TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA ĐÀO TẠO QUỐC TẾ

---------------o0o---------------

Ảnh có chứa biểu tượng, Nhãn hiệu, văn bản, Phông chữ

Mô tả được tạo tự động

**BÁO CÁO BÀI TẬP LỚN**

**PHÁT HIỆN BIỂN SỐ XE**

**Giảng viên hướng dẫn**: Cao Thị Luyên

**Lớp**: CNTT VA 2 K63

**Nhóm sinh viên thực hiện** : Nhóm 8

|  |  |
| --- | --- |
| **Họ và tên** | **Mã sinh viên** |
| Trần Hữu Tâm | 222631136 |
| Đoàn Hải Xuyên | 222631162 |
| Phạm Công Đăng | 222631087 |
| Nguyễn An Thạch | 222601139 |
| Đỗ Xuân Trường | 222631148 |

Hà Nội, 11 / 2025

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA ĐÀO TẠO QUỐC TẾ

---------------o0o---------------

Ảnh có chứa biểu tượng, Nhãn hiệu, văn bản, Phông chữ

Mô tả được tạo tự động

**BÁO CÁO BÀI TẬP LỚN**

**PHÁT HIỆN BIỂN SỐ XE**

**Giảng viên hướng dẫn**: Cao Thị Luyên

**Lớp**: CNTT VA 2

**Khóa**: 63

**Nhóm sinh viên thực hiện** : Nhóm 8

|  |  |
| --- | --- |
| Họ và tên | Mã sinh viên |
| Trần Hữu Tâm | 222631136 |
| Đoàn Hải Xuyên | 222631162 |
| Phạm Công Đăng | 222631087 |
| Nguyễn An Thạch | 222601139 |
| Đỗ Xuân Trường | 222631148 |

Hà Nội, 11 / 2025

# LỜI NÓI ĐẦU

Trong bối cảnh công nghệ ngày càng phát triển, việc ứng dụng xử lý ảnh vào các hệ thống giám sát và quản lý phương tiện trở nên phổ biến và cần thiết. Đề tài “Nhận diện biển số xe bằng xử lý ảnh” được thực hiện với mục tiêu xây dựng một chương trình có khả năng tự động phát hiện và trích xuất vùng chứa biển số xe từ hình ảnh đầu vào. Chương trình vận dụng các bước xử lý ảnh cơ bản và nâng cao để nhận diện chính xác vị trí biển số, sau đó hiển thị vùng biển số được khoanh vùng và lưu lại ảnh phần biển số riêng biệt. Thông qua đề tài, chúng em mong muốn củng cố kiến thức nền tảng về xử lý ảnh, đồng thời tiếp cận tư duy xây dựng hệ thống nhận dạng tự động – tiền đề cho các ứng dụng thông minh trong lĩnh vực giao thông.

# LỜI CẢM ƠN

Em xin cảm ơn cô Cao Thị Luyên đã hướng dẫn và hỗ trợ em trong quá trình làm bài tập lớn này. Nhờ có sự hướng dẫn của cô, em hiểu rõ hơn về cách áp dụng các kiến thức xử lý ảnh vào thực hành, đặc biệt là việc xây dựng từng bước trong quy trình phát hiện biển số xe. Quá trình thực hiện bài giúp em củng cố lại nhiều kiến thức đã học, rèn luyện tư duy lập trình và hiểu rõ hơn cách kết hợp các thuật toán để giải quyết một bài toán cụ thể.  
Em xin chân thành cảm ơn cô đã dành thời gian theo dõi và góp ý để em hoàn thành bài tập này tốt hơn.

# I. GIỚI THIỆU

## ****1.1.**** Tổng quan vấn đề

Bài tập này tập trung giải quyết bài toán “**Phát hiện biển số xe”** thông qua một quy trình xử lý ảnh truyền thống. Hệ thống được thiết kế để tự động tiếp nhận hình ảnh đầu vào, phân tích các đặc trưng cạnh (Sobel) và hìnhthái học (Morphology) để thực hiện hai mục tiêu cốt lõi:

* **Phát hiện:** Khoanh vùng chính xác vị trí chứa biển số xe trong ảnh.
* **Trích xuất:** Cắt và lưu trữ khu vực biển số đã phát hiện để phục vụ các bước xử lý tiếp theo.

## ****1.2. Lý do chọn đề tài****

* **Hiểu sâu bản chất thuật toán qua cài đặt thủ công:** Thay vì chỉ gọi các hàm thư viện có sẵn (như trong OpenCV), đề tài tập trung vào việc tự xây dựng lạicác thuật toán xử lý ảnh cốt lõi (như Gaussian Blur, Sobel, Otsu, Morphology) dựa trên tính toán ma trận với NumPy. Việc này giúp nắm vững nguyên lý toán học nền tảng và cơ chế hoạt động chi tiết bên trong của từng phép biến đổi ảnh..
* **Xây dựng Công cụ Gỡ lỗi:** Động lực quan trọng nhất được thể hiện rõ qua app.py là tạo ra một công cụ trực quan hóa bằng Streamlit. Công cụ này cho phép người dùng xem kết quả từng bước của pipeline xử lý ảnh, giúp dễ dàng đánh giá hiệu suất và xác định điểm yếu của từng thuật toán trong chuỗi xử lý.

