**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA KHOA HỌC MÁY TÍNH**

# BÁO CÁO ĐỒ ÁN

# NHẬN DẠNG BIỂN SỐ XE TỰ ĐỘNG

# AUTOMATIC LICENSE PLATE RECOGNITION

**CS231.M22.KHCL**

**GIẢNG VIÊN HƯỚNG DẪN: MAI TIẾN DŨNG**

**SINH VIÊN THỰC HIỆN: ĐÀO TRẦN ANH TUẤN – 20522107**

**TRẦN PHÚ VINH - 20522161**

**TP. HỒ CHÍ MINH, 6/2022**

# MỤC LỤC

# 1. GIỚI THIỆU BÀI TOÁN

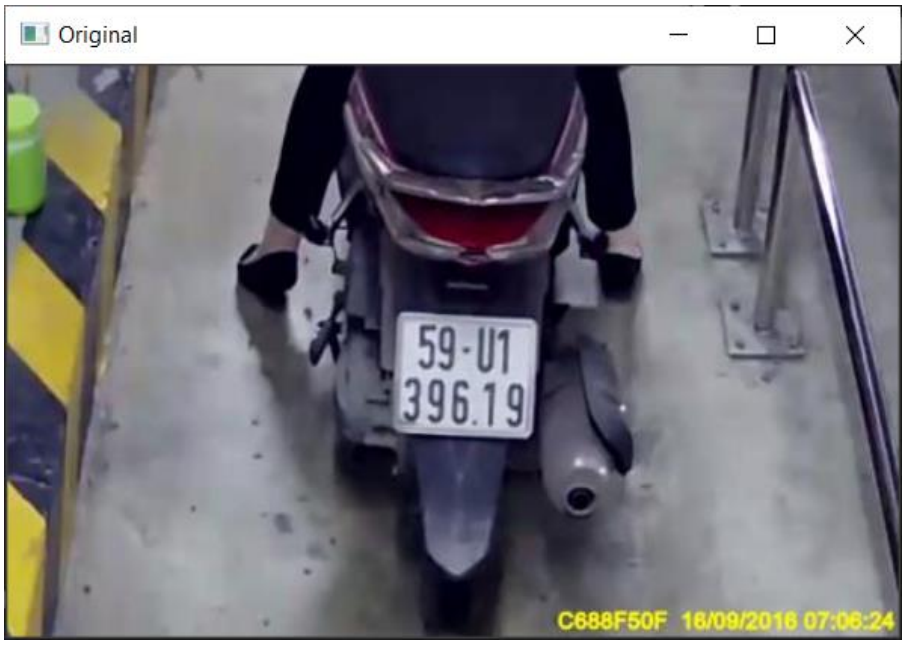
## 1.1. Mục đích bài toán

Hiện nay, các bãi giữ xe đã dần ứng dụng rộng rãi các phần mềm cho phép đọc và ghi biển số xe tự động giúp cho việc quản lí xe hiệu quả hơn và hạn chế sai sót của con người.

Trong phạm vi môn học Nhập môn Thị giác máy tính, nhóm chúng em muốn dùng các kiến thức về xử lí ảnh (OpenCV và Python) đã học và tự tìm kiếm thêm một số phương pháp để thử giải quyết công việc phát hiện và nhận dạng biển số tại các bãi giữ xe Việt Nam.

## 1.2. Input và Output của bài toán

- Input: Một bức ảnh có chứa một xe máy được gắn một biển số được cấp tại Việt Nam.




Hình 1: Hình ảnh Input.

- Output: vị trí của biển số trong bức ảnh đầu vào và các ký tự có trong biển số vừa được phát hiện.


Hình 2: Hình ảnh Output.

# 2. PIPELINE

Chúng em chia bài toán thành hai phần: phát hiện biển số và đọc các ký tự từ biển số vừa phát hiện được(nếu có).

## 2.1. Phát hiện biển số (License plate detection)

### 2.1.1. Handcraft Algorithm

Nhóm chúng em đề xuất trình tự các bước xử lý như sau:

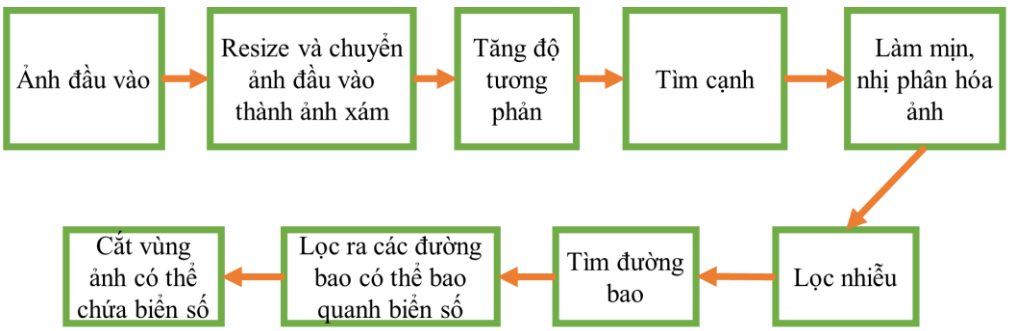



Figure 1: Sơ đồ quá trình phát hiện biển số xe máy Việt Nam.

