BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI:

NHẬN DẠNG BIỂN SỐ XE VÀ XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÃI GIỮ XE Ô TÔ

Giảng viên hướng dẫn: TS. Nguyễn Quốc Tuấn

KS. Trần Thị Dung

Sinh viên thực hiện : Lâm Chương

Mã số sinh viên : 565107106

Lớp : CQ.56.CNPM

Khoá : 56

TP. Hồ Chí Minh, năm 2019

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

BÁO CÁO ĐỒ ÁN TỐT NGHIỆP

ĐỀ TÀI:

NHẬN DẠNG BIỂN SỐ XE VÀ XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÃI GIỮ XE Ô TÔ

Giảng viên hướng dẫn: TS. Nguyễn Quốc Tuấn

KS. Trần Thị Dung

Sinh viên thực hiện: Lâm Chương

Lớp: CQ.56.CNPM

Khoá: 56

TP. Hồ Chí Minh, năm 2019

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
PHÂN HIỆU TẠI THÀNH PHỐ HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập – Tự do – Hạnh phúc

NHIỆM VỤ THIẾT KẾ TỐT NGHIỆP

BỘ MÔN: **CÔNG NGHỆ THÔNG TIN**

-------***-------

Mã sinh viên: 5651071006 **Họ tên SV:** Lâm Chương

Khóa: K56 **Lớp:** CQ.56.CNPM

1. Tên đề tài

NHẬN DIỆN BIỂN SỐ XE VÀ XÂY DỰNG ỨNG DỤNG QUẢN LÝ BÃI GIỮ XE Ô TÔ.

2. Mục đích, yêu cầu

a. Mục đích:

- Tìm hiểu về thuật toán Canny, các thư viện OpenCV, EmguCV, Aforge.net và ứng dụng tạo một chương trình phát hiện và nhận dạng biển số xe cũng như quản lý cho bãi gửi xe.
- Nhận diện biển số xe và xây dựng "Ứng dụng quản lý bãi giữ xe ô tô", nhằm tối ưu diện tích, nhân lực; khắc phục tình trạng thiếu bãi giữ xe khi nhu cầu về phương tiện đi lại tăng cao; hạn chế trộm cắp cho các bãi giữ xe.

b. Yêu cầu:

- **Yêu cầu chức năng**
  - Hệ thống cho phép: Ghi hình ảnh và nhận dạng biển số xe, tính thời gian xe trong bãi và chi phí sử dụng bãi đỗ xe.
  - Kiểm tra số xe và số ô đậu trong bãi.
- **Yêu cầu phi chức năng**
  - Tốc độ: tốc độ tối ưu, sử dụng băng thông hiệu quả, có thể làm việc tốt khi có nhiều dữ liệu, tốc độ khi tìm kiếm, tốc độ khi hiển thị.
  - Giao diện: Thân thiện với người dùng và dễ dàng thao tác.

3. Nội dung và phạm vi đề tài

a. Nội dung:

Tìm hiểu về bộ thư viện OpenCV. Tìm hiểu về các thuật toán xử lý ảnh và nhận dạng ký tự từ đó áp dụng một thuật toán để xây dựng thành một ứng dụng thực tế. Thông qua sự hỗ trợ của thư viện OpenCV, sẽ xây dựng ứng dụng đọc và lưu biển số xe dành cho bãi đỗ xe ôtô, bằng cách chụp hình ảnh có chứa biển số xe thông qua camera. Từ hình ảnh, sử dụng thuật toán xử lý ảnh để đọc ra các chữ số trên hình. Sau khi lấy được các chữ số sẽ thực hiện lưu trữ và tính toán thời gian gửi khi xe đi ra từ đó có thể tính được số tiền dựa vào thời gian gửi.

b. Phạm vi:

Nghiên cứu tổng quan một số thuật toán xử lý hình ảnh, nghiên cứu thuật toán Canny và thư viện Tesseract trong nhận dạng.

4. Công nghệ, công cụ và ngôn ngữ lập trình

- Công nghệ sử dụng: công nghệ nhận dạng biển số xe
- Công cụ: Visual Studio 2017 (sử dụng thư viện OpenCV và EmguCV).
- Ngôn ngữ lập trình: SQL Server, C#.

5. Các kết quả chính dự kiến sẽ đạt được và ứng dụng

- Nắm bắt được thuật toán Canny và thư viện Tesseract trong nhận dạng biển số.
- Tìm hiểu sơ lược thư viện OpenCV về xử lý hình ảnh và hoàn thành ứng dụng đọc ghi biển số xe cho bãi gửi xe bằng ngôn ngữ C# với sự hỗ trợ của hệ quản trị cơ sở dữ liệu SQL Server.

6. Giáo viên và cán bộ hướng dẫn

Họ tên: TS. Nguyễn Quốc Tuấn

Họ tên: GV. Trần Thị Dung

Đơn vị công tác: Trường Đại học Giao thông Vận tải Phân hiệu tại TP. Hồ Chí Minh

Điện thoại: Email: dungtt.utc2@gmail.com

Ngày ..... tháng ..... năm 2019	**Đã giao nhiệm vụ TKTN**
Trưởng BM Công nghệ Thông tin	**Giảng viên hướng dẫn**

Đã nhận nhiệm vụ TKTN

Sinh viên: Lâm Chương Ký tên:

Điện thoại: 0327802383 Email: lamchuong28@gmail.com

LỜI CÁM ƠN

Lời đầu tiên em xin gửi lời cảm ơn chân thành và sâu sắc nhất tới toàn thể quý thầy cô Trường Đại học Giao thông Vận tải Phân hiệu tại thành phố Hồ Chí Minh, hơn nữa là lời tri ân sâu sắc tới thầy cô trong bộ môn Công nghệ thông tin đã cho em những kiến thức vô cùng quý giá để em có thể hoàn thành đồ án này. Đặc biệt em xin gửi lời cảm ơn tới cô Trần Thị Dung – Giảng viên Trường Đại học Giao thông Vận Tải Phân hiệu tại thành phố Hồ Chí Minh và thầy Nguyễn Quốc Tuấn – Giảng viên Trường Đại học Giao thông Vận tải đã hỗ trợ, hướng dẫn em hoàn thành đồ án một cách tốt nhất.

Để đạt được nhưng kết quả như ngày hôm nay là cả một quá trình phấn đấu, cố gắng của bản thân, và sự giúp đỡ chỉ bảo tận tình của các quý thầy cô, sự khích lệ động viên của gia đình và bạn bè. Đó là may mắn của em.

Với điều kiện thời gian và kiến thức của bản thân còn hạn chế nên đồ án thực hiện còn nhiều thiếu sót. Em rất mong nhận được sự góp ý chân thành để em có thể phát triển hơn trong tương lai

Em xin chân thành cảm ơn!

TP. Hồ Chí Minh, ngày…tháng…năm…

Sinh viên thực hiện

Lâm Chương

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Tp. Hồ Chí Minh, ngày …… tháng …… năm ……

Giáo viên hướng dẫn

MỤC LỤC

DANH MỤC CHỮ VIẾT TẮT

STT	Mô tả	Ý nghĩa	Ghi chú
1	LOG	Laplacian of Gaussian	
2	OCR	Optical Character Recognition	
3	JFC	John F. Canny	
4	API	Application Programming Interface	
5	IDE	Integrated Development Environment	
6	RDBMS	Relational Database Management System	
7	IT	Information Technology	
8	OpenCV	Open Source Computer Vision Library	
9	NMS	Non-maximum suppression	

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU, SƠ ĐỒ

LỜI MỞ ĐẦU

Với sự phát triển của công nghệ ngày nay. Xử lý ảnh là một trong những chuyên ngành quan trọng và lâu đời của Công nghệ thông tin. Xử lý ảnh được áp dụng trong nhiều lĩnh khác nhau như y học, vật lý, hoá học, tìm kiếm tội phạm, tquân sự và một số lĩnh vực khác của đời sống xã hội.

Phần lớn, con người thu nhận thông tin bằng thị giác, cụ thể đó là các hình ảnh. Vì vậy xử lý ảnh là vấn đề không thể thiếu và hết sức quan trọng để thu được hình ảnh tốt hơn, đẹp hơn, nhằm đáp ứng các yêu cầu thông tin khác nhau của người nhận.

Với những tiến bộ trong bộ cảm biến hình ảnh, truyền hình kỹ thuật số, cơ sở dữ liệu hình ảnh, video, đa phương tiện và hệ thống. Cũng như với sự gia tăng của máy in màu, hiển thị hình ảnh màu sắc, các thiết bị DVD và đặc biệt là máy ảnh kỹ thuật số. Xử lý ảnh màu ngày cảng trở nên phổ biến và dễ dàng hơn.

Theo đánh giá trong bài báo [1], bản thân em nhận thấy *Phương pháp Candy* phù hợp với đề tài mà em đang thực hiện. Chính vì vậy, em tiến hành tìm hiểu về phương pháp Candy và thư viện OpenCV, xây dựng hệ thống phát hiện – nhận dạng – quản lý một bãi giữ xe ô tô. Nội dung của đồ án được trình bày bao gồm 5 chương với cấu trúc như sau:

Chương 1: **Tổng quan về đề tài.**

Chương 2: **Cơ sở lý thuyết.**

Chương 3: **Bài toán nhận dạng biển số xe.**

Chương 4: **Phân tích thiết kế ứng dụng.**

Chương 5: **Cài đặt và thực nghiệm.**

Vì lý do thời gian thực hiện và kiến thức có phần hạn hẹp nên mặc dù có sự hỗ trợ tận tình của các thầy cô, nhưng không tránh khỏi những sai sót. Vì lẽ đó nên những lời góp ý, đánh giá của quý thầy cô sẽ là mục tiêu và kế hoạch để em có thể cải thiện chất lượng của đề tài.

CHƯƠNG 1: TỔNG QUAN VỀ ĐỀ TÀI

1.1. Đặt vấn đề

Hiện nay, với sự gia tăng đáng kể của các phương tiện giao thông, nhu cầu về bãi đậu là hết sức cần thiết. Những yêu cầu về chất lượng bãi đậu cũng rất được quan tâm. Cùng với sự phát triển của khoa học công nghệ. Đạt được nhiều thành tựu to lớn trong tất cả các lĩnh vực và ngành nghề của đời sống xã hội. Vì vậy, việc thiết lập một bãi giữ xe thông minh là rất phù hợp, vừa đáp ứng nhu cầu vừa áp dụng công nghệ hiện đại. Ở Việt Nam, hệ thống nhận dạng biển số xe đã được nghiên cứu và đã đưa vào sử dụng ngày càng phổ biến. Vì vậy việc nghiên cứu và áp dụng các thuật toán vào việc xây dựng một hệ thống nhận dạng biển số xe được chính xác hơn, không phụ thuộc vào mắt thường hay chỉ hình ảnh đối chiếu đơn thuần. Sử dụng trí tuệ nhân tạo đưa các học thuật và nhận dạng một cách hiệu quả hơn, thuật toán nào tốt ưu hơn, có độ chính xác cao hơn sẽ được áp dụng rộng rãi. Đứng trước nhu cầu thực tế đó, không ít nhóm nghiên cứu, doanh nghiệp đã nghiên cứu, tìm hiểu và xây dựng thành công phần mềm cho các bãi đổ xe tự động, chung cư, các công ty…

Vì vậy, việc nghiên cứu thuật toán Canny để trình bày về xử lý ảnh, nhận dạng biển số xe, các vấn đề liên quan tới kiến thức, nền tảng xây dựng, sử dụng các ứng dụng hay ngôn ngữ cũng là điều đáng quan tâm hiện nay.

1.2. Mục tiêu đề tài

Nghiên cứu ứng dụng thuật toán Canny trong nhận dạng hình ảnh và xây dựng chương trình Demo cho thuật toán. Mục đích nghiên cứu là để viết các ứng dụng sử dụng cho việc nhận dạng hình ảnh sau đó xây dựng ứng dụng "Nhận dạng biển số xe cho bãi đỗ xe ô tô" nhằm tạo điều kiện thuận lợi cho người trông xe ở bãi xe, giúp đẩy nhanh tốc độ và tính chính xác về thời gian cho người trông xe.

Yêu cầu đối với bài toán này là phải nắm rõ thuật toán dùng để nhận dạng hình ảnh được sử dụng trong bài, có nghiên cứu thêm một số thuật toán nhận dạng khác sau đó đưa ra so sánh và nhận định. Từ đó có thể áp dụng thuật toán nào là tốt nhất để giải quyết bài toán của mình. Ngoài ra, ta cần phải nắm rõ một số kỹ thuật trong xử lý hình

ảnh để giải quyết một số vấn đề khó khăn khi gặp phải trong bài toán. Khi xây dựng chương trình, yêu cầu sự chính xác cao, tốc độ của thuật toán phải nhanh.

1.3. Phạm vi nghiên cứu

Nghiên cứu tổng quan một số thuật toán xử lý hình ảnh và nhận dạng số, ký tự.

Về lý thuyết: xác định thuật toán, phương pháp phù hợp để phát hiện và nhận dạng biển số xe cũng như các chức năng có liên quan.

Về ứng dụng: chương trình phần mềm phát hiện và nhận dạng biển số xe; so sánh biển số trước khi xe vào và sau khi xe ra; tính toán được thời gian xe vào và ra, chi phí.

Các điều kiện thực hiện: hình chụp trực diện từ phía sau đuôi xe, cường độ ánh sáng là 300÷500 lux (độ sáng tiêu chuẩn bình thường) hoặc ánh sáng ngoài trời ban ngày bình thường, tình trạng biển số tốt và không bị chói sáng. Đối tượng là các biển số xe hình chữ nhật (xe ô tô), biển số xe tốt (không bị biến dạng, chói sáng).

1.4. Một số module chức năng

- Phát hiện xe vào, xe ra và khoang vùng phát hiện là biển số xe đó.

- Nhận dạng biển số xe, trích xuất ký tự và lưu lại thông tin về xe.

- So sánh thông tin xe vào và thông tin xe ra (sau khi lấy được thông tin xe ra

- Tính thời gian và chi phí chủ xe phải trả..

- Tăng số ô đậu khi xe vào cũng như giảm số ô đậu khi xe ra để xuất thông báo khi nào bãi đầy.

- Thống kế tổng lượt xe trong ngày cũng như tổng chi phí thu được

CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

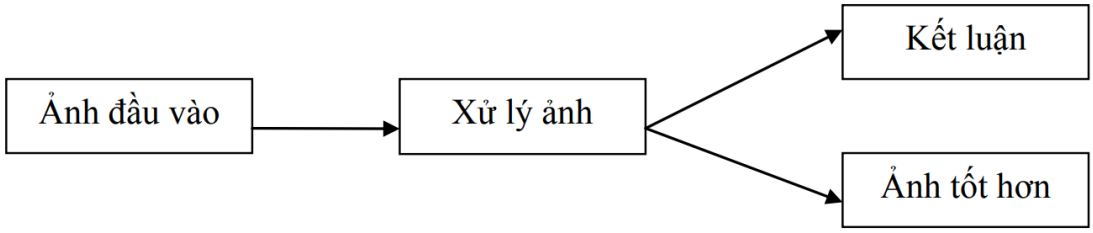
2.1. Tổng quan về xử lý ảnh và biên

2.1.1. Tổng quan về xử lý ảnh

2.1.1.1. Xử lý ảnh là gì?

Con người thu nhận thông tin qua các giác quan trong đó thị giác đóng vai trò quan trọng nhất. Cùng với sự phát triển nhanh của phần cứng máy tính, xử lý ảnh và đồ hoạ đã phát triển mạnh mẽ và ngày càng có nhiều ứng dụng trong cuộc sống. Nhờ thế mà xử lý ảnh từng bước đóng một vai trò quan trọng trong tương tác người máy

Quá trình xử lý nhận dạng ảnh là một quá trình thao tác nhằm biến đổi một ảnh đầu vào để cho ra một kết quả mong muốn. Xử lý ảnh thông thường gồm 3 bước:



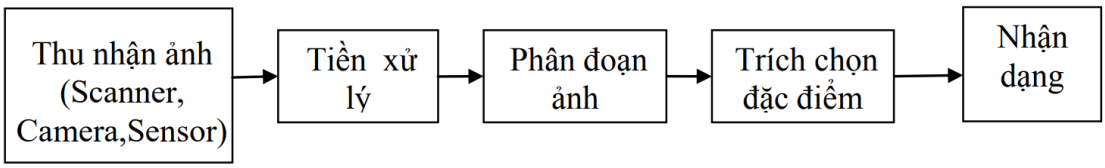
Hình 2.1. *Quá trình xử lý ảnh*

- Bước 1: Nhập một hình ảnh với một máy quét quang học hoặc trực tiếp thông qua nghệ thuật chụp ảnh số.

- Bước 2: Thao tác hoặc phân tích các hình ảnh bằng một cách nào đó. Giai đoạn này có thể bao gồm kỹ thuật nâng cao chất lượng hình ảnh và nén dữ liệu, hoặc hình ảnh có thể được phân tích để tìm ra các hình dáng mà mắt người không thể thấy được. Ví dụ: các nhà khí tượng học sử dụng xử lý ảnh để phân tích các ảnh vệ tinh.

- Bước 3: Kết quả đầu ra - hình ảnh có thể bị thay đổi bằng cách này hay cách khác, nó có thể là một ảnh "tốt hơn" (Ví dụ: ảnh mờ được xử lý để nhìn rõ hơn) hoặc một kết luận (Ví dụ: phân tích ảnh để trích chọn các đặc trưng vân tay hay ảnh một tai nạn giao thông phác họa hiện trường tại nạn).

2.1.1.2. Các quá trình cơ bản trong xử lý ảnh



Hình 2.2. *Các bước cơ bản trong quá trình xử lý*

Một ảnh đầu vào cần thông qua rất nhiều bước khác nhau để có được một ảnh đầu ra mong muốn. Các quá trình cơ bản của một hệ thống xử lý ảnh được thể hiện thông qua sơ đồ dưới đây:

a. Thu nhận ảnh:

Để thực hiện được quá trình đầu tiên trong hệ thống xử lý ảnh, ta cần sử dụng các thiết bị thu nhận ảnh để chuyển các thông tin dưới dạng hình ảnh thành các cấu trúc lưu trữ được trong máy tính và được hiển thị ra màn hình, máy in,... Ảnh có thể thu nhận từ vệ tinh qua các bộ cảm ứng, qua camera, máy chụp ảnh đơn sắc (màu), hay các tranh, ảnh được quét trên máy quét ảnh. Nếu ảnh thu nhận được chưa phải là dạng số hóa ta phải chuyển đổi hay số hóa ảnh trước khi chuyển sang giai đoạn tiếp theo.

b. Tiền xử lý:

Sau khi được thu nhận bởi các thiết bị thu nhận ảnh, ảnh sẽ được cải thiện về độ tương phản, nhiễu,… bởi các kỹ thuật xử lý ảnh để làm ảnh tốt hơn theo mục đích sử dụng nhằm phục vụ cho quá trình xử lý tiếp theo.