# II. CƠ SỞ LÝ THUYẾT

## 2.1. Các phương pháp tiếp cận

### 2.1.1. Phương pháp học sâu

Đây là hướng tiếp cận hiện đại, sử dụng các mạng nơ-ron tích chập (CNN) hoặc các mô hình phát hiện vật thể tiên tiến (như YOLO):

* **Ưu điểm:** Độ chính xác rất cao, hoạt động tốt trong môi trường phức tạp, đa dạng góc chụp và điều kiện ánh sáng.
* **Nhược điểm:** Yêu cầu thu thập dữ liệu và sử dụng GPU để huấn luyện; hoạt động theo cơ chế "hộp đen" khó giải thích quy trình trung gian.

### 2.1.2. Phương pháp xử lý ảnh truyền thống

Đây là phương pháp sử dụng các thuật toán dựa trên đặc trưng thủ công như cạnh, màu sắc và hình dạng hình học:

* **Nguyên lý:** Dựa trên quan sát rằng biển số xe thường có độ tương phản cao (chữ đen nền trắng hoặc ngược lại), hình dạng chữ nhật đặc trưng và mật độ cạnh dọc lớn (do các ký tự sắp xếp liền kề).
* **Ưu điểm:** không cần huấn luyện, quy trình minh bạch, dễ dàng tinh chỉnh dựa trên logic toán học.
* **Lý do lựa chọn:** đề tài sử dụng phương pháp này nhằm mục đích hiểu rõ bản chất của từng phép toán biến đổi ảnh.

## 2.2. Các thuật toán xử lý ảnh cốt lõi

### 2.2.1. Tiền xử lý ảnh

* **Resize:** Ảnh đầu vào được thay đổi kích thước về chiều rộng cố định
* **Grayscale:** Chuyển đổi ảnh màu (RGB/BGR) sang ảnh xám để giảm chiều dữ liệu. Do việc phát hiện biển số chủ yếu dựa vào sự thay đổi cường độ sáng (intensity) và cấu trúc biên dạng thay vì thông tin màu sắc.
* **Gaussian Blur:** Sử dụng bộ lọc Gaussian để làm mịn ảnh, giảm nhiễu hạt (noise). Bước này rất quan trọng để loại bỏ các chi tiết thừa có thể gây nhiễu cho quá trình phát hiện biên sau này.

### 2.2.2. Các phép toán hình thái học

Hình thái học là tập hợp các phép toán xử lý ảnh dựa trên hình dạng.

* **Blackhat Morphology:** Là hiệu của phép đóng (Closing) và ảnh gốc. Phép toán này được sử dụng để làm nổi bật các chi tiết nhỏ, tối màu trên nền sáng (hoặc ngược lại), giúp tăng cường độ tương phản của các ký tự và đường viền biển số.
* **Closing và Opening:**
  + Opening (Co -> Giãn): Giúp loại bỏ các điểm nhiễu nhỏ ra khỏi nền.
  + Closing (Giãn -> Co): Giúp lấp đầy các lỗ hổng nhỏ bên trong đối tượng và nối liền các thành phần đứt gãy, giúp vùng biển số trở thành một khối liền mạch.

### 2.2.3. Phát hiện biên Sobel

* Toán tử Sobel tính toán đạo hàm gần đúng của cường độ sáng hình ảnh. Trong bài toán này toán tử Sobel được sử dụng để tính toán độ lớn gradient từ cả hai phương ngang và dọc, giúp làm nổi bật toàn bộ đường viền của ký tự và khung biển số.

### 2.2.4. Otsu Thresholding

Đây là thuật toán tự động xác định ngưỡng tối ưu để chuyển ảnh xám thành ảnh nhị phân (đen-trắng).

* **Nguyên lý:** Thuật toán tìm giá trị ngưỡng sao cho phương sai trong lớp giữa hai nhóm điểm ảnh là lớn nhất.
* **Ứng dụng:** Otsu đặc biệt hiệu quả khi độ sáng của ảnh không đồng đều, giúp tách biệt vùng ứng viên biển số ra khỏi nền một cách tự động.

### 2.2.5. Phân tích thành phần liên thông

Sau khi có ảnh nhị phân, thuật toán Connected-Component Labeling được dùng để gom nhóm các điểm ảnh (pixels) lân cận nhau thành các đối tượng độc lập (blobs).

