

ĐỒ ÁN XỬ LÝ ẢNH: MÁY CHẤM TRẮC NGHIỆM OMR

Nhóm thực hiện: Nguyễn Thế Minh Nhật

Ngày 12 tháng 12 năm 2025

Mục lục

1	Giới thiệu chung	2
1.1	Các tính năng chính	2
2	Cấu trúc dự án	2
3	Cài đặt môi trường	3
3.1	Yêu cầu hệ thống	3
3.2	Cài đặt thư viện	3
4	Phương pháp thực hiện (Methodology)	3
4.1	Sơ đồ tổng quát (High-Level Pipeline)	3
4.2	Mô tả chi tiết từng giai đoạn	5
4.2.1	Giai đoạn 1: Thu thập và Chuẩn hóa đầu vào	5
4.2.2	Giai đoạn 2: Định vị và Biến đổi hình học	5
4.2.3	Giai đoạn 3: Trích xuất và Phân tích đặc trưng	5
4.2.4	Giai đoạn 4: Ra quyết định và Kết xuất	6
5	Hướng dẫn sử dụng	6
5.1	Bước 1: Chuẩn bị dữ liệu	6
5.2	Bước 2: Cấu hình đường dẫn	6
5.3	Bước 3: Chạy chương trình	6
6	Hạn chế và Hướng phát triển	7
6.1	Hạn chế hiện tại	7

1 Giới thiệu chung

Dự án này là một ứng dụng minh họa khả năng của thư viện OpenCV và các kỹ thuật Xử lý ảnh số để tự động chấm điểm phiếu trả lời trắc nghiệm (Optical Mark Recognition - OMR). Hệ thống được thiết kế để hoạt động nhanh chóng, chính xác và hỗ trợ nhiều định dạng đầu vào.

1.1 Các tính năng chính

- **Đa dạng định dạng đầu vào:** Xử lý tốt cả file scan PDF và các định dạng ảnh phổ biến (JPG, PNG).
- **Tự động nhận diện và căn chỉnh:** Tự động phát hiện khung viền phiếu trả lời trong ảnh chụp và thực hiện biến đổi phối cảnh (Warp Perspective) để bẻ thẳng tờ giấy trước khi xử lý.
- **Chấm điểm chính xác:** Hỗ trợ chấm điểm cho cấu trúc phiếu tiêu chuẩn (ví dụ: 20 câu, 4 lựa chọn A, B, C, D).
- **Trực quan hóa kết quả:** Vẽ vòng tròn xanh lên các câu trả lời được hệ thống nhận diện và hiển thị kết quả trực tiếp trên ảnh.
- **Báo cáo kết quả:** Xuất danh sách đáp án đã chọn ra màn hình console (hoặc file Excel trong phiên bản mở rộng).

2 Cấu trúc dự án

Cây thư mục của dự án được tổ chức như sau:

main.py: File thực thi chính của chương trình. Chịu trách nhiệm điều phối luồng xử lý từ đọc ảnh đến xuất kết quả.

requirements.txt: Danh sách các thư viện Python cần thiết để chạy dự án.

src/: Thư mục chứa mã nguồn xử lý lõi (Core modules).

- **pre_processing.py:** Các hàm tiền xử lý ảnh (làm mờ Gaussian, tách biên Canny, tìm contour...).
- **utils.py:** Các hàm tiện ích (sắp xếp tọa độ, xoay thẳng ảnh warp perspective...).
- **omr_logic.py:** Logic chính để sinh lưới tọa độ, đếm pixel và chấm điểm.
- **measure_tool.py:** Công cụ tương tác hỗ trợ đo đạc và lấy tọa độ mỏ neo (anchors) cho một mẫu phiếu mới.

data/raw/: Thư mục chứa ảnh chụp hoặc file PDF phiếu trả lời đầu vào.

