BÁO CÁO ĐỒ ÁN MÔN HỌC
HỆ THỐNG CHẤM THI VÀ SỐ HÓA THÔNG TIN TỰ ĐỘNG (SMARTGRADER)

Nhóm thực hiện: Nhóm 10 - Nguyễn Thế Minh Nhật

Mục lục

Chương 1

TỔNG QUAN ĐỀ TÀI

1.1 Đặt vấn đề

Trong bối cảnh chuyển đổi số giáo dục hiện nay, hình thức thi trắc nghiệm khách quan ngày càng trở nên phổ biến nhờ tính chính xác, công bằng và khả năng đánh giá diện rộng. Tuy nhiên, quy trình chấm thi truyền thống đang tồn tại hai rào cản lớn:

1. **Chấm thủ công:** Tốn nhiều thời gian, nhân lực và dễ xảy ra sai sót do yếu tố con người, đặc biệt khi số lượng bài thi lớn.
2. **Máy chấm thi chuyên dụng:** Mặc dù có tốc độ cao nhưng chi phí đầu tư thiết bị phần cứng rất đắt đỏ, đòi hỏi giấy thi đặc thù và khó triển khai linh hoạt tại các cơ sở giáo dục quy mô nhỏ hoặc trong các bài kiểm tra thường xuyên.

Xuất phát từ thực tế đó, việc phát triển một giải pháp phần mềm có khả năng tận dụng các thiết bị sẵn có (như camera điện thoại, máy scan văn phòng) để chấm thi là vô cùng cấp thiết. Đề tài **"SmartGrader - Hệ thống chấm thi và số hóa thông tin tự động"** được xây dựng nhằm giải quyết bài toán trên bằng cách ứng dụng các kỹ thuật Tiên tiến của Thị giác máy tính (Computer Vision) và Học sâu (Deep Learning).

1.2 Mục tiêu đề tài

Mục tiêu chính của đề tài là xây dựng một hệ thống phần mềm mã nguồn mở (Open Source) có khả năng tự động hóa quy trình chấm thi trắc nghiệm với độ chính xác cao và chi phí thấp. Các mục tiêu cụ thể bao gồm:

- **Tự động hóa quy trình tạo đề (Auto Sheet Generator):** Xây dựng module sinh phiếu thi chuẩn dưới dạng PDF bằng mã lệnh, đảm bảo tạo ra "Ground Truth"(tọa độ đáp án) chính xác tuyệt đối, loại bỏ sai số so với thiết kế thủ công [1].
- **Xử lý ảnh và Chấm điểm (Robust OMR Engine):** Phát triển thuật toán nhận dạng đánh dấu quang học (OMR) có khả năng thích nghi với điều kiện môi trường khắc nghiệt như: ánh sáng không đều, bóng đổ, hoặc giấy bị nghiêng, ố màu thông qua kỹ thuật Phân ngưỡng thích nghi (Adaptive Thresholding) [2].
- **Số hóa thông tin thí sinh (OCR & Validation):** Tích hợp mô hình Deep Learning (Easy-OCR) để nhận dạng chữ viết tay và chữ in (Tên, Số báo danh), kết hợp với các biểu thức chính quy (Regex) để làm sạch và chuẩn hóa dữ liệu đầu ra [3].

- **Xuất báo cáo tự động:** Hỗ trợ xuất kết quả dưới dạng hình ảnh trực quan (có đánh dấu Đúng/Sai) và file dữ liệu số để lưu trữ.

1.3 Phạm vi và Đối tượng nghiên cứu

1.3.1 Phạm vi nghiên cứu

Đề tài tập trung giải quyết các vấn đề kỹ thuật sau:

- Xử lý đầu vào là các tệp tin hình ảnh (.jpg, .png) hoặc tài liệu quét (.pdf) chứa phiếu trả lời trắc nghiệm.
- Hệ thống được xây dựng bằng ngôn ngữ **Python**, sử dụng các thư viện lõi gồm: **OpenCV** (xử lý ảnh), **EasyOCR** (nhận dạng chữ), và **ReportLab** (tạo phiếu thi).
- Giới hạn xử lý trên các phiếu thi có cấu trúc định dạng trước (Template-based), yêu cầu ảnh đầu vào phải nhìn thấy rõ 4 góc định vị của phiếu thi để thực hiện phép biến đổi hình học (Warp Perspective) [4].