* **Cơ chế:** Thuật toán quét ảnh hai lần; lần một để gán nhãn tạm thời và ghi nhận sự tương đương, lần hai để chuẩn hóa nhãn về gốc sử dụng cấu trúc dữ liệu Disjoint-set.
* **Lọc nhiễu:** Dựa trên các đặc trưng hình học của các blobs này (như diện tích, tỷ lệ khung hình - Aspect Ratio, độ đặc - Extent), hệ thống lọc bỏ các nhiễu nền và chỉ giữ lại đối tượng có khả năng là biển số nhất.

# III. TRIỂN KHAI

## ****3.1.**** Môi trường và công cụ phát triển

Hệ thống được xây dựng và kiểm thử trên môi trường sau:

* **Ngôn ngữ lập trình:** Python.
* **Thư viện xử lý ảnh:** NumPy (tính toán ma trận), OpenCV (đọc/ghi ảnh).
* **Framework giao diện:** Streamlit.

## 3.2. Cấu trúc tổ chức mã nguồn

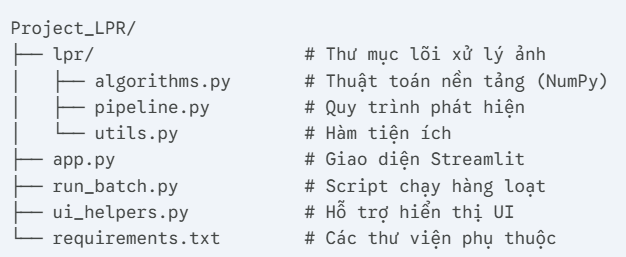
Dự án được tổ chức theo kiến trúc module hóa để đảm bảo tính minh bạch và dễ bảo trì:

1. **Thư mục lpr (Lõi xử lý):**

* **algorithms.py:** Chứa các thuật toán xử lý ảnh cơ bản được cài đặt thủ công như convolution, morphology, Otsu, Sobel.
* **pipeline.py:** Chứa logic quy trình phát hiện biển số, kết nối các thuật toán rời rạc thành chuỗi xử lý.
* **utils.py:** Các hàm tiện ích hỗ trợ (đọc ảnh, padding, xoay ảnh affine).

1. **Module ứng dụng:**

* **app.py:** Mã nguồn chính cho giao diện web Streamlit.
* **ui\_helpers.py:** Các hàm hỗ trợ chuyển đổi định dạng ảnh để hiển thị trên web.
* **run\_batch.py:** hỗ trợ chạy thử nghiệm hàng loạt trên tập dữ liệu lớn.



## 3.3. Triển khai

Quy trình phát hiện biển số được triển khai tuần tự qua 5 giai đoạn chính. Mỗi giai đoạn nhận đầu vào là kết quả của bước trước đó:

1. Giai đoạn 1 - Chuẩn hóa ảnh đầu vào:

* **Resize:** Ảnh đầu vào được thay đổi kích thước về chiều rộng cố định (mặc định 1000px) để giảm tải tính toán và đồng bộ kích thước kernel cho các bước sau.
* **Grayscale:** Chuyển ảnh sang ảnh xám, loại bỏ thông tin màu sắc, chỉ giữ lại cường độ sáng để tập trung vào cấu trúc biên dạng.

1. Giai đoạn 2 - Trích xuất đặc trưng:

Hệ thống sử dụng phương pháp kết hợp để tìm kiếm vùng ứng viên:

* **Gaussian Blur:** Áp dụng kernel 5x5 để giảm nhiễu hạt, tránh phát hiện biên sai.
* **Blackhat Morphology:** Làm nổi bật các chi tiết tối (ký tự) trên nền sáng (biển số). Bước này giúp tăng cường độ tương phản cục bộ một cách đáng kể.
* **Sobel Edge Detection:** Tính toán đạo hàm theo phương dọc và ngang để tìm biên.
* Tổng hợp: Kết quả của Blackhat và Sobel được gộp lại bằng phép lấy giá trị lớn nhất. Kỹ thuật này giúp giữ lại được cả các vùng có độ tương phản cao và các vùng có mật độ biên dày đặc.

1. Giai đoạn 3 - Phân đoạn:

* **Otsu Thresholding:** Tự động tính toán ngưỡng tối ưu từ lược đồ xám để chuyển ảnh tổng hợp thành ảnh nhị phân.
* **Opening:** Áp dụng phép co (Erosion) rồi giãn (Dilation) để loại bỏ các điểm nhiễu trắng nhỏ (muối tiêu) xuất hiện rải rác trong nền.
* **Closing:** Áp dụng phép giãn (Dilation) rồi co (Erosion) với kernel hình chữ nhật ngang. Bước này cực kỳ quan trọng để "hàn gắn" các ký tự rời rạc lại thành một khối liền mạch (blob).

