình 3.3. Biển số sau khi nhận diện

# Phầm IV. Cài đặt và thử nghiệm

## 4.1 IDE : Pycharm

PyCharm là môi trường phát triển tích hợp đa nền tảng (IDE) được phát triển bởi Jet Brains và được thiết kế đặc biệt cho Python. PyCharm có mặt trên cả 3 nền tảng Windows, Linux và Mac OS.

Pycharm là một giải pháp phù hợp cho Python developers vì IDE này hỗ trợ nhiều extensions, môi trường ảo (Virtual Environment), nhiều tính năng thông minh như bộ code completion, tự động thụt lề, phát hiện văn bản trùng lặp và kiểm tra lỗi. Ngoài ra còn có các tính năng tìm kiếm mã nguồn thông minh để tìm kiếm từng từ một trong nháy mắt.

Dowload Pycharm tại [*https://www.jetbrains.com/pycharm/download/?section=windows*](https://www.jetbrains.com/pycharm/download/?section=windows)

## Ngôn ngữ lập trình: Python

Python là [ngôn ngữ lập trình máy tính bậc cao](https://glints.com/vn/blog/ngon-ngu-lap-trinh-bac-cao/) thường được sử dụng để xây dựng trang web và phần mềm, tự động hóa các tác vụ và tiến hành [phân tích dữ liệu](https://glints.com/vn/blog/ky-nang-phan-tich-du-lieu/).

Python đã trở thành một yếu tố chính trong khoa học dữ liệu, cho phép các nhà phân tích dữ liệu và các chuyên gia khác sử dụng ngôn ngữ này để thực hiện các phép tính thống kê phức tạp, tạo trực quan hóa dữ liệu, xây dựng thuật toán [học máy](https://glints.com/vn/blog/hoc-machine-learning/), thao tác và phân tích dữ liệu cũng như hoàn thành các nhiệm vụ khác liên quan đến dữ liệu.

Dowload python tại:

*https://www.python.org/downloads/*

## 4.3 Thư viện

|  |  |  |
| --- | --- | --- |
| Thư viện | Công dụng | Cài đặt |
| Tkinter | Tkinter là một thư viện GUI (Graphical User Interface) trong Python, giúp tạo ra giao diện người dùng đồ họa. | Tkinter thường được đi kèm với Python, không cần cài đặt thêm. |
| OpenCV | OpenCV là một thư viện phổ biến cho xử lý ảnh và thị giác máy tính. Nó cung cấp các công cụ và thuật toán để thực hiện nhiều tác vụ như đọc ảnh, xử lý ảnh, nhận diện đối tượng, và nhiều công việc khác. | pip install opencv-python |
| NumPy | NumPy là thư viện tính toán khoa học trong Python. Nó hỗ trợ các phép toán trên mảng và ma trận, làm cho việc xử lý dữ liệu số trở nên thuận lợi hơn. | pip install numpy |
| Pillow | Pillow là một fork của PIL (Python Imaging Library), chú trọng vào thao tác và xử lý ảnh. Các module Image và ImageTk được sử dụng để làm việc với ảnh và chuyển đổi giữa định dạng ảnh và đối tượng Tkinter. | pip install Pillow |
| Math | Math là một thư viện cung cấp các hàm toán học cơ bản. | math thường được đi kèm với Python, không cần cài đặt thêm. |

## 4.4 Dữ liệu kiểm thử

Một số mẫu ảnh kiểm thử chương trình:











## 4.5 Xây dựng chương trình

|  |  |
| --- | --- |
| **Các bước làm** | **Thuật toán chính** |
| 1. Chuyển ảnh xám | cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY) |
| 2. Tăng độ tương phản | cv2.convertScaleAbs(imgGrayscale, alpha=2.0, beta=0) |
| 3. Giảm nhiễu bằng bộ lọc Gaussian | cv2.GaussianBlur(imgContrast, (5, 5), 0) |
| 4. Nhị phân hóa với ngưỡng động | cv2.adaptiveThreshold(imgGrayscale, 250, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY\_INV, ADAPTIVE\_THRESH\_BLOCK\_SIZE, ADAPTIVE\_THRESH\_WEIGHT) |
| 5. Phát hiện cạnh Canny | cv2.Canny(imgThresh, 250, 255) |
| 6. Lọc biển số với Contour | cv2.findContours(dilated\_image, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE) |
| 7. Xoay biển số | cv2.getRotationMatrix2D(ptPlateCenter, -angle, 1.0) |
| 8. Tìm vùng đối tượng | cv2.drawContours(mask, [screenCnt], 0, 255, -1) |
| 9. Tìm và tách ký tự | cv2.findContours(thre\_mor, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE) |
| 10. Nhận diện ký tự | Mô hình KNN (K-Nearest Neighbors) |
| 11. Hiển thị kết quả | cv2.imshow(window\_name, img) hoặc cv2.imshow(window\_name, cv2.cvtColor(roi, cv2.COLOR\_BGR2RGB)) |

