**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HOC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

### Báo cáo đồ án

(Cải tiến và bổ sung)

Automatic license plate recognition (ALPR)

**GVHD: Võ Hoài Việt**

Mục lục

[***Báo cáo đồ án*** 1](#_Toc9790407)

[**A Thành viên nhóm:** 3](#_Toc9790408)

[**B Mức độ hoàn thành:** 3](#_Toc9790409)

[**C Báo cáo:** 3](#_Toc9790410)

[I. Tổng quan cách cài đặt: 3](#_Toc9790411)

[II. Yêu cầu của hệ thống: 11](#_Toc9790412)

[III. Kết quả thực nghiệm: 11](#_Toc9790413)

[**D Hướng dẫn sử dụng:** 16](#_Toc9790414)

[**E Tham khảo:** 17](#_Toc9790415)

# **A Thành viên nhóm:**

Tên nhóm: Fusion

|  |  |  |  |
| --- | --- | --- | --- |
| STT | MSSV | Họ tên | Email |
| 1 | 1612174 | Phùng Tiến Hào | [tienhaophung@gmail.com](mailto:tienhaophung@gmail.com) |
| 2 | 1612269 | Võ Quốc Huy | [voquochuy304@gmail.com](mailto:voquochuy304@gmail.com) |
| 3 | 1612272 | Trần Nhật Huy | [nhathuy13598@gmail.com](mailto:nhathuy13598@gmail.com) |

# **B Mức độ hoàn thành:**

|  |  |  |
| --- | --- | --- |
| STT | Nội dung | Mức độ hoàn thành (%) |
| 1 | Phát hiện biển số (Plate detection) | 100 |
| 2 | Kiểm tra biển số (Validation) | 100 |
| 3 | Nhận dạng kí tự trong biển số (Character recognition) | 100 |
| Tổng cộng: | | **100** |

# **C Báo cáo:**

## Tổng quan cách cài đặt:

Các bước thực hiện hiện:

A close up of a map

Description automatically generated

Figure 1 Lưu đồ hoạt động của hệ thống

Chi tiết hơn:

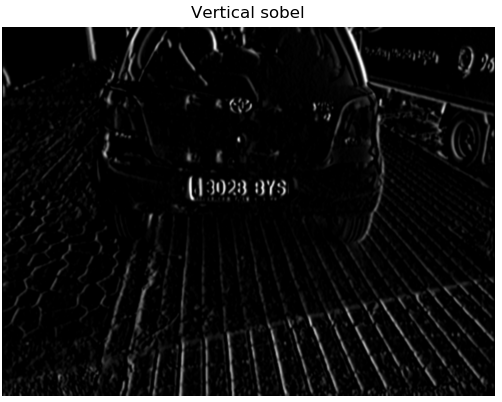
1. Scale ảnh về một kích thước chung: có width = 600 và height = (600 \* original\_height)/original\_width
2. Gray-scale image: để thực hiện tìm biên cạnh và phục vụ cho các bước sau



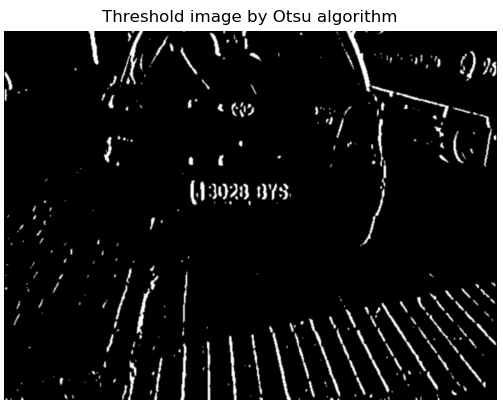
1. Làm mờ ảnh với Gaussian filter để giảm bớt nhiễu