1. Giai đoạn 4 - Phân tích thành phần liên thông và lọc ứng viên:

* **Labeling:** Thuật toán Connected-Component Labeling quét toàn bộ ảnh để gom nhóm các pixel liền kề thành các đối tượng độc lập.
* **Lọc ứng viên:** Duyệt qua từng đối tượng và giữ lại nếu thỏa mãn các điều kiện hình học của biển số:
  + Diện tích: Phải đủ lớn để chứa thông tin (> 5000px) nhưng không quá lớn (< 50000px) để loại bỏ nhiễu nhỏ hoặc các khối quá khổ.
  + Tỷ lệ khung hình: Kiểm tra xem đối tượng có dạng hình chữ nhật ngang hay không. Hệ thống chấp nhận 2 dải tỷ lệ: biển một dòng (AR từ 3.0 - 5.5) và biển hai dòng (AR từ 1.0 - 2.5).
  + Độ đặc: Tỷ lệ diện tích của đối tượng trên diện tích hình chữ nhật bao quanh nó phải lớn hơn ngưỡng (ví dụ 0.2) để loại bỏ các đường viền rỗng.

1. Giai đoạn 5: Trích xuất kết quả:

* Từ các ứng viên đạt chuẩn, hệ thống chọn ứng viên tốt nhất (thường là vùng có độ đặc cao nhất).
* Tiến hành cắt (crop) vùng ảnh này từ ảnh gốc để xuất ra kết quả cuối cùng.

## 3.4. Xây dựng ứng dụng trực quan hóa

Để phục vụ mục đích phân tích và tinh chỉnh thuật toán, một ứng dụng web được xây dựng bằng Streamlit với các chức năng:

* **Sidebar điều khiển:** Cho phép người dùng tải ảnh lên và điều chỉnh các tham số của pipeline.
* **Hiển thị lưới:** Kết quả của 10 bước xử lý trung gian được hiển thị song song trên giao diện, giúp nhìn trực quan từng bước xử lý ảnh.
* **Quản lý trạng thái:** Ứng dụng sử dụng cơ chế **session state** để lưu trữ ảnh và tham số, đảm bảo trải nghiệm mượt mà khi người dùng tương tác liên tục.

# IV. KẾT QUẢ THỰC NGHIỆM

## 4.1. Dữ liệu và điều kiện

* Nguồn dữ liệu: Ảnh xe từ camera giám sát Kztek.
* Số lượng ảnh thử nghiệm: 60 ảnh (ô tô, xe máy).
* Điều kiện chụp: đa dạng về góc, độ sáng, phản chiếu, bóng mờ.



## 4.2. Kết quả từng bước xử lý

Trường hợp phát hiện biển số:

|  |  |  |
| --- | --- | --- |
| **Bước** | **Kết quả** | **Nhận xét** |
| 0\_Resized |  | Ảnh gốc chụp được từ camera |
| 1\_Gray |  | Ảnh xám loại bỏ màu nền, giữ chi tiết ký tự. |
| 2\_Blurred |  | Làm mờ giảm nhiễu nhỏ. |
| 3\_Blackhat |  | Làm nổi vùng sáng (biển số). |
| 4\_Sobel\_Mag |  | Phát hiện biên ký tự rõ. |
| 5\_Combined |  | Vùng biển số nổi bật nhất. |
| 6\_Binary |  | Ảnh nhị phân hóa rõ vùng biển. |
| 7\_Opened |  | Nhiễu bị loại bỏ. |
| 8\_Closed |  | Các ký tự biển nối liền nhau. |
| 9\_Contours |  | Biển số được vẽ khung xanh (ROI). |
| 10\_Crop |  | Biển số cắt riêng. |

Trường hợp không phát hiện được biển số:

| Bước | Ảnh | Nhận xét |
| --- | --- | --- |
| 0\_Resized |  |  |
| 1\_Gray |  |  |
| 2\_Blurred |  | Làm mờ ký tự nhỏ |
| 3\_Blackhat |  | KHÔNG highlight biển số |
| 4\_Sobel\_Mag |  | Không bắt cạnh ký tự, chỉ bắt nền |
| 5\_Combined |  | Không có vùng biển số |
| 6\_Binary |  | Nhiễu nhiều, không tách được số |
| 7\_Opened |  | Xóa ký tự vì ký tự quá nhỏ |
| 8\_Closed |  | Không nối được blob |
| 9\_Contours |  | |  | | --- | |  |  |  | | --- | | Không đủ area, AR, extent | |
| 10\_Crop | Không có ảnh |  |

## 4.3. Kết quả tổng hợp

Sau khi tiến hành chạy thử nghiệm trên tập dữ liệu gồm 60 ảnh (bao gồm cả ô tô và xe máy), nhóm em đã thu được kết quả như bảng dưới đây:

|  |  |
| --- | --- |
| Tổng số ảnh | 60 |
| Số ảnh phát hiện đúng | 38 |
| Số ảnh phát hiện sai / Không phát hiện | 22 |
| Độ chính xác | 63.33% |

Kết quả thực nghiệm trên 60 ảnh cho thấy hệ thống đạt độ chính xác khoảng **63.33%**. Với phương pháp xử lý ảnh truyền thống, hệ thống hoạt động ổn định trên các ảnh có chất lượng tốt, ánh sáng tiêu chuẩn và góc chụp trực diện. Tuy nhiên, tỉ lệ lỗi vẫn còn khá cao (khoảng 36%), chủ yếu rơi vào các trường hợp ảnh thiếu sáng, bị lóa đèn hoặc biển số có kích thước quá nhỏ khiến thuật toán không bắt được biên dạng.