### 1) Resize và chuyển ảnh đầu vào thành ảnh xám

Từ ảnh ban đầu, chúng em sẽ giữ nguyên tỷ lệ của ảnh khi chuyển ảnh về kích thước mới: $new\_width = 600$ và $new\_height = height * \frac{600}{old\_width}$. Tiếp theo, chuyển ảnh đã được resize thành ảnh xám.


Hình 3: Ảnh sau khi được resize và chuyển thành ảnh xám.

## 2) Tăng độ tương phản

Nhóm em sử dụng hai phép biến đổi là **Black hat** và **Top hat (White hat)**

### a) Black hat

Phép Black hat sẽ làm nổi bật các đối tượng tối trong nền sáng.


Hình 4: Ảnh Black hat.

### b) Top hat (White hat)

Phép Top hat sẽ làm nổi bật các đối tượng sáng trong nền tối.


Hình 5: Ảnh Top hat.

Khi đã có ảnh Black hat và Top hat, nhóm em tiến hành tăng độ tương phản bằng cách cộng ảnh xám ban đầu với ảnh Top hat rồi trừ đi ảnh Black hat.


Hình 6: Ảnh xám được tăng độ tương phản.

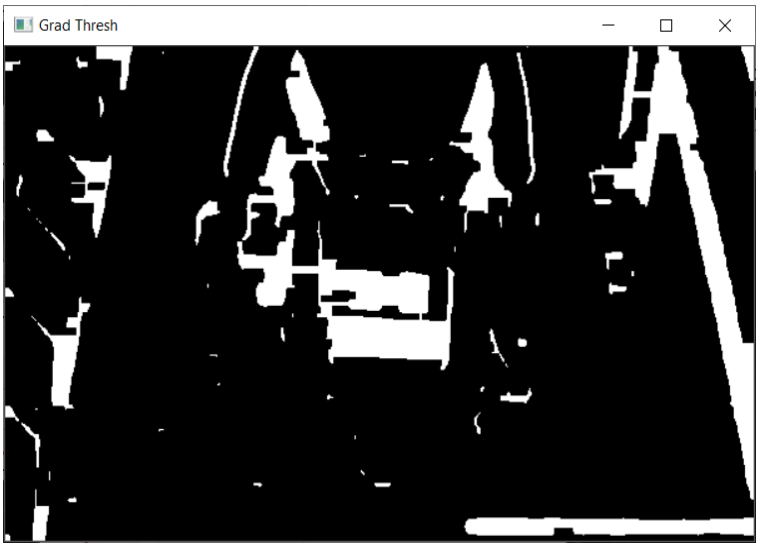
### 3) Tìm cạnh

Nhóm em tìm cạnh bằng phương pháp Sobel – Scharr.


Hình 7: Cạnh tìm được bằng Sobel – Scharr.

### 4) Làm mịn và nhị phân hóa ảnh

Tiếp đến, nhóm em tiến hành làm mịn ảnh bằng Gaussian blur và nhị phân hóa ảnh bằng Otsu.




Hình 8: Ảnh sau khi làm mịn và nhị phân hóa.

### 5) Lọc nhiễu

Sau khi nhị phân hóa, ảnh sẽ có các chấm trắng nhiễu ảnh hưởng đến kết quả tìm đường bao (contour). Do đó, nhóm em lọc nhiễu bằng phép toán hình thái MorphologyEx của openCV, cụ thể là phép mở (Open). Gồm hai bước: sử dụng phép toán hình thái Erode để xói mòn các chấm nhiễu, tiếp đến là Dilate để khiến các đối tượng không phải chấm nhiễu trở về kích thước ban đầu.


Hình 9: Ảnh sau khi lọc bỏ các chấm trắng gây nhiễu.

Trước khi tìm đường bao, nhóm em sẽ sử dụng một phương pháp khác là Mask để giữ lại phần sáng của ảnh, bao gồm biển số. Bắt đầu với ảnh phần sáng được tính bằng phương pháp threshold Otsu trên ảnh xám đã tăng độ tương phản, nhóm em dùng ảnh phần sáng này làm mask để thực hiện phép bitwise AND giữa ảnh được lọc nhiễu với chính nó. Nghĩa là ở các pixel của ảnh được lọc nhiễu sẽ AND với chính nó ở vị trí pixel của mask khác 0. Kết quả là các vùng sáng trong ảnh xám ban đầu được giữ lại trên ảnh được lọc nhiễu.