**Chi tiết bước làm 10.Nhận diện ký tự :**

**1. Quá trình đào tạo ( training)**

a. Xử lý Dữ liệu Huấn luyện:

* Mỗi hình ảnh của ký tự được chuyển thành một mảng numpy 1D và được lưu vào tệp flattened\_images.txt.
* Nhãn (label) của mỗi ký tự được lưu vào tệp classifications.txt.

b. Khởi Tạo Mô Hình KNN:

* Sử dụng cv2.ml.KNearest\_create() để tạo một đối tượng mô hình KNN.

c. Huấn Luyện Mô Hình:

* Dữ liệu huấn luyện (flattened\_images.txt và classifications.txt) được đưa vào mô hình thông qua phương thức train.
* Mô hình học từ dữ liệu và tạo một mô hình có khả năng dự đoán nhãn cho các ký tự mới.

**2. Quá trình dự đoán (Inference):**

a. Xử Lý Dữ liệu Dự đoán:

* Mỗi ký tự cần được nhận diện được chuyển thành mảng numpy 1D tương tự như trong quá trình đào tạo.

b. Dự Đoán Ký Tự:

* Dữ liệu ký tự mới được đưa vào mô hình thông qua phương thức findNearest.
* Mô hình dự đoán nhãn của ký tự dựa trên các ký tự đã được đào tạo.

c. Hiển Thị Kết Quả:

* Kết quả dự đoán (nhãn) sau đó được chuyển thành ký tự tương ứng và hiển thị trên ảnh.

**Chương trình trainning đầy đủ :**

*# GenData.py*import numpy as np  
import cv2  
  
  
*# module level variables ##########################################################################*MIN\_CONTOUR\_AREA = 40  
  
  
RESIZED\_IMAGE\_WIDTH = 20  
RESIZED\_IMAGE\_HEIGHT = 30  
  
*###################################################################################################*def main():  
 imgTrainingNumbers = cv2.imread("training\_chars.png") *# read in training numbers image  
 #imgTrainingNumbers = cv2.resize(imgTrainingNumbers, dsize = None, fx = 0.5, fy = 0.5)* imgGray = cv2.cvtColor(imgTrainingNumbers, cv2.COLOR\_BGR2GRAY) *# get grayscale image* imgBlurred = cv2.GaussianBlur(imgGray, (5,5), 0) *# blur  
  
 # filter image from grayscale to black and white* imgThresh = cv2.adaptiveThreshold(imgBlurred, *# input image* 255, *# make pixels that pass the threshold full white* cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, *# use gaussian rather than mean, seems to give better results* cv2.THRESH\_BINARY\_INV, *# invert so foreground will be white, background will be black* 11, *# size of a pixel neighborhood used to calculate threshold value* 2) *# constant subtracted from the mean or weighted mean* cv2.imshow("imgThresh", imgThresh) *# show threshold image for reference* imgThreshCopy = imgThresh.copy() *# make a copy of the thresh image, this in necessary b/c findContours modifies the image* imgContours, npaContours, npaHierarchy = cv2.findContours(imgThreshCopy, *# input image, make sure to use a copy since the function will modify this image in the course of finding contours* cv2.RETR\_EXTERNAL, *# retrieve the outermost contours only* cv2.CHAIN\_APPROX\_SIMPLE) *# compress horizontal, vertical, and diagonal segments and leave only their end points  
  
 # declare empty numpy array, we will use this to write to file later  
 # zero rows, enough cols to hold all image data* npaFlattenedImages = np.empty((0, RESIZED\_IMAGE\_WIDTH \* RESIZED\_IMAGE\_HEIGHT))  
   
  
 intClassifications = [] *# declare empty classifications list, this will be our list of how we are classifying our chars from user input, we will write to file at the end  
  