## 4.4. Giao diện người dùng

Giao diện người dùng được xây dựng bằng thư viện **Streamlit**, giúp việc kiểm thử từng bước xử lý của pipeline trở nên trực quan, dễ theo dõi và dễ điều chỉnh tham số. Giao diện gồm hai khu vực chính: **Sidebar điều khiển** và **khu vực hiển thị kết quả**.

### 4.4.1. Cấu trúc tổng thể của giao diện

Giao diện chia thành hai phần:

* **Sidebar**: chứa toàn bộ các tham số người dùng có thể điều chỉnh và chức năng upload ảnh.
* **Main Area**: hiển thị ảnh gốc và tất cả các bước trung gian của pipeline theo dạng lưới 4 cột để quan sát dễ dàng. Cách tổ chức này giúp người dùng vừa xem được thay đổi trực tiếp vừa điều chỉnh các tham số kiểm thử theo thời gian thực.



### 4.4.2. Chức năng của sidebar và main area

**Sidebar bao gồm:**

1. **Upload ảnh** (cho phép nhận file PNG, JPG, JPEG, BMP).
2. **Điều chỉnh tham số pipeline**, bao gồm:

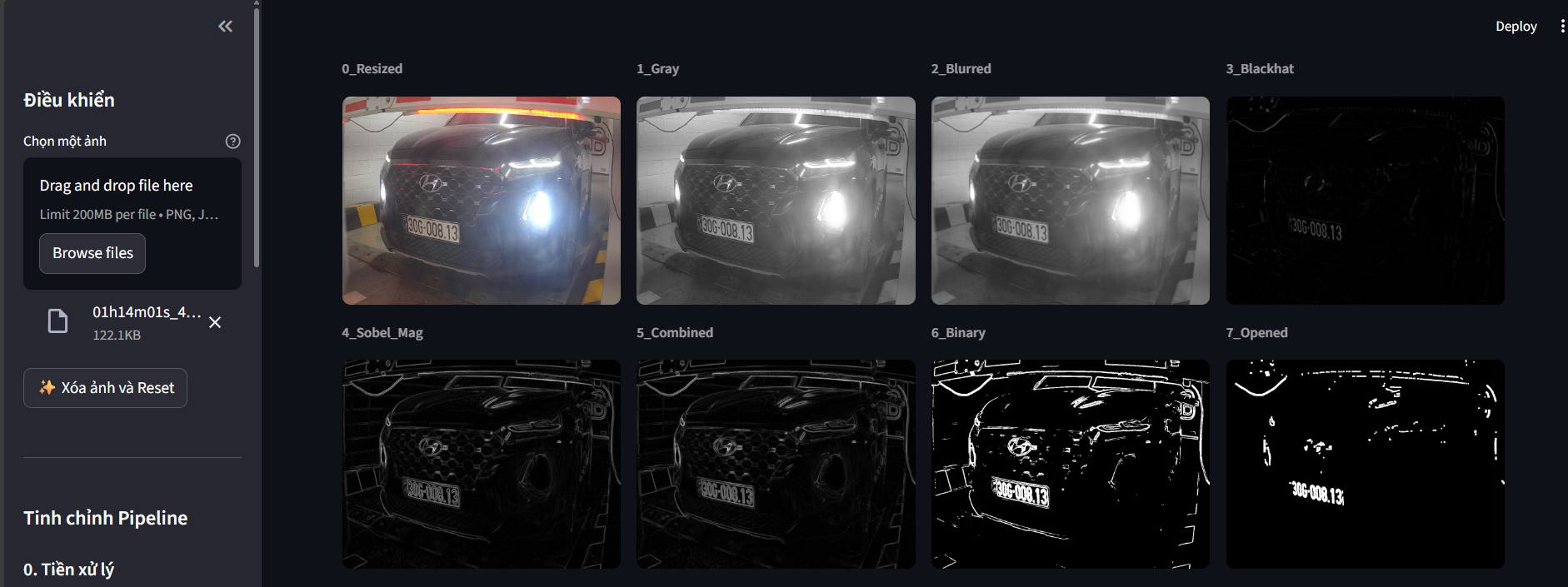
* Resize width
* Gaussian blur kernel và sigma
* Blackhat kernel size
* Opening kernel size
* Closing kernel size
* Các ngưỡng lọc blob:
* min\_area, max\_area
* dải tỉ lệ chiều rộng/chiều cao (aspect ratio)
* extent tối thiểu