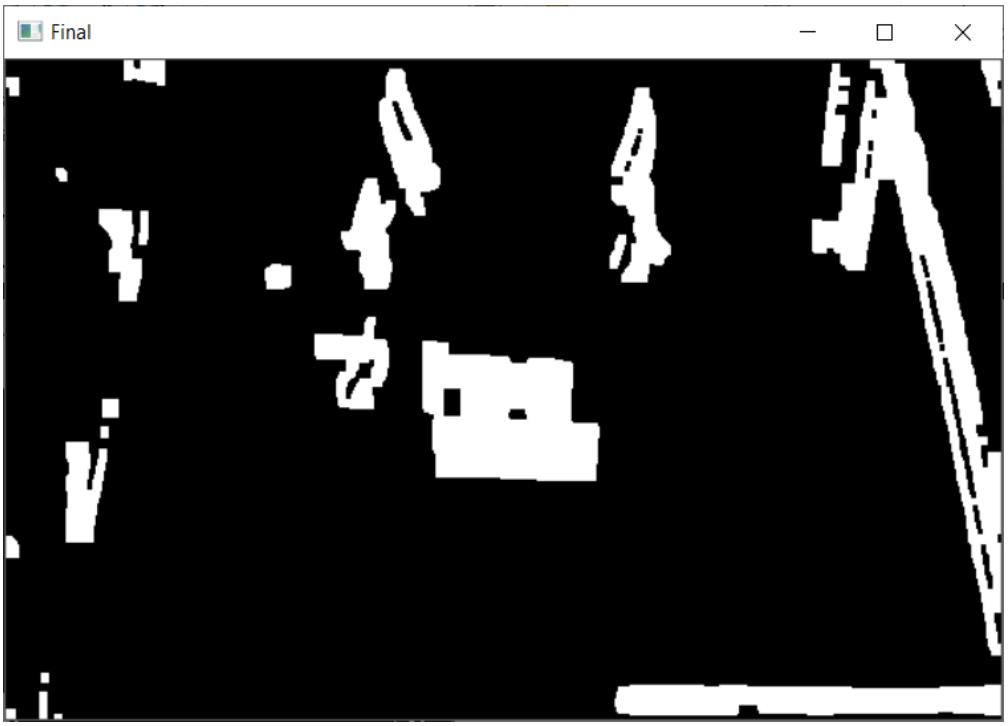

Hình 10: Quá trình áp dụng Mask.


Hình 11: Ảnh vùng sáng được dùng để làm Mask.


Hình 12: Ảnh được lọc nhiễu và giữ lại các vùng sáng.

Để lấp đầy các phần bị trống và nối vùng sáng của hàng trên với hàng dưới giúp cho việc tìm bao đóng của biển số được tốt hơn, chúng em sử dụng phép toán hình thái

(morphologyEx) đặc biệt là phép đóng (CLOSE). Gồm hai bước: sử dụng phép toán hình thái Dilate mở rộng các vùng trắng và lấp đầy lỗ trống, sau đó là Erode xói mòn vùng trắng về kích thước ban đầu.

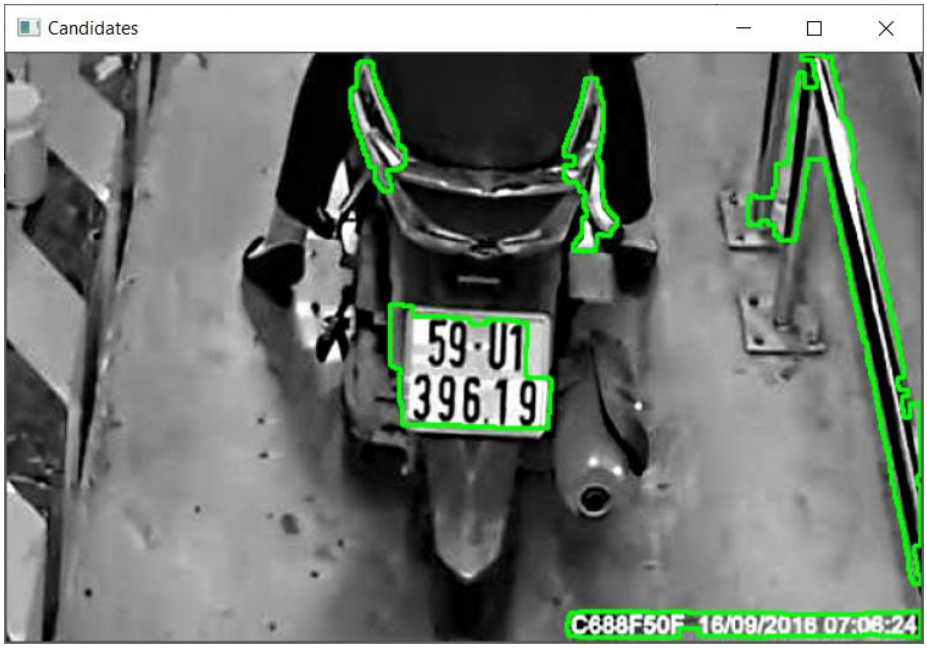



Hình 13: Sau khi dùng Close, các lỗ trống được lắp đầy.

**6) Tìm đường bao (contour)**

Đường bao có thể được giải thích đơn giản là một đường cong nối tất cả các điểm liên tục (dọc theo đường biên), có cùng màu hoặc cường độ. Các đường bao là một công cụ hữu ích để phân tích hình dạng, phát hiện và nhận dạng đối tượng. Nếu bức ảnh có nhiễu gần nhau thì thuật toán tìm đường bao sẽ bao tất cả các nhiễu này, tạo nên những đường bao lớn, không cần thiết. Các đường bao có chứa biển số nhưng bị ảnh hưởng bởi nhiễu xung quanh làm giảm độ chính xác ở bước cắt biển số.