 # possible chars we are interested in are digits 0 through 9, put these in list intValidChars* intValidChars = [ord('0'), ord('1'), ord('2'), ord('3'), ord('4'), ord('5'), ord('6'), ord('7'), ord('8'), ord('9'),  
 ord('A'), ord('B'), ord('C'), ord('D'), ord('E'), ord('F'), ord('G'), ord('H'), ord('I'), ord('J'),  
 ord('K'), ord('L'), ord('M'), ord('N'), ord('O'), ord('P'), ord('Q'), ord('R'), ord('S'), ord('T'),  
 ord('U'), ord('V'), ord('W'), ord('X'), ord('Y'), ord('Z')] *#Là mã ascii của mấy chữ này* for npaContour in npaContours: *# for each contour* if cv2.contourArea(npaContour) > MIN\_CONTOUR\_AREA: *# if contour is big enough to consider* [intX, intY, intW, intH] = cv2.boundingRect(npaContour) *# get and break out bounding rect  
  
 # draw rectangle around each contour as we ask user for input* cv2.rectangle(imgTrainingNumbers, *# draw rectangle on original training image* (intX, intY), *# upper left corner* (intX+intW,intY+intH), *# lower right corner* (0, 0, 255), *# red* 2) *# thickness* imgROI = imgThresh[intY:intY+intH, intX:intX+intW] *# crop char out of threshold image* imgROIResized = cv2.resize(imgROI, (RESIZED\_IMAGE\_WIDTH, RESIZED\_IMAGE\_HEIGHT)) *# resize image, this will be more consistent for recognition and storage* cv2.imshow("imgROI", imgROI) *# show cropped out char for reference* cv2.imshow("imgROIResized", imgROIResized) *# show resized image for reference* cv2.imshow("training\_numbers.png", imgTrainingNumbers) *# show training numbers image, this will now have red rectangles drawn on it* intChar = cv2.waitKey(0) *# get key press* if intChar == 27: *# if esc key was pressed* sys.exit() *# exit program* elif intChar in intValidChars: *# else if the char is in the list of chars we are looking for . . .* intClassifications.append(intChar) *# append classification char to integer list of chars (we will convert to float later before writing to file)  
 #Là file chứa label của tất cả các ảnh mẫu, tổng cộng có 32 x 5 = 160 mẫu.* npaFlattenedImage = imgROIResized.reshape((1, RESIZED\_IMAGE\_WIDTH \* RESIZED\_IMAGE\_HEIGHT)) *# flatten image to 1d numpy array so we can write to file later* npaFlattenedImages = np.append(npaFlattenedImages, npaFlattenedImage, 0) *# add current flattened impage numpy array to list of flattened image numpy arrays  
   
 # end if  
 # end if  
 # end for* fltClassifications = np.array(intClassifications, np.float32) *# convert classifications list of ints to numpy array of floats* npaClassifications = fltClassifications.reshape((fltClassifications.size, 1)) *# flatten numpy array of floats to 1d so we can write to file later* print ("\n\ntraining complete !!\n")  
  
 np.savetxt("classifications.txt", npaClassifications) *# write flattened images to file* np.savetxt("flattened\_images.txt", npaFlattenedImages) *#* cv2.destroyAllWindows() *# remove windows from memory* return  
  
*###################################################################################################*if \_\_name\_\_ == "\_\_main\_\_":  
 main()  
*# end if*

**Chương trình main đầy đủ:**

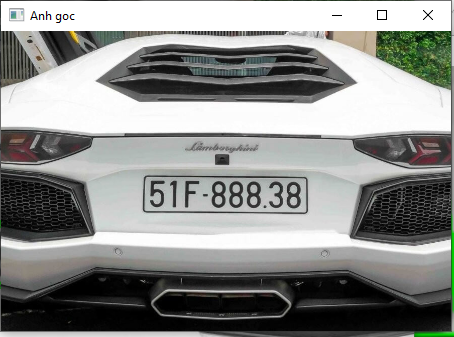
import tkinter as tk  
from tkinter import filedialog  
import cv2  
import numpy as np  
import math  
from PIL import Image, ImageTk  
import Preprocess  
  
class ImageProcessingApp:  
 def \_\_init\_\_(self, root):  
 self.root = root  
 self.root.title("License Plate Recognition")  
 self.root.geometry("600x600")  
 self.image\_path = tk.StringVar()  
 self.image\_label = tk.Label(self.root)  
 self.n = 1 *# Initialize n here  
  