1.3.2 Đối tượng nghiên cứu

- Các thuật toán xử lý ảnh: Canny Edge Detection, Contour Retrieval, Perspective Transform.
- Các kỹ thuật xử lý nhị phân: Otsu's Binarization, Adaptive Gaussian Thresholding.
- Mô hình nhận dạng quang học (OCR) hỗ trợ tiếng Việt và tiếng Anh.

1.4 Phương pháp tiếp cận

Hệ thống được thiết kế theo kiến trúc **MVC (Model-View-Controller)** để đảm bảo tính module hóa và dễ dàng bảo trì:

1. **Model (Xử lý dữ liệu):** Bao gồm các engine nòng cốt như `omr_engine.py` (xử lý chấm điểm trắc nghiệm) và `ocr_engine.py` (xử lý nhận dạng ký tự). Sử dụng phương pháp định hướng dữ liệu (Data-driven) thông qua file cấu hình `coordinates.json` và `config.py`.
2. **View (Hiển thị):** Module `renderer.py` chịu trách nhiệm vẽ lại kết quả chấm lên ảnh gốc và hiển thị giao diện kết quả.
3. **Controller (Điều phối):** Module `main.py` đóng vai trò điều phối luồng dữ liệu từ lúc đọc ảnh, gọi các thuật toán xử lý đến khi xuất báo cáo cuối cùng.

1.5 Cấu trúc báo cáo

Báo cáo được trình bày trong 6 chương với nội dung cụ thể như sau:

- **Chương 1: Tổng quan đề tài.** Trình bày lý do, mục tiêu, phạm vi và phương pháp tiếp cận của dự án SmartGrader.

- **Chương 2: Cơ sở lý thuyết.** Giới thiệu các kiến thức nền tảng về Xử lý ảnh (Grayscale, Edge Detection, Warping) và công nghệ OCR.
- **Chương 3: Phân tích và Thiết kế hệ thống.** Mô tả chi tiết kiến trúc phần mềm, sơ đồ luồng dữ liệu (Flowchart) và cấu trúc dữ liệu đầu vào/đầu ra.
- **Chương 4: Cài đặt và Hiện thực.** Đi sâu vào chi tiết cài đặt các thuật toán quan trọng: Sinh đề tự động, Chống nhiễu bằng Adaptive Threshold, và Hậu xử lý dữ liệu OCR bằng Regex.
- **Chương 5: Kết quả thực nghiệm.** Trình bày các kịch bản kiểm thử, hình ảnh kết quả thực tế và đánh giá độ chính xác của hệ thống.
- **Chương 6: Kết luận và Hướng phát triển.** Tổng kết các kết quả đạt được, phân tích hạn chế và đề xuất hướng cải tiến (tích hợp YOLO, GUI).

Chương 2

CƠ SỞ LÝ THUYẾT VÀ CÔNG NGHỆ SỬ DỤNG

Chương này trình bày các cơ sở lý thuyết về xử lý ảnh số (Digital Image Processing) và nhận dạng quang học (OCR) được áp dụng trong hệ thống SmartGrader. Đồng thời, chương cũng giới thiệu các thư viện và công cụ lập trình chính được sử dụng để hiện thực hóa các giải pháp kỹ thuật.

2.1 Tổng quan về Xử lý ảnh số (Digital Image Processing)

Trong dự án này, xử lý ảnh đóng vai trò tiền xử lý (preprocessing) quan trọng để chuẩn hóa dữ liệu đầu vào trước khi đưa vào các mô hình nhận dạng. Các kỹ thuật chính bao gồm:

2.1.1 Ảnh xám (Grayscale)

Ảnh xám là ảnh mà mỗi điểm ảnh (pixel) chỉ mang thông tin về cường độ ánh sáng, không có thông tin về màu sắc. Giá trị cường độ thường nằm trong khoảng từ 0 (đen) đến 255 (trắng). Việc chuyển đổi từ ảnh màu (RGB) sang ảnh xám giúp giảm khối lượng dữ liệu tính toán (từ 3 kênh màu xuống 1 kênh) và loại bỏ nhiễu màu không cần thiết.

Trong mã nguồn (`src/utils/image_utils.py`), quá trình này sử dụng công thức chuyển đổi tiêu chuẩn:

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \tag{2.1}$$

Hàm sử dụng: `cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)`.

