Báo cáo Phân tích Chi tiết Dự án: Hệ thống Chấm điểm Trắc nghiệm và Nhận dạng Ký tự Quang học Tự động (OMR & OCR)

Nhóm thực hiện: Nguyễn Thế Minh Nhật

Ngày 17 tháng 12 năm 2025

Mục lục

1 Giới thiệu Tổng quan Dự án

Trong bối cảnh chuyển đổi số giáo dục đang diễn ra mạnh mẽ, việc tự động hóa các quy trình thủ công như chấm thi và quản lý thông tin ngày càng trở nên cấp thiết. Dự án này ra đời nhằm giải quyết thách thức đó bằng cách xây dựng một hệ thống tích hợp thông minh, kết hợp cả Nhận dạng Dấu hiệu Quang học (OMR) và Nhận dạng Ký tự Quang học (OCR). Tầm quan trọng chiến lược của giải pháp này nằm ở khả năng tăng cường đáng kể hiệu suất, giảm thiểu sai sót do con người gây ra và tiết kiệm hàng giờ lao động cho các nhà giáo dục.

Mục tiêu cốt lõi của dự án là xây dựng một giải pháp toàn diện để tự động hóa hai nhiệm vụ chính: chấm điểm các phiếu trả lời trắc nghiệm (OMR) và trích xuất thông tin văn bản như tên, mã số sinh viên (OCR) từ các tệp PDF được quét. Hệ thống được thiết kế để tối ưu hóa độ chính xác và cung cấp kết quả một cách nhanh chóng, trực quan.

1.1 Các tính năng chính

- **Xử lý đầu vào từ file PDF:** Hệ thống có khả năng làm việc trực tiếp với định dạng PDF, một định dạng tài liệu quét phổ biến nhất hiện nay. Điều này mang lại sự tiện lợi tối đa cho người dùng, loại bỏ các bước chuyển đổi định dạng thủ công.
- **Tự động phát hiện và căn chỉnh tài liệu:** Một trong những tính năng thông minh nhất là khả năng tự động tìm kiếm khung viền của phiếu trả lời trong ảnh quét. Sau khi phát hiện, hệ thống áp dụng biến đổi phối cảnh để "làm phẳng" tài liệu, loại bỏ hiệu quả các biến dạng do góc quét hoặc giấy bị cong, đảm bảo độ chính xác cho các bước xử lý sau.
- **Cơ chế thiết lập tọa độ Hybrid:** Đây là một quyết định thiết kế chiến lược. Hệ thống kết hợp thông minh giữa việc thiết lập thủ công một lần cho OMR và tự động hoàn toàn cho OCR. Khối đáp án OMR có cấu trúc lưới cứng nhắc và có thể dự đoán, do đó việc nội suy tọa độ toàn bộ lưới chỉ từ hai điểm neo do người dùng chọn là phương pháp cực kỳ hiệu quả. Ngược lại, các trường thông tin OCR (tên, mã số) có thể thay đổi về vị trí và số lượng trên các mẫu phiếu khác nhau. Vì vậy, việc sử dụng thuật toán phát hiện đường viền tự động là một giải pháp linh hoạt và mạnh mẽ hơn, giúp hệ thống thích ứng mà không cần viết lại mã nguồn.
- **Chấm điểm OMR chính xác:** Dựa trên tọa độ đã thiết lập, hệ thống quét các ô trả lời, xác định lựa chọn của học sinh và so sánh với đáp án chuẩn để tính điểm cuối cùng.
- **Trích xuất thông tin OCR:** Hệ thống sử dụng thư viện EasyOCR để đọc và trích xuất nội dung văn bản từ các vùng được phát hiện tự động, cho phép số hóa các thông tin quan trọng của thí sinh.
- **Trực quan hóa và lưu trữ kết quả:** Hệ thống không chỉ trả về dữ liệu thô mà còn tạo ra các tệp đầu ra hữu ích, bao gồm ảnh phiếu trả lời đã được chấm điểm (đánh dấu đúng/sai), ảnh báo cáo điểm số và tệp JSON chứa dữ liệu OCR.

1.2 Các công nghệ sử dụng

Để hiện thực hóa các tính năng trên, dự án đã tận dụng một bộ công cụ và thư viện mạnh mẽ trong hệ sinh thái Python.