Một số tiến trình trong quá trình tiền xử lý là:

- *Điều chỉnh độ chiếu sáng:* khắc phục hậu quả của sự chiếu sáng không đồng đều.

- *Khử nhiễu:* Nhiễu được chia làm 2 loại cơ bản là nhiễu hệ thống và nhiễu ngẫu nhiên. Trong đó, nhiễu hệ thống là nhiễu có quy luật có thể khử bằng các phép biến đổi Fourier và loại bỏ các đỉnh điểm. Đối với nhiễu ngẫu nhiên - vết bẩn không rõ nguyên nhân thì được khắc phục bằng các phép lọc (lọc trung bình, lọc trung vị,..).

- *Hiệu chỉnh mức xám:* có thể tăng hay giảm số mức xám nhằm khắc phục tính không đồng bộ gây nên từ hiệu ứng của thiết bị thu nhận hình ảnh hoặc độ tương phản giữa các vùng ảnh.

- *Chuẩn hóa* độ lớn, hình dạng và màu sắc.

- *Nắn chỉnh hình học:* ảnh thu nhận thường bị biến dạng do các thiết bị quang học và điện từ, để khắc phục điều này người ta sử dụng các phép chiếu đươc xây dựng trên tập các điểm điều khiển.

c. Phân đoạn ảnh:

Phân đoạn ảnh là một quá trình thao tác ở mức thấp trong toàn bộ hệ thống xử lý ảnh. Quá trình này thực hiện việc phân vùng ảnh thành các vùng rời rạc và đồng nhất với nhau hay nói cách khác là xác định các biên của các vùng ảnh đó. Phân đoạn ảnh là chia ảnh thành các vùng không trùng lắp, mỗi vùng gồm 1 nhóm pixel liên thông và đồng nhất theo 1 tiêu chí nào đó. Ví dụ: đồng nhất về màu sắc, mức xám, kết cấu, độ sâu của các layer,… Sau khi phân đoạn mỗi pixel chỉ thuộc về một dùng duy nhất.

Để đánh giá chất lượng của quá trình phân đoạn là rất khó, vì vậy cần phải xác định rõ mục tiêu của quá trình phân đoạn là gì? Ví dụ: để nhận dạng chữ (mã vạch) trên phong bì thư cho mục đích phân loại bưu phẩm, cần chia các câu chữ về địa chỉ hoặc tên người gửi thành các từ, các chữ, các số (các vạch) riêng biệt để nhận dạng.

Kết quả của quá trình phân đoạn ảnh thường được cho dưới dạng dữ liệu điểm ảnh thô, trong đó hàm chứa biên của một vùng ảnh hoặc tập hợp tất cả các điểm ảnh thuộc về chính vùng ảnh đó. Trong cả hai trường hợp, sự chuyển đổi dữ liệu thô này thành một dạng thích hợp hơn cho việc xử lý trong máy tính là hết sức cần thiết, nghĩa là nên biểu diễn một vùng ảnh dưới dạng biên hay dưới dạng một vùng hoàn chỉnh gồm tất cả những điểm ảnh thuộc về nó.

- Biểu diễn dạng biên cho một vùng phù hợp với những ứng dụng chỉ quan tâm đến các đặc trưng hình dạng bên ngoài của đối tượng, ví dụ như các góc cạnh và điểm uốn trên biên

- Biểu diễn dạng vùng lại thích hợp cho những ứng dụng khai thác các tính chất bên trong của đối tượng. Ví dụ như vân ảnh hoặc cấu trúc xương của nó. Và trong một số ứng dụng thì cả hai cách biểu diễn trên đều cần thiết.

d. Trích chọn đặc điểm:

Dựa trên các thông tin thu nhận được qua quá trình phân đoạn, kết hợp với các kỹ thuật xử lý, thủ tục phân tích dữ liệu để đưa ra các đặc điểm đặc trưng, đối tượng ảnh cũng như các thông tin cần thiết trong quá trình xử lý. Nhờ đó việc nhận dạng các đối tượng ảnh chính xác hơn, tốc độ tính toán cao và dung lượng nhớ lưu trữ giảm xuống.

Các đặc điểm của đối tượng được trích chọn tùy theo mục đích nhận dạng trong hệ thống xử lý ảnh. Sau đây là một vài đặc điểm của ảnh:

- Đặc điểm không gian: Phân bố mức xám, phân bố xác suất, biên độ, điểm uốn v.v…

- Đặc điểm biến đổi: Các đặc điểm loại này được trích chọn bằng việc thực hiện lọc vùng (zonal filtering). Các bộ vùng được gọi là "mặt nạ đặc điểm" (feature mask) thường là các khe hẹp với hình dạng khác nhau (chữ nhật, tam giác, cung tròn v.v..)

- Đặc điểm biên và đường biên: Đặc trưng cho đường biên của đối tượng và do vậy rất hữu ích trong việc trích trọn các thuộc tính bất biến được dùng khi nhận dạng đối tượng. Các đặc điểm này có thể được trích chọn nhờ toán tử gradient, toán tử la bàn, toán tử Laplace, toán tử "chéo không" (zero crossing) v.v..

e. Nhận dạng:

Nhận dạng ảnh là quá trình cuối cùng của hệ thống xử lý ảnh - quá trình liên quan đến các mô tả đối tượng mà người ta muốn đặc tả nó. Quá trình này thường đi sau quá trính trích chọn đặc điểm trong hệ thống xử lý ảnh.

Có 2 kiểu nhận dạng ảnh cơ bản:

- Nhận dạng theo tham số (mô tả tham số).

- Nhận dạng theo cấu trúc (mô tả theo cấu trúc).

Hiện nay, người ta đã áp dụng kỹ thuật nhận dạng khá thành công với nhiều đối tượng khác nhau như: nhận dạng ảnh vân tay, khuôn mặt, nhận dạng chữ (chữ cái, chữ số, chữ có dấu), nhận dạng chữ in (đánh máy) phục vụ cho việc tự động hóa quá trình đọc tài liệu, tăng tốc độ và chất lượng nhận thông tin từ máy tính. Ngoài ra kỹ thuật nhận dạng dựa vào kỹ thuật mạng nơ ron đang được áp dụng và cho kết quả khả quan.

2.1.2. Tổng quan về biên

2.1.2.1. Biên và các kiểu biên cơ bản

a. Một số khái niệm về biên

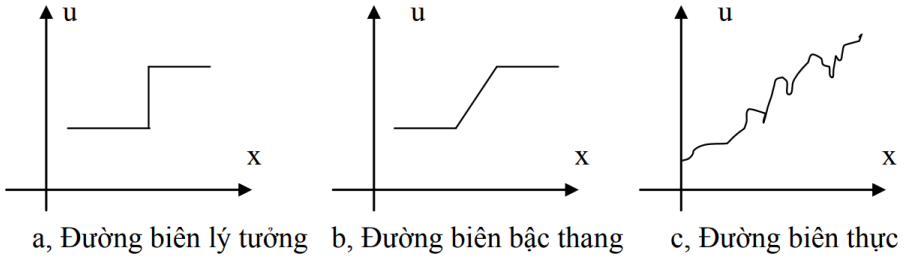
Cho tới nay chưa có định nghĩa chính xác về biên và mỗi định nghĩa được sử dụng trong một số trường hợp nhất định. Biên có thể được tạo ra bởi bóng tối, kết cấu hình học... Biên cũng có thể được định nghĩa là không liên tục ở cường độ hình ảnh do sự thay đổi trong cấu trúc hình ảnh. Biên trong một hình ảnh thường xảy ra với độ phân giải hoặc quy mô khác nhau và đại diện cho quá trình chuyển đổi của mức xám khác nhau, hay mức độ gradient. Tuy nhiên, nhìn chung biên có thể được định nghĩa như sau:

Điểm biên: một điểm ảnh được coi là điểm biên nếu có sự thay đổi đột ngột về mức xám. Ví dụ: đối với ảnh đen trắng, một điểm được gọi là điểm biên nếu nó là điểm đen có ít nhất một điểm trắng bên cạnh.

Đường biên (đường bao của ảnh - boundary) của đối tượng: được tạo thành bởi một tập các điểm biên liên tiếp.

Mỗi một biên là một thuộc tính gắn liền với một điểm riêng biệt và được tính toán từ những điểm lân cận nó. Đó là một biến Vector bao gồm 2 thành phần:

- Độ lớn của Gradient.
- Hướng của biên với góc φ, lệch so với hướng của Gradient ψ một góc -90^0



Hình 2.3. *Đường bao của ảnh*

Mô hình biểu diễn đường biên: theo toán học, điểm ảnh có sự biến đổi mức xám u(x) một cách đột ngột theo hình dưới:

b. Các biên cơ bản

- **Biên lý tưởng**

Việc phát hiện biên một cách lý tưởng là việc xác định được tất cả các đường bao trong đối tượng. Biên là sự thay đổi đột ngột về mức xám nên sự thay đổi này càng lớn thì càng dễ dàng nhận ra biên.

Một biên được coi là biên lý tưởng khi có sự thay đổi cấp xám lớn giữa các vùng trong ảnh. Biên này thường chỉ xuất hiện khi có sự thay đổi cấp xám qua một điểm ảnh.

- **Biên bậc thang**

Biên dốc xuất hiện khi sự thay đổi cấp xám trải rộng qua nhiều điểm ảnh. Vị trí của biên được xem như vị trí chính giữa của đường dốc nối giữa cấp xám thấp và cấp xám cao. Tuy nhiên đây chỉ là đường dốc trong toán học, từ khi ảnh được kỹ thuật số hóa thì đường dốc không còn là đường thẳng mà thành những đường lởm chởm, không trơn.

- **Biên thực**

Trên thực tế, ảnh thường có biên không lý tưởng, có thể do các nguyên nhân sau:

- Hình dạng không sắc nét.

- Nhiễu: kết quả của nhiễu trên ảnh gây ra một sự biến thiên ngẫu nhiên giữa các điểm ảnh. Sự xuất hiện ngẫu nhiên của các điểm ảnh có mức xám chênh lệch cao làm cho các đường biên dốc trở lên không trơn chu mà trở thành các đường biên gồ ghề, mấp mô, không nhẵn, đây chính là đường biên trên thực tế.

2.1.2.2. Vai trò của biên trong nhận dạng:

Đường biên là một loại đặc trưng cục bộ tiêu biểu trong phân tích nhận dạng ảnh. Người ta sử dụng đường biên làm phân cách các vùng xám (màu) cách biệt. Ngược lại, người ta cũng dùng các vùng ảnh để tìm đường phân cách.

Như đã đề cập tới ở phần tổng quan về một hệ thống nhận dạng và xử lý ảnh, quá trình nhận dạng có hai giai đoạn cần thực hiện:

- Giai đoạn học: Các đặc điểm của đối tượng mẫu được lưu trữ (gọi là học mẫu) và tập các phần tử mẫu được chia thành các lớp.

- Giai đoạn nhận dạng: Khi có đối tượng cần nhận dạng, các đặc điểm của đối tượng sẽ được trích chọn và sử dụng hàm quyết định để xác định đối tượng cần nhận dạng thuộc lớp nào.

2.1.3. Các phương pháp phát hiện biên[1]

Các phương pháp phát hiện biên truyền thống thường dựa trên kết quả của phép tích chập (convolution) giữa bức ảnh cần nghiên cứu f (x, y) và một bộ lọc 2D (filter) h(x, y) thường được gọi là mặt nạ (mask).

$$h(x,y) * f(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} h(k_1,k_2) f(x-k_1, y-k_2) dk_1 dk_2 \quad (1)$$

Nếu $h(x,y)$ và $f(x,y)$ có dạng rời rạc thì công thức (1) sẽ được viết lại thành

$$h(n_1,n_2) * f(n_1,n_2) = \sum_{k_1=-\infty}^{\infty} \sum_{k_2=-\infty}^{\infty} h(k_1,k_2) f(n_1-k_1, n_2-k_2) \quad (2)$$

Trên thực tế người ta hay dùng $h(n_1,n_2)$ là ma trận [3x3] như sau:

$$h = \begin{pmatrix} h(-1,1) & h(0,1) & h(1,1) \\ h(1,0) & h(0,0) & h(1,0) \\ h(-1,-1) & h(0,-1) & h(1,-1) \end{pmatrix} \quad (3)$$

Cấu trúc và giá trị của các toán tử phát hiện biên sẽ xác định hướng đặc trưng mà toán tử nhạy cảm với biên. Có một số toán tử thích hợp cho các đường biên có hướng nằm ngang, một số toán tử lại thích hợp cho việc tìm kiếm biên dạng thẳng đứng hay theo hướng đường chéo. Có nhiều phương pháp phát hiện biên đang được áp dụng, tuy nhiên ta có thể phân thành hai nhóm cơ bản là phát hiện biên dùng Gradient và phương pháp Laplacian. Phương pháp phát hiện biên dùng Gradient (sử dụng các toán tử Roberts, Prewitt, Sobel, Canny) dựa vào tính giá trị cực đại và cực tiểu của đạo hàm bậc nhất của ảnh. Phương pháp Laplacian sẽ tìm kiếm những điểm có giá trị 0 khi lấy đạo hàm bậc hai của ảnh (Mars-Hildreth).

2.1.3.1. Phương pháp Gradient

Phương pháp Gradient là phương pháp dò biên cục bộ bằng cách tìm kiếm cực đại và cực tiểu khi lấy đạo hàm bậc nhất của ảnh trong không gian hai chiều.

Gradient là một vector có các thành phần biểu thị tốc độ thay đổi giá trị của điểm ảnh theo 2 hướng x và y, hay có thể nói là nó đại diện cho sự thay đổi về hướng và độ lớn của một vùng ảnh.

Đạo hàm bậc nhất theo hướng ngang và dọc được tính theo (4)

$$\Delta f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \quad (4)$$

Biên độ của gradient vector hay độ lớn tổng cộng của giá trị đạo hàm nằm tại biên là kết hợp của cả hai giá trị này theo công thức (5)

$$\Delta f = |\Delta f| = \sqrt{G_x^2 + G_y^2} \quad (5)$$

Hướng của gradient vector được xác định theo

$$\text{angle of } \nabla \mathbf{f} = tan^{-1}\left(\frac{G_y}{G_x}\right) \quad (6)$$

Hướng của biên sẽ vuông góc với hướng của gradient vector này.

a. Toán tử Sobel

Trên thực tế Sobel sử dụng hai mặt nạ có kích thước [3 x 3] trong đó một mặt nạ chỉ đơn giản là sự quay của mặt nạ kia đi một góc 90^0 như ở hình 2. Các mặt nạ này được thiết kế để tím ra các đường biên theo chiều đứng và chiều ngang một cách tốt nhất. Khi thực hiện phép convolution giữa ảnh và các mặt nạ này ta nhận được các gradient theo chiều đứng và chiều ngang Gx, Gy. Toán tử Sobel có dạng như **Hình 2.4**

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

Hình 2.4. *Toán tử Sobel*

b. Toán tử Prewitt

Phương pháp Prewitt gần giống với Sobel. Đây là phương pháp lâu đời nhất, cổ điển nhất. Toán tử Prewitt được mô tả trên **Hình 2.5**

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Hình 2.5. *Toán tử Prewitt*

c. Toán tử Robert

Tương tự như Sobel, ta tính đường biên ngang và dọc một cách riêng rẽ dùng 2 mặt nạ như hình 4, sau đó tổng hợp lại để cho đường biên thực của ảnh. Tuy nhiên do mặt nạ của Robert khá nhỏ nên kết quả là bị ảnh hưởng khá nhiều của nhiễu.

0	0	0
0	−1	0
0	0	1

0	0	0
0	0	−1
0	1	0

Hình 2.6. *Toán tử Robert*

d. Phương pháp Canny

Phương pháp này sử dụng hai mức ngưỡng cao và thấp. Ban đầu ta dùng mức ngưỡng cao để tìm điểm bắt đầu của biên, sau đó chúng ta xác định hướng phát triển của biên dựa vào các điểm ảnh liên tiếp có giá trị lớn hơn mức ngưỡng thấp. Ta chỉ loại bỏ các điểm có giá trị nhỏ hơn mức ngưỡng thấp. Các đường biên yếu sẽ được chọn nếu chúng được liên kết với các đường biên khỏe.

Phương pháp Canny bao gồm các bước sau:

Bước 1. Trước hết dùng bộ lọc Gaussian (3.4) để làm mịn ảnh.

$$G'(x) = \left(-\frac{x}{\sigma^2}\right) e^{-\left(\frac{x^2}{2\sigma^2}\right)} \quad (7)$$

Bước 2. Sau đó tính toán gradient (8) và (9) của đường biên của ảnh đã được làm mịn.

$$C_x[x,y] = -\left(\frac{j}{\sigma^2}\right) e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \quad (8)$$

$$C_y[x,y] = -\left(\frac{i}{\sigma^2}\right) e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)} \quad (9)$$

Bước 3. Tiếp theo là loại bỏ những điểm không phải là cực đại.

Bước 4. Bước cuối cùng là loại bỏ những giá trị nhỏ hơn mức ngưỡng.

Phương pháp này hơn hẳn các phương pháp khác do ít bị tác động của nhiễu và cho khả năng phát hiện các biên yếu. Nhược điểm của phương pháp này là nếu chọn ngưỡng quá thấp sẽ tạo ra biên không đúng, ngược lại nếu chọn ngưỡng quá cao thì nhiều thông tin quan trọng của biên sẽ bị loại bỏ. Căn cứ vào mức ngưỡng đã xác định trước, ta sẽ quyết định những điểm thuộc biên thực hoặc không thuộc biên. Nếu mức ngưỡng càng thấp, số đường biên được phát hiện càng nhiều (nhưng kèm theo là nhiễu và số các đường biên giả cũng xuất hiện càng nhiều). Ngược lại nếu ta đặt mức ngưỡng càng cao, ta có thể bị mất những đường biên mờ hoặc các đường biên sẽ bị đứt đoạn.