2.1.2 Làm trơn ảnh (Gaussian Blur)

Để giảm nhiễu hạt (noise) thường xuất hiện khi chụp ảnh bằng điện thoại, hệ thống sử dụng bộ lọc Gaussian. Kỹ thuật này làm mờ ảnh bằng cách tích chập (convolution) ảnh gốc với một hạt nhân (kernel) Gaussian. Trong hàm `warp_document`, hệ thống sử dụng kernel kích thước $(5, 5)$ để loại bỏ các chi tiết thừa trước khi tìm biên, giúp thuật toán Canny hoạt động hiệu quả hơn.

2.1.3 Phát hiện biên (Canny Edge Detection)

Đây là thuật toán phổ biến để tìm ra các cạnh của đối tượng trong ảnh. Thuật toán hoạt động dựa trên việc tìm cực đại cục bộ của gradient (đạo hàm bậc nhất) cường độ sáng. Quy trình

Canny bao gồm các bước:

1. Khử nhiễu bằng Gaussian Blur.
2. Tính gradient cường độ và hướng của biên.
3. Non-maximum Suppression: Loại bỏ các điểm không phải cực đại.
4. Hysteresis Thresholding: Sử dụng hai ngưỡng (`threshold1` và `threshold2`) để lọc biên.

Trong dự án, các ngưỡng này được cấu hình trong file `config.py` với giá trị `CANNY_THRESHOLD_1 = 30` và `CANNY_THRESHOLD_2 = 100` để tối ưu việc bắt khung giấy thi.

2.1.4 Biến đổi phối cảnh (Perspective Transform)

Ảnh chụp từ thực tế thường bị nghiêng hoặc biến dạng hình thang. Kỹ thuật Perspective Transform (hay Warping) giúp "nắn"lại ảnh về góc nhìn trực diện (top-down view). Quá trình này yêu cầu xác định được 4 điểm góc của tờ giấy (tl, tr, br, bl). Ma trận biến đổi M kích thước 3×3 được tính toán sao cho:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = M \times \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.2}$$

Trong module `src/utils/image_utils.py`, hàm `four_point_transform` chịu trách nhiệm thực thi kỹ thuật này, đảm bảo các ô trắc nghiệm luôn nằm đúng vị trí tọa độ mẫu.

2.1.5 Phân ngưỡng thích nghi (Adaptive Thresholding)

Đây là kỹ thuật quan trọng nhất trong module chấm thi (`omr_engine.py`). Khác với phân ngưỡng toàn cục (Global Thresholding) sử dụng một giá trị ngưỡng duy nhất cho toàn bộ ảnh, Adaptive Thresholding tính toán ngưỡng riêng cho từng vùng nhỏ.

Trong dự án, phương pháp **Adaptive Gaussian** được sử dụng:

- Giá trị ngưỡng $T(x, y)$ là tổng trọng số Gaussian của các điểm lân cận trừ đi hằng số C.
- Cấu hình hiện tại: Block Size = 51, $C = 10$.

Ưu điểm: Kỹ thuật này giúp hệ thống hoạt động tốt ngay cả khi ảnh bị bóng đổ (shadow) hoặc ánh sáng không đều, đảm bảo tách biệt rõ ràng vết mực tô (đen) và nền giấy (trắng).

2.2 Nhận dạng quang học (OCR) và Xử lý ngôn ngữ

2.2.1 Mô hình EasyOCR

EasyOCR là một thư viện mã nguồn mở hỗ trợ nhận dạng văn bản trên hơn 80 ngôn ngữ, bao gồm tiếng Việt. Kiến trúc của EasyOCR là sự kết hợp của Deep Learning:

- **Detection (Phát hiện chữ):** Sử dụng mạng CRAFT (Character Region Awareness for Text Detection) để khoanh vùng vị trí văn bản.
- **Recognition (Nhận dạng):** Sử dụng mạng CRNN (Convolutional Recurrent Neural Network) kết hợp với LSTM và CTC Loss để đọc nội dung văn bản từ vùng đã cắt.

Trong module `ocr_engine.py`, EasyOCR được cấu hình để đọc song ngữ `['vi', 'en']`, phục vụ việc trích xuất tên thí sinh và mã đề.