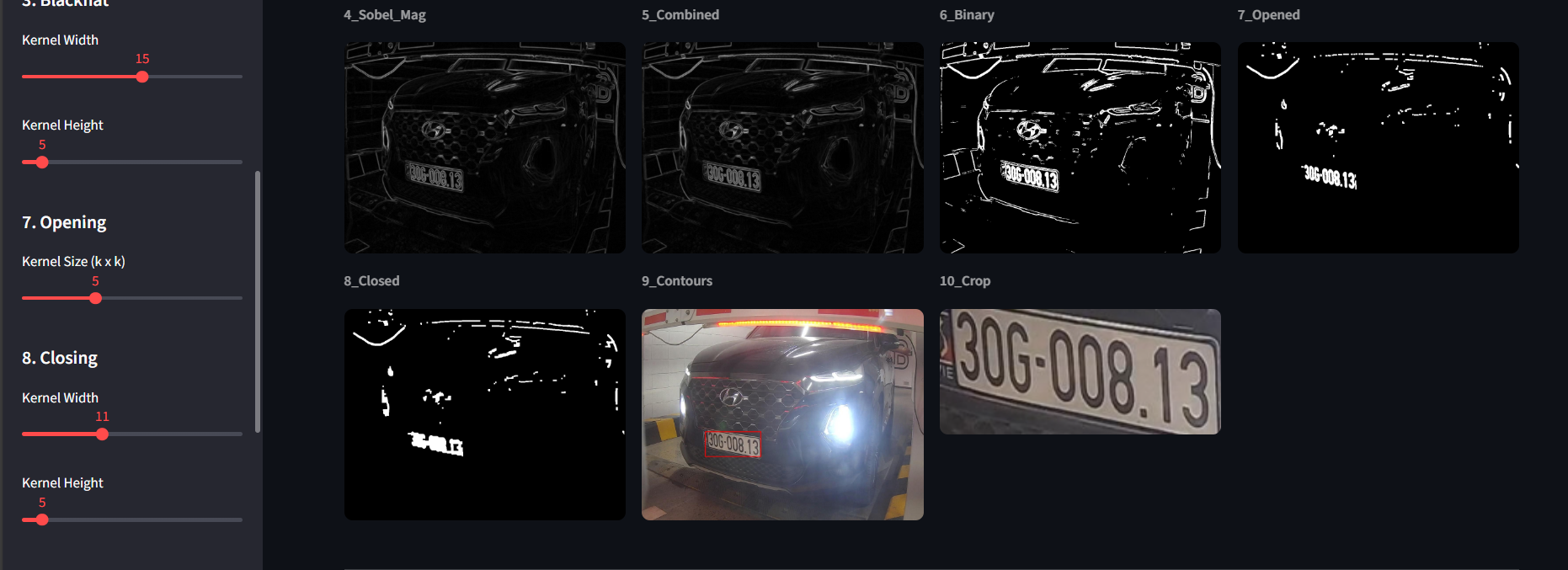
Các tham số này thay đổi trực tiếp đầu ra của pipeline, giúp người dùng quan sát ảnh hưởng của từng giá trị.

**Main Area:**

Sau khi người dùng upload ảnh, tất cả các kết quả từ các bước trong pipeline sẽ được hiển thị theo thứ tự:

1. Ảnh đã resize
2. Ảnh grayscale
3. Gaussian blur
4. Blackhat
5. Sobel
6. Combined
7. Binary
8. Opended
9. Closed
10. Contours
11. Crop vùng biển số





### 4.4.3. Quy trình hoạt động

1. Người dùng upload ảnh → ảnh được đọc dưới dạng byte.
2. Giao diện chuyển đổi ảnh sang định dạng xử lý và gửi vào hàm detect\_plate\_manual\_steps().
3. Pipeline thực hiện lần lượt các bước xử lý (blur, blackhat, threshold, morphology, lọc blob...).
4. Mỗi bước trả về ảnh trung gian → giao diện hiển thị ngay trên **main area.**
5. Khi người dùng điều chỉnh tham số, giao diện tự rerun và cập nhật toàn bộ kết quả. Nhờ cơ chế này, giao diện đóng vai trò là công cụ kiểm thử trực quan cho toàn bộ pipeline.

### 4.4.4. Quản lý trạng thái

Trong Streamlit, mỗi lần người dùng thay đổi tham số, ứng dụng sẽ rerun. Để hạn chế mất dữ liệu, giao diện sử dụng:

1. st.session\_state['image\_bytes']: lưu ảnh đang được xử lý
2. st.session\_state['file\_name']: lưu tên file
3. Các giá trị tham số hiện tại của người dùng
4. Cách quản lý này giúp:

* Không bị mất ảnh mỗi khi rerun
* Chỉ xử lý lại ảnh khi người dùng upload ảnh mới
* Tránh pipeline chạy lại không cần thiết
* Giữ giao diện ổn định và thân thiện.