Phương pháp Canny có các ưu điểm sau:

- Cực đại hóa tỷ số tín hiệu trên nhiễu làm cho việc phát hiện các biên thực càng chính xác.

- Đạt được độ chính xác cao của đường biên thực.

- Làm giảm đến mức tối thiểu số các điểm nằm trên đường biên nhằm tạo ra các đường biên mỏng, rõ.

2.1.3.2. Laplacian of Gaussian (LOG)

Dùng phương pháp gradient sẽ cho kết quả là ảnh nhận được có cấu trúc không rõ nét do tạo nên những đường biên dày, không sắc nét. Để nhận được các đường biên mỏng và rõ nét, ta phải tiến hành các bước xử lý tiếp theo như loại bỏ những điểm không

phải là cực trị (*nonmaximum suppression*) đồng thời áp dụng kỹ thuật liên kết biên (*edge linking*). Ngoài ra ta còn gặp phải vấn đề là làm thế nào để xác định được mức ngưỡng một cách chính xác. Việc chọn đúng giá trị ngưỡng phụ thuộc rất nhiều vào nội dung của từng bức ảnh. Nếu ta tăng gấp đôi kích thước của một bức ảnh mà không thay đổi giá trị cường độ của các điểm ảnh, ta sẽ nhận được gradients bị suy giảm đi một nửa. Mặt khác kích thước của mặt nạ (*masks*) cũng ảnh hưởng nhiều đến giá trị của gradients trong ảnh.

Phương pháp gradient chỉ thích hợp cho các vùng ảnh độ tương phản thay đổi có tính nhảy bậc, điều này gây khó khăn cho phát hiện các đường thẳng. Để khắc phục nhược điểm này ta thường dùng đạo hàm bậc hai. Phương pháp Laplacian cho phép xác định đường biên dựa vào giá trị 0 của đạo hàm bậc hai của ảnh. Laplacian của một ảnh tại điểm I(x,y) được tính theo (10):

$$L(x,y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2} \qquad (10)$$

Laplacian được kết hợp với bộ lọc làm mịn ảnh để tìm biên.Xét công thức sau:

$$h(r) = -e^{-\frac{r^2}{2\sigma^2}} \qquad (11)$$

Ở đây $r^2 = x^2 + y^2$ và σ là độ lệch chuẩn (standard deviation). Nếu thực hiện phép tích chập của hàm này với ảnh cần tìm biên, kết quả là ảnh sẽ bị mờ đi, mức độ mờ phụ thuộc vào giá trị của σ. Laplacian của h tức đạo hàm bậc hai của h theo r là:

$$\nabla^2 h(r) = -\left[\frac{r^2-\sigma^2}{\sigma^4}\right] - e^{-\frac{r^2}{2\sigma^2}} \qquad (12)$$

Hàm này thường được gọi là Laplacian of a Gaussian (LoG) do (11) có dạng Gaussian.

Trong phương pháp này, bộ lọc Gaussian được kết hợp với Laplacian cho phép hiển thị những vùng ảnh có cường độ thay đổi nhanh do đó làm tăng hiệu quả phát hiện biên. Nó cho phép làm việc với một diện tích rộng hơn xung quanh điểm ảnh đang được nghiên cứu nhằm phát hiện chính xác hơn vị trí của đường biên. Nhược điểm của phương pháp này là không xác định được hướng của biên do sử dụng hai bộ lọc Laplacian quá khác nhau có dạng như trên hình.

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Hình 2.7. *Toán tử Laplacian*

2.1.3.3. Kết quả thực nghiệm

Ta tiến hành so sánh hiệu quả phát hiện biên khi áp dụng các toán tử nêu trên dùng MATLAB. Chất lượng phát hiện biên được đánh giá thông qua các tiêu chí như sai số trung bình bình phương (MSE) và tỷ số tín hiệu/nhiễu (PSNR).

a. Sai số trung bình bình phương MSE đánh giá mức độ sai khác giữa biên nhận được do tính toán và biên thực thông qua công thức:

$$MSE = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_1(i,j) - f_2(i,j))^2 \quad \text{và}$$

$$RMSE = \sqrt{MSE} \quad (13)$$

Với $\boldsymbol{f}_1(i, j)$ và $\boldsymbol{f}_2(i, j)$ là các điểm ảnh trên biên theo tính toán và trên biên thực.

b. Tỷ số tín hiệu/nhiễu được tính theo công thức

$$PSNR = 10\log(255^2/MSE) \quad (14)$$

Kết quả tính toán đối với 2 bức ảnh “moon.jpg” và “Lena.jpg” được cho trong bảng 1 và bảng 2.

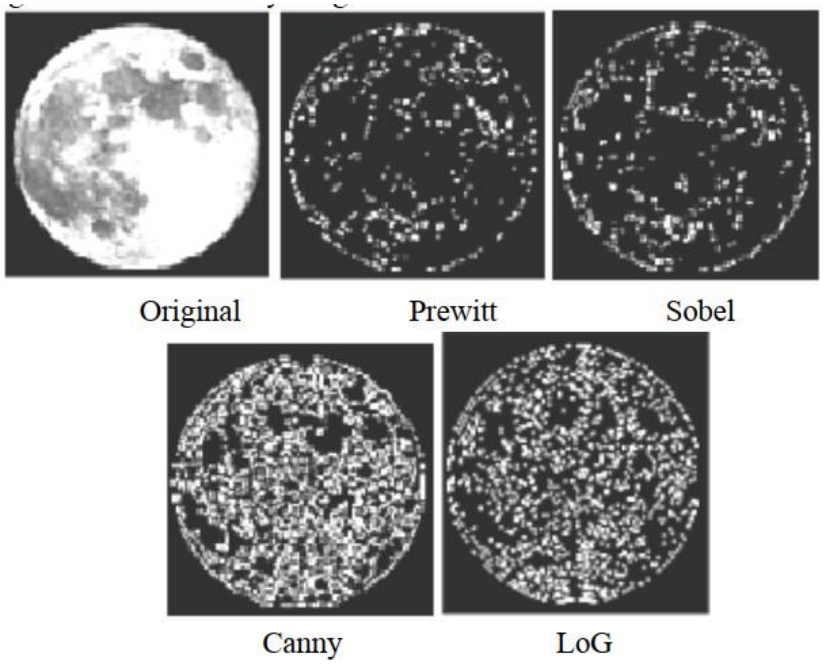
STT	Phương pháp	MSE	PSNR
1	Sobel	9.9033e+003	8.1730
2	Prewitt	9.9035e+003	8.1729
3	Canny	9.8804e+003	8.1830
4	LoG	9.8881e+003	8.1797

Bảng 2.1. *So sánh MSE và PSNR của ảnh mặt trăng (moon.jpg)*

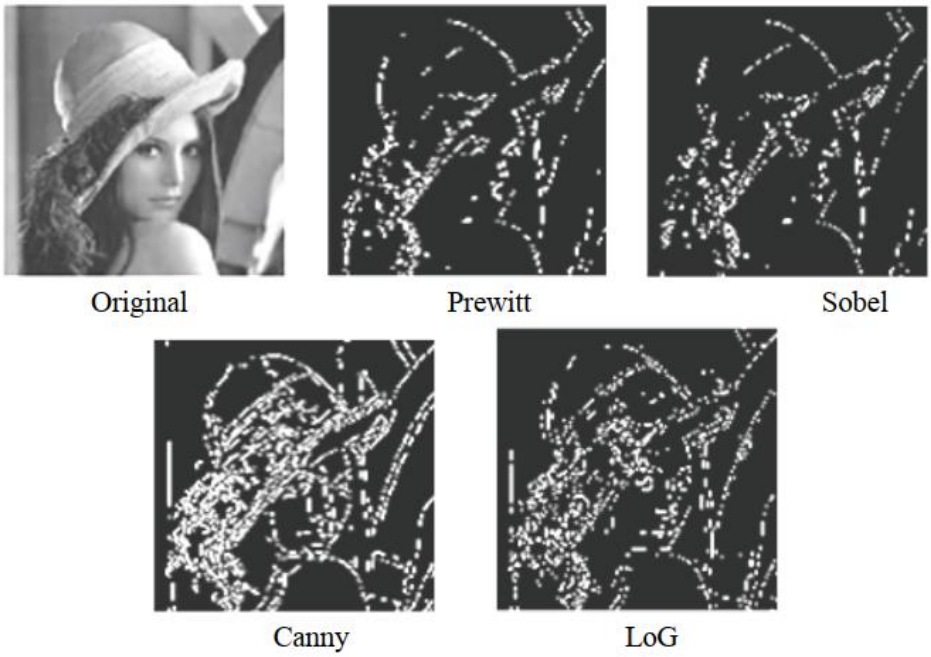
STT	Phương pháp	MSE	PSNR
1	Sobel	1.1028e+004	7.7058
2	Prewitt	1.1028e+004	7.7058
3	Canny	1.1017e+004	7.7103
4	LoG	1.1022e+004	7.7081

Bảng 2.2. *So sánh MSE và PSNR của ảnh cô gái Lena(Lena.jpg)*

Qua số liệu ở bảng 1 và bảng 2 thấy phương pháp Canny cho kết quả MSE có giá trị nhỏ nhất và PSNR cho giá trị lớn nhất so với các phương pháp khác. Hình 5 và hình 6 cho ta kết quả khi áp dụng các toán tử nêu trên cho hai bức ảnh có nội dung khác nhau. Ta thấy rằng phương pháp Canny cho phép hiện đầy đủ hơn các biên có thể có trên các bức ảnh. Phương pháp LoG cho ta thấy rõ hơn các đường thẳng thuộc biên, đồng thời chỉ số MSE và PSNR cũng đạt được tương đối tốt.



Hình 2.8. *Kết quả phát hiện biên sử dụng các toán tử Prewitt, Sobel, Canny và LoG cho ảnh "moon.jpg"*



Hình 2.9. *Kết quả phát hiện biên sử dụng các toán tử Prewitt, Sobel, Canny và LoG cho ảnh "Lena.jpg"*

2.2. Tổng quan về nhận dạng ký tự quang học

Nhận dạng ký tự quang học (*Optical Character Recognition*, viết tắt là OCR) [2] là phần mềm máy tính tạo ra để chuyển đổi các hình ảnh có chứ các ký tự (hình ảnh đó có được do quét bằng máy scan, camera hay đơn giản chỉ là một hình ảnh đưa vào) thành các dạng văn bản, tài liệu. OCR được hình thành dựa trên lĩnh vực nghiên cứu về nhận dạng mẫu, trí tuệ nhân tạo và thị giác máy tính. Hiện nay OCR đã được ứng dụng trong rất nhiều lĩnh vực và cũng được nhiều nhà nghiên cứu tiến hành khai phá ứng dụng của nó nhưng vẫn còn rất nhiều ứng dụng có thể phát triển thêm nhờ OCR, điều này là một thách thức và cũng là động lực cho các nhà nghiên cứu về máy học.

Nhận dạng ký tự quang học (dùng các kỹ thuật quang học chẳng hạn như gương và ống kính) và nhận dạng ký tự số (sử dụng máy quét và các thuật toán máy tính) lúc đầu người ta xem như hai lĩnh vực khác nhau. Bởi vì chỉ có rất ít các ứng dụng tồn tại với các kỹ thuật quang học thực sự, bởi vậy thuật ngữ **Nhận dạng ký tự quang học** được mở rộng và bao gồm luôn ý nghĩa nhận dạng ký tự số.

Đầu tiên hệ thống nhận dạng yêu cầu phải được huấn luyện với các mẫu ký tự cụ thể. Các hệ thông "thông minh" với độ chính xác nhận dạng cao đói với hầu hết các

phông chữ hiện nay đã trở lên phổ biến. Một số hệ thống còn có khả năng tái tạo lại các định dạng của tài liệu gần giống với bản gốc bao gồm: hình ảnh, các cột, bảng biểu, các thành phần không phải là văn bản.

ABBYY - một hãng công nghệ hàng đầu trên thế giới về lĩnh vực **Nhận dạng ký tự quang học** đã tiến hành nghiên cứu và triển khai công nghệ nhận dạng Tiếng Việt vào tháng 4 năm 2009. Với công nghệ này độ chính xác trong việc nhận dạng tài liệu chữ in Tiếng Việt lên tới hơn 99% (cứ nhận dạng 100 ký tự thì có chứa đến 1 ký tự sai). Công nghệ của ABBYY chấp nhận hầu hết các định dạng ảnh đầu vào như: PDF, TIFF, JPEG, GIF, PNG, BMP, PCX, DCX, DjVu... Kết quả nhận dạng được lưu trữ dưới các định dạng MS Word, MS Excel, HTML, TXT, XML, PDF, PDF 2 lớp, trong đó định dạng PDF 2 lớp là một định dạng hoàn hảo cho việc lưu trữ và khai thác tài liệu.

Và Tesseract là một OCR (Optical Character Recognition) hàng đầu hiện nay.

2.2.1. Tổng quan về Tesseract [3] :

Tesseract là một OCR (Optical Character Recognition) hàng đầu hiện nay. Công cụ này được phân phối với bản quyền mã nguồn mở Apache 2.0. Nó hỗ trợ nhận diện kí tự trên các tập tin hình ảnh và xuất ra dưới dạng kí tự thuần, html, pdf, tsv, invisibletext-only pdf. Người dùng có thể sử dụng trực tiếp hoặc lập trình viên có thể sử dụng các chức năng thông qua API.

Tesseract được phát triển bởi Hewlett-Packard Laboratories Bristol tại HewleettPackard Co, Greeley Colorado từ 1985 đến 1994. Sau đó, nó được cập nhật một số thay đổi nhỏ và tạm ngưng phát triển từ sau 1998. Đến năm 2005, Tesseract được phân bố dưới dạng mã nguồn mở bởi HP và được phát triển bởi Google từ năm 2006.

Hiện tại, Tesseract đã phát triển đến version 3.0x và có thể hoạt động trên 3 hệ điều hành phổ biến là Window, Mac và Linux. Công cụ này hỗ trợ nhận diện kí tự của hơn 100 ngôn ngữ khác nhau, bao gồm cả tiếng Việt. Không những thế, chúng ta có thể huấn luyện chương trình dùng Tesseract để có thể nhận diện một ngôn ngữ nào đó. Bên cạnh đó, mã nguồn mở này không hỗ trợ GUI, nên bạn sẽ cần tới ứng dụng của bên thứ ba nếu muốn sử dụng chức năng này.

Đối với các lập trình viên, họ có thể sử dụng các API của Tesseract để xây dựng ứng dụng của mình. Thư viện đó gọi là labtesseract và được cung cấp cho ngôn ngữ C/C++. Trong trường hợp sử dụng ngôn ngữ khác thì cần phải sử dụng các gói hỗ trợ tương ứng:

- .NET: charlesw/tesseract, http://code.google.com/p/tesseractdotnet/
- Python: tesserocr, pyocr, …
- Java: tess4j 10

Tesseract có ưu điểm là dễ dàng sử dụng, giúp người dùng tiết kiệm thời gian. Trong khi đó nhược điểm chủ yếu là độ chính xác chưa cao, với những hình ảnh có màu nền mà màu chữ không có nhiều chênh lệch, hay các hình chụp chữ viết tay thì kết quả nhận dạng không khả quan

2.2.2. Huấn luyện dữ liệu trên tesseract[4]

Tesseract ban đầu được thiết kế để nhận dạng các từ tiếng Anh trên ngôn ngữ hệ Latinh. Sau này, nhờ sự cố gắng của nhiều nhà phát triển mà các phiên bản của Tesseract đã có thể nhận diện các ngôn ngữ khác ngoài hệ Latinh như tiếng Trung, tiếng Nhật và tương thích với các ký tự trong bảng mã UTF-8. Việc nhận dạng các ngôn ngữ mới trên Tesseract có thể thực hiện được nhờ vào việc huấn luyện dữ liệu. Từ phiên bản 3.0 trở đi, Tesseract đã có thể hỗ trợ thêm nhiều dạng ngôn ngữ mới và mở rộng thêm việc huấn luyện theo font chữ. Bởi vì ban đầu, bộ Tesseract được huấn luyện để nhận diện từ chính xác nhất trên một số loại font mặc định, nếu sử dụng các font chữ khác để nhận diện thì có thể kết quả sẽ không có độ chính xác cao khi làm việc với các loại font được cài đặt sẵn trong dữ liệu huấn luyện. Để thực hiện quá trình huấn luyện thì ta phải sử dụng công cụ có sẵn của Tesseract. Mặc định trong luận văn này, sử dụng công cụ Tesseract 3.01 cho việc thực hiện huấn luyện ngôn ngữ và font mới.

Để huấn luyện dữ liệu trên Tesseract (hoặc ngôn ngữ mới) thì ta cần một tập các tập tin dữ liệu chứa trong thư mục tessdata, sau đó kết hợp các tập tin này thành tập tin duy nhất. Các tập tin có trong thư mục tessdata có quy tắc đặt tên theo dạng:

tên_ngôn_ngữ.tên_tập tin. Ví dụ các tập tin cần thiết khi thực hiện việc huấn luyện tiếng Anh:

- tessdata/eng.config.
- tessdata/eng.unicharset: Tập ký tự của ngôn ngữ huấn luyện.
- tessdata/eng.unicharambigs.
- tessdata/eng.inttemp: Danh mục cho tập hợp các ký tự.
- tessdata/eng.pffmtable: Tập tin dạng hộp – sử dụng để xác định ký tự có trong tập tin huấn luyện.
- tessdata/eng.normproto: Như tập tin pffmtable.
- tessdata/eng.punc-dawg.
- tessdata/eng.number-dawg.
- tessdata/eng.freq-dawg: Danh sách các từ tổng quát.
- tessdata/eng.word-dawg: Danh sách các từ thông thường.
- tessdata/eng.user-word: Danh sách từ của người dùng (tùy chọn có thể có hoặc không).

Bước cuối cùng sẽ tổng hợp dữ liệu từ bước trên và phát sinh ra tập tin dữ liệu duy nhất có dạng:

- tessdata/eng.traineddata.

Các tập tin cần thiết cho việc huấn luyện dữ liệu sẽ được phát sinh khi ta sử dụng công cụ có sẵn để qua quá trình huấn luyện.

Tổng hợp lại ta có:

- Sinh hình ảnh huấn luyện
- Tạo các tập tin *.box
- Bắt đầu chạy huấn luyện Tesseract
- Clustering (tập hợp lại)
- Thêm dữ liệu từ điển (tùy biến)
- Tổ hợp kết quả lại với nhau:

Trong việc ứng dụng Tesseract enginer cho nhận dạng ký hiệu toán học, thực chất các vấn đề khó khăn nhất nằm ở khâu huấn luyện hơn là các vấn đề về lập trình và tích hợp phần mềm.