2.2.2 Biểu thức chính quy (Regular Expression - Regex)

Do kết quả OCR thường bị nhiễu (ví dụ: nhận dạng nhầm vết bẩn thành dấu chấm), Regex được sử dụng như một bộ lọc hậu xử lý (Post-processing) để chuẩn hóa dữ liệu. Các mẫu Regex (Pattern) chính được sử dụng trong dự án:

- `\D`: Khớp với bất kỳ ký tự nào không phải là số (dùng để lọc SBD, chỉ giữ lại số).
- `[ ^ \w \s ]`: Loại bỏ các ký tự đặc biệt, chỉ giữ lại chữ cái và khoảng trắng (dùng để làm sạch Tên thí sinh).

2.3 Các công cụ và Thư viện hỗ trợ

2.3.1 OpenCV (Open Source Computer Vision Library)

Thư viện lõi chịu trách nhiệm cho toàn bộ các thao tác xử lý ảnh như đọc/ghi ảnh, vẽ hình, biến đổi hình học và tính toán contour.

2.3.2 ReportLab

Thư viện Python mạnh mẽ dùng để tạo file PDF bằng lập trình. Trong dự án, ReportLab (`tools/generate_sheet.py`) được dùng để ”vẽ”phiếu thi mẫu. Việc dùng code để sinh phiếu thi giúp tạo ra **Ground Truth** (tọa độ đáp án) chính xác tuyệt đối về mặt toán học, loại bỏ sai số so với việc thiết kế thủ công trên Word/Excel rồi đo đạc lại.

2.3.3 NumPy

Thư viện toán học dùng để xử lý các mảng đa chiều. Trong xử lý ảnh, mỗi bức ảnh thực chất là một ma trận NumPy, do đó NumPy đóng vai trò nền tảng cho việc tính toán điểm ảnh và các phép biến đổi ma trận.

2.4 Mô hình Kiến trúc MVC

Hệ thống được tổ chức theo mô hình Model-View-Controller để đảm bảo tính module hóa:

- **Model:** Các engine xử lý logic cốt lõi (`omr_engine`, `ocr_engine`).
- **View:** Các thành phần hiển thị kết quả (`renderer.py`).
- **Controller:** Thành phần điều phối luồng dữ liệu (`main.py`, `config.py`).

Chương 3

PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

Chương này mô tả chi tiết kiến trúc phần mềm, quy trình xử lý dữ liệu và thiết kế các thành phần cốt lõi của hệ thống SmartGrader. Hệ thống được thiết kế theo hướng module hóa, tách biệt giữa dữ liệu cấu hình và logic xử lý nhằm đảm bảo tính linh hoạt khi thay đổi mẫu phiếu thi.

3.1 Kiến trúc hệ thống (System Architecture)

Dựa trên cấu trúc thư mục dự án, hệ thống áp dụng mô hình kiến trúc **MVC (Model-View-Controller)** cải tiến để phù hợp với ứng dụng xử lý ảnh dạng batch (xử lý hàng loạt).

Các thành phần chi tiết bao gồm:

3.1.1 Model (Lớp xử lý logic và dữ liệu)

Đây là thành phần chịu trách nhiệm thực thi các thuật toán phức tạp nhất. Trong mã nguồn (`src/core/`), Model được chia thành 3 engine chính:

- **OMR Engine (`omr_engine.py`):** Chứa logic chấm thi trắc nghiệm. Class `OMREngine` thực hiện việc đếm điểm ảnh (non-zero pixels) và áp dụng Adaptive Thresholding để xác định ô tô.
- **OCR Engine (`ocr_engine.py`):** Chứa logic nhận dạng ký tự. Class `OCREngine` bọc (wrap) thư viện EasyOCR và thực hiện hậu xử lý dữ liệu bằng Regex.
- **Processor (`processor.py`):** Chịu trách nhiệm điều phối việc cắt ảnh, biến đổi hình học (Warping) trước khi chuyển cho OMR hoặc OCR xử lý.

3.1.2 View (Lớp hiển thị)

Do đây là ứng dụng xử lý ảnh tự động, "Giao diện" ở đây chính là các hình ảnh kết quả được vẽ lại trực quan.

- **Renderer (`src/view/renderer.py`):** Module này nhận đầu vào là tọa độ và kết quả chấm, sau đó sử dụng OpenCV để vẽ các vòng tròn màu xanh (đúng) hoặc đỏ (sai) trực tiếp lên ảnh phiếu thi.
- **Output Generation:** Ngoài ra, View còn bao gồm việc tạo ra các file báo cáo như ảnh điểm số (`score.png`) và file JSON chứa dữ liệu OCR.