Công nghệ/Thư viện	Vai trò trong Dự án
Python	Ngôn ngữ lập trình chính, đóng vai trò kết nối tất cả các thành phần của hệ thống.
OpenCV	Thư viện thị giác máy tính cốt lõi, được sử dụng cho hầu hết các tác vụ từ tiền xử lý, phát hiện cạnh, biến đổi phối cảnh.
NumPy	Nền tảng cho tính toán khoa học, cung cấp cấu trúc mảng hiệu suất cao để biểu diễn và thao tác trên ma trận điểm ảnh.
EasyOCR	Thư viện OCR mạnh mẽ được sử dụng để nhận dạng và trích xuất văn bản, hỗ trợ đa ngôn ngữ bao gồm cả tiếng Việt.
pdf2image	Tiện ích giúp chuyển đổi các trang của tệp PDF thành đối tượng hình ảnh mà OpenCV có thể xử lý.
Poppler	Thư viện render PDF, là một phụ thuộc hệ thống bắt buộc để pdf2image có thể hoạt động.

Để hiểu cách các công nghệ này được tổ chức và phối hợp, phần tiếp theo sẽ đi sâu vào cấu trúc thư mục và mã nguồn của dự án.

2 Phân tích Cấu trúc Dự án

Một cấu trúc dự án được tổ chức tốt là nền tảng cho việc đảm bảo tính dễ bảo trì, khả năng mở rộng và sự rõ ràng của mã nguồn. Dự án này tuân thủ các nguyên tắc thiết kế module hóa, tách biệt rõ ràng các chức năng vào các tệp và thư mục riêng biệt.

Cấu trúc cây thư mục của dự án được trình bày như sau:

`main.py`: Tệp thực thi chính của chương trình. Nó đóng vai trò là bộ điều phối, kiểm soát toàn bộ luồng hoạt động từ việc kiểm tra cấu hình, gọi các module xử lý cho đến khi hiển thị kết quả cuối cùng.

`requirements.txt`: Tệp văn bản liệt kê tất cả các thư viện Python cần thiết để dự án có thể hoạt động. Điều này giúp đơn giản hóa quá trình cài đặt môi trường.

`src/`: Thư mục chứa toàn bộ mã nguồn cốt lõi của ứng dụng, được chia thành các module chức năng.

- `config.py`: Tệp cấu hình tập trung. Chứa tất cả các đường dẫn tệp, tham số xử lý ảnh và các hằng số quan trọng, giúp việc tùy chỉnh trở nên dễ dàng mà không cần sửa đổi logic chính.
- `pre_processing.py`: Chịu trách nhiệm cho giai đoạn tiền xử lý, bao gồm đọc tệp PDF, phát hiện đường viền tài liệu và thực hiện biến đổi phối cảnh để làm phẳng ảnh.
- `omr_logic.py`: Chứa logic chuyên biệt cho việc chấm điểm trắc nghiệm, từ việc tạo lưới tọa độ các ô đáp án đến việc phân tích điểm ảnh để xác định câu trả lời.

- `ocr_logic.py`: Quản lý việc trích xuất văn bản bằng cách sử dụng EasyOCR trên các vùng ảnh được chỉ định.
- `measure_tool.py`: Một công cụ tương tác, chỉ được kích hoạt trong lần chạy đầu tiên để người dùng thiết lập các điểm neo cho khối OMR và tự động phát hiện vùng OCR.
- `utils.py`: Chứa các hàm tiện ích chung được sử dụng bởi nhiều module khác, chẳng hạn như hàm sắp xếp điểm và hàm biến đổi hình học.
- `ui.py`: Chịu trách nhiệm cho tất cả các tương tác với người dùng, bao gồm hiển thị hộp thoại, vẽ kết quả lên ảnh và hiển thị cửa sổ kết quả cuối cùng.

`data/`: Thư mục lưu trữ tất cả dữ liệu đầu vào.

- `raw/`: Nơi chứa tệp PDF phiếu trả lời cần xử lý.
- `answer/`: Chứa tệp answer_key.csv với đáp án đúng cho phần trắc nghiệm.
- `template/`: Lưu trữ tệp coordinates.json sau khi người dùng thiết lập tọa độ lần đầu.

`output/`: Thư mục nơi tất cả các kết quả đầu ra (ảnh, tệp JSON) được lưu trữ.

`poppler-25.11.0/`: Chứa thư viện Poppler cần thiết để xử lý tệp PDF trên hệ điều hành Windows.

Sau khi đã hiểu các thành phần riêng lẻ và vai trò của chúng, phần tiếp theo sẽ trình bày cách chúng phối hợp với nhau trong một luồng xử lý hoàn chỉnh.

