# **B. Phát hiện ký tự biển số xe**

Mục tiêu: Nhận dạng và trích xuất thông tin ký tự từ biển số xe đã được hiệu chỉnh.

Công nghệ sử dụng: PaddleOCR, một thư viện nhận dạng ký tự mạnh mẽ, hỗ trợ đa ngôn ngữ, bao gồm cả tiếng Việt, giúp tăng độ chính xác khi nhận dạng ký tự trên biển số xe.

Quy trình:

- Sử dụng ảnh biển số xe đã được hiệu chỉnh từ bước trước làm đầu vào.

- Áp dụng PaddleOCR để phát hiện và nhận dạng ký tự trên biển số xe.

- Xử lý kết quả nhận dạng, bao gồm cả việc loại bỏ những ký tự nhận dạng sai và hiệu chỉnh kết quả cuối cùng.

## **2.4. Các cải tiến tối ưu**

### **Tối ưu hóa mô hình YOLOv8**

Giảm kích thước mô hình: Thực hiện các kỹ thuật như pruning (cắt tỉa) và quantization (lượng tử hóa) để giảm kích thước mô hình mà không làm giảm đáng kể độ chính xác.

Tăng tốc độ xử lý: Sử dụng TensorRT trên Jetson Nano và tối ưu hóa các thư viện trên Raspberry Pi 4 để tăng tốc độ suy luận.

### **Tối ưu hóa quy trình xử lý ảnh**

Áp dụng các kỹ thuật xử lý ảnh nâng cao: Sử dụng các phương pháp tiền xử lý ảnh như histogram equalization và noise reduction để cải thiện chất lượng ảnh đầu vào, giúp tăng độ chính xác của mô hình.

* Histogram Equalization: là một kỹ thuật xử lý hình ảnh làm tăng độ tương phản của hình ảnh bằng cách điều chỉnh độ tương phản của hình ảnh thông qua sử dụng *biểu đồ thang độ xám* (Histogram) của nó.

Histogram là 1 phương pháp biểu diễn phân bố độ sáng (intensity distribution) của các pixel trong một ảnh. Histogram của một ảnh được tính bằng cách đếm số lượng pixel có cùng giá trị màu hoặc cường độ sáng trong ảnh (với ảnh màu thì cần phải chọn kênh màu cụ thể để tính, hoặc đưa ảnh về ảnh xám – gray scale). Histogram của ảnh (khi chưa chuẩn hóa - unnormalized histogram ) được tính như sau:

với (với L là mức thang độ xám, thường là 255).

Chuẩn hóa Histogram - normalized histogram:

với M, N lần lượt là số hàng và cột của ảnh.

Hầu hết trong các trường hợp, người ta thường làm việc với normalized histogram.

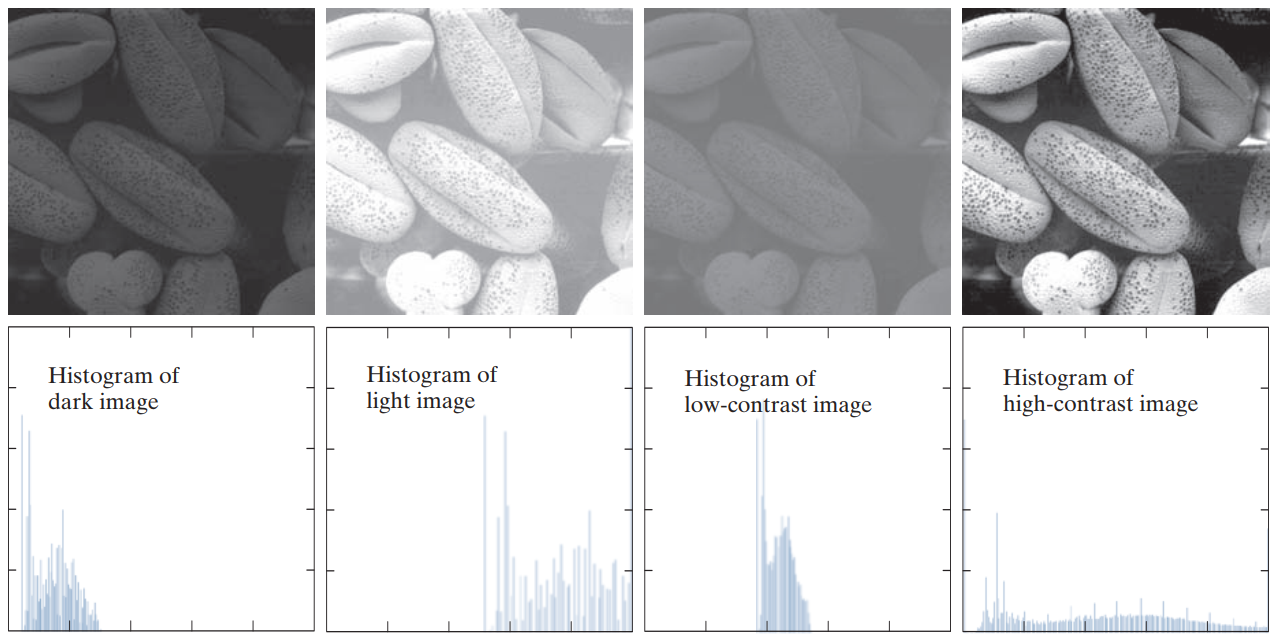
Ví dụ dưới đây về histogram của các ảnh tối, sáng, tương phản thấp, tương phản cao tương ứng:

+ Với hình ảnh tối, biểu đồ có các cột tập trung ở bên trái tương ứng với màu tối.

+ Với hình ảnh sáng, biểu đồ được tập trung ở bên phải chứa các pixel màu trắng.

+ Với hình ảnh có độ tương phản thấp, biểu đồ có các cột tập trung vào nhau và xung quanh khoảng giữa.

+ Với hình ảnh có độ tương phản cao, biểu đồ sẽ có các giá trị san đều.



Ý tưởng cơ bản của Histogram Equalization là làm phẳng phân bố histogram của ảnh. Equalization ngụ ý việc ánh xạ phân bố của histogram ban đầu sang một phân bố khác (có các giá trị intensity rộng và đều hơn). Cách tính Histogram Equalization:

* Tính toán Histogram: Đầu tiên, chúng ta tính toán histogram của ảnh, đại diện cho phân bố của các mức sáng trong ảnh. Histogram là một biểu đồ thống kê của số lượng pixel có cùng mức sáng.
* Tính toán Cumulative Distribution Function (CDF): Tiếp theo, chúng ta tính toán hàm phân phối tích lũy (CDF) của histogram. CDF cho biết tỷ lệ phần trăm pixel có mức sáng nhỏ hơn hoặc bằng mức sáng của mỗi cường độ.

,

trong đó: là số lượng pixel có cường độ sáng (intensity) I trong ảnh,

là giá trị histogram tích lũy tại cường độ sáng intensity j, ví dụ j trong khoảng [0, 255].

* Equalization: Sau đó, chúng ta ánh xạ lại mỗi giá trị cường độ ban đầu sang một giá trị mới bằng cách sử dụng CDF. Quá trình này tạo ra một ánh xạ phi tuyến tính từ mức sáng ban đầu của mỗi pixel sang mức sáng mới, làm phẳng phân bố của histogram.