3 Cài đặt môi trường

3.1 Yêu cầu hệ thống

Để chạy được dự án, máy tính cần cài đặt:

1. **Python 3.x** (Khuyến dùng bản 3.8 trở lên).
2. **Poppler:** Phần mềm hỗ trợ thư viện `pdf2image` để đọc file PDF.
 - Tải về và giải nén.
 - Thêm đường dẫn thư mục `bin` của Poppler vào biến môi trường (Environment Variable) của Windows.

3.2 Cài đặt thư viện

Mở terminal (CMD hoặc VS Code terminal) tại thư mục dự án và chạy lệnh sau:

```
1 pip install -r requirements.txt
2
```

Các thư viện chính bao gồm: `opencv-python`, `numpy`, `pdf2image`, `matplotlib`, `pandas`.

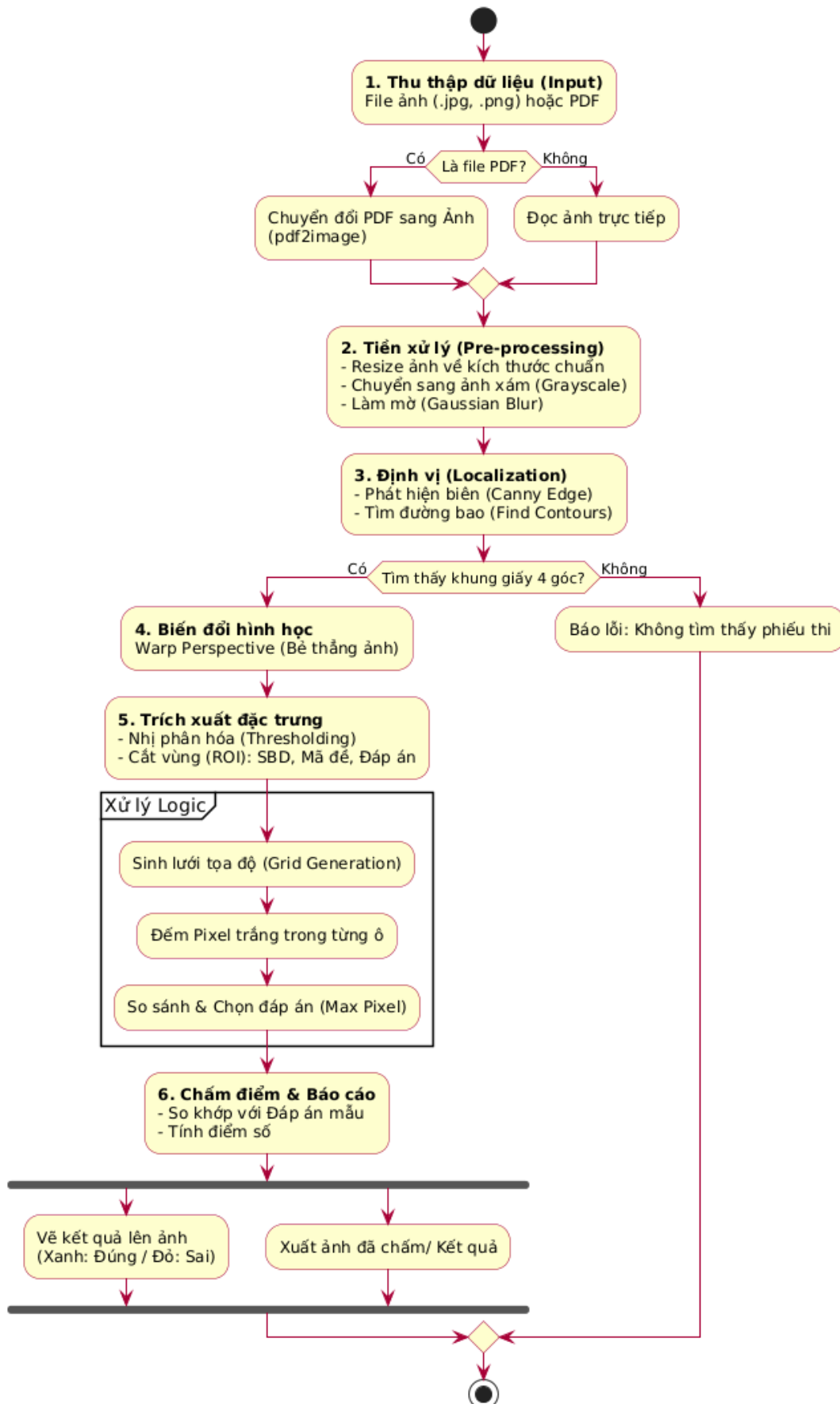
4 Phương pháp thực hiện (Methodology)