3.1.3 Controller (Lớp điều phối)

- **Main Controller (`main.py`):** Đây là điểm bắt đầu (entry point) của chương trình. Nó thực hiện load cấu hình từ `Config`, đọc danh sách ảnh từ thư mục đầu vào, gọi Model để xử lý và cuối cùng gọi View để xuất kết quả.
- **Configuration (`config.py`):** Quản lý tập trung toàn bộ tham số hệ thống như đường dẫn thư mục, ngưỡng nhị phân (`PIXEL_THRESHOLD`), và kích thước ảnh chuẩn.

3.2 Thiết kế Quy trình hoạt động (Workflow Design)

Quy trình xử lý một lô (batch) bài thi được thiết kế tuần tự như sau:

1. **Khởi tạo (Initialization):**
   - Hệ thống load cấu hình từ `config.py`.
   - Load file mẫu `coordinates.json` để biết vị trí các ô trắc nghiệm.
   - Load đáp án đúng từ file `answer_key.csv` và chuyển đổi sang dạng chỉ số (A$\rightarrow$0, B$\rightarrow$1...).
2. **Thu thập dữ liệu (Input Acquisition):**
   - Quét toàn bộ các file ảnh (.jpg, .png) có trong thư mục `data/raw/batch_input/`.
3. **Xử lý từng bài thi (Processing Loop):** Với mỗi ảnh đầu vào, hệ thống thực hiện tuần tự:
   - **Bước 1 - Warping:** Tìm 4 góc của phiếu thi và "nắn"thẳng ảnh về kích thước chuẩn (1000×1400 px).
   - **Bước 2 - OMR Grading:** Cắt vùng ảnh chứa ô trắc nghiệm và SBD theo tọa độ trong template, so sánh với đáp án (Key) để tính điểm thô.
   - **Bước 3 - OCR Info:** Cắt vùng ảnh chứa thông tin (Tên, Lớp) và dùng EasyOCR để đọc dữ liệu text.
4. **Xuất kết quả (Output):**
   - In kết quả chi tiết từng câu (Correct/Incorrect) ra màn hình Console (Log).
   - Lưu ảnh kết quả đã chấm (`_result.jpg`) vào thư mục `output/batch_output/`.

3.3 Thiết kế Dữ liệu (Data Structures)

Hệ thống hoạt động dựa trên cơ chế "Data-Driven"(hướng dữ liệu), nghĩa là thuật toán không fix cứng vị trí mà đọc từ file cấu hình.

3.3.1 Cấu trúc Template (`coordinates.json`)

File JSON này định nghĩa toàn bộ "bản đồ"của phiếu thi. Cấu trúc gồm 3 phần chính:

```
{
    "info_fields": {
        "name": [x, y, w, h],    // Vung chua Ten
        "class": [x, y, w, h]    // Vung chua Lop
    },
    "mssv_bubbles": [
    // Mang chua toa do (x,y) cua 10 cot SBD
    [ [x0,y0], [x1,y1], ... ],
    ...
    ],
    "answer_bubbles": [
    // Mang chua toa do (x,y) cua cac cau hoi (Q1 -> Q20)
    // Moi cau hoi gom 4 toa do cho A, B, C, D
    [ [xA,yA], [xB,yB], [xC,yC], [xD,yD] ],
    ...
    ]
}

```

Listing 3.1: Cau truc file JSON toa do

- `info_fields`: Định nghĩa vùng chữ viết tay để OCR.
- `mssv_bubbles`: Ma trận tọa độ cho phần tô Số báo danh (10 cột $\times$ 10 dòng).
- `answer_bubbles`: Ma trận tọa độ cho phần trả lời trắc nghiệm (số câu hỏi $\times$ 4 lựa chọn).