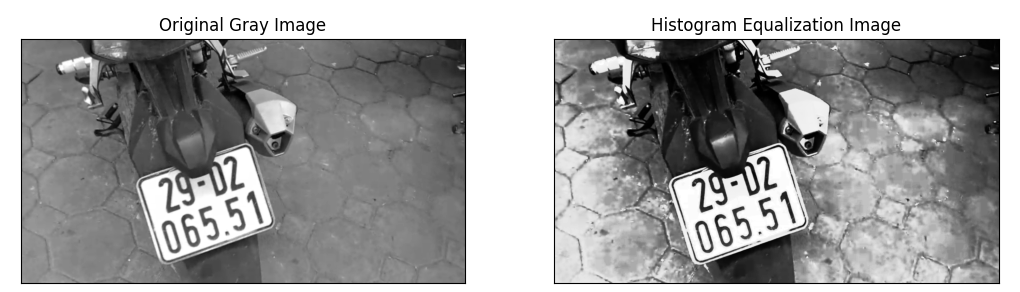
,

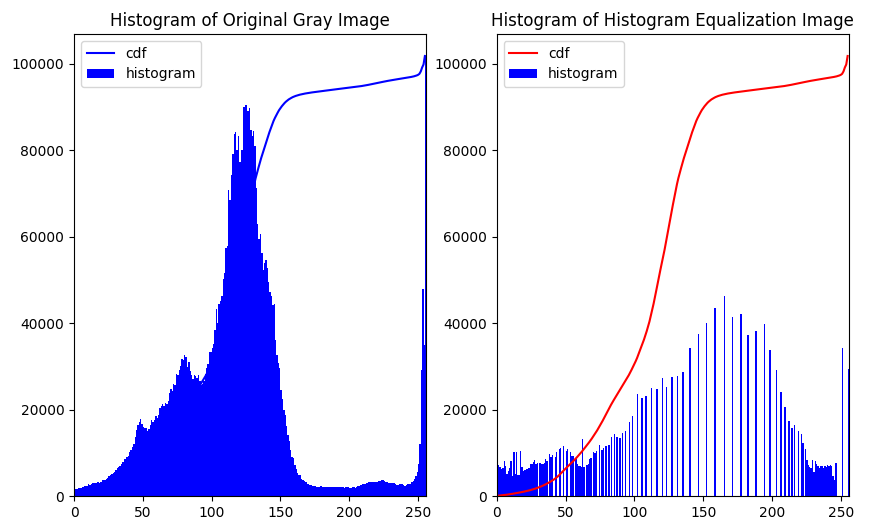
trong đó là giá trị hàm phân phối tích lũy tại cường độ i,

*minCDF* là giá trị nhỏ nhất của *CDF*, chính là số pixel trong ảnh.

* Ánh xạ lại ảnh: Cuối cùng, chúng ta sử dụng ánh xạ đã tính được để cập nhật lại mỗi pixel trong ảnh gốc, từ đó tạo ra ảnh đã được equalize.

Dưới đây là ví dụ 1 ảnh biển số xe ban đầu ở thang độ xám và sau khi thực hiện Histogram Equalization:





* Noise Reduction:

Nhiễu luôn xuất hiện trong ảnh kỹ thuật số trong các bước thu nhận, mã hóa , truyền và xử lý ảnh… Giảm nhiễu hay lọc nhiễu một quy trình tiêu chuẩn được sử dụng trong hầu hết mọi hệ thống xử lý hình ảnh. Việc lựa chọn bộ lọc phù hợp phụ thuộc vào kiểu dữ liệu cũng như loại nhiễu trong ảnh. Một số bộ lọc nhiễu phổ biến trong xử lý ảnh:

|  |  |
| --- | --- |
| Nhiễu | Bộ lọc phù hợp |
| Salt & Pepper (nhiễu muối tiêu) | Median filter (bộ lọc trung vị) |
| Poisson (hoặc nhiễu Shot, một loại nhiễu có thể được mô hình hóa bằng quy trình Poisson) | Mean filter (bộ lọc trung bình) |
| Gaussian (1 loại nhiễu phổ biến, xuất hiện nhiều trong các nguồn tự nhiên, tuân theo phân phối chuẩn) | Mean/Gaussian/BilAtERAL filter |
| Speckle (nhiễu đốm, thường xuất hiện trong các loại ảnh y tế như chụp CT, X-ray) | Weiner |

Với bộ ảnh biển số xe, với mục đích tối ưu hóa quy trình xử lý ảnh để đưa ra kết quả tối ưu nhất, đồng thời hạn chế những phương pháp quá phức tạp làm chậm quá trình xử lý ảnh, ở đây chỉ nói về các phương pháp phổ biến bao gồm Mean filters, Gaussian Blur và Median filter (phương pháp thuộc Order-statistic filters):

* Gaussian Blur: Công thức của Gaussian Filter được biểu diễn bằng hàm Gauss (Gaussian function). Hàm Gaussian có tính chất làm mịn và đồng nhất, và thường được sử dụng trong xử lý ảnh để làm mờ hình ảnh hoặc giảm nhiễu. Gaussian function là một hàm mũ có dạng:

Trong đó, G(x,y) là giá trị của hàm Gaussian tại điểm (x,y).