 # Create and place widgets* self.create\_widgets()  
  
 def create\_widgets(self):  
 *# Create a button to choose an image* choose\_button = tk.Button(self.root, text="Chọn ảnh", command=self.choose\_image)  
 choose\_button.pack(pady=10)  
  
 *# Create a label to display the selected image path* path\_label = tk.Label(self.root, textvariable=self.image\_path)  
 path\_label.pack(pady=10)  
  
 *# Create a label to display the selected image* self.image\_label.pack(pady=10)  
  
 *# Create a button to process the selected image* process\_button = tk.Button(self.root, text="Kiểm tra", command=self.process\_image)  
 process\_button.pack(pady=10)  
  
  
  
 def choose\_image(self):  
 *# Open a file dialog to choose an image* file\_path = filedialog.askopenfilename(title="Chọn ảnh", filetypes=[("Image files", "\*.png;\*.jpg;\*.jpeg")])  
  
 *# Update the image path variable and display it* self.image\_path.set(file\_path)  
  
 *# Display the selected image* if file\_path:  
 image = Image.open(file\_path)  
 image = image.resize((300, 200)) *# Resize the image for display* photo = ImageTk.PhotoImage(image)  
 self.image\_label.config(image=photo)  
 self.image\_label.image = photo  
  
 *# def show\_image(self, img, window\_name):  
 # cv2.imshow(window\_name, img)  
 # cv2.waitKey(0)* def show\_image(self, img, window\_name, width=450, height=300):  
 *# Resize ảnh để hiển thị trong khung có kích thước cố định* resized\_img = cv2.resize(img, (width, height))  
 cv2.imshow(window\_name, resized\_img)  
 *# cv2.waitKey(0)  
 # cv2.destroyAllWindows()* def process\_image(self):  
 *# Thực hiện xử lý ảnh với đường dẫn đã chọn* selected\_path = self.image\_path.get()  
 if selected\_path:  
 *# Thực hiện xử lý ảnh như trong main.py* img = cv2.imread(selected\_path)  
 img = cv2.resize(img, dsize=(1920, 1080))  
  
 *###################### If you want to try increasing the contrast #############  
 # img2 = cv2.imread("1.jpg")  
 # imgGrayscaleplate2, \_ = Preprocess.preprocess(img)  
 # imgThreshplate2 = cv2.adaptiveThreshold(imgGrayscaleplate2, 250, cv2.ADAPTIVE\_THRESH\_GAUSSIAN\_C, cv2.THRESH\_BINARY\_INV, ADAPTIVE\_THRESH\_BLOCK\_SIZE ,ADAPTIVE\_THRESH\_WEIGHT )  
 # cv2.imshow("imgThreshplate2",imgThreshplate2)  
 ###############################################################  
  
 ######## Upload KNN model ######################* npaClassifications = np.loadtxt("classificationS.txt", np.float32)  
 npaFlattenedImages = np.loadtxt("flattened\_images.txt", np.float32)  
 npaClassifications = npaClassifications.reshape(  
 (npaClassifications.size, 1)) *# reshape numpy array to 1d, necessary to pass to call to train* kNearest = cv2.ml.KNearest\_create() *# instantiate KNN object* kNearest.train(npaFlattenedImages, cv2.ml.ROW\_SAMPLE, npaClassifications)  
 *#########################  
  
 ################ Image Preprocessing #################* imgGrayscaleplate, imgThreshplate = Preprocess.preprocess(img)  
 canny\_image = cv2.Canny(imgThreshplate, 250, 255) *# Canny Edge* kernel = np.ones((3, 3), np.uint8)  
 dilated\_image = cv2.dilate(canny\_image, kernel, iterations=1) *# Dilation  
 # cv2.imshow("dilated\_image",dilated\_image)  
  