# V. THẢO LUẬN

## 5.1. Điểm mạnh

* **Triển khai từ đầu:** Ưu điểm lớn nhất của dự án này là hầu hết các thuật toán xử lý ảnh cốt lõi (như chuyển đổi không gian màu, tích chập, lọc Gaussian, Sobel, Otsu, Morphology, Connected Components Labeling) đều được lập trình thủ công dựa trên lý thuyết, sử dụng thư viện nền tảng như NumPy. Điều này không chỉ giúp hiểu sâu bản chất các phép toán mà còn tạo ra một hệ thống minh bạch, dễ dàng tùy chỉnh và phát triển cho các mục đích nghiên cứu và học thuật.
* **Kiến trúc module linh hoạt:** Hệ thống được thiết kế với các module tách biệt rõ ràng (algorithms.py, pipeline.py, utils.py). Kiến trúc này cho phép dễ dàng thử nghiệm, thay thế hoặc cải tiến từng thuật toán riêng lẻ mà không ảnh hưởng đến toàn bộ pipeline.
* **Giao diện trực quan:** Ứng dụng web được xây dựng bằng Streamlit cung cấp một giao diện trực quan và thân thiện, cho phép người dùng dễ dàng tải ảnh lên và theo dõi kết quả của từng bước xử lý trung gian. Đây là một công cụ hữu ích cho việc gỡ lỗi (debug) và trình diễn (demo) hiệu quả của thuật toán.
* **Khả năng xử lý hàng loạt:** Script run\_batch.py cho thấy hệ thống có thể dễ dàng được mở rộng để xử lý tự động hàng loạt ảnh, một yêu cầu thiết thực trong các ứng dụng thực tế.

## 5.2. Hạn chế

Mặc dù đạt được những kết quả ban đầu khả quan, hệ thống hiện tại vẫn tồn tại một số điểm yếu và hạn chế cần được chỉ ra:

* **Độ nhạy cảm với điều kiện môi trường:** Thuật toán hiện tại hoạt động tốt nhất trong điều kiện ảnh chụp chuẩn (ánh sáng đều, biển số sạch, góc chụp thẳng). Hiệu suất có thể suy giảm đáng kể trong các trường hợp sau:
* **Ánh sáng yếu hoặc ngược sáng:** Ảnh hưởng trực tiếp đến bước nhị phân hóa, có thể khiến biển số bị mất hoặc nhiễu nặng.
* **Biển số bị bẩn, mờ hoặc méo mó:** Các phép morphology và lọc theo diện tích/tỉ lệ có thể không đủ để tách biển số khỏi nền.
* **Phụ thuộc vào tham số cố định:** Pipeline sử dụng nhiều tham số được "hard-code" (ví dụ: kích thước kernel cho blur, blackhat, opening, closing; ngưỡng diện tích và tỷ lệ khung hình). Các tham số này được tối ưu cho một tập dữ liệu cụ thể và có thể không tổng quát hóa tốt cho các loại biển số khác nhau (ví dụ: biển số xe máy so với xe ô tô, biển số dài 1 dòng so với 2 dòng) hoặc điều kiện chụp khác.
* **Hiệu suất tính toán:** Việc triển khai các thuật toán từ đầu bằng NumPy, mặc dù dễ hiểu, có thể chậm hơn so với việc sử dụng các hàm tối ưu hóa cao của thư viện như OpenCV (được viết bằng C++). Điều này có thể là một vấn đề đối với các ứng dụng đòi hỏi xử lý thời gian thực.
* **Hạn chế trong giai đoạn phát hiện:**
  + Phương pháp kết hợp Blackhat và Sobel, mặc dù hiệu quả trong nhiều trường hợp, có thể không đủ mạnh để phân biệt biển số với các đối tượng có kết cấu tương tự trong ảnh (ví dụ: lưới tản nhiệt, văn bản trên biển quảng cáo).
  + Thuật toán chỉ trả về một biển số duy nhất (ứng viên có extent cao nhất). Điều này khiến nó không thể xử lý các ảnh có nhiều biển số xe cùng lúc.
* **Thiếu giai đoạn nhận dạng ký tự:** Pipeline hiện tại mới chỉ dừng lại ở việc **phát hiện và cắt** vùng biển số. Hệ thống chưa có bước nhận dạng ký tự để chuyển đổi ảnh biển số đã cắt thành chuỗi văn bản, là đầu ra cuối cùng và quan trọng nhất của một hệ thống LPR hoàn chỉnh.