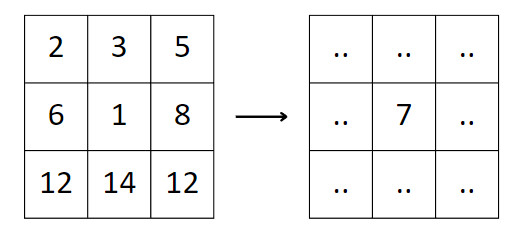
là độ lệch chuẩn (standard deviation), quyết định độ rộng và mức độ phân tán của hàm Gauss. Độ lệch chuẩn càng lớn, phân bố của hàm Gauss càng rộng và đồng đều.

(x,y) là tọa độ của điểm trong không gian 2 chiều.

* Arithmetic Mean Filter: là trường hợp đơn giản nhất của Mean filter, phương pháp này tính toán trung bình các giá trị intensity của pixel trong 1 bộ lọc trượt qua ảnh (sliding window) có kích thước . Bộ lọc này làm mịn các biến thể cục bộ trong hình ảnh và giảm nhiễu do làm mờ.

trong đó là ảnh đầu vào, bộ lọc có kích thước ,

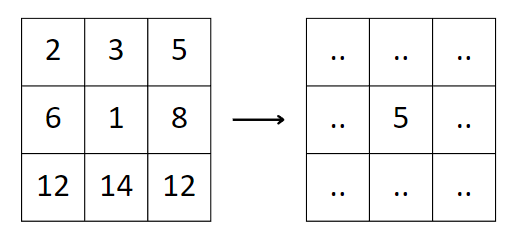
là ảnh được phục hồi.



Ví dụ với cửa số 3×3, tổng các giá trị intensity trong của sổ được chia cho 9.

* Geometric Mean Filter: bộ lọc này đạt được độ mịn tương đương với Arithmetic Mean Filter, nhưng nó có xu hướng mất ít chi tiết hình ảnh hơn trong quá trình này.

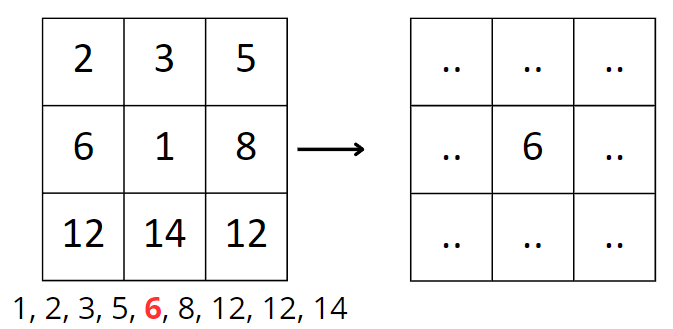
trong đó biểu thị phép nhân, ở đây giá trị intensity của mỗi pixel trong cửa sổ được nhân với nhau, sau đó lấy lũy thừa bậc .



Ví dụ với cửa số 3×3 như sau.

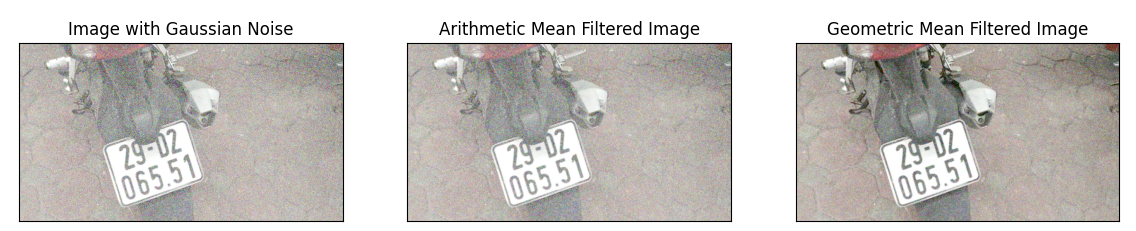
Ngoài 2 phương pháp trên, Mean filter còn có các phương pháp khác như Harmonic Mean Filter (hoạt động tốt đối với nhiễu muối - salt noise nhưng không tốt đối với nhiễu tiêu - pepper noise) và Contraharmonic Mean Filter (phù hợp để giảm hoặc loại bỏ hầu như ảnh hưởng của nhiễu muối tiêu).