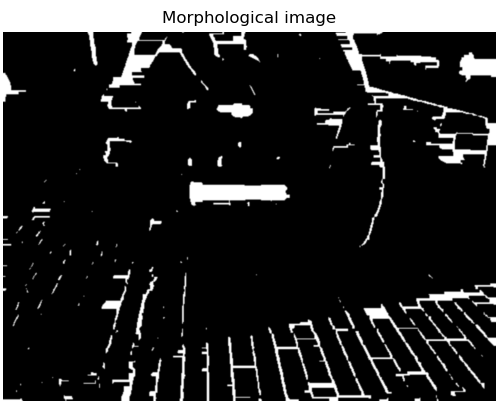
1. Tìm biên cạnh dọc bằng Sobel filter



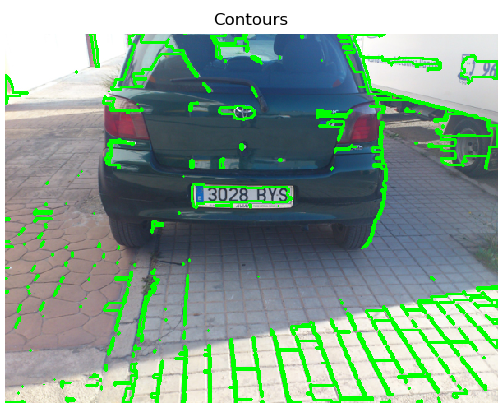
1. Threshold ảnh bằng phương pháp Otsu để nó tự tìm một ngưỡng tối ưu



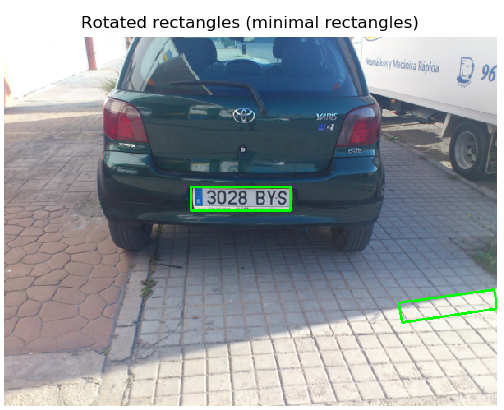
1. Áp dụng morphological operation, cụ thể là phép “Clossing”. Tức là trước tiên sẽ làm giản nỡ (dilation) ảnh sau đó làm xói mòn (erosion) ảnh để giảm nhiễu và xóa các lỗ chấm đen trên đối tượng. Với kích thước kernel = 19x3 (đây là kích thước thực nghiệm) chỉ áp dụng để nhận diện biển số xe ô tô với tối đa 8 kí tự.



1. Tìm contours để phân vùng ảnh (segmentation)



1. Tìm các bounding box hình chữ nhật (có tính góc nghiêng không quá 30 độ)



1. Kiểm tra và xác thực các bounding box:

Các công viên liên quan:

* Kiểm ra màu trắng có chiểm đáng kể trong vùng đó không
* Kiểm tra tỉ lệ khung hình cho phần vùng biển só:
  + Kích thước biển ô tô của ngước ngoài (cụ thể châu Tây Ban Nha hoặc Châu Âu): chiều rộng 52 cm, chiều cao 11 cm.
  + Do đó, tỉ lệ khung hình: 52/11 = 4.7272 với độ lỗi chấp nhận là 40%.
  + Dao động của tỉ lệ khung hình là min = 4.7272\*(1 – 0.4), max = 4.7272\*(1 + 0.4)
* Kiểm tra diện tích vùng trong khoảng từ: min = 15\*aspect\_ratio\*15, max = 125\*aspect\_ratio\*125
* Kiêm tra góc quay: Nếu bounding box xoay quá 30 độ thì bị loại.



1. Tận dụng phần nền biển số là màu trắng, tính cường độ màu trung bình của phần ảnh biển số đó xem có gần trắng hay không (> 100)



1. Nhận dạng kí tự trong biển số bằng thư viện mở Tesseract – được phát triển bởi Google



## Yêu cầu của hệ thống:

Để đảm bảo hệ thống hoạt động tốt và chính xác thì chất lượng ảnh đầu vào đóng vai trò rất quan trọng. Cụ thể:

* Ảnh phải được chụp trực diện và cách xa khoảng 1.5-2 (m).
* Điều kiện sáng đầy đủ
* Ảnh bị nhiễu ít
* Góc nghiêng vừa phải

Ngôn ngữ sử dụng: Python

Các thư viện cần thiết để chạy chương trình:

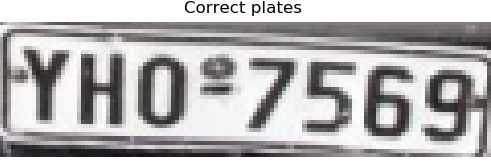
* Thư viện OpenCV
* Thư viện PIL
* Thư viện Matplotlib
* Thư viện Numpy
* Thư viện Pytesseract