3 Luồng Hoạt động của Hệ thống

Luồng xử lý dữ liệu từ đầu đến cuối là xương sống của ứng dụng. Phần này sẽ trình bày chi tiết từng bước mà hệ thống thực hiện, từ khi nhận tệp PDF đầu vào cho đến khi xuất ra kết quả cuối cùng, thông qua hai kịch bản hoạt động chính.

3.1 Kịch bản 1: Thiết lập Lần đầu (First-Time Setup)

Kịch bản này xảy ra khi hệ thống được chạy lần đầu tiên và không tìm thấy tệp cấu hình tọa độ (coordinates.json).

1. **Khởi tạo:** Tệp main.py kiểm tra sự tồn tại của đường dẫn config.COORDINATES_PATH. Khi không tìm thấy, nó gọi hàm ui.prompt_coordinate_setup() để hiển thị một hộp thoại, hỏi người dùng có muốn bắt đầu quá trình thiết lập hay không.

2. **Tải và Xử lý Ảnh:** Nếu người dùng đồng ý, hàm measure_tool.get_coordinates_and_rois() được gọi. Bên trong hàm này, pre_processing.get_warped_image_from_pdf() được thực thi để chuyển đổi PDF thành ảnh, phát hiện, căn chỉnh và trả về hai hình ảnh quan trọng: ảnh phiếu trả lời đã được làm phẳng (warped_standard) và ảnh chứa các vùng bên ngoài phiếu (outside_image).

3. **Giai đoạn 1 - Thiết lập OMR (Thủ công):** Một cửa sổ tương tác hiển thị ảnh warped_standard. Người dùng được hướng dẫn click chuột vào tâm của ô đáp án đầu tiên (góc trên-trái, ví dụ: Câu 1, Lựa chọn A) và sau đó là tâm của ô đáp án cuối cùng (góc dưới-phải) của khối trắc nghiệm.

4. **Giai đoạn 2 - Phát hiện Vùng OCR (Tự động):** Ngay sau khi hai điểm neo được chọn, cửa sổ tương tác sẽ đóng lại. Hệ thống tự động xử lý ảnh outside_image ở chế độ nền. Nó sử dụng các thuật toán xử lý ảnh như cv2.threshold và cv2.findContours để tự động xác định các đường viền bao quanh các khối văn bản (như tên, MSSV, mã đề).

5. **Lưu trữ Tọa độ:** Tọa độ của hai điểm neo OMR và các hình chữ nhật bao quanh các vùng OCR được phát hiện sẽ được cấu trúc và lưu vào tệp coordinates.json. Tệp này sẽ được tái sử dụng cho tất cả các lần chạy sau.

3.2 Kịch bản 2: Vận hành Thông thường (Subsequent Runs)

Khi tệp coordinates.json đã tồn tại, hệ thống sẽ hoạt động ở chế độ hoàn toàn tự động.

1. **Tải Cấu hình:** main.py đọc tệp coordinates.json để lấy ra tọa độ hai điểm neo cho OMR (bubble_anchors) và danh sách các vùng chữ nhật cho OCR (ocr_regions).

2. **Xử lý PDF:** Tương tự như kịch bản 1, hàm pre_processing.get_warped_image_from_pdf() được gọi để lấy ảnh phiếu trả lời đã được căn chỉnh (warped_img) và ảnh vùng bên ngoài (outside_img).

3. **Thực thi Chấm điểm:** Hàm điều phối chính run_grading_process() được gọi, truyền vào các tọa độ và hình ảnh đã được chuẩn bị.

4. **Xử lý OMR:** Bên trong run_grading_process(), các hàm logic OMR được thực thi tuần tự:

   - `omr_logic.generate_column_coordinates` sử dụng hai điểm neo để nội suy và tính toán tọa độ của tất cả các ô đáp án còn lại.
   - `omr_logic.grade_with_coordinates` phân tích điểm ảnh tại mỗi tọa độ để xác định câu trả lời của học sinh.
   - `omr_logic.calculate_score` so sánh kết quả với đáp án và tính điểm cuối cùng.

5. **Xử lý OCR:** Hàm ocr_logic.extract_text_from_regions được gọi, nhận vào ảnh outside_img và danh sách các vùng OCR đã tải từ tệp cấu hình để trích xuất văn bản.

6. **Tổng hợp và Hiển thị:** Cuối cùng, ui.save_output_files được sử dụng để lưu tất cả các tệp kết quả (ảnh, JSON) vào thư mục output/, và ui.show_final_results hiển thị các ảnh kết quả trên màn hình đồng thời in thông tin OCR ra console.