2.2.3. Quá trình huấn luyện ngôn ngữ và font mới[4]

Để trải qua quá trình huấn luyện ngôn ngữ hoặc loại font mới trên Tesseract ta cần thực hiện thông qua các giai đoạn sau:

- Phát sinh các tập tin hình ảnh cho việc huấn luyện:

Đây là bước đầu tiên nhằm xác định tập ký tự sẽ được sử dụng trong việc huấn luyện. Trước hết ta cần chuẩn bị sẵn một tập tin văn bản chứa các dữ liệu huấn luyện (trường hợp cụ thể là một đoạn văn bản). Việc tạo ra tập tin huấn luyện cần theo các quy tắc sau:

- Bảo đảm số lần xuất hiện ít nhất của các ký tự trong mẫu từ khoảng 5 đến 10 lần cho một ký tự.
- Nên có nhiều mẫu cho các từ xuất hiện thường xuyên, ít nhất là 20 lần.
- Các dữ liệu huấn luyện nên được chia theo kiểu font, mỗi tập tin huấn luyện chỉ nên chứa 1 loại font nhưng có thế huấn luyện nhiều loại font cho nhiều tập tin. Không nên kết hợp nhiều loại font trong riêng một tập tin huấn luyện.
- Sau khi đã chuẩn bị mẫu văn bản dùng cho việc huấn luyện thì ta cần phá tsinh ra ảnh từ tập tin đó. Dùng các phần mềm để chuyển tập tin mẫu văn bản sang dạng tập tin ảnh hoặc in mẫu văn bản sau đó quét thành tập tin hình ảnh dạng.tif với độ phân giải là 300dpi. Tập tin cuối cùng trước khithực hiện việc huấn luyện là tập tin ảnh dạng .tif.

- Tạo các tập tin dạng hộp .box:

Một dạng tập tin để Tesseract có thể huấn luyện dựa trên các dữ liệu hình ảnh đã có bước đầu là tập tin dạng hộp – box. Tập tin dạng hộp là tập tin văn bản chứa 1 dãy các ký tự tuần tự từ đầu đến cuối trong tập tin hình ảnh, mỗi hàng chứa thông tin của 1 ký tự, tọa độ và đường bao quanh ký tự đó trong tập tin ảnh.

Để tạo ra tập tin dạng hộp ta sẽ dùng cách gõ lệnh (trên Windows là CMD và Linux là Terminal) sau (yêu cầu người dùng phải cài đặt công cụ Tesseract để có thể chạy được các lệnh này):

Sau khi thực hiện câu lệnh trên thì ta sẽ tạo ra được các tập tin dạng hộp .box.

- Chạy công cụ Tesseract trên máy tính để thực hiện việc huấn luyện dữ liệu. Sau khi được tập tin .box thì chúng ta cần 1 trình chỉnh sửa tập tin dạng hộp để kiểm tra lại và chỉnh sửa lại các thông số của từng ký tự cho khớp với văn bản ban đầu trong tập tin ảnh huấn luyện. Ở đây nhóm em dùng phần mềm TextBoxEditor để chỉnh sửa trực tiếp tập tin dạng hộp.

- Sau khi kiểm tra và chỉnh sửa lại các ký tự cho chính xác trong tập tin dạng hộp thì thực hiện lệnh tiếp theo: Nếu thành công thì tại giai đoạn này, Tesseract sẽ phát sinh ra tập tin .tr

➢ Ước lượng tập ký tự của ngôn ngữ cần huấn luyện:

Tesseract cần biết hết các tập ký tự có thể xuất hiện trong dữ liệu. Ta dùng lệnh sau:

Sau khi thực hiện, tập tin unicharset sẽ được tạo ra.

➢ Xác định kiểu font trong dữ liệu (từ phiên bản 3.0.1 trở đi):

Đây là tính năng mới chỉ có từ phiên bản Tesseract 3.0.1 trở đi. Với tính năng này người dùng có thể huấn luyện dữ liệu với nhiều loại font khác nhau thay vì chỉ có thể dùng các font mặc định sẵn ở các phiên bản trước. Ta cần tạo tập tin font_properties để quy định thông số các kiểu font ta đã sử dụng trong các mẫu văn bản huấn luyện.

Cấu trúc của tập tin font_properties là mỗi hàng chứa tên 1 loại font huấn luyện và các đặc tính của font đó: <tên loại font><in nghiêng><in đậm><bình thường><in hoa><fraktur> (đánh dấu có thuộc tính bằng bit 1 hoặc không có dùng bit 0). Ví dụ cấu trúc tập tin font_properties với dữ liệu huấn luyện là tiếng Anh.

➢ Gom nhóm dữ liệu:

Tại giai đoạn này thì các đường nét khung của ký tự đã được rút trích ra và chúng ta cần gom nhóm lại các dữ liệu ban đầu để tạo ra mẫu thử (prototype). Hình dạng, đường nét của các ký tự sẽ được gom nhóm lại nhờ vào chương trình mftraining và cntraining có sẵn trong công cụ Tesseract:

Với lệnh mftraining sẽ tạo ra tập tin dữ liệu: inttemp (chứa hình dạng mẫu), pffmtable và Microfeat nhưng ít khi sử dụng).

Cuối cùng dùng công cụ cntraining sẽ tạo ra tập tin dữ liệu normproto.

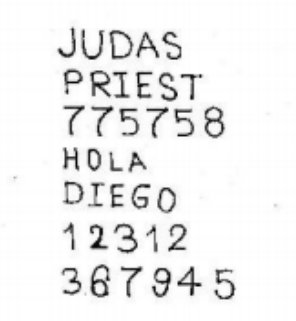
- Tạo tập tin unicharambigs.
- Kết hợp các tập tin lại tạo thành tập tin huấn luyện dữ liệu:

Cuối cùng sau khi đã có đủ các tập tin huấn luyện cần thiết (inttemp, pffmtable, normproto, Microfeat) thì ta đổi tên các tập tin lại cho đúng dạng với tiền tố lang. trước tên tập tin với lang là 3 ký tự đại diện cho ngôn ngữ huấn luyện theo chuẩn ISO 639-2 Kết quả là tạo ra tập tin lang.trainedata. Bỏ tập tin này vào thưc mục tessdata của Tesseract thì Tesseract đã có thể nhận diện được ngôn ngữ hoặc font chữ mới.

2.2.4. Một số thử nghiệm[3]

Tiến hành thử nghiệm trên ba loại hình ảnh: Hình chụp từ chữ viết tay (1), hình chụp từ chữ đánh máy (2) và hình từ tập tin pdf (3).

Hình chữ viết tay



Hình 2.10. *Ví dụ về hình chữ viết tay*

- Kết quả:

JUDA$
PRIEST
775758
HOLA
DIEGO
12312
387945

Hình 2.11. *Kết quả chữ viết tay*

- Tỉ lệ sai : 1/33 chiếm 3,03%

Hình chữ đánh máy

Hình 2.12. *Ví dụ về hình chữ đánh máy*

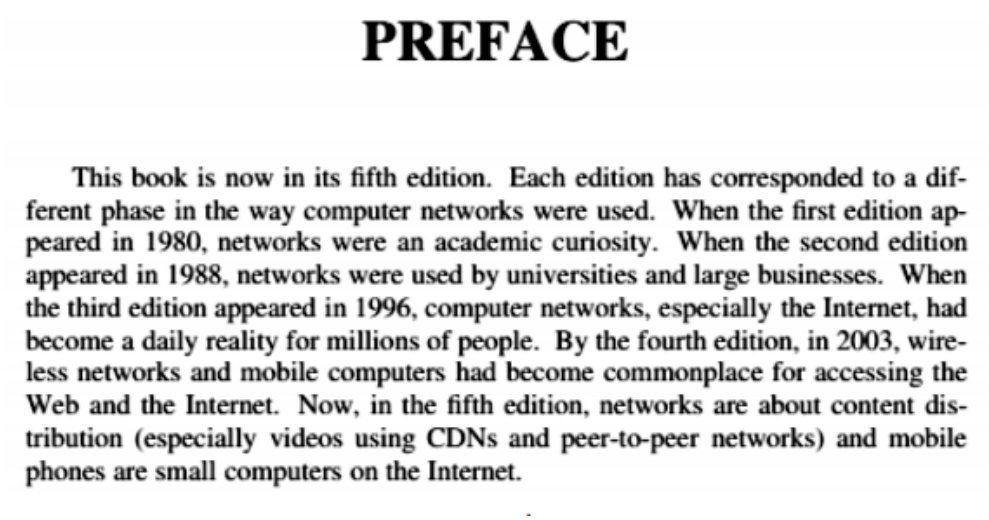
- Kết quả:

ESTA67
ES767
UNA4567
PRU EBA5887

Hình 2.13. *Kết quả chữ đánh máy*

- Tỉ lệ sai : 1/28 chiếm 3,57%

Hình ảnh tập tin .pdf



PREFACE

This book is now in its fifth edition. Each edition has corresponded to a different phase in the way computer networks were used. When the first edition appeared in 1980, networks were an academic curiosity. When the second edition appeared in 1988, networks were used by universities and large businesses. When the third edition appeared in 1996, computer networks, especially the Internet, had become a daily reality for millions of people. By the fourth edition, in 2003, wireless networks and mobile computers had become commonplace for accessing the Web and the Internet. Now, in the fifth edition, networks are about content distribution (especially videos using CDNs and peer-to-peer networks) and mobile phones are small computers on the Internet.

Hình 2.14. *Ví dụ về hình dạng pdf*

- Kết quả:

PREFACE
This book is now In "5 mm edllmn Eden edmon has cormsponded In a dlf—
teaenr phase rn me way camplllnt networks were used When the firs! edman ap
peared in man. networks weae an academic cum: Iy When me second edmorr
appeared In 1933. networks were used by unlvcrslues and large businesses When
lhe nrrrd ednmn appeared in 1995, compuler networks. especially lhe Inrer-rrer,
had
become a duly reamy rar mrnmna cl penplc By lhe rrnrrnr edllmn. in 2003. wu':—
less nclwmks and mohllc compumeus had become commonplace for accessing rhe
Web and me unerrrer. Now, In [he mun edllkm, networks are about content
u1bullan(espeda.Ily videos using cum and pecuopccr networks) and mobile
phones are small mmpulers on the xnrerner

Hình 2.15. *Kết quả về hình dạng pdf*

- Tỉ lệ sai trên 50% so với văn bản gốc. Văn bản càng dài độ chính xác càng giảm dần.

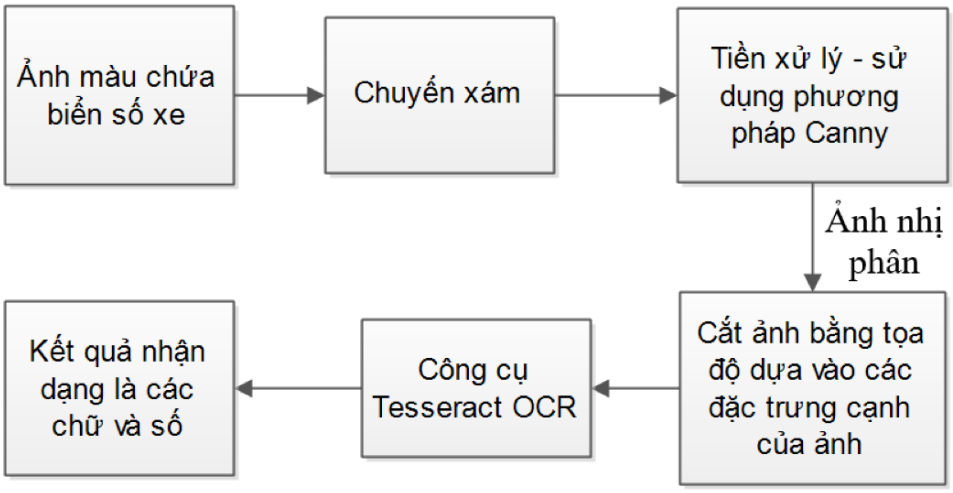
Kết luận : Tesseract dùng để nhận dạng kí tự trên một tập tin hình và chuyển kí tự thành tập tin thành văn bản. Bên cạnh những ưu điểm vượt trội của mình, Tesseract cũng có một số những hạn chế như nhầm lẫn giữa chữ hoa và chữ thường, nhầm lẫn giữa các kí tự có hình dáng tương tự, đúng từ nhưng sai trong ngữ cảnh.

CHƯƠNG 3: BÀI TOÁN NHẬN DẠNG BIỂN SỐ XE Ô TÔ

3.1. Giới thiệu:

Trong bài toán nhận dạng biển số xe ô tô, đầu vào là hình ảnh chứa vùng biển số xe ô tô. Công việc chúng ta cần làm là bóc tách khu vực chứa biển số xe. Sau đó, đưa khu vực chứa biển số xe vào quá trình huấn luyện và nhận dạng. Để bóc tách được khu vực chứa biển số xe ta phải thực hiện quá trình tiền xử lý ảnh: xám hóa, giảm nhiễu, nhị phân hóa, lọc ngưỡng. Để nhận dạng được các ký tự chữ và số, ta sử dụng công cụ Tesseract OCR.

Bài toán được thực hiện theo quy trình như sau:



Hình 3.1. *Quy trình xử lý*

3.2. Ảnh màu đầu vào và chuyển xám:

Ảnh màu: Ảnh màu theo lý thuyết của Thomas là ảnh tổ hợp từ 3 màu cơ bản: đỏ (R), lục (G), lơ (B) và thường thu nhận trên các dải băng tần khác nhau. Với ảnh màu, cách biểu diễn cũng tương tự như với ảnh đen trắng, chỉ khác là các số tại mỗi phần tử của ma trận biểu diễn cho ba màu riêng rẽ gồm: đỏ (red), lục (green) và lam (blue). Để biểu diễn cho một điểm ảnh màu cần 24 bit. 24 bit này được chia thành ba khoảng 8 bit. Mỗi màu cũng phân thành L cấp màu khác nhau (thường $L=256$). Mỗi khoảng này biểu diễn cho cường độ sáng của một trong các màu chính.

Hình 3.2. *Ảnh màu chứa biển số xe ô tô*

Ảnh đen trắng: Ảnh đen trắng chỉ bao gồm 2 màu: màu đen và màu trắng. Người ta phân mức đen trắng đó thành L mức. Nếu sử dụng số bit B=8 bit để mã hóa mức đen trắng (hay mức xám) thì L được xác định : $L=2^B$ (trong ví dụ của ta $L=2^8= 256$ mức). Nếu L bằng 2, B=1, nghĩa là chỉ có 2 mức: mức 0 và mức 1, còn gọi là ảnh nhị phân. Mức 1 ứng với màu sáng, còn mức 0 ứng với màu tối. Nếu L lớn hơn 2 ta có ảnh đa cấp xám. Nói cách khác, với ảnh nhị phân mỗi điểm ảnh được mã hóa trên 1 bit, còn với ảnh 256 mức, mỗi điểm ảnh được mã hóa trên 8 bit. Như vậy, với ảnh đen trắng: nếu dùng 8 bit (1 byte) để biểu diễn mức xám, số các mức xám có thể biểu diễn được là 256. Mỗi mức xám được biểu diễn dưới dạng là một số nguyên nằm trong khoảng từ 0 đến 255, với mức 0 biểu diễn cho mức cường độ đen nhất và 255 biểu diễn cho mức cường độ sáng nhất.

Hình 3.3. *Ảnh đã chuyển xám*

3.3. Tiền xử lý ảnh:

Để bóc tách biển số xe ô tô ra ảnh gốc và chuyển về dạng nhị phân, ta sử dụng phương pháp Canny.

3.2.1. Giới thiệu phương pháp Canny – Tiền xử lý để lấy biển số ra từ ảnh

Mục đích của việc phát hiện biên nói chung là giảm thiểu đáng kể số lượng dữ liệu trong một hình ảnh, trong khi các đặc tính cấu trúc vẫn được giữ để sử dụng cho việc xử lý hình ảnh hơn nữa. Có rất nhiều phương pháp phát hiện biên đã được đề cập ở trên nhưng phương pháp Canny – được phát triển bởi John F. Canny (JFC) vào năm 1986, là một trong những công cụ xử lý hình ảnh thông dụng nhất. Mặc dù nó khá là cũ, song đã trở thành một trong những phương pháp phát hiện biên tiêu chuẩn và vẫn được sử dụng trong nghiên cứu.

Mục tiêu (ràng buộc) của JFC để phát triển thuật toán đó là tối ưu những vấn đề liên quan đến các tiêu chuẩn sau:

- Phát hiện: Xác suất phát hiện những điểm biên thực sự phải là cực đại hóa, trong khi xác suất lỗi phát hiện những điểm không phải biên cần được giảm thiểu. Điều này tương ứng với tối đa hóa tỷ lệ báo hiệu nhiễu.

- Cục bộ hóa: Các biên phát hiện càng gần biên thực càng tốt. Có nghĩa là độ chênh lệch cấp xám giữa các điểm trên cùng một biên càng nhỏ càng tốt.

- Số lượng trả lời – hiệu suất: biên không được nhận ra nhiều, trong khi chỉ có một biên tồn tại -> giảm số lượng biên được phát hiện không được nhận ra.

3.2.2. Các bước thực hiện

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 (object). Do đó, để phát hiện cạnh trong ảnh, giải thuật Canny là một trong những giải thuật phổ biến, nổi tiếng nhất trong Xử lý ảnh.

Thuật toán được tiến hành qua 4 bước riêng biệt sau:

- Bước 1: Giảm nhiễu: làm trơn ảnh để loại bỏ nhiễu bằng cách nhân chập ảnh với bộ lọc Gauss.

- Bước 2: Tìm gradient: Tính toán góc và chiều dài của gradient. Biên nên được đánh dấu là nơi mà gradient của ảnh có chiều dài lớn.

- Bước 3: Thực hiện "Non-maximum suppression": Chỉ cực đại cục bộ những điểm được đánh dấu là biên (có mức xám cao).

- Bước 4: Lọc ngưỡng: Những biên tiềm năng được xác định bởi ngưỡng cao và ngưỡng thấp. (2 ngưỡng cao thấp)

3.2.3. Mô tả các bước thực hiện

a. Bước 1: Giảm nhiễu

Đây là điều không thể tránh khỏi vì tất cả các hình ảnh chụp từ máy quay sẽ chứa một số nhiễu. Để ngăn nhầm lẫn nhiễu với các biên, nhiễu phải được giảm bớt. Do đó ảnh trước tiên được làm mịn bằng cách áp dụng một bộ lọc Gauss. Cách thức tiến hành giống như ở Laplace of Gauss.