Hệ thống chấm thi trắc nghiệm (OMR) được xây dựng dựa trên một quy trình xử lý ảnh tuần tự (Pipeline) gồm 4 giai đoạn chính, đảm bảo chuyển đổi dữ liệu thô từ ảnh chụp thành kết quả điểm số chính xác.

4.1 Sơ đồ tổng quát (High-Level Pipeline)

Quy trình tổng thể của hệ thống được mô tả qua sơ đồ khối dưới đây:

OMR System Pipeline (Quy trình Xử lý OMR)



4.2 Mô tả chi tiết từng giai đoạn

4.2.1 Giai đoạn 1: Thu thập và Chuẩn hóa đầu vào

Đây là bước đầu tiên nhằm đưa dữ liệu từ nhiều nguồn khác nhau về một định dạng thống nhất để xử lý.

- **Tiếp nhận dữ liệu (Data Ingestion):** Hệ thống hỗ trợ hai loại đầu vào:
 - *Ảnh đơn*: Các định dạng phổ biến như .jpg, .png từ máy ảnh hoặc điện thoại.
 - *Tập tin PDF*: Sử dụng thư viện pdf2image để chuyển đổi (rasterize) từng trang tài liệu PDF thành ma trận điểm ảnh (Numpy array) để xử lý tuần tự.
- **Tiền xử lý (Preprocessing):**
 - *Resize*: Ảnh đầu vào được thay đổi kích thước (Resize) về chiều rộng chuẩn (ví dụ: 1000px) để giảm tải tính toán và chuẩn hóa tọa độ xử lý.
 - *Grayscale Conversion*: Chuyển đổi không gian màu từ RGB sang ảnh xám (Grayscale), loại bỏ thông tin màu sắc không cần thiết.
 - *Gaussian Blur*: Áp dụng bộ lọc Gauss (kernel 5×5) để làm mịn ảnh, giảm nhiễu hạt (noise) tần số cao, giúp tăng độ chính xác cho bước phát hiện biên.

4.2.2 Giai đoạn 2: Định vị và Biến đổi hình học

Mục tiêu của giai đoạn này là trích xuất vùng phiếu trả lời và loại bỏ các biến dạng hình học do góc chụp nghiêng.

- **Phát hiện biên (Edge Detection):** Sử dụng thuật toán **Canny** với ngưỡng (threshold) thích hợp để phát hiện các cạnh trong ảnh.
- **Tìm vùng bài thi (ROI Localization):**
 - Sử dụng thuật toán tìm đường bao (`findContours`) để xác định các đối tượng khép kín.
 - Lọc và chọn contour có diện tích lớn nhất và có 4 đỉnh (xấp xỉ đa giác bằng `approxPolyDP`). Đây được giả định là khung viền của tờ phiếu thi.
- **Biến đổi phối cảnh (Perspective Transform):**
 - Xác định tọa độ 4 góc của tờ phiếu trong ảnh gốc.
 - Sử dụng ma trận biến đổi phối cảnh (**Warp Perspective**) để ánh xạ vùng ảnh phiếu thi về một hình chữ nhật phẳng (Top-down view), loại bỏ hoàn toàn biến dạng phối cảnh.