4 Cài đặt và Hướng dẫn sử dụng

4.1 Yêu cầu về môi trường

1. **Python:** Phiên bản 3.8 trở lên.

2. **Poppler:** Do hệ thống cần xử lý file PDF, Poppler là một yêu cầu bắt buộc.

   - Dự án đã đi kèm thư mục `poppler-25.11.0` cho Windows.

- **Quan trọng:** Người dùng cần thêm đường dẫn đầy đủ tới thư mục `poppler-25.11.0/Librar` vào biến môi trường `Path` của hệ thống. Nếu không có bước này, chương trình sẽ báo lỗi không tìm thấy Poppler.

4.2 Cài đặt các thư viện Python

Mở một cửa sổ dòng lệnh (terminal) trong thư mục gốc của dự án và chạy lệnh sau để cài đặt tất cả các thư viện cần thiết:

```
pip install -r requirements.txt

```

Lưu ý: Trong lần chạy đầu tiên, thư viện ***`easyocr`*** *có thể sẽ tự động tải về các model nhận dạng. Đây là quá trình chỉ diễn ra một lần và cần có kết nối Internet.*

4.3 Cấu hình hệ thống

Tất cả các cấu hình quan trọng đều được tập trung tại file `src/config.py`.

- `PDF_PATH`: Đường dẫn tới file PDF phiếu làm bài đầu vào.
- `ANSWER_KEY_PATH`: Đường dẫn tới file `.csv` chứa đáp án của phần trắc nghiệm.
- `COORDINATES_PATH`: Đường dẫn để lưu file template tọa độ sau lần thiết lập đầu tiên.

Người dùng cần đảm bảo các file dữ liệu (phiếu làm bài, đáp án) được đặt đúng vị trí và các đường dẫn trong file config là chính xác.

4.4 Hướng dẫn vận hành

1. **Chạy chương trình:**

```
python main.py

```

2. **Thiết lập lần đầu (First-Time Setup):** Nếu hệ thống không tìm thấy file tọa độ template, một hộp thoại sẽ hiện lên hỏi quyền được bắt đầu quá trình thiết lập.
   - **Giai đoạn 1 (Thủ công):** Một cửa sổ hiển thị ảnh phiếu sẽ mở ra. Người dùng cần click chuột vào 2 vị trí theo hướng dẫn trên console: tâm của ô đáp án **trên cùng bên trái** và tâm của ô **dưới cùng bên phải** của khối trắc nghiệm.
   - **Giai đoạn 2 (Tự động):** Ngay sau khi 2 điểm được chọn, cửa sổ sẽ đóng lại. Hệ thống sẽ tự động chạy nền để tìm các vùng văn bản OCR.
   - Toàn bộ tọa độ sau đó được lưu vào file `coordinates.json`.
3. **Các lần chạy sau:** Chương trình sẽ tự động đọc file `coordinates.json` đã lưu và thực hiện toàn bộ quy trình chấm điểm, nhận dạng mà không cần bất kỳ tương tác nào.

5 Phân tích Chi tiết Mã nguồn theo Module

Đây là phần cốt lõi của báo cáo, đi sâu phân tích từng module mã nguồn để làm rõ các thuật toán, cấu trúc dữ liệu và các quyết định thiết kế quan trọng.

5.1 Module Tiền xử lý (src/pre_processing.py và src/utils.py)

- **Mục đích:** Module này đóng vai trò là bước chuẩn bị dữ liệu đầu vào. Nhiệm vụ của nó là biến đổi một ảnh quét có thể bị nghiêng, méo hoặc chụp từ một góc thành một ảnh phẳng, được chuẩn hóa về kích thước và sẵn sàng cho việc phân tích OMR và OCR.