 ###########################################  
  
 # Hiển thị từng bước xử lý trên các cửa sổ riêng biệt* self.show\_image(img, "Anh goc")  
 cv2.waitKey(0)  
  
 self.show\_image(imgGrayscaleplate, "1. Anh xam")  
 cv2.waitKey(0)  
  
 imgContrast = cv2.convertScaleAbs(imgGrayscaleplate, alpha=2.0, beta=0)  
 self.show\_image(imgContrast, "2. Tang cuong tuong phan")  
 cv2.waitKey(0)  
  
 imgGaussian = cv2.GaussianBlur(imgContrast, (5, 5), 0)  
 self.show\_image(imgGaussian, "3. Giam nhieu - Gaussian")  
 cv2.waitKey(0)  
  
 self.show\_image(imgThreshplate, "4. Anh nhi phan voi nguong dong")  
 cv2.waitKey(0)  
  
 canny\_image = cv2.Canny(imgThreshplate, 250, 255) *# Canny Edge* self.show\_image(canny\_image, "5. Phat hien Canny")  
 cv2.waitKey(0)  
  
  
 *###### Draw contour and filter out the license plate #############* contours, hierarchy = cv2.findContours(dilated\_image, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)  
 contours = sorted(contours, key=cv2.contourArea, reverse=True)[:10] *# Lấy 10 contours có diện tích lớn nhất* screenCnt = []  
 for c in contours:  
 peri = cv2.arcLength(c, True) *# Tính chu vi* approx = cv2.approxPolyDP(c, 0.06 \* peri, True) *# làm xấp xỉ đa giác, chỉ giữ contour có 4 cạnh* [x, y, w, h] = cv2.boundingRect(approx.copy())  
 ratio = w / h  
 if (len(approx) == 4):  
 screenCnt.append(approx)  
  
 detected = 0  
 if screenCnt:  
 detected = 1  
  
 if detected == 1:  
 for screenCnt in screenCnt:  
 cv2.drawContours(img, [screenCnt], -1, (0, 255, 0), 3) *# Khoanh vùng biển số xe  
  
 ############## Find the angle of the license plate #####################* (x1, y1) = screenCnt[0, 0]  
 (x2, y2) = screenCnt[1, 0]  
 (x3, y3) = screenCnt[2, 0]  
 (x4, y4) = screenCnt[3, 0]  
 array = [[x1, y1], [x2, y2], [x3, y3], [x4, y4]]  
 sorted\_array = array.sort(reverse=True, key=lambda x: x[1])  
 (x1, y1) = array[0]  
 (x2, y2) = array[1]  
 doi = abs(y1 - y2)  
 ke = abs(x1 - x2)  
 angle = math.atan(doi / ke) \* (180.0 / math.pi)  
  
 *####################################  
  
 ########## Crop out the license plate and align it to the right angle ################* mask = np.zeros(imgGrayscaleplate.shape, np.uint8)  
 new\_image = cv2.drawContours(mask, [screenCnt], 0, 255, -1, )  
  
 *# Cropping* (x, y) = np.where(mask == 255)  
 (topx, topy) = (np.min(x), np.min(y))  
 (bottomx, bottomy) = (np.max(x), np.max(y))  
  
 roi = img[topx:bottomx, topy:bottomy]  
 imgThresh = imgThreshplate[topx:bottomx, topy:bottomy]  
 ptPlateCenter = (bottomx - topx) / 2, (bottomy - topy) / 2  
  
 if x1 < x2:  
 rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, -angle, 1.0)  
 else:  
 rotationMatrix = cv2.getRotationMatrix2D(ptPlateCenter, angle, 1.0)  
  
 roi = cv2.warpAffine(roi, rotationMatrix, (bottomy - topy, bottomx - topx))  
 imgThresh = cv2.warpAffine(imgThresh, rotationMatrix, (bottomy - topy, bottomx - topx))  
 roi = cv2.resize(roi, (0, 0), fx=3, fy=3)  
 imgThresh = cv2.resize(imgThresh, (0, 0), fx=3, fy=3)  
  
 *####################################  
  
 #################### Prepocessing and Character segmentation ####################* kerel3 = cv2.getStructuringElement(cv2.MORPH\_RECT, (3, 3))  
 thre\_mor = cv2.morphologyEx(imgThresh, cv2.MORPH\_DILATE, kerel3)  
 cont, hier = cv2.findContours(thre\_mor, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)  
  
 *##################### Filter out characters #################* char\_x\_ind = {}  
 char\_x = []  
 height, width, \_ = roi.shape  
 roiarea = height \* width  
  
 for ind, cnt in enumerate(cont):  
 (x, y, w, h) = cv2.boundingRect(cont[ind])  
 ratiochar = w / h  
 char\_area = w \* h  
  
 if (0.01 \* roiarea < char\_area < 0.09 \* roiarea) and (0.25 < ratiochar < 0.7):  
 if x in char\_x:  
 x = x + 1  
 char\_x.append(x)  
 char\_x\_ind[x] = ind  
  