4.2.3 Giai đoạn 3: Trích xuất và Phân tích đặc trưng

Giai đoạn này tập trung vào việc xác định vị trí các ô trả lời và trạng thái tô của chúng.

- **Nhị phân hóa (Binarization):** Áp dụng phân ngưỡng thích nghi (**Adaptive Thresholding**) hoặc Otsu để tách biệt nét bút chì/mực (Foreground) ra khỏi nền giấy (Background). Ảnh kết quả là ảnh nhị phân (Đen/Trắng).

- **Xác định vùng đáp án (Zone Segmentation):**

- Dựa trên cấu trúc mẫu phiếu, ảnh được chia cắt thành các vùng cột câu hỏi.
- Sử dụng phương pháp **Template Grid Mapping**: Áp dụng lưới tọa độ định sẵn (được sinh ra từ các điểm mỏ neo) để xác định vị trí tâm của từng ô tròn trắc nghiệm (A, B, C, D).

- **Đếm điểm ảnh (Pixel Counting)**: Tại mỗi vị trí ô tròn, hệ thống sử dụng mặt nạ (Mask) để đếm số lượng điểm ảnh trắng (Non-zero pixels). Đây là chỉ số quan trọng để xác định ô đó có được tô hay không.

4.2.4 Giai đoạn 4: Ra quyết định và Kết xuất

- **Logic chọn đáp án (Scoring Logic):**

- So sánh số lượng pixel của các ô trong cùng một câu hỏi.
- Ô có số pixel lớn nhất và vượt qua ngưỡng nhiễu (Noise Threshold) được xác định là đáp án người dùng chọn.

- **Chấm điểm và Báo cáo:**

- So khớp đáp án người dùng với bộ đáp án chuẩn (Answer Key) để tính điểm.
- **Trực quan hóa**: Vẽ vòng tròn màu xanh (Đúng) hoặc đỏ (Sai) lên ảnh bài thi để người dùng dễ dàng kiểm tra.
- **Xuất dữ liệu**: Kết quả chấm được lưu trữ và xuất ra định dạng Excel (.xlsx) phục vụ công tác quản lý.

5 Hướng dẫn sử dụng

5.1 Bước 1: Chuẩn bị dữ liệu

Đặt file ảnh chụp ('.jpg', '.png') hoặc file scan ('.pdf') của phiếu trả lời vào thư mục:
data/raw/

5.2 Bước 2: Cấu hình đường dẫn

Mở file main.py và cập nhật biến IMAGE_PATH trở nên tên file bạn vừa chép vào.

```
1 IMAGE_PATH = "data/raw/OMR_Sample.pdf"
2
```

5.3 Bước 3: Chạy chương trình

Thực thi lệnh sau trong terminal:

```
1 python main.py
2
```

Chương trình sẽ tiến hành xử lý, hiển thị các cửa sổ gồm ảnh gốc, ảnh đã cắt, và kết quả chấm điểm. Kết quả chi tiết cũng sẽ được in ra màn hình console.

6 Hạn chế và Hướng phát triển

6.1 Hạn chế hiện tại

Mặc dù hệ thống hoạt động ổn định trên các mẫu thử nghiệm, nhưng vẫn tồn tại một số hạn chế:

- **Độ nhạy sáng:** Kết quả nhị phân hóa (Thresholding) có thể bị ảnh hưởng nếu ảnh chụp có bóng đổ quá lớn hoặc ánh sáng không đều.
- **Cấu trúc cố định:** Hiện tại logic chấm điểm đang được cấu hình cứng cho mẫu phiếu 20 câu. Việc thay đổi sang mẫu 50 hay 100 câu cần can thiệp vào code (file `main.py`).
- **Xử lý đơn trang:** Hệ thống chưa hỗ trợ tự động tách và xử lý hàng loạt trang trong một file PDF nhiều trang.