- **Phân tích các hàm chính:**

  - `get_warped_image_from_pdf()`: Đây là hàm chính của module, thực hiện một chuỗi các bước xử lý:

    1. Chuyển đổi PDF: Sử dụng `convert_from_path` từ thư viện `pdf2image` để đọc trang đầu tiên của tệp PDF và chuyển nó thành một đối tượng ảnh mà OpenCV có thể xử lý.
    2. Xử lý ảnh cơ bản: Gọi hàm `preprocess_image()` để thực hiện một chuỗi xử lý nhân quả: đầu tiên, chuyển đổi sang ảnh xám để đơn giản hóa việc xử lý; sau đó áp dụng làm mờ Gaussian để giảm nhiễu, một bước quan trọng để thuật toán Canny hoạt động ổn định hơn; và cuối cùng là phát hiện cạnh bằng Canny để làm nổi bật các đường viền chính của tài liệu.
    3. Tìm đường viền tài liệu: Gọi hàm `find_document_contour()` để tìm đường viền lớn nhất có 4 đỉnh trong ảnh cạnh, được giả định là đường viền của trang giấy.
    4. Tách vùng bên ngoài: Sử dụng `utils.get_outside_of_contour()` để tạo một mặt nạ từ đường viền đã tìm thấy, sau đó tách biệt vùng ảnh nằm bên ngoài phiếu trả lời. Vùng này sẽ được dùng cho OCR.
    5. Biến đổi phối cảnh: Sử dụng `utils.four_point_transform()` để "cắt" và "kéo" 4 đỉnh của đường viền tài liệu thành một hình chữ nhật hoàn hảo, tạo ra một ảnh phẳng, không bị biến dạng.

  - `four_point_transform()` (từ `utils.py`): Đây là một thuật toán kinh điển trong xử lý ảnh. Nó yêu cầu sắp xếp 4 điểm đầu vào theo một thứ tự nhất quán (trên-trái, trên-phải, dưới-phải, dưới-trái) thông qua hàm `order_points`. Sau đó, nó tính toán ma trận biến đổi phối cảnh bằng `cv2.getPerspectiveTransform` và áp dụng ma trận này lên ảnh gốc để tạo ra ảnh kết quả đã được làm phẳng.

5.2 Module Công cụ Thiết lập (src/measure_tool.py)

- **Mục đích:** Module này hoạt động như một công cụ tương tác, chỉ chạy một lần, cho phép người dùng thiết lập hệ thống một cách linh hoạt. Nó tạo ra một "template" tọa độ có thể tái sử dụng, giúp hệ thống thích ứng với các mẫu phiếu trả lời khác nhau mà không cần phải viết lại mã nguồn.

- **Phân tích luồng hoạt động:**

– **Giai đoạn 1 - Lấy điểm neo OMR:** Hàm `anchor_point_callback` được liên kết với sự kiện click chuột (`cv2.EVENT_LBUTTONDOWN`). Khi người dùng click, tọa độ con trỏ chuột được ghi lại. Hàm này chỉ cho phép ghi lại đúng hai tọa độ, tương ứng với hai điểm neo cần thiết cho khối OMR.
– **Giai đoạn 2 - Tự động phát hiện vùng OCR:** Logic này thể hiện một quyết định kiến trúc quan trọng. Sau khi hai điểm neo OMR được chọn, hệ thống lấy ảnh `outside_image` (vùng bên ngoài phiếu trả lời) được trả về từ module tiền xử lý. Việc tách biệt này (separation of concerns) cực kỳ hiệu quả: nó ngăn logic phát hiện OCR bị nhầm lẫn bởi các đường kẻ và ô tròn bên trong phiếu trả lời, cho phép nó tập trung hoàn toàn vào các vùng thông tin liên quan. Hàm chuyển ảnh sang nhị phân (`cv2.threshold`), tìm các đường viền (`cv2.findContours`), và sau đó lọc các đường viền theo diện tích để tự động xác định các vùng văn bản.
– **Lưu trữ:** Dữ liệu cuối cùng, bao gồm hai điểm neo OMR và danh sách các hình chữ nhật OCR, được cấu trúc thành một dictionary và lưu vào tệp `coordinates.json`.

6 Kết quả thực nghiệm

6.1 Dữ liệu đầu vào

Hệ thống được thử nghiệm với file `Mau_de_thi_co_dap_an.pdf`, là một file scan phiếu trả lời trắc nghiệm có kèm các thông tin của thí sinh.

6.2 Kết quả xử lý OMR

Sau khi xử lý, hệ thống tạo ra hai ảnh kết quả cho phần OMR:

- `scoring_result.png`: Ảnh phiếu làm bài với các đáp án được khoanh tròn. Màu xanh cho đáp án đúng, màu đỏ cho đáp án sai. Nếu sai, một chấm xanh nhỏ sẽ được đặt ở đáp án đúng.
- `score.png`: Một ảnh riêng biệt hiển thị điểm số cuối cùng và tổng số câu trả lời đúng.