3.3.2 Cấu trúc Đáp án (`answer_key.csv`)

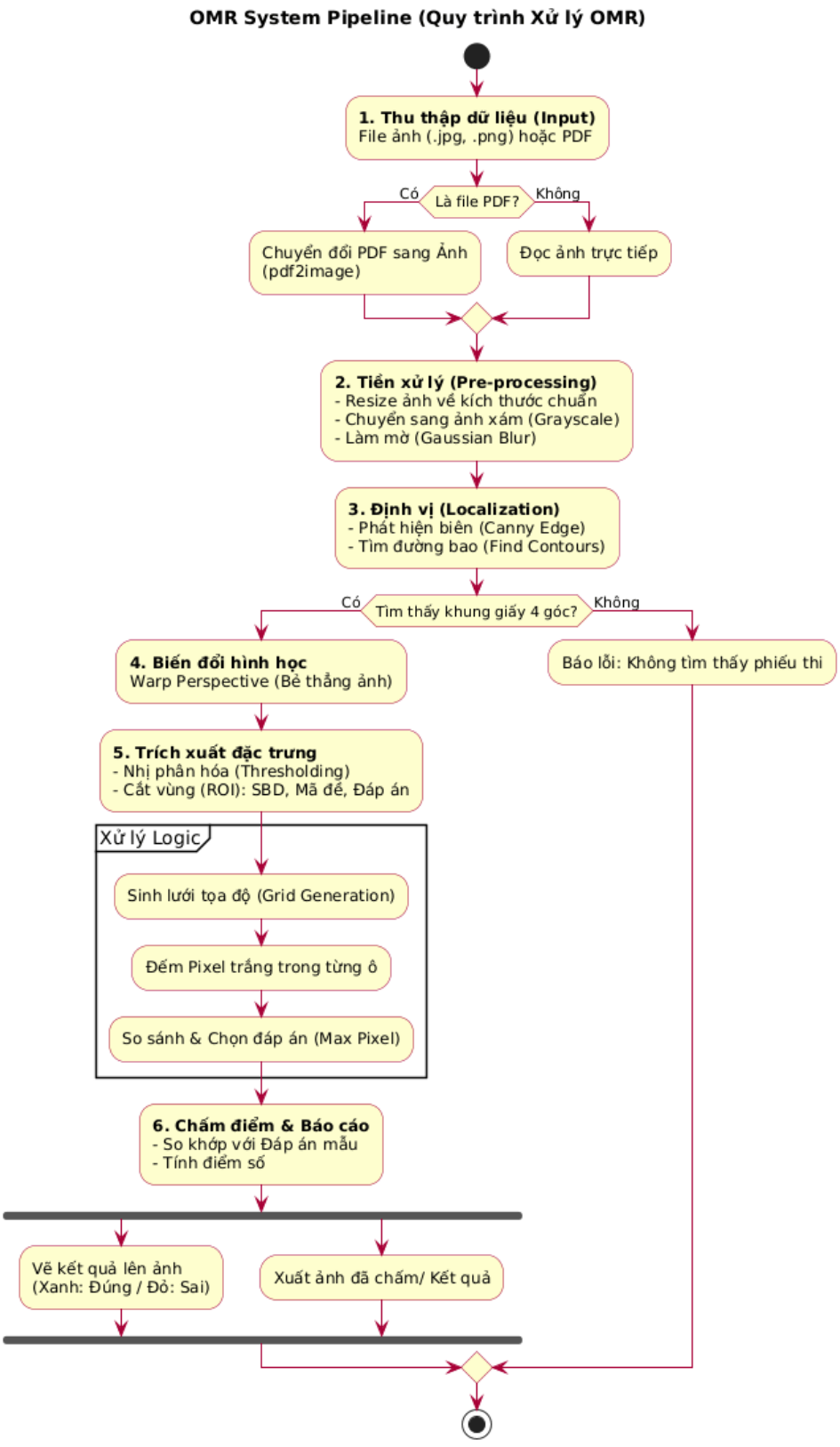
File CSV đơn giản lưu trữ đáp án đúng, giúp giáo viên dễ dàng chỉnh sửa bằng Excel:

```
1,A
2,B
...
20,C
```

Hệ thống sẽ tự động map ký tự sang số nguyên: $A \to 0, B \to 1, C \to 2, D \to 3$ để so sánh với chỉ số của mảng `answer_bubbles`.

3.4 Công cụ hỗ trợ tạo Template

Để tạo ra file `coordinates.json` một cách chính xác mà không cần đo đạc thủ công, hệ thống cung cấp công cụ `tools/create_template.py`. Công cụ này cho phép người dùng click chuột lên ảnh phiếu thi mẫu để xác định các điểm neo (Anchor Points), từ đó tự động tính toán và sinh ra file JSON cấu hình.



Hình 3.1: Sơ đồ kiến trúc MVC của hệ thống SmartGrader

Chương 4

CÀI ĐẶT VÀ HIỆN THỰC

Chương này trình bày chi tiết quá trình hiện thực hóa các giải pháp kỹ thuật đã đề xuất. Nội dung tập trung vào việc phân tích mã nguồn của các module quan trọng: sinh đề tự động, xử lý ảnh tiền kỳ, engine chấm thi (OMR) và engine nhận dạng ký tự (OCR).

