**TRƯỜNG ĐẠI HỌC SƯ PHẠM HÀ NỘI**

**KHOA CÔNG NGHỆ THÔNG TIN**

**---\*\*\*---**



**Tài liệu đặc tả sản phẩm:**

**Đề tài: Phát hiện và xử lý mã QR trong ảnh**

**Học phần: Xử lý ảnh**

**Giảng viên: Phạm Thọ Hoàn**

**Sinh viên thực hiện: Lê Huy Hoàng - 715105097**

1. **Mục tiêu:**

Sử dụng ngôn ngữ lập trình Python để giải quyết bài toán phát hiện và xử lý các mã QR trong ảnh đầu vào, sản phẩm sẽ cung cấp khả năng xử lý ảnh để tăng cường hiệu suất phát hiện mã QR, bao gồm các bước như cân bằng sáng, làm mờ và làm nét ảnh.

1. **Các chức năng và luồng xử lý:**
   1. ***Các chức năng:***

***2.1.1. Phát hiện mã QR trong ảnh:***

Từ ảnh đầu vào, ứng dụng sẽ trả về QR được phát hiện cùng với nội dung của QR được giải mã.

***2.1.2. Xử lý ảnh để tăng cường khả năng nhận diện mã QR:***

Trong một số trường hợp, ảnh đầu vào có chất lương chưa tốt (quá tối, mờ, nhiễu,...) thì không thể phát hiện được mã QR trực tiếp trong ảnh gốc thì khi đó, ảnh đầu vào sẽ được đi qua các bước tiền xử lý để nâng

***2.1.3. Hiển thị kết quả và xử lý mã QR:***

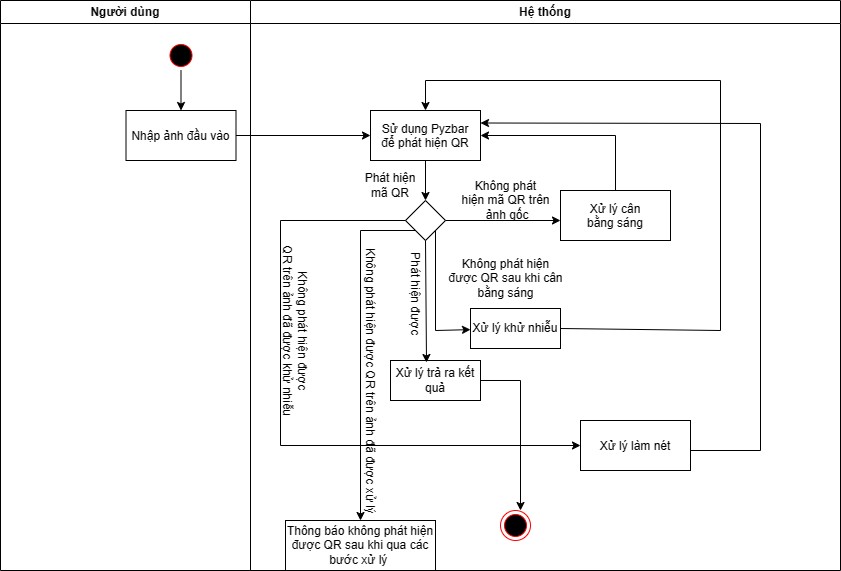
Vùng chứa mã QR được phát hiện trong ảnh sẽ được khoanh lại, và cắt ra, điều này thuận tiện cho việc quan sát mã QR được phát hiện.

* 1. ***Luồng xử lý:***

***2.1 Chi tiết luồng xử lý***

Từ ảnh đầu vào, ứng dụng sẽ sử dụng thư viện để phát hiện và giải mã mã QR, nếu phát hiện mã QR, thì mã QR sẽ được đi tới các bước xử lý trả ra kết quả (Khoanh vùng, cắt, xoay). Nếu không phát hiện được QR trên ảnh gốc, thì QR sẽ được đi qua các bước tiền xử lý: Cân bằng sáng (CLAHE), khử nhiễu làm mịn ảnh (Median Filtering), Làm nét ảnh (Sharpening), bất kể kết sau khi được xử lý của bộ lọc nào cũng được pyzbar xử lý, nếu phát hiện được thì đi tới bước xử lý trả về kết quả như trên và thông báo là QR được phát hiện sau khi xử lý bởi bộc lọc đó, và nếu không phát hiện được thì thông báo không phát hiện được QR sau khi được xử lý bởi bộ lọc đó và đi qua bộ lọc xử lý khác rồi lặp lại như trên

***2.2 Sơ đồ hoạt động:***



1. **Các thành phần sản phẩm:**

***3.1. Các phương thức xử lý đầu vào và đầu ra ảnh:***

* detect\_qr\_from\_image(image\_path): Phát hiện mã QR từ ảnh đầu vào đi kèm 1 bộ lọc cân bằng sang CLAHE dựa theo cân bằng Histogram với đầu vào là đường dẫn ảnh gốc
* view(qrCodes\_processed, image, img): Chịu trách nhiệm xử lý và hiển thị đầu ra kết quả (khoanh vùng và cắt QR, kết quả giải mã QR) với đầu vào là list QR code được xử lý bởi pyzbar , ảnh qua tiền xử lý và ảnh gốc

***3.2. Các phương thức tiền xử lý ảnh:***

* medianImg3x3(img): Định nghĩa bộ lọc trung vị 3x3 của ảnh. Bộ lọc này được áp dụng để làm mờ ảnh và giảm nhiễu.
* sharpen\_image(image): Làm nét ảnh bằng cách áp dụng một kernel cụ thể là Laplacian (thường được sử dụng để làm nét và làm nổi bật các đặc trưng cạnh trong ảnh)

***3.3. Các hàm phụ trợ:***

* rotate\_qr\_code(newImg): Biến đổi thành ảnh xám từ đầu vào, làm nghịch đảo ảnh xám (QR sẽ sáng hơn so với viền), sau đó sử dụng phương pháp OTSU để tự động xá định ngưỡng nhị phân trên Histogram của ảnh. Dùng minAreaRect() (cv2) để tính toán góc nghiêng mã qr để tính ra góc cần xoay, sau đó xoay ảnh bằng ma trận biến đổi affine được tính bằng getRotationMatrix2D() (cv2)

1. **Công nghệ và thư viện:**

* OpenCV (cv2): Sử dụng để xử lý và phân tích ảnh
* Numpy: Sử dụng để thao tác và xử lý mảng nhiều chiều
* Pyzbar: Thư viện Python cho việc phát hiện và giải mã QR

1. **Quy trình làm việc:**

***5.1. Tạo môi trường phát triển:***

* Cài đặt IDE, Python và các thư viện cần thiết.

***5.2. Triển khai code:***

* Viết mã cho các chức năng đã mô tả.
* Kiểm tra và xác nhận tính đúng đắn của từng chức năng.

***5.3. Kiểm thử và sửa lỗi:***

* Kiểm tra tính ổn định và hiệu suất của ứng dụng.
* Xử lý các lỗi phát sinh và tối ưu hóa mã nguồn.

***5.4. Viết tài liệu đặc tả***