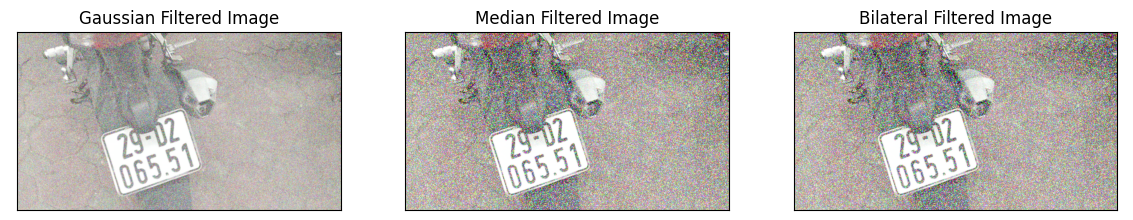
* Median filters: là phương pháp phổ biến nhất thuộc loại Order-statistic filters. Ý tưởng cơ bản của Median Filter là thay mỗi pixel trong hình ảnh bằng giá trị trung vị của các pixel trong một vùng lân cận xung quanh nó.



Ví dụ với bộ lọc cửa số 3×3 như sau, 6 là giá trị trung vị của cửa sổ này.

Dưới đây là 1 hình ảnh nhiễu được xử lý qua các phương pháp trên:





Đa luồng xử lý: Áp dụng đa luồng để xử lý song song việc tiền xử lý ảnh và suy luận mô hình, giảm thiểu thời gian chờ và tối ưu hóa tốc độ xử lý tổng thể.

### **Tối ưu hóa hiệu suất trên thiết bị nhúng**

Quản lý tài nguyên thiết bị: Monitor và tối ưu hóa việc sử dụng CPU, RAM, và bộ nhớ lưu trữ để đảm bảo mô hình chạy mượt mà trên thiết bị nhúng.

Tối ưu hóa năng lượng: Áp dụng các chiến lược như dynamic voltage and frequency scaling (DVFS) để quản lý hiệu quả việc tiêu thụ năng lượng của thiết bị, quan trọng cho các ứng dụng cần hoạt động lâu dài trên pin.

# **CHƯƠNG 3: ĐÁNH GIÁ TỐI ƯU PHƯƠNG PHÁP**

NHẬN DẠNG BIỂN SỐ XE TRÊN HỆ NHÚNG

## **3.1. Triển khai nhận dạng biển số xe trên hệ nhúng**

### **3.1.1. Giới thiệu về thiết bị biên**

Giới thiệu sơ bộ về Raspberry Pi 4 và Jetson Nano: Cấu hình, điểm mạnh, điểm yếu.

|  |  |  |
| --- | --- | --- |
|  | Raspberry Pi 4 | Jetson Nano |
| Cấu hình | * Hãng sản xuất: Raspberry Pi Foundation. * CPU: Broadcom BCM2711, ARM Cortex-A72 64-bit SoC. * GPU: Broadcom VideoCore VI. * RAM: 2GB, 4GB hoặc 8GB LPDDR4. * Cổng kết nối: HDMI, USB 3.0, USB-C, Ethernet, Wi-Fi, Bluetooth. | * Hãng sản xuất: NVIDIA. * CPU: ARM Cortex-A57 quad-core. * GPU: NVIDIA Maxwell GPU với 128 CUDA cores. * RAM: 4GB LPDDR4. * Cổng kết nối: HDMI, USB 3.0, USB 2.0, Ethernet, GPIO. |
| Hỗ trợ Hệ điều hành | * Raspbian (dựa trên Debian), Ubuntu, và các hệ điều hành khác như Windows 10 IoT Core. | * Ubuntu Linux. * Phần mềm hỗ trợ: NVIDIA JetPack SDK, NVIDIA CUDA, cuDNN, TensorRT |
| Ứng dụng phổ biến | * Dự án IoT, máy tính nhúng, học máy, điều khiển robot, media center, server nhỏ,… * Thích hợp cho các ứng dụng phổ biến như máy chủ nhỏ, media center, DIY. | * Học máy, nhận diện hình ảnh, xử lý video, IoT, robot, tự động hóa, và các ứng dụng AI. * Thích hợp cho các ứng dụng yêu cầu hiệu suất tính toán cao như AI và xử lý hình ảnh. |
| Giá cả | * Từ 1tr700k – 2tr700k (RAM 2GB, 4GB, 8GB) | * Khoảng 4tr700k với máy tính AI NVIDIA (khoảng 5tr400k với các phụ kiện đi kèm như Adaptor, vỏ Case, thẻ nhớ) |
| Ưu/Nhược điểm | * Giá cả: Raspberry Pi 4 thường rẻ hơn so với Jetson Nano, đặc biệt là khi tính đến các phiên bản có giá thấp như Raspberry Pi 4 2GB. * Cộng đồng và Hỗ trợ: Raspberry Pi có cộng đồng lớn và phong phú với nhiều tài liệu và hỗ trợ từ người dùng. | * Hiệu suất: Jetson Nano thường có hiệu suất cao hơn so với Raspberry Pi 4, đặc biệt là trong các ứng dụng AI và học máy nhờ vào GPU NVIDIA. * Cộng đồng và Hỗ trợ: Jetson Nano có cộng đồng nhỏ hơn nhưng chuyên sâu, đặc biệt là trong lĩnh vực AI và máy học. |