## Kết quả thực nghiệm:

Mẫu dữ liệu test gồm 66 ảnh trong nhiều điều kiện khác nhau như: độ sáng, nhiễu, ảnh chụp góc nghiêng,…

|  |  |  |
| --- | --- | --- |
| Công việc | Số lượng mẫu bị thất bại | Độ chính xác (%) |
| Phát hiện biển số | 9 | 86.36 |
| Nhận dạng kí tư | 45 | 30 |

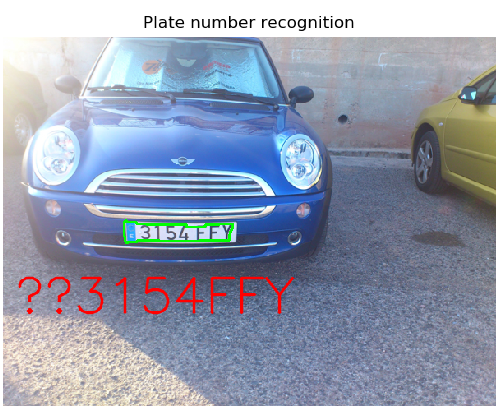
**Một số ảnh phát hiện và nhận dạng kí tự đúng:**







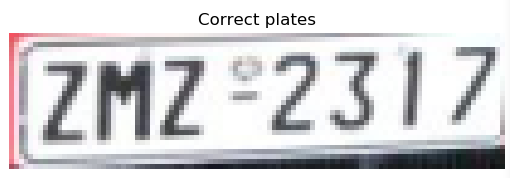


**Một số ảnh phát hiện sai hoặc nhận dạng sai (hoặc không ra kết quả: None)**









**Nhận xét:**

* Kết quả phát hiện biển số vẫn khá thi nhưng chưa thực sự chính xác vì có trường hợp đã phát hiện ra vùng ảnh là biển số nhưng khi qua bước kiểm thực lần nữa thì đã bị loại. Điều này nhóm sẽ cải tiến bằng mô hình SVM để phân lớp và xác định biển số.
* Đa số ảnh bị nhận dạng kí tự sai hoặc không thể nhận dạng kí tự được.
* Bên cạnh đó, nhóm sẽ cố gắng thay thế phương pháp truyền thống bằng việc dùng mô hình mạng deep learning YOLO để giúp phát hiện biển số tốt hơn và chạy với thời gian thực.
* Hiện tại, công việc nhận dạng kí tự được thực hiện bởi thư viên Tesseract của google. Nhóm không hiểu ví sao kết quả nhận dạng lại quá thấp. Điều này trong đợt cải tiến sắp tới nhóm sẽ cố gắng cải thiện.

# **D Cải tiến và bổ sung:**

Ở đây, chúng tôi có hai phiên bản:

1. **Phiên bản 1:** Bản cải tiến và bổ sung của đợt nộp lần đầu

A close up of text on a white background

Description automatically generated

Những phần bổ sung và cải tiến được tô màu đỏ nhạt.

**Chi tiết hơn như sau:**

**FloodFill:** với mục đích là giúp cắt ảnh và trích xuất ra vùng ảnh biển số chính xác hơn. Nó giúp loại bỏ các phần thừa không phải biển số trong vùng ảnh đó.

* Ý tưởng: ta sẽ gieo một số hạt giống (seed) ở xung quanh tâm của rotated rectangle (bounding box có tính góc nghiêng, được tạo bởi contours) và sẽ floodFill để fill các vùng đã đặt hạt giống và loang ra. Tân dụng background của biển số là màu trắng, do đó ta sẽ tìm được bounding box sẽ ôm sát biển số hơn.
* Thêm nữa, ta còn thực hiện kiểm định sơ bộ vùng ảnh đó có phải biển số (dựa vào tỉ lệ khung hình) và thực hiện phép biển đổi AFFINE như xoay vùng ảnh đó lại cho nó song song với trục hoành. Mục đích: giúp cân chỉnh ảnh cho việc phân lớp đối tương và nhận dạng kí tự.
* Cuối cùng, ta thực hiện xén ảnh. Tuy nhiên, mỗi ảnh biển số thì có kích thước khác nhau, điều kiện sáng khác nhau. Do đó, ta cần làm thêm bước resize kích thước ảnh về (Width = 144, Height = 33) và cân bằng sáng cho ảnh.