***Bộ lọc Gauss[8]** là cách làm mờ một ảnh bằng hàm Gaussian. Phương pháp này được ứng dụng một cách rộng rãi và hiệu quả trong các phần mềm xử lý đồ họa. Nó cũng là công cụ phổ biến để thực hiện quá trình tiền xử lý (preprocessing) hình ảnh dùng làm dữ liệu đầu vào tốt cho các phân tích cao cấp hơn như trong Computer Vision, hoặc cho các giải thuật được thực hiện trong một tỉ lệ khác của hình được cho. Nó có thể giúp làm giảm nhiễu (Noise) và mức độ chi tiết (không mong muốn) của hình ảnh. Như vậy phát biểu một cách thực hành hơn **Bộ lọc Gauss** là một loại bộ lọc làm mờ ảnh, sử dụng lý thuyết hàm Gaussian (cũng được biết đến như là dạng phân tán chuẩn (Normal Distribution) trong thống kê để tính toán việc chuyển đổi (Transformation) mỗi Pixel của hình.

b. Bước 2: Tìm Gradient

Các thuật toán Canny về cơ bản tìm thấy các biên nơi mà cường độ mức xám của hình ảnh thay đổi nhiều nhất. Những vùng này được tìm thấy bằng cách xác định gradient của ảnh. Gradient tại mỗi điểm ảnh trong ảnh được làm mịn, được xác định bằng cách áp dụng những phướng pháp dựa theo toán tử Sobel. Bước đầu tiên là đạo hàm các kết quả ở bước 1 theo hướng x và y với mặt nạ 3x3:

$$H_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad H_y = \begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Các biên độ gradient (còn được gọi là những cường độ biên) sau đó có thể được xác định như là một thước đo khoảng cách Euclide bằng cách áp dụng luật của Pythagoras[9] như trong phương trình sau:

$$S' = \sqrt{S_x^2 + S_y^2}$$

Đôi khi nó được đơn giản hóa bằng cách áp dụng thước đo khoảng cách Manhattan để giảm bớt sự phức tạp tính toán:

$$|S'| = |S_x| + |S_y|$$

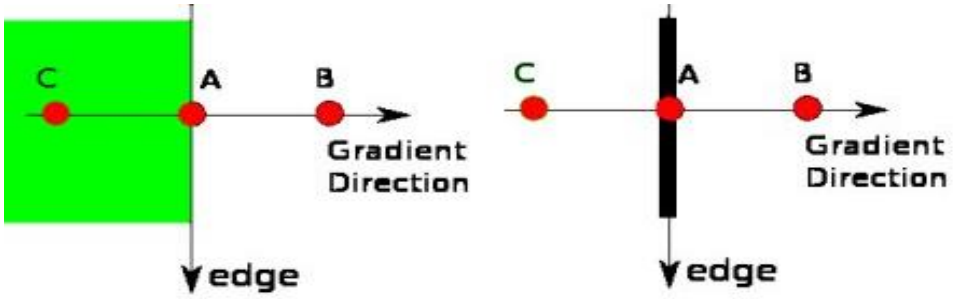
trong đó S_x và S_y là gradient theo 2 hướng x và y tương ứng và hướng của biên θ như sau:

$$\theta = tan^{-1}\frac{S_y}{S_x}$$

Ảnh S' tìm được là kết quả của bước thứ 2.

c. Bước 3: Tiến hành "Non-maximum suppression"

Tức là loại bỏ các pixel ở vị trí không phải cực đại toàn cục. Ở bước này, ta dùng một filter 3x3 lần lượt chạy qua các pixel trên ảnh gradient. Trong quá trình lọc, ta xem xét xem độ lớn gradient của pixel trung tâm có phải là cực đại (lớn nhất trong cục bộ - local maximum) 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**. Ví dụ: nếu hướng gradient đang là 0 độ, ta sẽ so pixel trung tâm với pixel liền trái và liền phải nó. Trường hợp khác nếu hướng gradient là 45 độ, ta sẽ so sánh với 2 pixel hàng xóm là góc trên bên phải và góc dưới bên trái của pixel trung tâm.

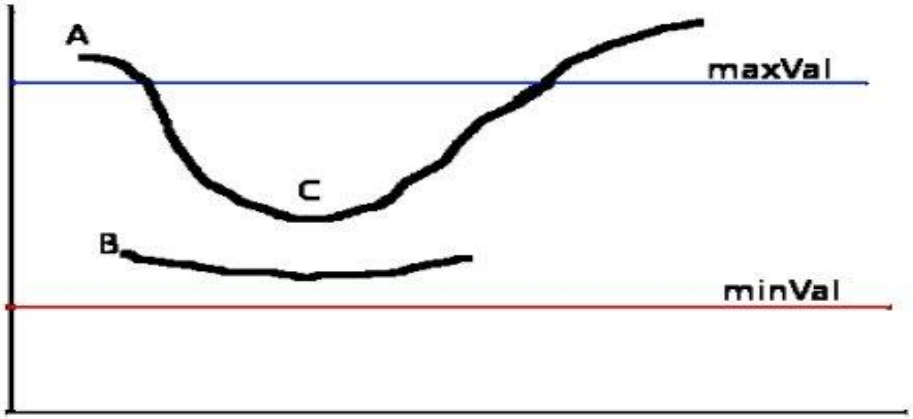


Hình 3.4. *Hình mô tả quá trình NMS*

Tương tự cho 2 trường hợp hướng gradient còn lại. Kết thúc bước này ta được một mặt nạ nhị phân (ảnh nhị phân).

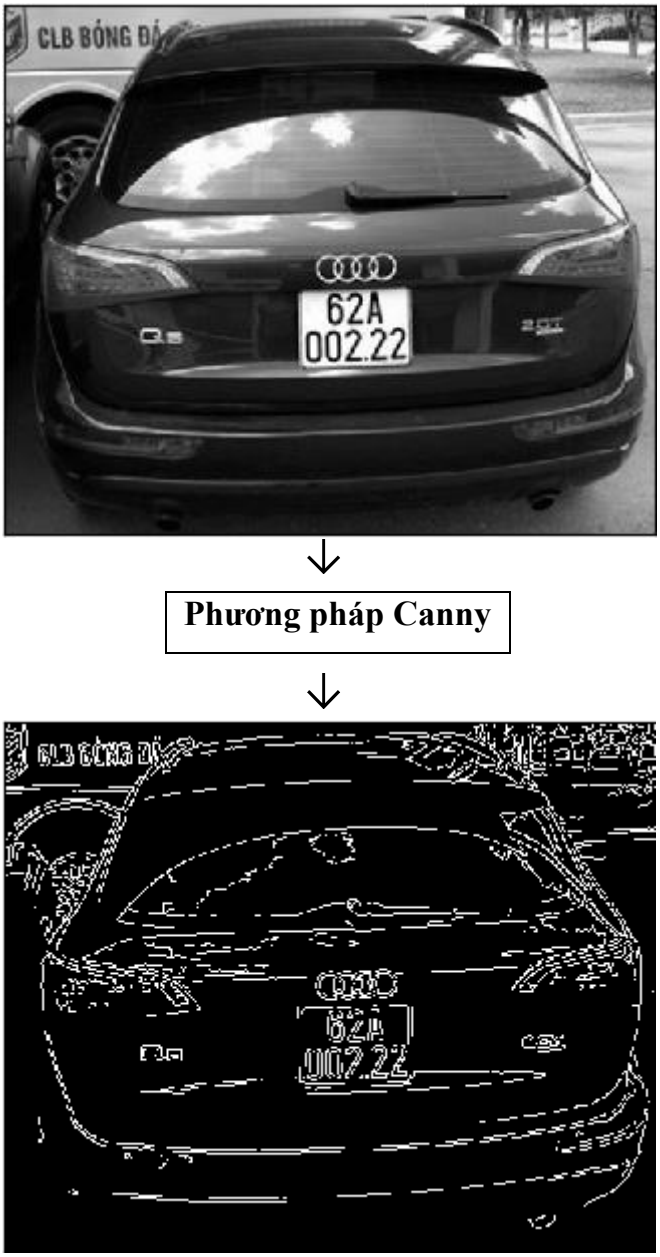
d. Bước 4: Lọc ngưỡng

Lọc ngưỡng: ta sẽ xét các pixel dương trên mặt nạ nhị phân kết quả của bước trước. Nếu giá trị gradient vượt ngưỡng **maxVal** thì pixel đó chắc chắn là cạnh. Các pixel có độ lớn gradient nhỏ hơn ngưỡng **minVal** sẽ bị loại bỏ. Còn các pixel nằm trong khoảng 2 ngưỡng trên sẽ được xem xét rằng nó có nằm liên 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. Sau bước này ta có thể áp dụng thêm bước hậu xử lý loại bỏ nhiễu (tức những pixel cạnh rời rạc hay cạnh ngắn) nếu muốn.



Hình 3.5. *Hình minh họa về ngưỡng lọc*

3.2.4. Kết quả sau khi sử dụng phương pháp Canny



Hình 3.6. *Kết quả khi sử dụng phương pháp Canny*

Vì phương pháp Canny này đã tích hợp trong thư viện LaptrinhVBLibs, chúng ta gọi ra và thực hiện. Sau khi có được ảnh ở dạng nhị phân, ta dùng tọa độ để cắt ảnh. Khi phát hiện cạnh, sẽ cung cấp luôn tọa đọ của những cạnh đó và từ đó vẽ được hình tứ giác ở đây sẽ là hình vuông hoặc hình chữ nhật chứa ảnh đó. Được lưu vào đối tượng Rectangle. Dùng hàm Bitmap ch = grayframe.Clone(up[i],grayframe.PixelFormat) cắt

ra convert về một bitmap. Ta tách được biển số ra khỏi ảnh. Tiếp theo, dùng công cụ Tesseract OCR để nhận dạng.

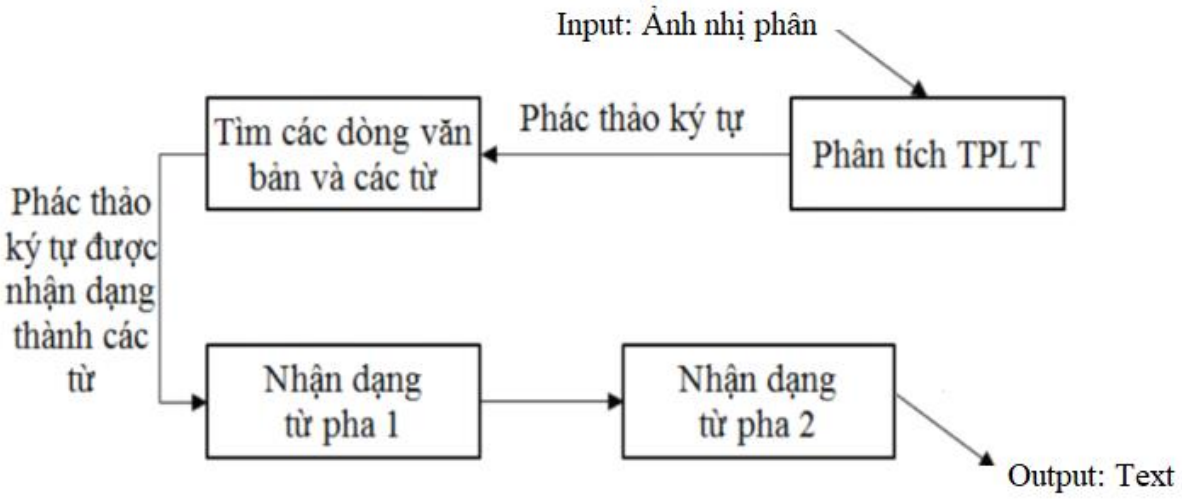
Ta được kết quả như sau:

Hình 3.7. *Kết quả khi tách biển số*

3.3. Công cụ nhận dạng ký tự Tesseract OCR

Để nhận dạng được các ký tự sau khi đã tách được biển số ra khỏi ảnh gốc, ta sử dụng công cụ nhận dạng **Tesseract OCR**, một OCR hàng đầu hiện nay.

Cơ chế hoạt động của Tesseract OCR[3]



Hình 3.8. *Cấu trúc của Tesseract*

Với cơ chế như thế nào mà Tesseract có thể mang đến sự hiệu quả cũng như được sử dụng khá nhiều trong việc nhận dạng ký tự như hiện nay. Về cơ bản, quá trình nhận diện sẽ diễn ra từng bước trải qua bốn bước chính như phân tích layout, tìm kiếm dòng, tìm kiếm ký tự, nhận diện ký tự và chỉnh sửa kết quả.

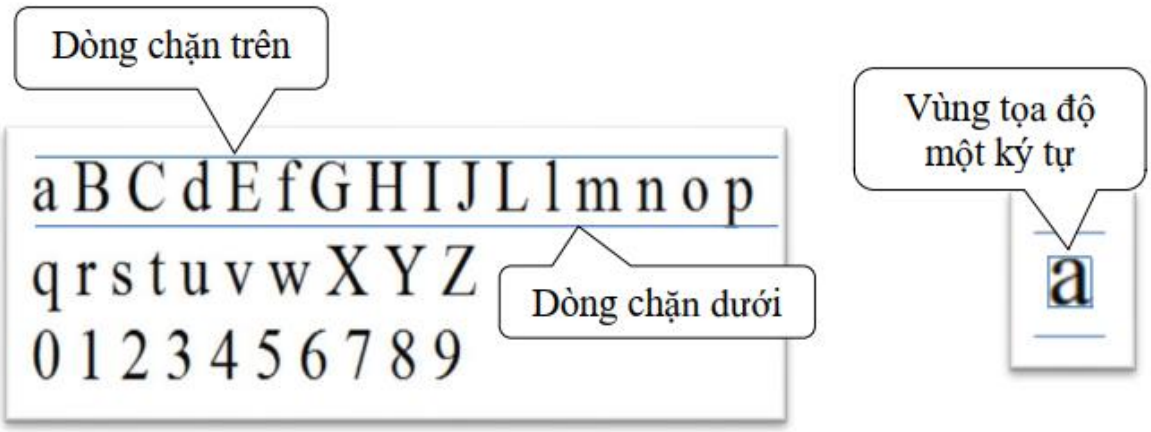
3.3.1. Phân tích layout

Trước tiên, hình ảnh sẽ được phân tích để tìm ra các vùng kết nối (connected component). Bước này cho phép OCR dễ dàng nhận biết những vùng ký tự ngược để có thể nhận diện những ký tự bên trong. Trong Tesseract, những vùng chứa ký tự này được gọi là Blob.

3.3.2. Tìm kiếm dòng và ký tự

Để nhận dạng, ta phải nhận dạng được các dòng và các ký tự trong ảnh. Sau khi qua tiền xử lý, dựa trên các đặc trưng văn bản sẽ tách được các dòng và từ, ký tự trong ảnh. Mỗi dòng văn bản luôn có tọa độ chặn dưới và chặn trên, trong khi đó mỗi ký tự có tọa độ chặn dưới, chặn trên, giới hạn trái và giới hạn phải.

Giải thuật xác định tọa độ chặn trên và dưới của mỗi dòng văn bản được mô tả tóm tắt như sau: Từ tọa độ ban đầu (0,0), duyệt theo chiều ngang để tìm điểm có màu đen, nếu đã hết dòng mà vẫn chưa thấy thì bắt đầu duyệt lại từ đầu dòng kế tiếp. Nếu tìm thấy điểm có màu đen thì ghi nhận tung độ điểm đó là tung độ của dòng chặn trên và dừng duyệt. Tương tự với tọa độ chặn dưới, xuất phát từ điểm có hoành độ là 0 và tung độ bằng tung độ của dòng chặn trên, cũng duyệt theo chiều ngang, nếu sau hết dòng không thấy điểm đen nào thì ghi nhận tung độ dòng đang xét là tung độ của dòng chặn dưới, còn nếu tìm thấy điểm đen thì lại xét lại từ dòng kế tiếp



Hình 3.9. *Tách dòng cơ sở và xác chọn vùng ký tự*

Lặp lại các bước trên để tìm tọa độ chặn trên và chặn dưới cho dòng còn lại**.**

Giải thuật xác định vùng tọa độ cho mỗi ký tự như sau: Có được tọa độ giới hạn mỗi dòng, xác định được tọa độ chặn dưới và chặn trên của mỗi ký tự trên dòng đó. Trong khi đó, để tìm tọa độ giới hạn trái và phải của một ký tự, bắt đầu từ đầu dòng chặn trên, duyệt theo chiều dọc tới tung độ của dòng chặn dưới, nếu gặp điểm màu đen thì ghi nhận hoành độ điểm đó là hoàng độ của cột giới hạn trái, nếu không thấy điểm màu đen thì tiếp tục lại từ đầu cột kế tiếp. Tương tự với tọa độ giới hạn phải, bắt đầu từ đầu dòng chặn trên, duyệt theo chiều dọc tới tung độ của dòng chặn dưới, nếu sau hết cột không

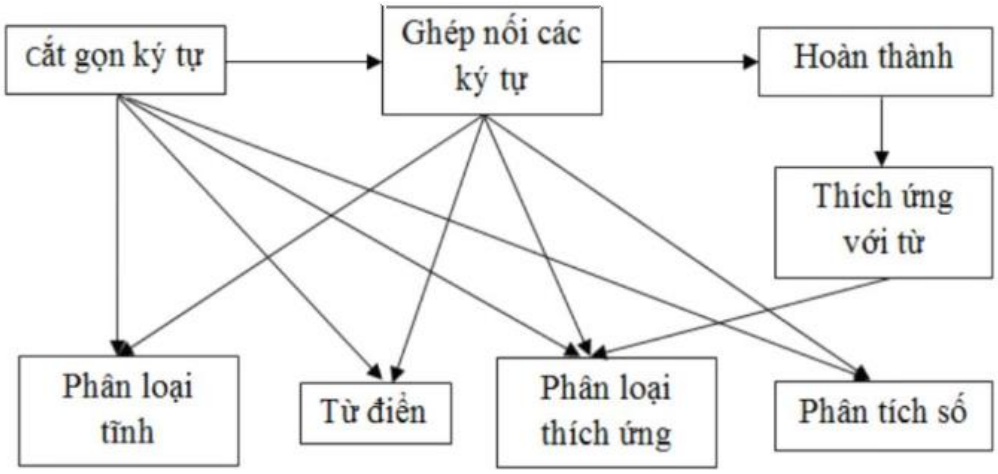
tìm thấy điểm màu đen thì hoành độ cột đang xét là hoành độ cột giới hạn phải của ký tự, nếu tìm thấy thì tiếp tục lại từ đầu cột kế tiếp.