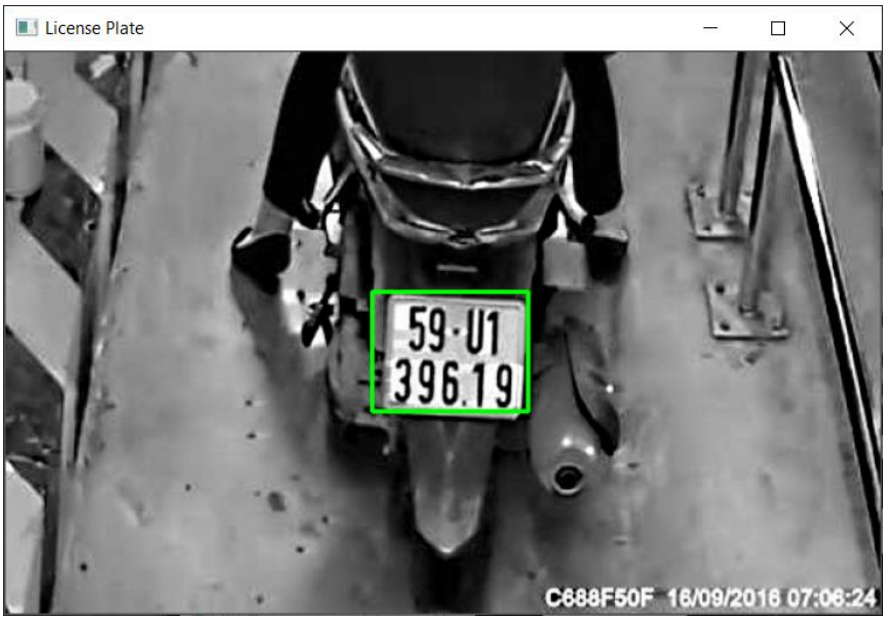
Nhóm em tìm đường bao trên ảnh bằng OpenCV, sau đó lấy ra 5 đường bao có diện tích lớn nhất. Nhóm chúng em cho rằng 5 đường bao này có thể bao gồm biển số do bước lọc nhiễu giúp hạn chế các đường bao có kích thước lớn nhưng không chứa biển số.




Hình 14: 5 đường bao có diện tích lớn nhất trong hình.

### 7) Lọc ra các đường bao

5 đường bao lớn nhất được xấp xỉ thành hình chữ nhật và tính lệ cạnh. Tỉ lệ cạnh của biển số xe máy Việt Nam là 190/140 ≈ 1.35. Ở đây, nhóm em lấy sai số là $1.35 \pm 0.25$. Các đường bao thỏa mãn có thể là các đường bao bao quanh vùng biển số.




Hình 15: Đường bao được lọc ra có chứa biển số.

### 8) Cắt vùng ảnh có chứa biển số

Cắt ảnh từ hình chữ nhật đã được xấp xỉ ở bước trước đó, chúng em dùng hàm cv.minArea để tạo ra được hình chữ nhật bao quanh vùng contour một cách tối ưu nhất để cắt ảnh. Việc này sẽ giúp cho việc phân đoạn kí tự ở giai đoạn sau được tốt hơn.


Hình 16: Cắt ảnh biển số tối ưu.

### 2.1.2. Haar – Cascades Algorithm

Ngoài thuật toán Handcraft mà nhóm tự đề xuất, chúng em còn sử dụng thuật toán Haar – Cascades được OpenCV tích hợp sẵn để so sánh với thuật toán Handcraft ở khẩu phát hiện biển số.

Haar – Cascades là thuật toán phân loại dựa trên đặc trưng Haar được đề xuất vào năm 2001 bởi Paul Viola và Michael Jones trong bài báo "Rapid Object Detection using a Boosted Cascade of Simple Features". Ban đầu, Haar-cascades được dùng để nhận dạng khuôn mặt, nhưng mô hình vẫn có thể được train để nhận dạng các đối tượng khác. Đây là thuật toán máy học dựa vào việc huấn luyện trên rất nhiều ảnh chứa và không chứa đối tượng cần phân loại.

Nhóm xài mô hình Haar-cascades được train sẵn từ: https://github.com/v-haopt12/ALPR-project.

## 2.2. Nhận dạng ký tự trên biển số (Character Recognition)

Về phần nhận dạng các ký tự trên biển số, với ảnh đầu vào là ảnh đầu ra của phần phát hiện biển số, chúng em xử lý theo 2 cách như sau:

### 2.2.1. Sử dụng phân đoạn ảnh để phát hiện kí tự:

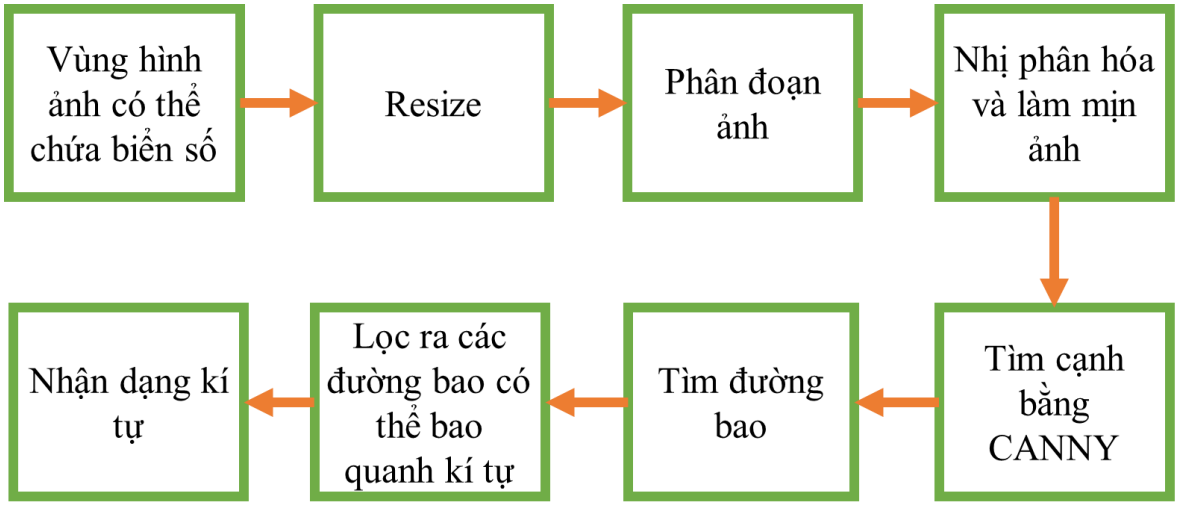



Figure 2: Sơ đồ xử lý ảnh, phần đoạn ảnh để nhận dạng ký tự trên biển số.

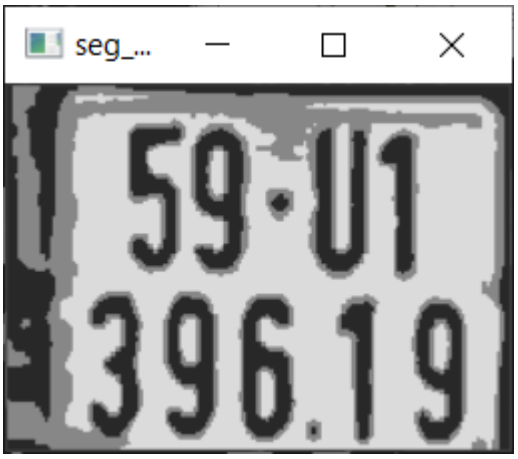
**1) Resize**

Ảnh được resize giữ nguyên tỉ lệ $new\_width = 200$ và $new\_height = height * \frac{200}{old\_width}$.


Hình 17: Ảnh cắt ra được resize.

### 2) Phân đoạn ảnh

Nhóm em chọn K-Means để phân đoạn ảnh thành 3 màu: trắng (nền biển số), xám (nhiễu), đen (màu ký tự trên biển).




Hình 18: Ảnh được phân đoạn thành 3 phần: xám, trắng, đen.

### 3) Nhị phân hóa và làm mịn ảnh

Ảnh vừa phân đoạn được nhị phân hóa và làm mịn bằng Gaussian blur

### 4) Tìm cạnh

Để tìm cạnh, chúng em chọn phương pháp Canny.


Hình 19: Cạnh tìm được bằng Canny.

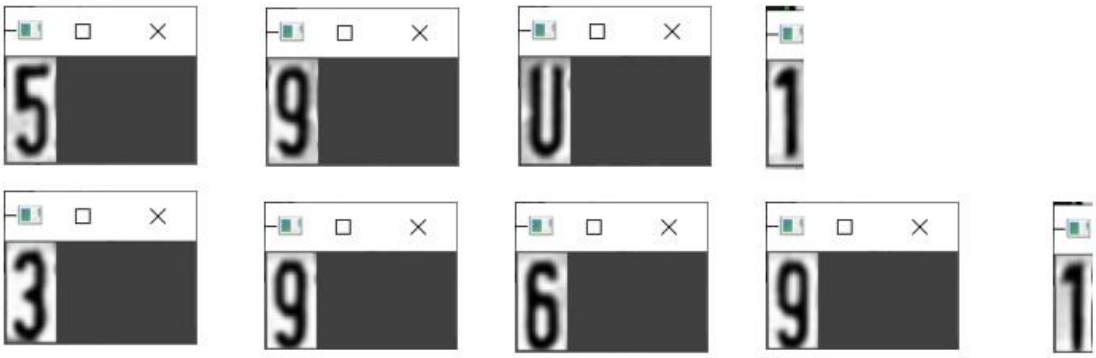
**5) Tim đường bao**

Kế đến, tìm đường bao trên ảnh vừa tìm cạnh. Để các đường bao bao quanh ký tự thuận tiện cho việc nhận dạng.


Hình 20: Các đường bao tìm được lẫn lộn số và nền.

**6) Lọc ra các đường bao**

Tuy nhiên, các đường bao vừa tìm được sẽ lẫn lộn nền với kí tự, do đó nhóm em sẽ xấp xỉ các đường bao thành hình chữ nhật và giữ lại đường bao nào có tỉ lệ phù hợp. Tỉ lệ nhóm tụi em chọn là chiều cao: 55, chiều rộng 22. Đường bao thích hợp được dùng để cắt ký tự cho bước sau.



Hình 21: Các ký tự được cắt ra từ đường bao qua bước chọn lọc.