**Bộ phân lớp SVM:** dùng để phân lớp xem vùng ảnh có là biển số hay không (0: không phải, 1: biển số).

* Ở đây, nhóm sử dụng pretrained model của tác giá cuốn sách Mastering OpenCV with Practical Computer Vision Projects (6th ed., pp. 161-188). Tác giả đã huấn luyện model này trên tập dataset gồm 75 ảnh dương và 35 ảnh âm với kích thước 144x33 pixels.
* Dữ liệu tuy không nhiều nhưng vẫn đủ cho việc phân lớp và họ sử dụng pixel trong ảnh để làm đặc trưng huấn luyện.

**Character extraction**: bước này đóng vai trò khá quan trọng trong khâu nhận dạng kí tự. Để tăng độ chính xác của nhận dạng kí tự, ta cần phải thực hiện tách các kí tự trong ảnh ra thành từng ảnh kí tự riêng lẻ và đem vào Tesseract để nhận dạng.

Cách thức: Tìm các contour và chọn lọc lại các contour thích hợp dựa vào diện tích. Sau đó, đóng bounding box cho các contour ấy rồi trích xuất vùng ảnh tương ứng.

**Character image processing**: Thêm vào đó, ta cần phải làm thêm các bước xử lý như: threshold ảnh, dùng phép biến đổi hình thái như Opening (Erosion followed Dilation) để giảm nhiễu và tăng độ chính xác cho nhận dạng kí tự.

**Tìm hiểu về Tesseract**

1) Giới thiệu:

Tesseract là một OCR engine rất thông dụng, được phát triển ban đầu bởi tập đoàn Hewlett - Packard vào thập niên 80 và trở thành open source vào năm 2005. Google tiếp tục phát triển dự án này vào 2006.

Tesseract v4 có hỗ trợ nhận diện ký tự bằng deep learning.

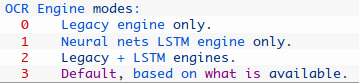
Nền tảng bên dưới của Tesseract có sử dụng Long Short-Term Memory (LSTM) network, là một loại Recurrent Neural Network (RNN).

Tesseract có hỗ trợ nhận diện hơn 100 ngôn ngữ.

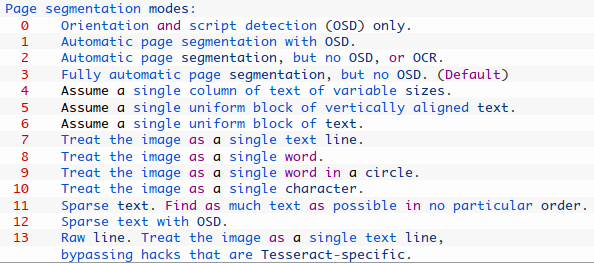
Để có kết quả tốt, cần preprocess ảnh input trước khi đưa vào tesseract.

2) Các flag cần quan tâm khi sử dụng thư viện Tesseract:

* -l: ngôn ngữ sử dụng.
* --oem: OCR engine mode, xác định loại thuật toán sử dụng. Có 4 loại:



* --psm: Page Segmentation Mode, xác định loại phân đoạn tran. Có 14 loại:

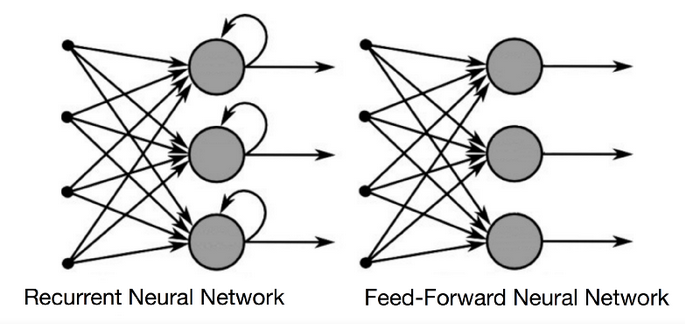


3) Recurrent Neural Network (RNN):

RNN là một loại neural network rất “mạnh”.

RNN thích hợp cho dữ liệu tuần tự, nối tiếp nhau (sequential data) như một câu văn, một lời nói, ... Nó được gọi như vậy là do nhớ được các input và tận dụng các input này để predict trong các lần tiếp theo.

RNN được phân biệt với feedforward network là do nó có feedback loop liên kết trở lại các node. Các thông tin khi predict sẽ được lưu lại. Có thể nghĩ rằng RNN có memory.