 *############ Character recognition ##########################* char\_x = sorted(char\_x)  
 strFinalString = ""  
 first\_line = ""  
 second\_line = ""  
  
 for i in char\_x:  
 (x, y, w, h) = cv2.boundingRect(cont[char\_x\_ind[i]])  
 cv2.rectangle(roi, (x, y), (x + w, y + h), (0, 255, 0), 2)  
  
 imgROI = thre\_mor[y:y + h, x:x + w] *# Crop the characters* imgROIResized = cv2.resize(imgROI, (20, 30)) *# resize image* npaROIResized = imgROIResized.reshape(  
 (1, 20 \* 30))  
  
 npaROIResized = np.float32(npaROIResized)  
 \_, npaResults, neigh\_resp, dists = kNearest.findNearest(npaROIResized, k=3)  
 strCurrentChar = str(chr(int(npaResults[0][0])))  
 cv2.putText(roi, strCurrentChar, (x, y + 50), cv2.FONT\_HERSHEY\_DUPLEX, 2, (255, 255, 0), 3)  
  
 if (y < height / 3):  
 first\_line = first\_line + strCurrentChar  
 else:  
 second\_line = second\_line + strCurrentChar  
  
 print("\n License Plate " + str(self.n) + " is: " + first\_line + " - " + second\_line + "\n")  
 roi = cv2.resize(roi, None, fx=0.75, fy=0.75)  
 *# cv2.imshow(str(self.n), cv2.cvtColor(roi, cv2.COLOR\_BGR2RGB))  
 # self.n += 1* cv2.namedWindow(str(self.n), cv2.WINDOW\_NORMAL)  
 cv2.resizeWindow(str(self.n), 980, 280)  
 cv2.imshow(str(self.n), cv2.cvtColor(roi, cv2.COLOR\_BGR2RGB))  
 self.n += 1  
  
 img = cv2.resize(img, None, fx=0.5, fy=0.5)  
 cv2.imshow('Loc bien Contour', img)  
 cv2.waitKey(0)  
 *#ad* cv2.destroyAllWindows()  
 else:  
 print("Vui lòng chọn ảnh trước khi kiểm tra.")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 root = tk.Tk()  
 app = ImageProcessingApp(root)  
 root.mainloop()

# Phần V. Kết luận

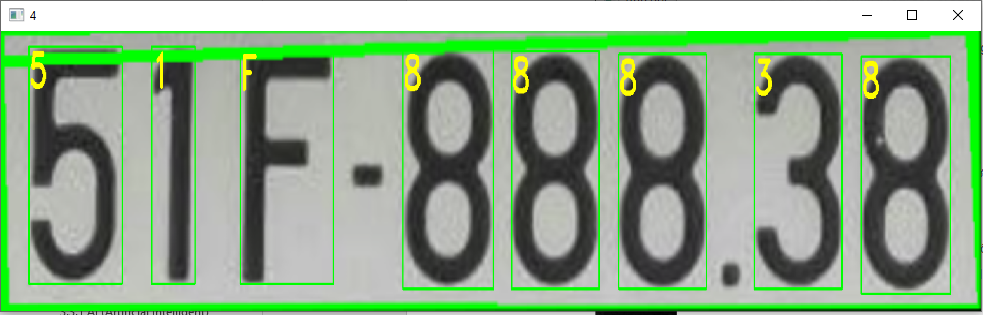
## 5.1 Kiểm thử chương trình :

* Độ chính xác của thuật toán là: 70%
* Số lượng ảnh test: 51
* Nguồn dữ liệu: ảnh trên mạng và ảnh tự chụp
* Các trường hợp nhận diện sai: ảnh có cường độ sáng quá lớn, ảnh có nhiều vật nhiễu, ảnh bị bóng, bị mờ