3.3.3. Nhận dạng từ

Sau khi đã xác định được các dòng ký tự cùng các đối số tương ứng, dòng ký tự sẽ được chia nhỏ thành các từ dựa vào các ký tự phân cách. Lúc này, văn bản cố định sẽ được chia nhỏ và tiến hành nhận diện. Trong khi đó, văn bản không cố định hoặc chưa chắc chắn thì sẽ được chia nhỏ thành các từ dù chưa chắc chắn. Nhưng nhờ vào bước nhận diện, chúng ta sẽ thu được kết quả cuối cùng chính xác hơn.

Bước vào quá trình nhận diện, input của chúng ta sẽ được đánh giá, phân tích hai lần. Ở lần đầu tiên, OCR sẽ nhận diện ký tự với kết quả phân tích ở bước trước đó. Các kết quả nhận diện thoả mãn yêu cầu sẽ được đưa vào tập tin huấn luyện để hỗ trợ cho quá trình nhận diện lần thứ hai với các kết quả chưa đạt yêu cầu. Đương nhiên, việc xác nhận kết quả có thoả mãn yêu cầu hay không cần phải dựa trên nhiều tiêu chí vì nhận diện nội dung phải trải qua một quá trình lặp đi lặp lại gồm các bước nhận diện ký tự, ghép ký tự và so khớp với từ điển. Các tiêu chí đó bao gồm khoảng cách của các ký tự, độ phù hợp với từ điển và khoảng cách đến các dấu câu.

Sơ đồ nhận dạng một từ là quy trình phân tích một từ được chia ra thành các ký tự:



Hình 3.10. *Quá trình nhận dạng từ*

Mỗi ký tự cần nhận dạng có những đặc trưng riêng, có khoảng 50 tới 100 đặc trưng điển hình trong mỗi ký tự. Mỗi đặc trưng chứa 3 tham số là hoành độ, tung độ, và góc xoay. Trong khi đó mỗi ký tự mẫu có từ 10 tới 20 đặc trưng, mỗi đặc trưng có 4 tham số là hoành độ, tung độ, góc xoay, độ dài.

Văn bản luôn tồn tại độ dư thừa ký tự và từ vựng, vậy chức năng phân loại ký tự tạo ra danh sách rút gọn chứa các ký tự mà ký tự đối sánh có thể trùng khớp. Các lớp ký tự mẫu sinh ra các lớp véc tơ bít tương ứng với những đặc trưng của từng ký tự.

Những đặc trưng của ký tự nhận dạng (Features of character) được đối sánh với lớp véc tơ bít của ký tự mẫu, và tính toán sự khác nhau giữa các đặc trưng của chúng. Bên cạnh đó có tham số thứ hai là độ dài của ký tự nhận dạng.

Hệ số đánh giá đối sánh là tích hai tham số trên, cặp đối sánh nào có hệ số nhỏ nhất thì xem như chúng là tương tự nhau.

Chức năng phân loại tĩnh (static classifier) phù hợp với các ký tự có phông chữ bất kỳ, nhưng nó chủ yếu được dùng để nhận dạng các ký tự riêng như ký tự chú giải, dấu ngăn cách hay kết thúc câu... trong khi chức năng phân loại thích ứng (Adaptive classifier) dùng để nhận dạng các ký tự theo phông chữ chuẩn.

Bộ từ điển dùng để lưu trữ dữ liệu cho quá trình phân loại và nhận dạng. Mỗi ngôn ngữ có một bộ từ điển chứa các ký tự theo các phông chữ khác nhau với thuộc tính như chuẩn – normal , đậm – bold, nghiêng – italic và thuộc tính kết hợp. Từ điển cũng lưu trữ các từ hay sử dụng, từ chữ cái, từ số, từ chữ hoa, từ chữ thường.

Cuối cùng, OCR sẽ xử lý những dấu cách không rõ ràng cùng với xem xét các giả thiết khác cho việc định vị những ký tự in hoa nhỏ để đi đến kết quả cuối cùng.

3.3.4. Kết quả sau khi sử dụng công cụ Tesseract OCR

Hình 3.11. *Hình gốc*

Hình 3.13. *Xử lý để nhận dạng*

Hình 3.12. *Những số nhân dạng được*

CHƯƠNG 4: PHÂN TÍCH THIẾT KẾ ỨNG DỤNG

4.1. Đặc tả và phân tích yêu cầu

4.1.1. Bài toán

Bài toán mà chúng là xây dựng ứng dụng phát hiện biển số xe và đọc được biển số cho bãi gửi xe sử dụng bộ thư viện xử lí ảnh OpenCV, phương pháp Canny để xử lí ảnh và nhận dạng ký tự quang học Tesseract OCR.

Chương trình có đầy đủ tính năng như: nhận dạng biển số xe, tính thời gian gửi khi xe ra khỏi bãi, thống kê số lượt xe gửi trong ngày,…và phải đảm bảo độ chính xác cao trong nhận dạng, tốc độ quét số nhanh, số lần sai không quá nhiều.

4.1.2. Yêu cầu bài toán

Để thể quản lý bãi giữ xe thì ứng dụng quản lý bãi giữ xe phải nắm rõ thuật toán dùng để nhận dạng ký tự số được sử dụng trong bài, có nghiên cứu thêm một số thuật toán nhận dạng ký tự số khác sau đó đưa ra so sánh và nhận định. Từ đó có thể áp dụng thuật toán nào là tốt nhất để giải quyết bài toán của mình. Ngoài ra ta cần phải nắm rõ một số kỹ thuật trong xử lý hình ảnh để giải quyết một số vấn đề khó khăn khi gặp phải trong bài toán. Khi xây dựng chương trình, yêu cầu sự chính xác cao, tốc độ của thuật toán phải nhanh.

4.1.3. Yêu cầu hệ thống

- **Yêu cầu về chức năng**

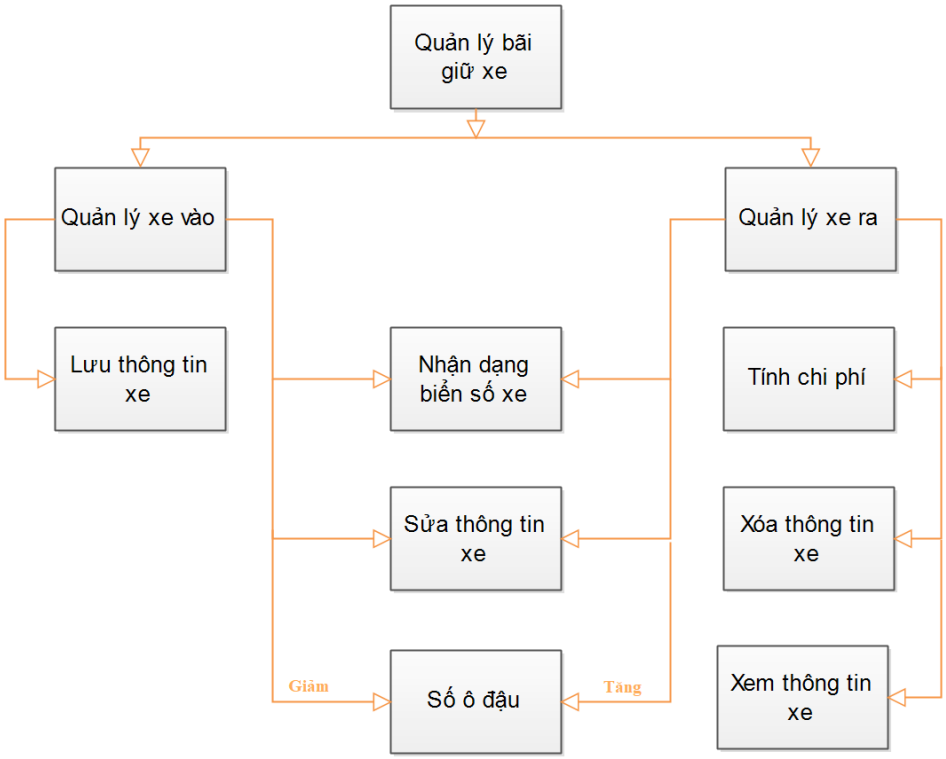
  - Chức năng bắt đầu và thoát.
  - Chức năng nhận dạng.
  - Chức năng chỉnh sửa biển số khi gặp sai số.
  - Tính thời gian và chi phí.
  - Thông báo tình trạng chỗ trống trong bãi giữ xe.

- **Yêu cầu phi chức năng**

  - Giao diện và bố cục phải trình bày sao cho người dùng dễ thao tác nhất.
  - Tốc độ xử lý phải nhanh, không bắt người dùng chờ quá lâu.

4.2. Xây dựng hệ thống

4.2.1. Các chức năng của hệ thống



Hình 4.1. *Sơ đồ chức năng*

- Quản lý xe vào

    + Nhận dạng biển số.

    + Sử thông tin biển số.

    + Lưu thông tin biển số.

    + Giảm số ô đậu.

- Quản lý xe ra

    + Nhận dạng biển số.

    + Sửa thông tin biển số.

    + Tính chi phí.

    + Xóa thông tin biển số.

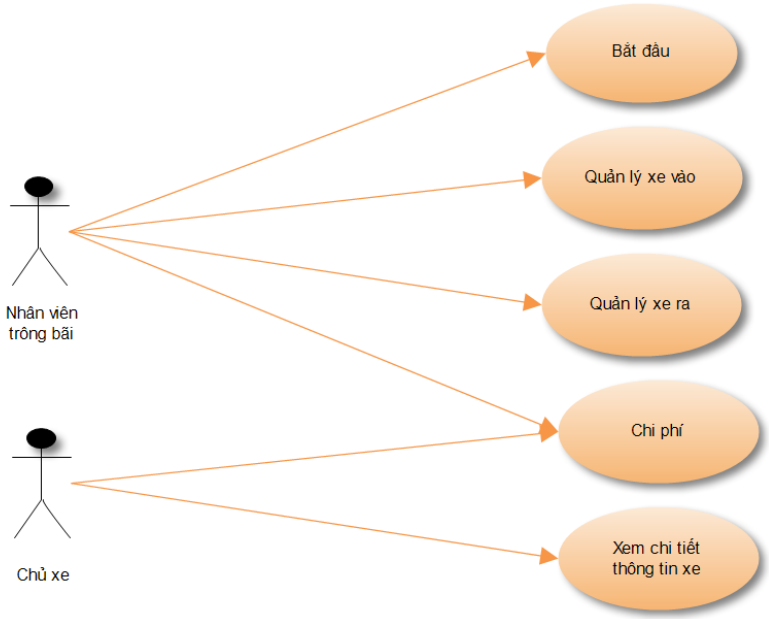
    + Tăng số ô đậu.

4.2.2. Xây dựng lược đồ Usecase

4.2.2.1. Các tác nhân tham gia

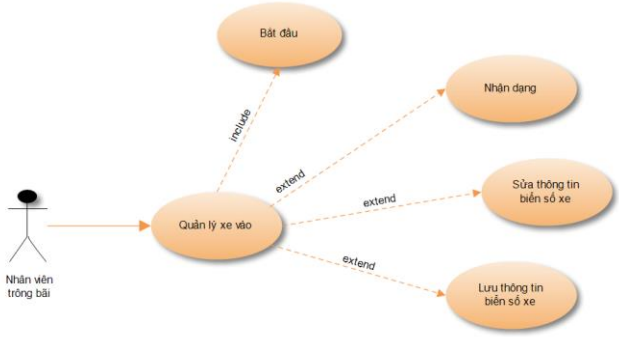
Đối tượng	Ý nghĩa
Nhân viên trông bãi	Là người được chủ xe thuê để trông coi bãi, là tác nhân được ủy quyền có quyền hạn cao nhất trong hệ thống. Nhân viên cho xe vào, nhận dạng, quản lý thông tin xe, thu phí.
Chủ xe	Chạy xe vào gửi và trả phí khi xe ra.

4.2.2.2. Usecase tổng quát

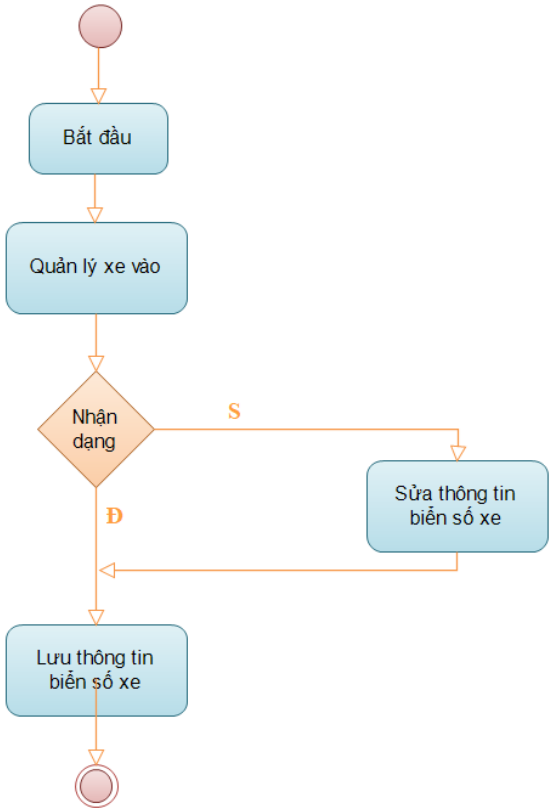


Hình 4.2. *Usecase tổng quát*

4.2.2.3. Usecase quản lý xe vào



Hình 4.3. *Usecase quản lý xe vào*

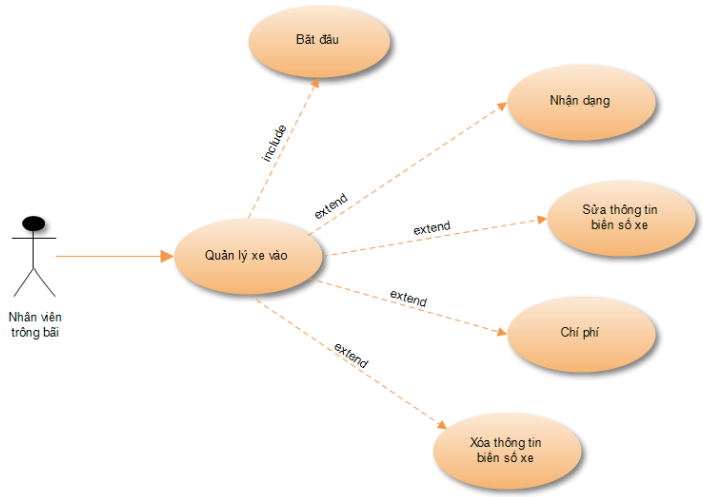


Hình 4.4. *Sơ đồ hoạt động quản lý xe vào*

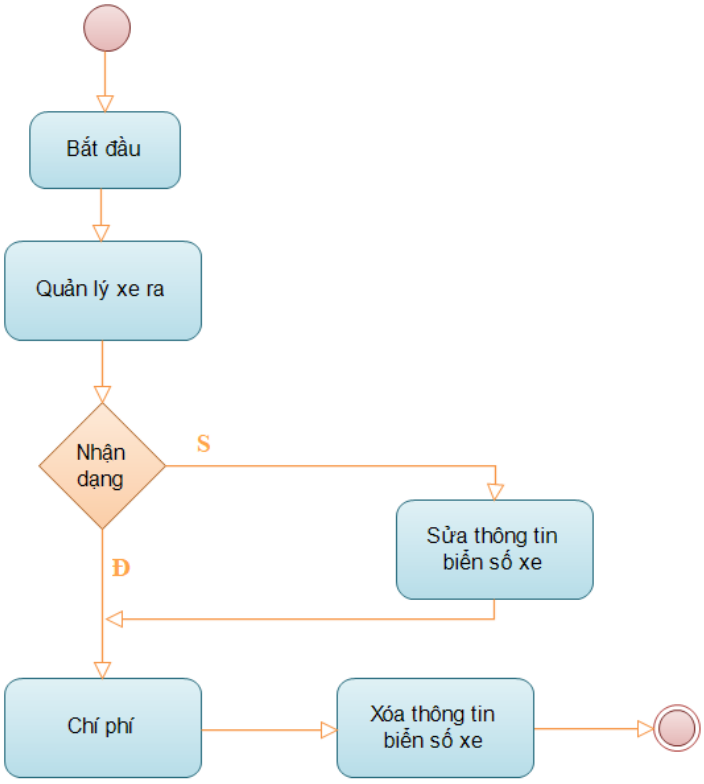
Đặt tả usecase quản lý xe vào

Tên usecase	Quản lý phòng
Tác nhân chính	Nhân viên trông bãi
Mức	1
Người chịu trách nhiệm	Nhân viên trông bãi
Điều kiên tiên quyết	Bắt đầu thành công
Đảm bảo tối thiểu	Bắt đầu thành công
Đảm bảo thành công	Hiển thị giao diện quản lý
Kích hoạt	Chọn các chức năng bên trái giao diện.