(Ví dụ về các hình ảnh kết quả có thể được chèn vào đây để minh họa)

6.3 Kết quả xử lý OCR

Đối với phần OCR, các kết quả bao gồm:

- `outside_area.png`: Đây là ảnh trực quan hóa vùng được sử dụng cho việc nhận dạng OCR. Nó là ảnh gốc với phần phiếu trắc nghiệm đã bị tô đen, chỉ để lại các vùng chứa thông tin văn bản.
- `ocr_results.json`: File JSON chứa kết quả văn bản được trích xuất. Mỗi vùng văn bản được tự động phát hiện sẽ tương ứng với một khóa (ví dụ: `"auto_roi_1"`) và giá trị là chuỗi ký tự đọc được.

Ví dụ nội dung file `ocr_results.json`:

```
{
    "auto_roi_1": "Ho va Ten: Nguyen Van A",
    "auto_roi_2": "SBD: 123456"
}

```

6.4 Đánh giá

- **Độ chính xác OMR:** Phụ thuộc nhiều vào chất lượng ảnh scan và độ đậm của nét tô. Với các ảnh scan rõ nét, hệ thống cho độ chính xác gần như tuyệt đối.
- **Độ chính xác OCR:** Phụ thuộc vào chất lượng của thư viện EasyOCR và độ rõ của chữ viết tay hoặc chữ in. Kết quả tốt với chữ in và có thể giảm độ chính xác với chữ viết tay quá nghệ thuật hoặc mờ.
- **Hiệu năng:** Thời gian xử lý cho mỗi phiếu tương đối nhanh, chủ yếu phụ thuộc vào thời gian tải model của EasyOCR trong lần đầu và tốc độ xử lý của CPU.

7 Kết luận và Hướng phát triển

7.1 Kết luận

Đồ án đã xây dựng thành công một hệ thống hoàn chỉnh có khả năng tự động hóa hai tác vụ quan trọng trong chấm thi: chấm điểm trắc nghiệm (OMR) và trích xuất thông tin thí sinh (OCR). Bằng cách kết hợp các thuật toán xử lý ảnh kinh điển của OpenCV và sức mạnh của model học sâu từ EasyOCR, hệ thống đã giải quyết được bài toán đặt ra với các kết quả chính:

- Tự động hóa toàn bộ quy trình từ đọc file PDF đến xuất kết quả.
- Giảm thiểu tối đa sự can thiệp của người dùng nhờ cơ chế thiết lập template thông minh và tự động phát hiện vùng OCR.
- Cung cấp kết quả đầu ra đa dạng, trực quan và dễ dàng lưu trữ, quản lý.

Dự án là một minh chứng rõ ràng cho tiềm năng của Xử lý ảnh trong việc giải quyết các bài toán thực tế trong lĩnh vực giáo dục và xa hơn nữa.

7.2 Hạn chế của hệ thống

- **Phụ thuộc vào cấu trúc phiếu:** Logic sinh tọa độ OMR hiện tại được thiết kế cho một mẫu phiếu nhất định. Việc thay đổi sang một mẫu phiếu có bố cục hoàn toàn khác (ví dụ: 2 cột câu hỏi) sẽ cần can thiệp vào mã nguồn.
- **Chất lượng ảnh hưởng lớn:** Chất lượng của ảnh scan (độ phân giải, độ tương phản, ánh sáng) ảnh hưởng trực tiếp đến độ chính xác của cả OMR và OCR.
- **Giao diện người dùng:** Hệ thống vẫn hoạt động chủ yếu trên giao diện dòng lệnh và các cửa sổ của OpenCV, chưa có một giao diện đồ họa (GUI) hoàn chỉnh.

7.3 Hướng phát triển

- **Xây dựng giao diện đồ họa (GUI):** Phát triển một ứng dụng desktop (sử dụng PyQt, Tkinter) hoặc web (sử dụng Flask, Django) để người dùng có thể dễ dàng tải lên file, cấu hình và xem kết quả một cách thân thiện hơn.
- **Nâng cao khả năng thích ứng:** Phát triển các thuật toán thông minh hơn để hệ thống có thể tự động nhận dạng bố cục của các loại phiếu trắc nghiệm khác nhau mà không cần cấu hình lại.
- **Xử lý theo lô (Batch Processing):** Cho phép người dùng tải lên nhiều file cùng lúc và hệ thống tự động xử lý hàng loạt.
- **Cải thiện độ chính xác OCR:** Thử nghiệm tích hợp các engine OCR khác hoặc tinh chỉnh model để cải thiện độ chính xác với chữ viết tay.