# VI. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## 6.1. Kết luận

Đề tài đã xây dựng thành công hệ thống phát hiện biển số xe sử dụng các kỹ thuật xử lý ảnh số truyền thống. Pipeline xử lý bao gồm các giai đoạn: tiền xử lý, trích xuất đặc trưng (Edge/Morphology), nhị phân hóa (Otsu) và phân tích thành phần liên thông (CCL). Kết quả thực nghiệm cho thấy hệ thống hoạt động ổn định, khoanh vùng chính xác biển số trong điều kiện ánh sáng tiêu chuẩn và góc chụp trực diện. Nhóm cũng đã xây dựng thành công ứng dụng trực quan hóa bằng Streamlit, hỗ trợ đắc lực cho việc tinh chỉnh tham số và gỡ lỗi thuật toán.

Tuy nhiên, nhóm nhận thức rõ những hạn chế cố hữu của phương pháp truyền thống khi đối mặt với dữ liệu thực tế phức tạp: độ nhạy cao với nhiễu ánh sáng, khó khăn trong việc xử lý biển số bị che khuất, mờ nhòe hoặc có góc nghiêng lớn. Đây là động lực để nhóm đề xuất các hướng cải tiến tiếp theo.

## 6.2. Hướng phát triển

1. **Nâng cấp mô hình nhận diện:**

* **Chuyển đổi sang kiến trúc học sâu:** Thay thế các bộ lọc thủ công bằng các mô hình mạng nơ-ron tích chập (CNN). Cụ thể, tích hợp YOLO (v8/v11) cho bài toán phát hiện đối tượng để tăng độ chính xác trong môi trường thiếu sáng và bối cảnh phức tạp.
* **Nhận diện ký tự:** sử dụng các mô hình chuyên dụng như LPRNet hoặc CRNN (CNN + LSTM + CTC Loss). Phương pháp này giúp nhận diện chuỗi ký tự liền mạch, xử lý tốt các trường hợp biển số bị nghiêng, biến dạng mà phương pháp hình thái học truyền thống không giải quyết được.
* **Tăng cường dữ liệu:** Xây dựng quy trình thu thập và gán nhãn dữ liệu tự động, áp dụng các kỹ thuật sinh dữ liệu nhân tạo hoặc biến đổi hình học để làm giàu tập huấn luyện.

1. **Tối ưu hóa hiệu năng:**

* **Xử lý đa luồng:** Tách biệt luồng đọc dữ liệu camera và luồng xử lý AI để tận dụng tối đa tài nguyên phần cứng và đảm bảo FPS ổn định.
* **Nén và tăng tốc mô hình:** Áp dụng kỹ thuật lượng tử hóa hoặc chuyển đổi mô hình sang định dạng ONNX/TensorRT. Mục tiêu là giảm độ trễ để hệ thống có thể vận hành mượt mà ngay cả trên các thiết bị biên có cấu hình thấp.

1. **Đóng gói và tích hợp hệ thống:**

* Đóng gói module nhận diện thành một RESTful API độc lập (sử dụng FastAPI/Flask). Điều này cho phép tích hợp linh hoạt vào các hệ thống lớn như quản lý bãi đỗ xe thông minh hay giám sát giao thông trên nền tảng Web/Mobile.
* Sử dụng Docker để đóng gói môi trường thực thi, đảm bảo tính nhất quán và dễ dàng triển khai trên nhiều nền tảng hạ tầng khác nhau.

# TÀI LIỆU THAM KHẢO

1. A Threshold Selection Method from Gray-Level Histograms – Otsu, N. (Bài báo gốc về Thuật toán Otsu (otsu\_threshold), phương pháp nhị phân hóa).
2. Sequential Operations in Digital Picture Processing – Rosenfeld, A., & Pfaltz, J. L. (Nền tảng về thuật toán phân tích thành phần liên thông (Connected Components Labeling)).
3. Automatic License Plate Recognition: A State-of-the-Art Review – Anagnostopoulos, C. N., et al. (Bài báo khảo sát về các kỹ thuật LPR dựa trên phát hiện cạnh và hình thái).
4. Deep Learning for Vision Systems – Elgendy, M. (Tài liệu chuyên sâu để thảo luận về cải tiến hệ thống bằng Học sâu (Deep Learning)).

# PHỤ LỤC: CHI TIẾT KẾT QUẢ THỬ NGHIỆM TRÊN TẬP DỮ LIỆU

|  |  |
| --- | --- |
| Ảnh gốc | Biển số |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  |  |
|  |  |
|  | Phát hiện nhầm biển số xe |
|  |  |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  |  |
|  |  |
|  |  |
|  |  |
|  | Không phát hiện biển số xe |
|  |  |
|  |  |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  |  |
|  | Phát hiện nhầm biển số xe |
|  | Không phát hiện biển số xe |
|  |  |
|  |  |
|  |  |
|  |  |
|  | Không phát hiện biển số xe |
|  |  |
|  | Không phát hiện biển số xe |
|  |  |
|  |  |
|  |  |
|  | Không phát hiện biển số xe |
|  |  |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  |  |
|  | Không phát hiện biển số xe |
|  |  |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  | Không phát hiện biển số xe |
|  |  |