4.2.2.4. Usecase quản lý xe ra



Hình 4.5. *Usecase quản lý xe vào*

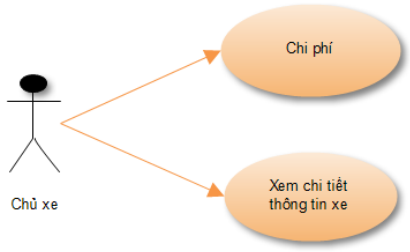


Hình 4.6. *Sơ đồ hoạt động quản lý xe ra*

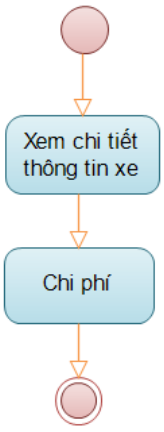
Đặc tả usecase quản lý người thuê

Tên usecase	Quản lý xe ra
Tác nhân chính	Nhân viên trông bãi
Mức	1
Người chịu trách nhiệm	Nhân viên trông bãi
Điều kiên tiên quyết	Bắt đầu thành công
Đảm bảo tối thiểu	Có xe trong bãi
Đảm bảo thành công	Hiển thị giao diện quản lý
Kích hoạt	Chọn các chức năng bên phải giao diện

4.2.2.5. Usecase chi phí



Hình 4.7. *Usecase chi phí*

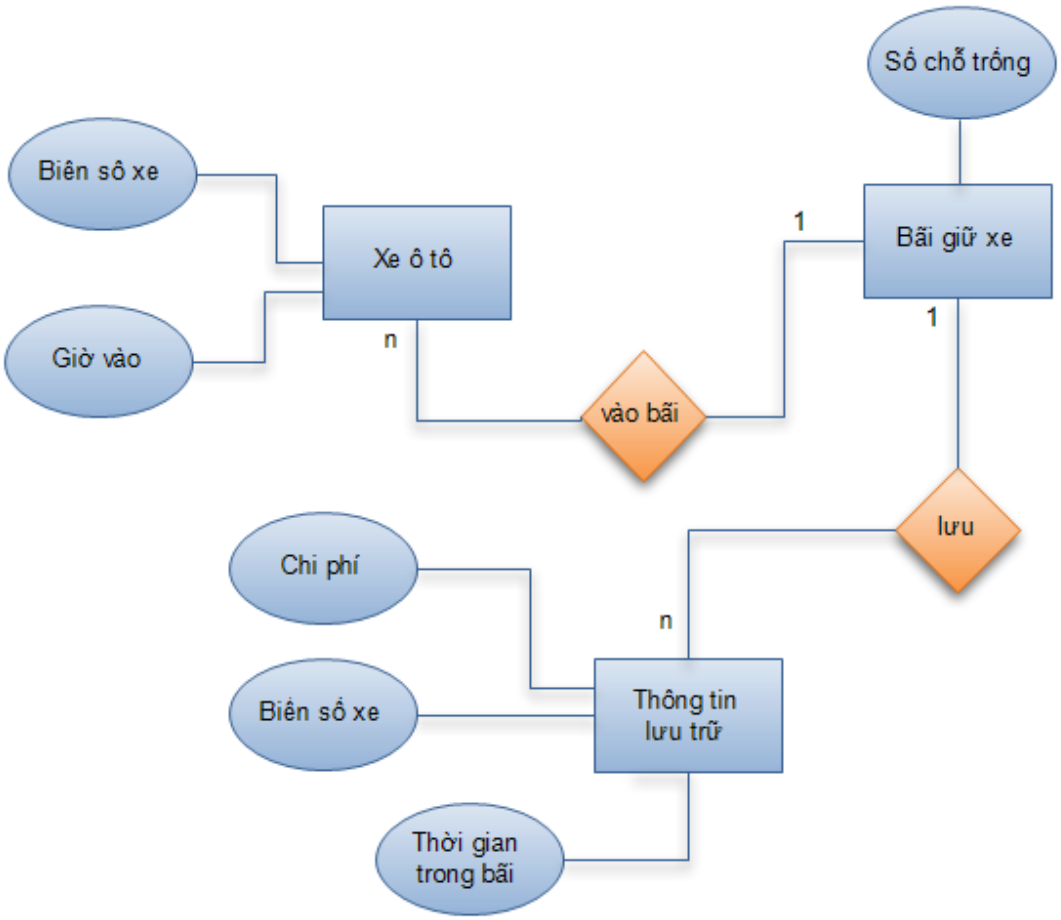


Hình 4.8. *Sơ đồ hoạt động chi phí*

Đặc tả usecase tra cứu

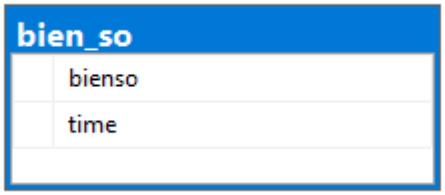
Tên usecase	Quản lý thống kê
Tác nhân chính	Nhân viên trông vãi và chủ xe
Mức	1
Người chịu trách nhiệm	Nhân viên trong bãi
Điều kiên tiên quyết	Có thông tin xe trong bãi
Đảm bảo tối thiểu	Thông tin đúng với xe ra
Đảm bảo thành công	Hiển thị thông tin xe ra và mức phí
Kích hoạt	Chọn chức năng thanh toán.

4.3. Mô hình ERD



Hình 4.9. *Mô hình ERD*

4.4. Cơ sở dữ liệu cần



Hình 4.10. *Cơ sở dữ liệu*

CHƯƠNG 5 : CÀI ĐẶT VÀ THỰC NGHIỆM

5.1. Cài đặt phần mềm

5.1.1. Công cụ cài đặt

Để cài đặt được ứng dụng, ta sử dụng các công cụ lập trình:

Phần mềm Visual Studio:

Microsoft Visual Studio[5] là một môi trường phát triển tích hợp (IDE) từ Microsoft. Nó được sử dụng để phát triển chương trình máy tính cho Microsoft Windows, cũng như các trang web, các ứng dụng web và các dịch vụ web. Visual Studio sử dụng nền tảng phát triển phần mềm của Microsoft như Windows API, Windows Forms, Windows Presentation Foundation, Windows Store và Microsoft Silverlight. Nó có thể sản xuất cả hai ngôn ngữ máy và mã số quản lý.

Visual Studio bao gồm một trình soạn thảo mã hỗ trợ IntelliSense cũng như cải tiến mã nguồn. Trình gỡ lỗi tích hợp hoạt động cả về trình gỡ lỗi mức độ mã nguồn và gỡ lỗi mức độ máy. Công cụ tích hợp khác bao gồm một mẫu thiết kế các hình thức xây dựng giao diện ứng dụng, thiết kế web, thiết kế lớp và thiết kế giản đồ cơ sở dữ liệu. Nó chấp nhận các plug-in nâng cao các chức năng ở hầu hết các cấp bao gồm thêm hỗ trợ cho các hệ thống quản lý phiên bản (như Subversion) và bổ sung thêm bộ công cụ mới như biên tập và thiết kế trực quan cho các miền ngôn ngữ cụ thể hoặc bộ công cụ dành cho các khía cạnh khác trong quy trình phát triển phần mềm.

Visual Studio hỗ trợ nhiều ngôn ngữ lập trình khác nhau và cho phép trình biên tập mã và gỡ lỗi để hỗ trợ (mức độ khác nhau) hầu như mọi ngôn ngữ lập trình. Các ngôn ngữ tích hợp gồm có C, C++ và C++/CLI (thông qua Visual C++),VB.NET (thông qua Visual Basic.NET), C# (thông qua Visual C#) và F# (như của Visual Studio 2010). Hỗ trợ cho các ngôn ngữ khác như J++/J#, Python và Ruby thông qua dịch vụ cài đặt riêng lẻ. Nó cũng hỗ trợ XML/SSLT, HTML/XHTML, JavaScript và CSS.

Microsoft cung cấp phiên bản "Express" (đối với phiên bản Visual Studio 2013 trở về trước) và "Community" (đối với bản Visual Studio 2015 trở về sau) là phiên bản miễn phí của Visual Studio.

SQL Server[6]: SQL Server là một hệ thống quản lý cơ sở dữ liệu quan hệ, (Relational Database Management System viết tắt là RDBMS), hỗ trợ một số lượng lớn các quy trình xử lý giao dịch, ứng dụng doanh nghiệp và ứng dụng phân tích trong các công ty IT. SQL Server là một trong 3 công nghệ dữ liệu dẫn đầu hiện này cùng với Oracle Database và IBM's DB2.

Cũng giống như các phần mềm RDBMS khác, Microsoft SQL Server được xây dựng bên trên lớp SQL - ngôn ngữ lập trình tiêu chuẩn hóa mà quản trị viên cơ sở dữ liệu (DBAs) và các chuyên gia CNTT sử dụng để quản lý cơ sở dữ liệu và truy vấn dữ liệu nằm bên trong. SQL Server thường gắn với Transact-SQL (T-SQL), một cài đặt SQL của Microsoft bổ sung một bộ chương trình mở rộng ngôn ngữ lập trình chuẩn.

Bộ thư viện xử lý hình ảnh OpenCV:

OpenCV[7] (Open Source Computer Vision Library) là một mã nguồn mở thị giác máy tính và thư viện phần mềm máy học. OpenCV được xây dựng để cung cấp một cơ sở hạ tầng chung cho các ứng dụng máy tính và đẩy nhanh việc sử dụng nhận thức của máy móc trong các sản phẩm thương mại. Là sản phẩm được cấp phép BSD, OpenCV giúp các doanh nghiệp dễ dàng sử dụng và sửa đổi mã.

Thư viện có hơn 2500 thuật toán được tối ưu hóa, trong đó bao gồm một bộ toàn diện của thuật toán máy tính và cổ điển hiện đại và máy học. Các thuật toán này có thể được sử dụng để phát hiện và nhận dạng khuôn mặt, xác định các đối tượng, phân loại hành động của con người trong video, theo dõi các chuyển động của máy ảnh, theo dõi các đối tượng di chuyển, giải phóng các mô hình 3D của các đối tượng, tạo các điểm mây 3D từ các camera âm thanh nổi, ghép các hình ảnh với nhau để tạo ra độ phân giải cao hình ảnh của toàn bộ cảnh, tìm hình ảnh tương tự từ cơ sở dữ liệu hình ảnh, loại bỏ mắt đỏ khỏi những bức ảnh chụp bằng flash, theo dõi chuyển động của mắt, nhận diện cảnh quan và thiết lập dấu hiệu để phủ lên nó với sự hiện thực tăng cường,… OpenCV có hơn 47 nghìn người sử dụng và số lượng tải ước tính trên 14 triệu. Thư viện được sử dụng rộng rãi trong các công ty, nhóm nghiên cứu và các cơ quan chính phủ.

Nó có giao diện C++, C, Python, Java và MATLAB và hỗ trợ Windows, Linux, Android và Mac OS. OpenCV chủ yếu hướng tới các ứng dụng thị giác thời gian thực và tận dụng các hướng dẫn của MMX và SSE khi có. Một giao diện CUDA và OpenCL đầy đủ tính năng đang được tích cực phát triển ngay bây giờ. Có hơn 500 thuật toán và khoảng 10 lần nhiều chức năng soạn hoặc hỗ trợ các thuật toán đó. OpenCV được viết tự nhiên trong C + + và có một giao diện sẵn mà làm việc liên tục với thành phần.

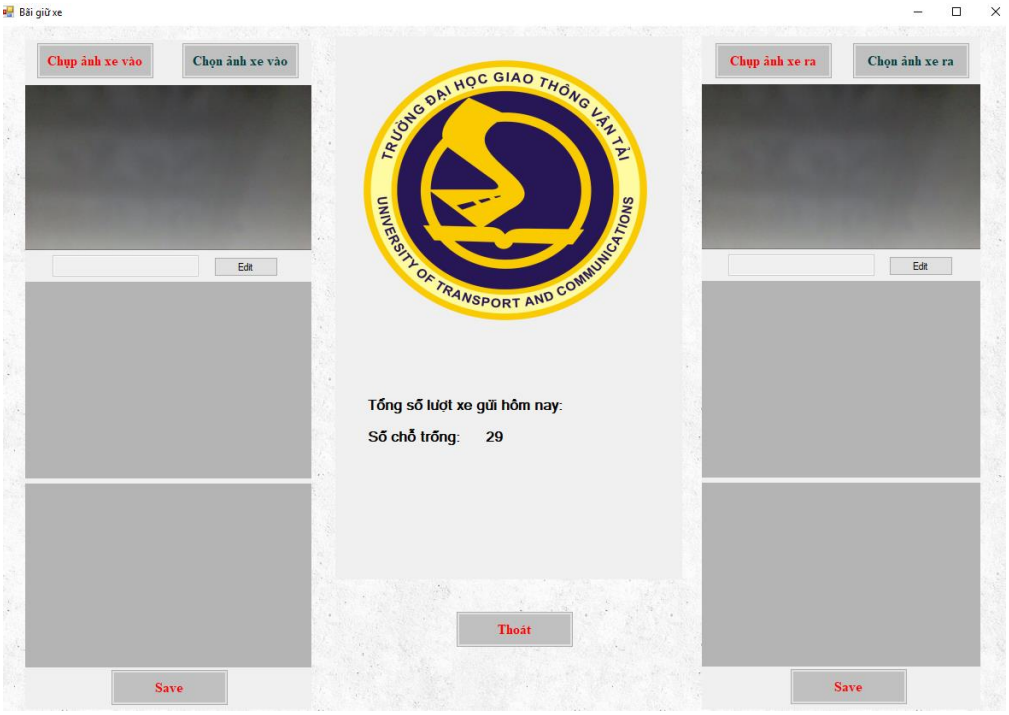
5.1.2. Thiết kế giao diện

Phần mềm có giao diện như sau:

Đây là giao diện đăng nhập sau khi mở phần mềm, bạn cần đăng nhập để truy cập vào hệ thống quản lý.

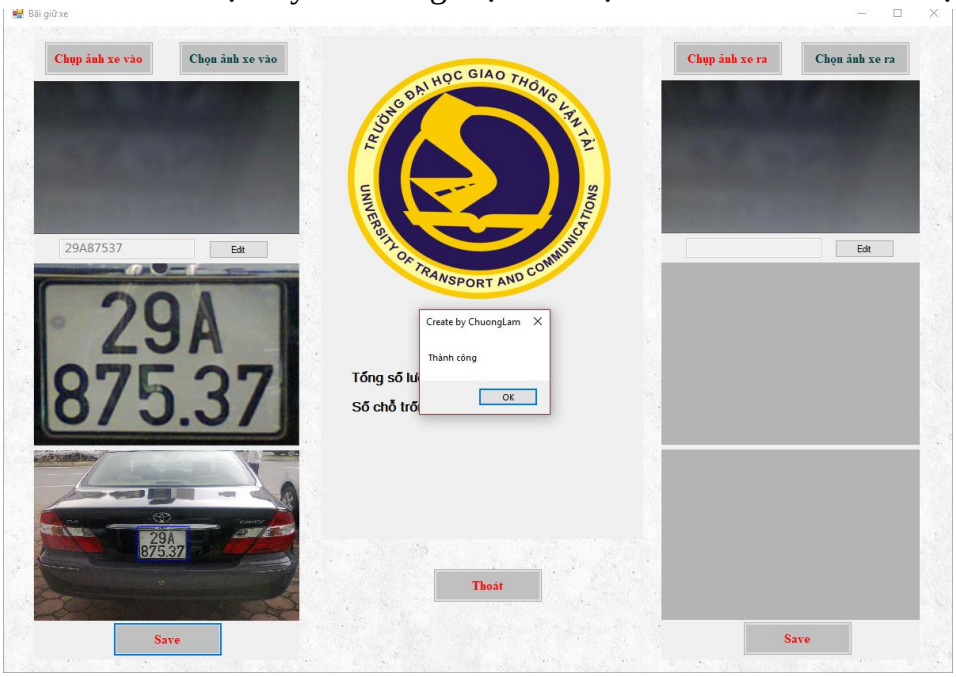
Hình 5.1. *Màn hình đăng nhập*

Sau khi click chọn vào button “Bắt đầu” (Hình 5.), giao diện chính của chương trình se xuất hiện.



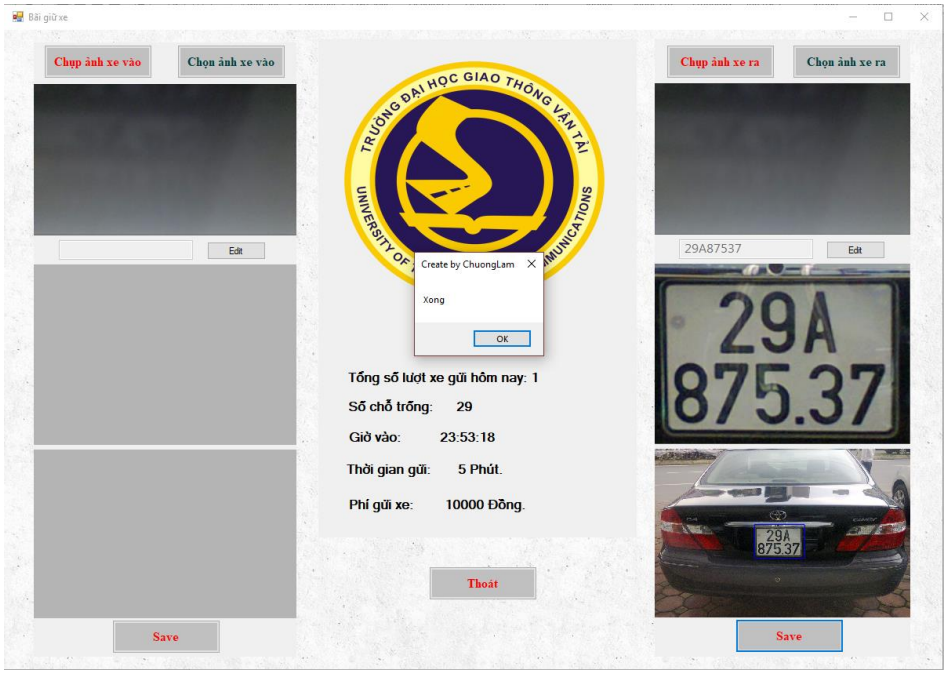
Hình 5.2. *Giao diện chính của chương trình*

Vào tới giao diện chính, ta có thể thấy 2 khung ảnh ở trên cùng, đây là nơi chụp ảnh từ camera hoặc lấy ảnh trong một thư mục bất kì để tiến hành nhận dạng.



Hình 5.3. *Chọn ảnh từ thư mục bất kì để nhận dạng và lưu trữ*

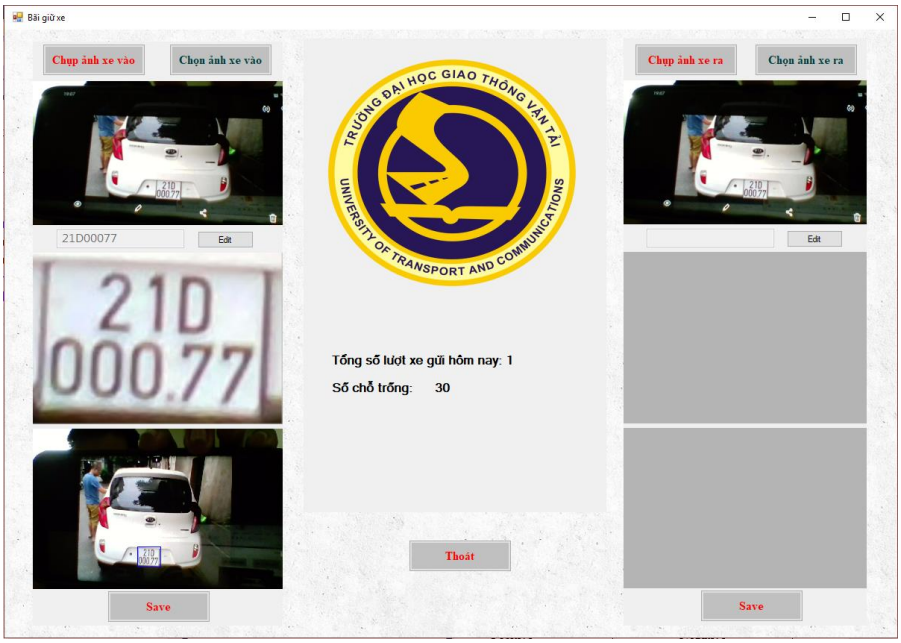
Hình 5.3. Mô tả việc chọn một ảnh từ thư mục bất kì để nhận dạng và lưu trữ thông tin trong cơ sở dữ liệu bao gồm biển số xe và giờ vào. Trường hợp sai số, ta có thể sửa đổi khi nhấn vào button “Edit”.