**Chương trình xử lý ra kết quả đúng :**



Hình 5. Ảnh ban đầu



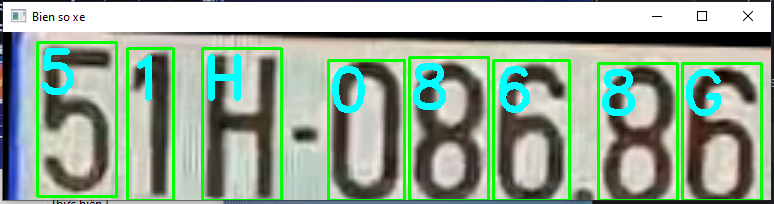
Hình 5. Ảnh kết quả đúng

Biển số xe ban đầu: 51F-88838, biển số xe chương trình đọc: 51F-88838

**Chương trình xử lý ra kết quả sai :**

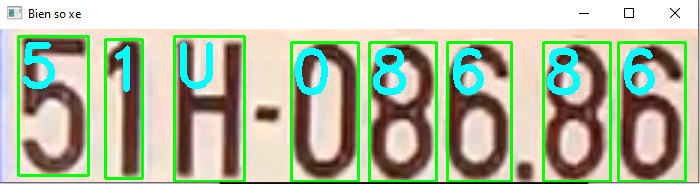


Hình 5. Ảnh ban đầu



Hình 5. Ảnh kết quả sai

Biển số xe ban đầu: 51H-08686, biển số xe chương trình đọc: 51H-0868G



Hình 5. Ảnh kết quả sai

Biển số xe ban đầu: 51H-08686, biển số xe chương trình đọc: 51U-08686

## 5.2 Ưu nhược điểm

Từ những kết quả thu được ở chương trên em nhận thấy phương pháp nhận diện biển số xe bằng xử lý ảnh và thuật toán KNN có những ưu và khuyết điểm sau:

* Ưu điểm:
  + Dễ cài đặt và sử dụng.
  + Khá nhẹ nên máy tính với cấu hình yếu cũng có thể xử lý mượt mà so với các thuật toán khác như CNN, SVM.
  + Phù hợp cho đối tượng sinh viên muốn tìm hiểu căn bản về xử lý ảnh hay trí tuệ nhân tạo.
* Khuyết điểm:
  + Khả năng nhận diện của KNN còn thấp, khi tập dữ liệu quá nhiều sẽ tăng thời gian xử lý vì phải quét hết tập dữ liệu train.
  + Nhận diện kém với sự phản chiếu của biển số, sự di ảnh, chói sáng từ môi trường ngoài, những biển có phần chữ số không rõ ràng, với biển số xe ô tô

Vì vậy tốt nhất cần đặt camera cố định, với môi trường ánh sáng xung quanh được cài đặt trước. Phông nền cần hạn chế tối da nhưng chi tiết lóe sáng gây nhiễu. Phương pháp này vẫn còn cần sự giám sát của con người nhiều chứ chưa thể hoàn toàn tự động.

# Tài liệu tham khảo

1. M. J. Ahmed, M. Sarfaz, A. Zidouri, and K. G. AI-Khatib, “License plate recognition system,” *Proc. IEEE Int. Conf. Electron. Circuits, Syst.*, vol. 2, no. January, pp. 898–901, 2003, doi: 10.1109/ICECS.2003.1301932.
2. C. N. E. Anagnostopoulos, “License plate recognition: A brief tutorial,” *IEEE Intell. Transp. Syst. Mag.*, vol. 6, no. 1, pp. 59–67, 2014, doi: 10.1109/MITS.2013.2292652.
3. A. Badr, M. M. Abdel, A. M. Thabet, and A. M. Abdelsadek, “Automatic number plate recognition system,” *Ann. Univ. Craiova, Math. Comput. Sci. Ser.*, vol. 38, no. 1, pp. 62–71, 2011, doi: 10.5120/ijca2018917277.
4. S. L. Chang, L. S. Chen, Y. C. Chung, and S. W. Chen, “Automatic License Plate Recognition,” *IEEE Trans. Intell. Transp. Syst.*, vol. 5, no. 1, pp. 42–53, 2004, doi: 10.1109/TITS.2004.825086.
5. N. D. Linh, N. Van Nhan, and D. Van Dat, “12.pdf,” *Tap chi thong tin khoa hoc va cong nghe Quang Binh*, 2018.
6. D. V. R. Mohan, M. T. Communication, S. Srkr, and E. College, “Number Plate Recognition by using open CV- Python,” pp. 4987–4992, 2019.
7. Nguyễn Vĩnh An, “So sánh một số phương pháp phát hiện biên,” *Tạp chí khoa học Trường Đại học Quốc gia Hà Nội*, vol. 31, no. 2, pp. 1–7, 2015.