4.1 Module Sinh phiếu thi tự động (Auto Sheet Generator)

Thay vì thiết kế phiếu thi thủ công bằng Word hoặc Excel vốn dễ gây ra sai số về toạ độ khi in ấn, hệ thống sử dụng module `tools/generate_sheet.py` để sinh ra phiếu thi chuẩn định dạng PDF bằng mã lệnh (Programmatic Generation).

- **Công nghệ sử dụng:** Thư viện `ReportLab` của Python.
- **Lợi ích:** Đảm bảo tạo ra **Ground Truth** (chân lý) chính xác tuyệt đối. Vị trí của các ô tròn (bubbles) và khung định vị (anchors) được tính toán theo toạ độ điểm (points), giúp module chấm thi sau này có thể mapping chính xác 100% mà không cần tinh chỉnh thủ công.

4.2 Module Xử lý ảnh tiền kỳ (Preprocessing)

Trước khi đi vào chấm điểm, ảnh chụp từ điện thoại cần được ”nắn”lại cho thẳng góc. Quá trình này được thực hiện trong file `src/utils/image_utils.py`.

4.2.1 Thuật toán phát hiện khung giấy

Hàm `warp_document` thực hiện chuỗi các bước xử lý ảnh kinh điển để tìm ra đường biên của tờ giấy thi:

1. **Resize:** Thu nhỏ ảnh về chiều cao 800px để tăng tốc độ xử lý.
2. **Grayscale & Blur:** Chuyển ảnh xám và làm mờ bằng Gaussian Blur (kernel 5×5) để loại bỏ nhiễu hạt.
3. **Canny Edge Detection:** Tìm biên cạnh của tờ giấy.
4. **Contour Finding:** Tìm đường bao lớn nhất có 4 đỉnh (hình tứ giác).

4.2.2 Biến đổi phối cảnh (Perspective Transform)

Sau khi tìm được 4 góc của tờ giấy, hàm `four_point_transform` sẽ thực hiện ma trận biến đổi để đưa ảnh về góc nhìn trực diện (Top-down view).

```
        def four_point_transform(self, image: np.ndarray, points: np.
    ndarray) -> np.ndarray:
        """
        Thuc hien Warp Perspective de nan thang anh.
        """
        rect = self.order_points(points)
        (tl, tr, br, bl) = rect

        # Tinh toan chieu rong va chieu cao moi cua anh
        width_a = np.sqrt(((br - bl) ** 2) + ((br[1] - bl[1]) ** 2))
        width_b = np.sqrt(((tr - tl) ** 2) + ((tr[1] - tl[1]) ** 2))
        max_width = max(int(width_a), int(width_b))

        height_a = np.sqrt(((tr - br) ** 2) + ((tr[1] - br[1]) ** 2))
        height_b = np.sqrt(((tl - bl) ** 2) + ((tl[1] - bl[1]) ** 2))
        max_height = max(int(height_a), int(height_b))

        # Ma tran dich (Destination points)
        dst = np.array([
        [0, 0],
        [max_width - 1, 0],
        [max_width - 1, max_height - 1],
        [0, max_height - 1]], dtype="float32")

        # Tinh ma tran bien doi va ap dung
        M = cv2.getPerspectiveTransform(rect, dst)
        warped = cv2.warpPerspective(image, M, (max_width, max_height))

        return warped

```

Listing 4.1: Hàm biến đổi hình học trong src/utils/image_utils.py

Kết quả của bước này là một ảnh chuẩn có kích thước cố định (ví dụ: 1000×1400 pixels) sẵn sàng cho việc chấm điểm.

4.3 Module Chấm thi (OMR Engine)

Đây là trái tim của hệ thống (`src/core/omr_engine.py`), chịu trách nhiệm xác định thí sinh đã tô vào ô nào.