### **3.1.2. Lựa chọn phần mềm và công cụ phát triển**

Lý do chọn lựa các công cụ này và cách chúng hỗ trợ triển khai trên hệ nhúng.

|  |  |  |
| --- | --- | --- |
|  | Raspberry Pi 4 | Jetson Nano |
| Lý do chọn lựa | * Sử dụng rộng rãi trong các dự án IoT nhờ vào tính linh hoạt, giá thành thấp, không yêu cầu cấu hình máy chủ mạnh. | * Phù hợp cho các ứng dụng nhận diện hình ảnh và video phức tạp trong IoT, yêu cầu hiệu suất tính toán cao. |
| Hỗ trợ triển khai | * Raspberry Pi 4 có các cổng kết nối để kết nối với các cảm biến và module điều khiển, có GPU để tăng tốc độ tính toán (mặc dù không mạnh mẽ như các GPU chuyên nghiệp). | * Jetson Nano cũng có các cổng như Raspberry, có GPU mạnh hơn và hỗ trợ các framework AI như TensorFlow và PyTorch để triển khai các mô hình học máy. |

Ví dụ với bài toán nhận diện biển số xe, Jetson Nano có thể ứng dụng cho việc giám sát giao thông khi phải xử lý hình ảnh phức tạp, rõ nét và nhiều biển số xe có thể xuất hiện trong 1 khung hình trong khi Raspberry Pi 4 có thể không phù hợp vì hạn chế về hiệu suất tính toán; Raspberry Pi 4 có thể ứng dụng trong các hầm gửi xe có số lượng biển số xe xuất hiện trong 1 khung hình ít, hình ảnh đơn giản, tiết kiệm chi phí phần cứng.

### **3.1.3. Cài đặt và Triển khai mô hình nhận dạng**

Cách mô hình được tối ưu hóa để chạy trên hệ thống nhúng.

Bước cài đặt phần mềm và mô hình trên Raspberry Pi 4 và Jetson Nano.

Mô tả quá trình cấu hình và khắc phục sự cố.

## **3.2. Đánh giá hiệu năng**

### **3.2.1. Tiêu chí đánh giá**

Định nghĩa các tiêu chí đánh giá hiệu năng: thời gian xử lý, độ chính xác, tiêu thụ năng lượng, v.v.

Phương pháp đánh giá: cách thức thu thập và phân tích dữ liệu.

### **3.2.2. So sánh hiệu năng giữa Raspberry Pi 4 và Jetson Nano**

Phân tích kết quả đánh giá: so sánh và đối chiếu hiệu năng giữa hai thiết bị.

Ưu điểm và nhược điểm của mỗi thiết bị trong việc triển khai mô hình nhận dạng.

## **3.3. Kết quả thực nghiệm**

Trình bày dữ liệu thực nghiệm: số lượng biển số được nhận dạng chính xác, tỷ lệ lỗi.

Hình ảnh, biểu đồ so sánh kết quả nhận dạng trên cả hai thiết bị.

Phân tích nguyên nhân của kết quả: tác động của phần cứng, phần mềm, và mô hình đến hiệu suất nhận dạng.

Đề xuất cách cải thiện hiệu suất và độ chính xác của hệ thống nhận dạng.

Tham khảo

(Book) Image Processing 4rd ed. - R. Gonzalez, R. Woods