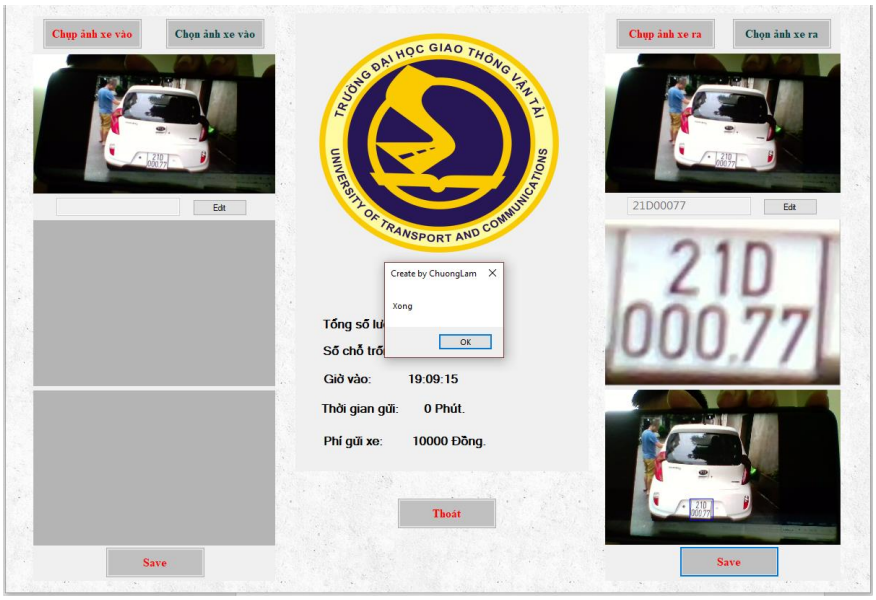
Hình 5.4. *Nhận dạng, tính chí và xóa thông tin xe*

Hình 5.4. Mô tả việc lấy một ảnh để nhận dạng, so sánh với thông tin xe ô tô đã được lưu trong cơ sở dữ liệu. Sau khi so sánh, nếu đúng chủ xe thanh toán chi phí và cho xe ra, ngược lại hệ thống sẽ xuất thông báo xe không có trong bãi.

Tương tự với việc sử dụng camera để chụp ảnh, được thể hiện ở Hình 5.5. Và Hình 5.6.



Hình 5.6. *Xử dụng camera cho xe vào*



Hình 5.5. *Xử dụng camera cho xe ra*

5.2. Đánh giá và kiểm thử

5.2.1. Cài đặt chương trình hoàn thiện

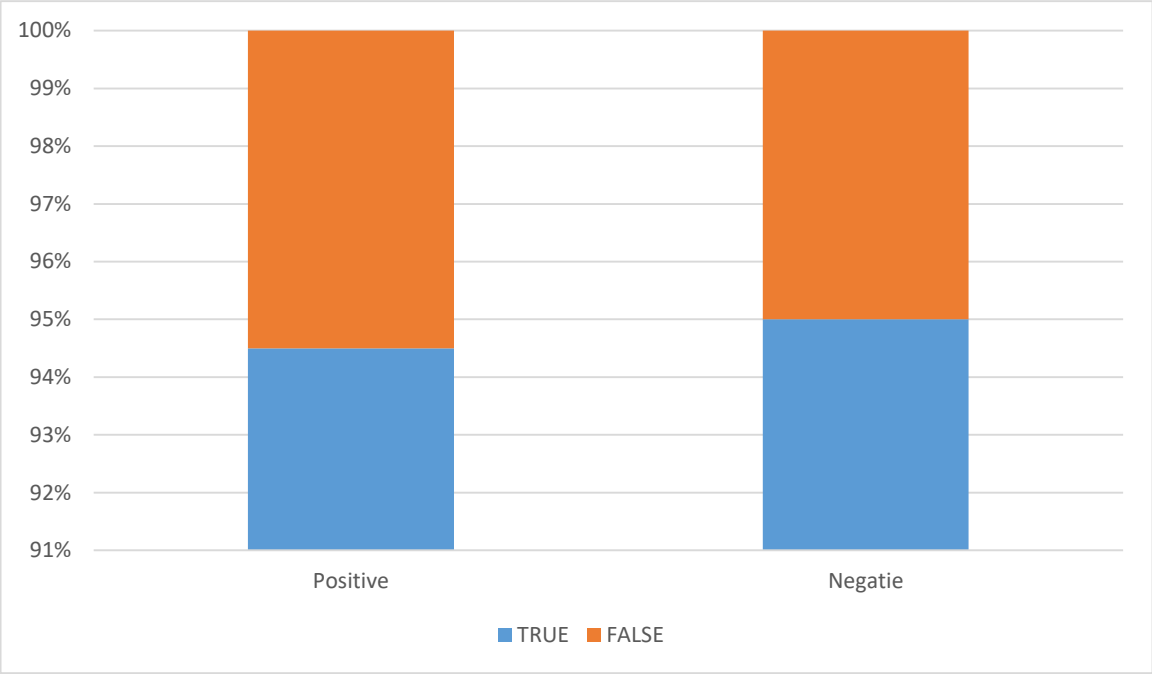
5.2.2. Đánh giá số liệu và kết luận

Cơ sở dữ liệu test bao gồm 25 xe ô tô.

Kết quả thực nghiệm 200 số (trích ra từ 25 xe ô tô) tất cả kế quả như sau:

	Positive	Negative
True	94.5%	95%
False	5.5%	5%

Kết quả trình bày dưới dạng biểu đồ:



Biểu đồ 5.1. *Kết quả nhận dạng*

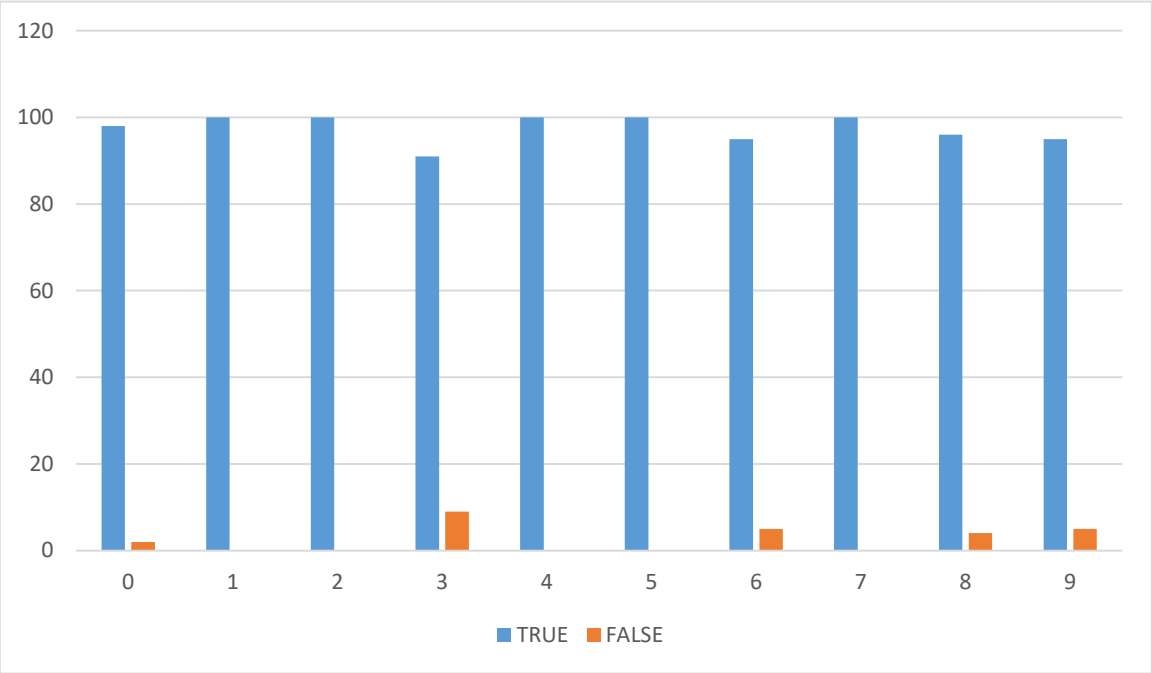
Trong đó có một số trường hợp Negative nhận dạng nhầm lẫn là giữa các số: 9 và 8, 6 và 5, 0 và 6, 0 và 9, 3 và 1, B và D, D và P. Các trường hợp nhầm lẫn này là do các cặp số có đặc trưng gần giống nhau, cặp số 3 và 1 trong quá trình tiền xử lý và thu thập dữ liệu số 1 hơi phình to ra, sau khi tính toán đặc trưng thì sẽ dễ nhầm lẫn với số 3, và đối với cặp 5 và 6 thì do số 5 sát cạnh gần biên dễ lầm lẫn thành số 6 khi tính toán đặc trưng. Điều này sẽ là mục tiêu phát triển trong tương lai, điều chỉnh tập huấn luyện và tiền xử lý để có kết quả tốt hơn.

Trong các lần nhận dạng thì kết quả nhận dạng của từng số là khác sau:

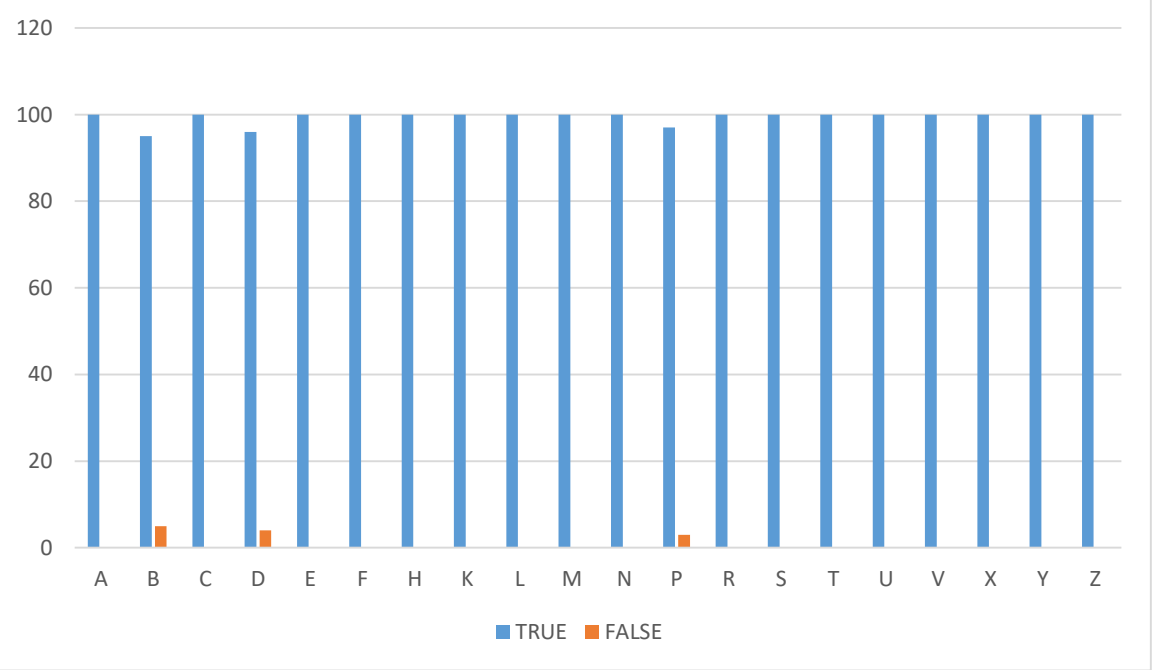
	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
True	98%	100%	100%	91%	100%	100%	95%	100%	96%	95%
False	2%	0%	0%	9%	0%	0%	5%	0%	4%	5%

	A	**B**	**C**	**D**	**E**	**F**	**H**	**K**	**L**	**M**
True	100%	95%	100%	96%	100%	100%	100%	100%	100%	100%
False	0%	5%	0%	4%	0%	0%	0%	0%	0%	0%
	N	**P**	**R**	**S**	**T**	**U**	**V**	**X**	**Y**	**Z**
True	100%	97%	100%	100%	100%	100%	100%	100%	100%	100%
False	0%	3%	0%	0%	0%	0%	0%	0%	0%	0%

Kết quả dạng biểu đồ:



Biểu đồ 5.3. *Biểu đồ nhận dạng cho ký tự số*



Biểu đồ 5.2. *Biểu đồ nhận dạng cho ký tự chữ*

Dựa vào các kết quả cho ta thấy, phương pháp Canny và nhận dạng ký quang học Tesseract OCR nhận dạng khá chính xác, thích hợp cho việc phân lớp để nhận dạng. Nhưng bên cạnh đó cũng có một số vấn đề cần cải thiện để có kết quả tốt hơn. Trong các ký tự thuật toán nhận dạng thì số 0, 3,6, 8, 9, B, D, P nhận dạng sai nhiều, do bài toán

tính toán dựa trên đặc trưng Pixel và số lượng đặc trưng còn ít nên tỉ lệ các số nhầm lẫn nhau vẫn còn. Các chữ số như 0, 9, 6 hay 6, 8 và 8, 9 hoặc 2, 7 có các nét gần như nhau, bên cạnh đó do điều kiện môi trường, camera không chuẩn cũng sẽ dễ gây ra nhầm lẫn. Điều này trong phát triển thêm trong tương lai.

24 xe ô tô nhận dạng đúng hết cả dãy số, 6 ô tô còn lại nhận dạng sai từ 1 đến 2 số. Tuy nhiên bài toán yêu cầu phải cho kết quả chính xác cả dãy số để tính tiền cho chủ xe ô tô nên ta sẽ tính nó là trường hợp sai.

Sau khi hoàn thành bài toán nhận dạng biển số xe ô tô ta thấy có một số vấn đề:
Về ưu điểm: kết quả chương trình chạy khá chính xác trong các trường hợp tốt, dựa theo kết quả thử nghiệm thì tỉ lệ đúng khá cao, chương trình chạy nhanh, không tốn nhiều thời gian của người dùng, dễ thao tác. Tiết kiệm chi phí cho chủ bãi xe khi không sử dụng thẻ từ.

Nhược điểm: Giao diện chưa được hoàn mỹ, biển số còn bẩn, sẽ nhận dạng khó, máy quét cần có độ lấy nét nhanh, chương trình sẽ không nhận dạng được trong các trường hợp: dãy số trên ô tô bị mờ do bám bụi, bị khuyết, mất nét, chương trình lấy nét hơi chậm, khi gặp điều kiện ánh sáng không tốt như quá tối hay quá sáng chói sẽ làm chương trình không đọc được dãy số hoặc đọc mà bị sai nhiều.

Một số trường hợp có chất lượng hình ảnh tốt sẽ cho ra kết quả chính xác.

Hình 5.7. *Nhận dạng chính xác*

KẾT LUẬN

Kết quả của nghiên cứu đã chỉ ra được một số vấn đề về bài toán nhận dạng ký tự như: tìm hiểu một vài phương pháp phát hiện cạnh và thuật toán trong nhận dạng ký tự từ đó lựa chọn thuật toán nhận dạng phù hợp, ứng dụng của bài toán trong thực tế, các bước để xây dựng một bài toán nhận dạng ký tự.

Qua quá trình nghiên cứu cho thấy việc ứng dụng nhận dạng ký tự Tesseract có nhiều ưu khuyết điểm. Về ưu điểm, phân lớp khá tốt, kết quả chính xác khá cao, chạy nhanh, đáp ứng được yêu cầu của người dùng. Bên cạnh đó nó cũng có một số nhươc điểm như: nhận dạng với kết quả không cao trong trường hợp không tốt như: ký tự mờ, ký tự nhiễu (đang trạng thái chuyển số) hoặc điều kiện ánh sáng không tốt và một số trường hợp trong lúc tính toán đặc trưng sẽ làm cho một số ký tự số có đường nét gần giống nhau thì sẽ nhận dạng sai. Chương trình vẫn chưa hoàn thiện một số chức năng. Và sẽ phát triển thêm trong tương lai.

Từ những ưu và khuyết điểm trên sẽ là hướng phát triển trong tương lai. Qua phân tích nhược điểm thì đồ án sẽ phát triển thêm ở các bước tiền xử lý ảnh, giúp giảm thiểu các trường hợp không lý tưởng sẽ nhận dạng được tốt hơn. Trong bước trích chọn đặc trưng sẽ thử nghiệm một vài phương pháp tính toán đặc trưng khác để giảm thiểu các ký tự số có nét giống nhau sẽ nhận dạng sai lẫn nhau.

TÀI LIỆU THAM KHẢO

[1]. N.V. An/Tạp chí Khoa học ĐHQGHN: Khoa học Tự nhiên và Công nghệ, Tập 31, Số 2 (2015) 1-7.

[2]. https://vi.wikipedia.org/wiki/Nhận_dạng_ký_tự_quang_học (01/06/2019)

[3]. Trần Thanh Phước/Tạp chí Khoa học Công nghệ, Số 02/2014.

[4]. https://congdongopencv.blogspot.com/2017/10/bo-nhan-dang-ky-tu-quang-hoc-tesseract.html (04/06/2019)

[5]. https://vi.wikipedia.org/wiki/Microsoft_Visual_Studio (10/06/2019)

[6]. https://en.wikipedia.org/wiki/Microsoft_SQL_Server (10/06/2019)

[7]. https://en.wikipedia.org/wiki/OpenCV (01/05/2019)

[8]. https://en.wikipedia.org/wiki/Gaussian_blur (02/06/2019)

[9]. https://vi.wikipedia.org/wiki/Pythagoras (02/06/2019)