**7) Nhận dạng ký tự**

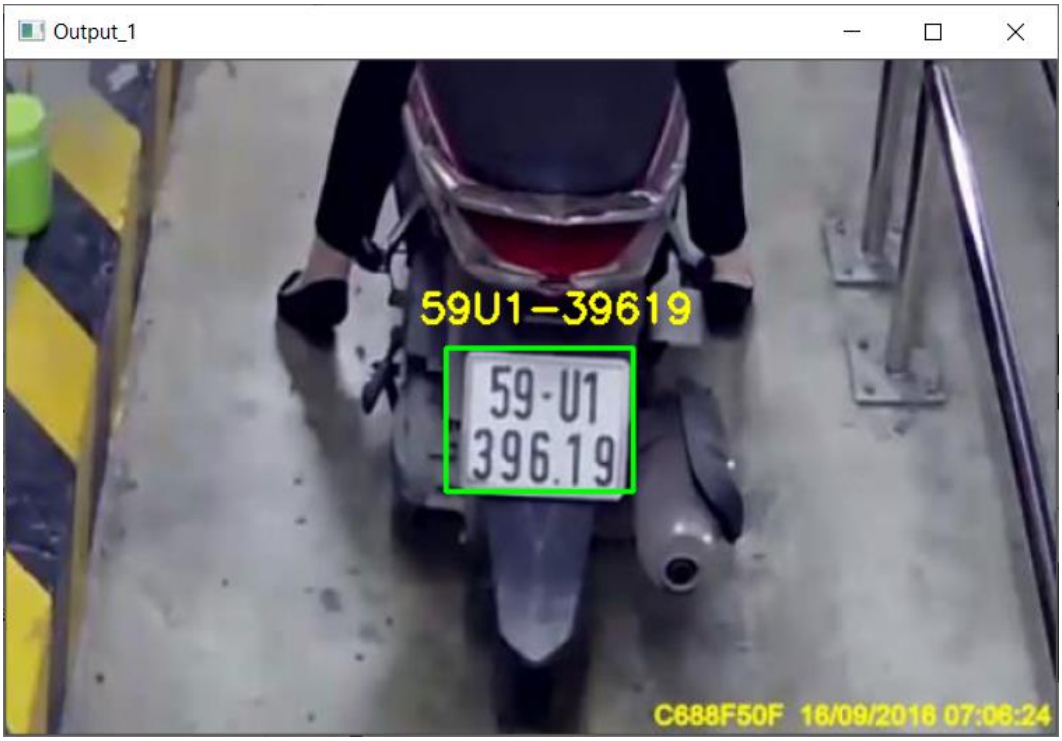
Để nhận dạng ký tự, tụi em chọn thư viện EasyOCR. Việc nhận dạng ký tự tụi em thực hiện trên từng ký tự vừa cắt được. Thực hiện nhận dạng như vậy sẽ tận dụng được tính năng allow_list của EasyOCR, buộc EasyOCR chỉ được phép nhận dạng

các kí tự trong ký tự cho sẵn. Điều này khiến việc nhận dạng được tốt hơn, tránh nhầm lẫn giữa chữ và số (điển hình là Z và 2). Việc này khả thi do biển số xe máy ở Việt Nam thường có một kí tự chữ ở vị trí thứ 3 hàng trên (xe trên 50 phân khối) và hai ký tự chữ ở vị trí thứ 3 và 4  hàng trên (xe 50 phân khối).


Hình 22: Kết quả nhận dạng từng ký tự của EasyOCR.

**8) Kết quả cuối cùng**

Gộp kết quả của hai bước phát hiện biển số và nhận dạng ký tự, nhóm em thu được ảnh sau:




Hình 23: Kết quả phát hiện và nhận dạng ký tự biển số xe Việt Nam.

### 2.2.2. Không dùng phân đoạn ảnh

Tuy nhiên, biển số của Việt Nam còn có các biển nước ngoài hiếm gặp, nên việc dùng allow_list trên từng vị trí ký tự trên biển số bằng EasyOCR không bao quát được trường hợp này. Do đó, chúng em tiến hành đưa thẳng output của phần phát hiện biển số mà không qua xử lý vào mô hình EasyOCR để nhận dạng ký tự trong ảnh.

Ưu điểm của cách này thời gian thực thi nhanh hơn so với cách phân đoạn ảnh (kết quả sẽ trình bày ở bên dưới) và có thể bao quát nhiều loại biển số hơn. Tuy nhiên, cách này sẽ bị hạn chế bởi vấn đề sự nhầm lẫn giữa chữ và số, ví dụ Z-2, L-4,… do không có được sự phân biệt hàng kí tự trên và dưới của biển số.

# 3. DỮ LIỆU

Do chúng em tự đề xuất thuật toán chỉ chủ yếu xài hàm OpenCV và sử dụng thuật toán Haar – Cascades được train sẵn nên cả hai thuật toán không cần phải train. DO đó tập dữ liệu Green Parking có thể coi là tập để đánh giá (validation hoặc test).

Tập dữ liệu bao gồm:

- 1747 ảnh biển số xe do GreenParking cung cấp trên trang (https://miai.vn/thu-vien-mi-ai/) có kích thước 472 x 303, cường độ sáng trung bình của mỗi ảnh 111.
- File ground truth chứa tên từng ảnh cùng với tọa độ ground truth bounding box bao sát biển số xe và thông tin chính xác trên biển số.

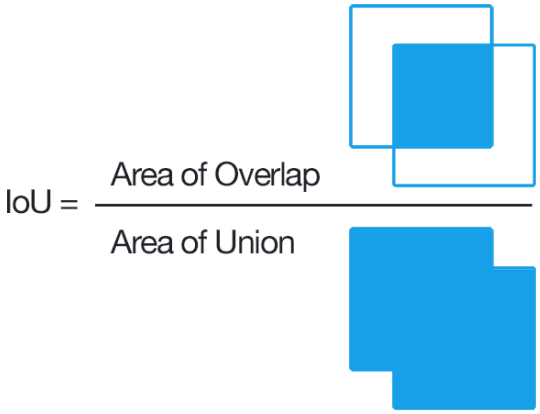
# 4. ĐÁNH GIÁ

Nhóm chúng em sử dụng thang đo:

- Phần License Plate Detection: IoU với threshold 0.5, Precision, Recall, F1
- Phần Character Recognition: Accuracy: kết quả nhận dạng ký tự của thuật toán phải giống hoàn toàn với ground truth mới tính là đúng, nếu không sẽ tính là sai.