Do có memory, RNN có khả năng nhớ những thứ quan trọng về input nó nhận được, giúp nó dự đoán chính xác trong những lần tiếp theo.

4) Long Short-Term Memory (LSTM) Network:

Tuy RNN có thể nhớ được nhưng trí nhớ của nó là “ngắn hạn”.

LSTM là một biến thể của RNN, nó giúp “mở rộng” memory. Như tên gọi của nó (tạm dịch: nhiều trí nhớ ngắn hạn), LTSM lưu trữ các thông tin trong lần predict trước và theo một cơ chế để sử dụng các thông tin này trong các lần predict sau. LSTM có thể đọc, ghi và xóa thông tin trong memory.

Mạng này bao gồm các “cells” có nhiệm vụ trong việc quyết định nên “nhớ” những thông tin gì, và thông tin nào là cần thiết được sử dụng trong mỗi lần dự đoán. Các việc quyết định này được thực hiện thông qua các “gates”. Các “gates” này có thể là hàm sigmoid, tanh, hay là elemenwise. multiplication,...

Các kết quả thực nghiệm:

1. **Phiên bản 2:** Bản nhận dạng biển số cho xe moto ở Việt Nam bằng mạng CNN, cụ thể hơn là dùng bộ phân lớp dựa vào dặc trưng Haar.

**Qui trình xử lý của hệ thống nhận dạng biển số cho xe máy:**

**A close up of a device

Description automatically generated**

1. **Giới thiệu về Haar cascade:**

# **E Hướng dẫn sử dụng:**

Chương trình được chạy bằng command line.

Cú pháp:

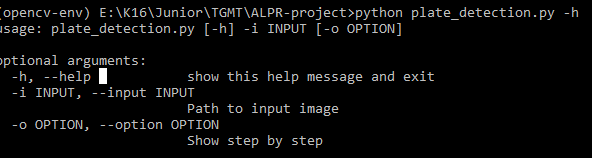
python <Ten chuong trinh.py> -i <Input image> -o <Option: 0 hoặc 1>

Ý nghĩa:

* <Ten chuong trinh.py>: file chạy chương trình
* <Input image>: đường dẫn đến ảnh input
* <Option>: Lựa chọn hiển thị tất cả các bước làm hay không. Nếu có thì nhập 1, không thì nhập 0. Mặc định chương trình sẽ để là 0 khi bạn không nhập giá trị này.

Lưu ý: Chương trình dùng argparse để giúp command line trở nên trực quan và dễ sử dụng hơn khi truyền tham sao. Bạn có thể nhập lệnh help để trở giúp:

python plate\_detection.py -h



Ví dụ: Để chạy chương trình ALPR với tập ảnh “.\test\_images\IMG\_0378.jpg”:

python 1612174\_1612269\_1612274\_Lab03.py -i “.\test\_images\IMG\_0378.jpg” – o 0

# **F Tham khảo:**

|  |  |
| --- | --- |
| [1] | Baggio, D. L. (2012). 5. Number Plate Recognition Using SVM and Neural Networks.  In Mastering OpenCV with Practical Computer Vision Projects (6th ed., pp. 161-188). Birmingham, UK: Packt Publishing |
| [2] | <https://github.com/kagan94/Automatic-Plate-Number-Recognition-APNR.git> |
| [3] | <https://github.com/MicrocontrollersAndMore/OpenCV_3_License_Plate_Recognition_Python.git> |
| [4] | <https://docs.opencv.org/3.4.3/d7/d4d/tutorial_py_thresholding.html> |
| [5] | <https://docs.opencv.org/3.1.0/d4/d73/tutorial_py_contours_begin.html> |
| [6] | <https://docs.opencv.org/trunk/d9/d61/tutorial_py_morphological_ops.html> |
| [7] | <https://docs.opencv.org/3.1.0/dd/d49/tutorial_py_contour_features.html> |
| [8] | <https://www.pyimagesearch.com/2017/07/10/using-tesseract-ocr-python/> |
| [9] | <https://cvisiondemy.com/license-plate-detection-with-opencv-and-python/> |
| [10] | Dataset: <http://www.medialab.ntua.gr/research/LPRdatabase.html?fbclid=IwAR0d_5jeUecAGPX2Dw23p5GFpArKSRAsLSZSD-jOU_RGhGbMOU2JydKsRq8> |