### 4.1. IoU (Intersection over Union)

Iou chỉ ra độ khớp giữa bounding box dự đoán và ground truth box. Nếu IoU lớn hơn một ngưỡng nhất định (thường là 0.5) sẽ được đánh giá là tốt.




Hình 24: Minh họa cách tính IoU

### 4.2. Precision và Recall

**Precision**: trả lời cho câu trong các bounding box được dự đoán thì tỷ lệ trường hợp bounding box có IoU ≥ 0.5.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\ detections}$$

**Recall**: đo lường tỷ lệ bounding box được dự đoán có IoU ≥ 0.5 trên tổng số ground truth box.

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\ ground\ truths}$$

Trong đó:

- **TP** (true positive): những bbox có IoU >= 0.5
- **FP** (false positive) những bbox có IoU < 0.5
- **FN** (false negative): mô hình không dự đoán được bbox.

## 4.3. F1 – score

F1 – score là trung bình điều hòa giữa precision và recall. Do đó, nó đại diện hơn trong việc đánh giá độ chính xác trên đồng thời cả hai thang đo precision và recall.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

## 4.4. Cấu hình thực hiện đánh giá

Đánh giá mô hình được thực hiện trên Colab với cấu hình:

- CPU: 2 core Intel(R) Xeon(R) CPU @ 2.20GHz
- GPU: Tesla T4

Phần detection được thực hiện trên CPU.

Phần recognition được thực hiện trên GPU (có thể thực hiện được trên CPU).

## 4.5. Kết quả

### 4.5.1. Phát hiện biển số

| | **Handcraft** | **Haar – Cascades** |
|---|---|---|
| Thời gian chạy (s) | **20 (s)** | 48.6 (s) |
| Average IoU | 0.7654 | **0.7744** |
| Precision | 0.9298 | **1.0** |
| Recall | **0.8663** | 0.4865 |
| F1 - score | **0.8969** | 0.6546 |

Bảng 1: Kết quả phát hiện biển số

### 4.5.2. Nhận dạng ký tự trên biển

Việc nhận dạng ký tự được thực hiện trên output phát hiện được từ thuật toán Handcraft do nhóm muốn tập trung vào thuật toán tự đề xuất, còn Haar – Cascades chỉ dùng để so sánh ở bước phát hiện biển số.

| Input | **Có segmentation** | **Không segmentation** |
|---|---|---|
| Thời gian chạy (s) | 292 (s) | **58.3 (s)** |
| Accuracy | **0.5513** | 0.5064 |

Bảng 2: So sánh giữa hai phương pháp xử lý trước khi nhận dạng ký tự

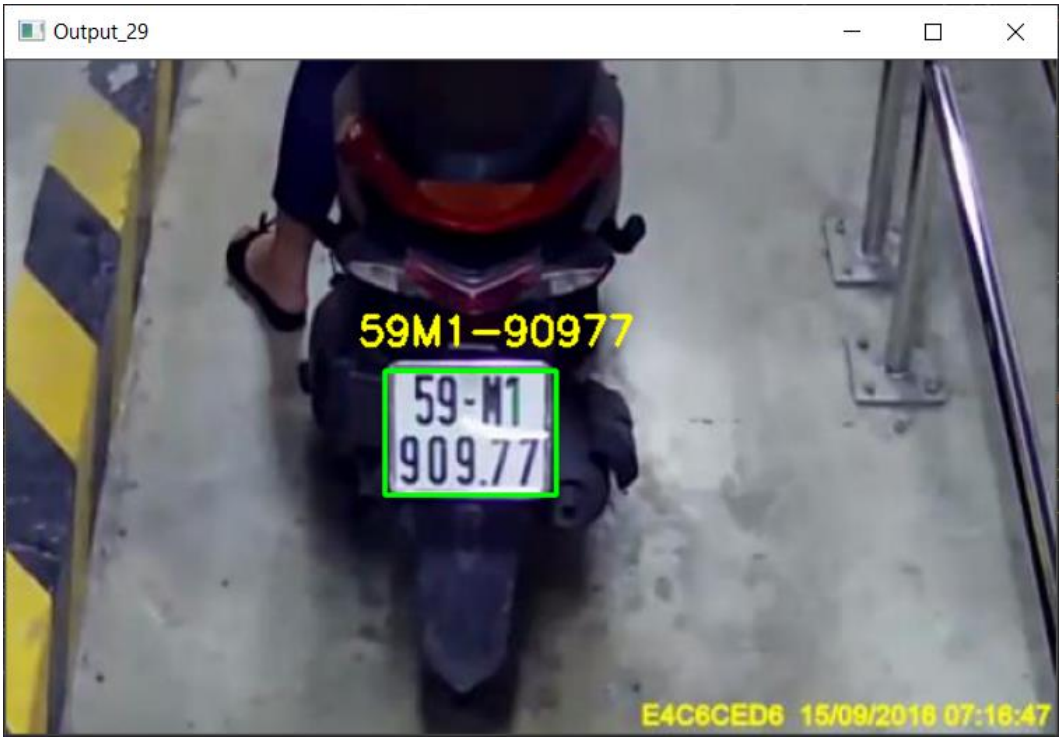
### 4.5.3. Một số hình ảnh kết quả đã làm tốt


Hình 25: Ký tự trên biển bị chói


Hình 26: Biển số bám bụi.


Hình 27: Ký tự trên biển bị chói


Hình 28: Hình ảnh bị mờ




Hình 29: Biển số có vùng bị chói


Hình 30: Biển số có viền dày

# 5. NHẬN XÉT

## 5.1. Ưu điểm

- Do thuật toán nhóm tự đề xuất tập trung vào OpenCV và Python nên chương trình dễ dàng cài đặt và sử dụng.
- Có thể không cần GPU để thực thi. Việc phát hiện biển số hoàn toàn do CPU đảm nhiệm còn việc nhận dạng ký tự do EasyOCR đảm nhiệm có thể thực hiện trên cả GPU và CPU
- Các thuật toán, phép biến đổi đều được OpenCV hỗ trợ, do đó phù hợp cho người mới tìm hiểu về xử lí ảnh.

## 5.2. Hạn chế

Thuật toán đề xuất của nhóm em vẫn còn nhiều hạn chế:

- Không phát hiện được biển số dính bùn, bị chói, gắn gầm, gắn bên hông, …
- Bounding box của những biển có vật liệu xung quanh cùng độ sáng với biển số thường bị phát hiện dư so với ground truth box hoặc có thể không phát hiện được.
- Ảnh phải được chụp đối diện vùng biển số, với góc < 15°, cách từ 2m - 3m, ảnh không quá tối hoặc quá sáng.
- Phần nhận dạng kí tự còn nhầm lẫn một vài trường hợp: X – H, M – N, V – Y, C – G, 1 – 7,…
- Mô hình chưa thể phát hiện được biển ngoại giao do nhóm chúng em chưa tìm được biển ngoại giao trên xe máy để điều chỉnh mô hình.


Hình 31: Mô hình phát hiện sai do có vật liệu kim loại chói phía trên biển số


Hình 32: Mô hình nhận dạng sai ký tự do biển số bị chói từ đèn chiếu hậu


Hình 33: Mô hình phát hiện dư vùng biển số do ốp lốc máy bên trái cùng độ sáng với biển số. Ký tự G bị lẫn lộn thành C


Hình 34: Biển số bị chói do đèn chiếu hậu nên thuật toán không phát hiện được biển số dẫn đến không nhận dạng được ký tự

# 6. CÀI ĐẶT VÀ SỬ DỤNG

Yêu cầu thư viện:

- OpenCV: 4.5.1.x
- imultils: 0.5.4
- pyparsing: 3.0.7
- pytorch (có thể tải bản cho CPU): 1.11.0
- EasyOCR (git clone từ phần tài liệu tham khảo)
- sklearn: 1.0.2
- numpy: 1.22.3

Cách sử dụng:

- git clone project của nhóm tại link: https://github.com/daotrananhtuan09102002/CS231.M22.KHCL-Project
- Cài đặt môi trường và các thư viện cần thiết:
- Dùng lệnh sau để chạy chương trình:

**python ANPR_Ver.1\ocr_license_plate.py --input .\GreenParking\ -s 1**

Trong đó

- Đường dẫn --input có thể đổi thành đường dẫn đến folder chứa ảnh mà mình cần phát hiện và nhận dạng biển số.
- Cờ -s cho phép chọn 2 giá trị 1 và 0 tương ứng với việc dùng character segmentation hoặc không dùng.
- Có thể dùng thêm cờ -d 1 để có thể coi từng bước xử lí ảnh.

- Dùng lệnh sau nếu muốn sử dụng HaarCascade để phát hiện biển số:

**python ANPR_Ver.2\anpr2.py**

Lưu ý: nếu muốn áp dụng cho ảnh khác cần phải vào file ANPR_Ver.2/anpr.2.py để chỉnh lại đường dẫn folder chứa ảnh.

# TÀI LIỆU THAM KHẢO


Thang đo cho phần evaluation:

- Evaluating Object Detection Models: Guide to Performance Metrics | Manal El Aidouni
- https://github.com/rafaelpadilla/Object-Detection-Metrics
- https://phamdinhkhanh.github.io/2020/08/13/ModelMetric.html
- https://towardsdatascience.com/a-single-number-metric-for-evaluating-object-detection-models-c97f4a98616d

EasyOCR: https://github.com/JaidedAI/EasyOCR

Phần license plate detection:

- https://pyimagesearch.com/2020/09/21/opencv-automatic-license-number-plate-recognition-anpr-with-python/#pyuni-reco-header
- https://github.com/v-haopt12/ALPR-project
- https://github.com/mrzaizai2k/VIETNAMESE_LICENSE_PLATE
- https://github.com/KostyaKulakov/Russian_System_of_ANPR
- https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
- https://pyimagesearch.com/2021/05/12/image-gradients-with-opencv-sobel-and-scharr/
- https://www.youtube.com/watch?v=NApYP_5wlKY
- https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html