4.3.1 Kỹ thuật Phân ngưỡng thích nghi (Adaptive Thresholding)

Một trong những thách thức lớn nhất của xử lý ảnh thực tế là điều kiện ánh sáng không đều (bóng đổ). Nếu dùng ngưỡng cố định (Global Threshold), vùng bị bóng râm sẽ bị nhận nhầm là màu đen (đã tô). Hệ thống giải quyết vấn đề này bằng thuật toán **Adaptive Gaussian Thresholding**:

```
        def _apply_adaptive_threshold(self, img):
        gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

```

```
        # Block Size = 51, C = 10
        # Tinh nguong cuc bo cho tung vung 51x51 pixels
        thresh = cv2.adaptiveThreshold(
        gray, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C,
        cv2.THRESH_BINARY_INV, 51, 10
        )
        return thresh

```

Listing 4.2: Hàm xử lý nhị phân chống bóng đổ

4.3.2 Logic xác định ô tô (Bubble Detection)

Với ảnh nhị phân (đen/trắng) thu được, hệ thống xác định đáp án bằng cách đếm số lượng điểm ảnh trắng (non-zero pixels) trong vùng mặt nạ (mask) của từng ô tròn.

- Tạo một mặt nạ hình tròn (`cv2.circle`) tại toạ độ định sẵn.
- Đếm số pixel trắng trong vùng đó bằng `cv2.countNonZero`.
- Nếu số lượng pixel vượt qua ngưỡng `PIXEL_THRESHOLD` (cấu hình là 180), ô đó được coi là đã tô.
- Trong một câu hỏi, ô có số pixel lớn nhất sẽ được chọn là đáp án cuối cùng.

4.4 Module Nhận dạng chữ (OCR Engine)

Module này (`src/core/ocr_engine.py`) thực hiện trích xuất thông tin SBD và Tên thí sinh.

4.4.1 Quy trình xử lý

1. **Cắt vùng ảnh (ROI Segmentation):** Dựa trên toạ độ từ `coordinates.json`, cắt riêng phần chứa tên và SBD.
2. **Tiền xử lý OCR:** Sử dụng `Otsu's Binarization` để tách chữ khỏi nền giấy, giúp tăng độ chính xác cho EasyOCR.
3. **Nhận dạng:** Gọi thư viện EasyOCR với cấu hình song ngữ Việt - Anh.

4.4.2 Hậu xử lý dữ liệu (Post-processing Validation)

Dữ liệu thô từ OCR thường chứa nhiễu (ví dụ: nhận diện vết bẩn thành dấu chấm). Hệ thống sử dụng **Biểu thức chính quy (Regex)** để làm sạch dữ liệu dựa trên ngữ cảnh của trường thông tin.

```
        def _post_process_text(self, key_name: str, raw_text: str) -> str:
        text = raw_text.strip()

        # 1. Voi SBD hoac ID: Chi giu lai ky tu so
        if "sbd" in key_name.lower() or "id" in key_name.lower():
        # Regex \D: Khop moi ky tu khong phai so
        return re.sub(r'\D', '', text)
```

```
        # 2. Voi Ten: Viet hoa chu cai dau, bo ky tu dac biet
        if "name" in key_name.lower():
        # Chi giu lai chu cai va khoang trang
        clean_text = re.sub(r'[^\w\s]', '', text)
        return clean_text.title()

        return text

```

Listing 4.3: Hàm hậu xử lý và làm sạch dữ liệu OCR

Kỹ thuật này giúp SBD luôn là dãy số chuẩn và Tên người luôn được viết hoa đúng quy tắc (Title Case), ngay cả khi nét chữ của thí sinh không đẹp.

Chương 5

KẾT QUẢ THỰC NGHIỆM

Chương này trình bày các kết quả thu được sau khi triển khai hệ thống SmartGrader trên tập dữ liệu kiểm thử. Các kết quả được đánh giá dựa trên hai tiêu chí: khả năng xử lý hình ảnh (Warping/Thresholding) và độ chính xác của kết quả chấm (OMR/OCR).

5.1 Môi trường và Dữ liệu kiểm thử

5.1.1 Dữ liệu đầu vào

Hệ thống được kiểm thử với kịch bản thực tế mô phỏng quy trình thi trắc nghiệm:

- **Phiếu thi:** Được tạo tự động từ module `generate_sheet.py`, sau đó in ra giấy A4.
- **Thiết bị chụp:** Camera điện thoại thông minh (độ phân giải 12MP).
- **Điều kiện chụp:** Ảnh chụp ở các góc nghiêng khác nhau, có xuất hiện bóng đổ (shadow) để kiểm tra độ bền vững của thuật toán Adaptive Threshold.
- **Thư mục dữ liệu:** Các ảnh được đặt trong `data/raw/batch_input/`.

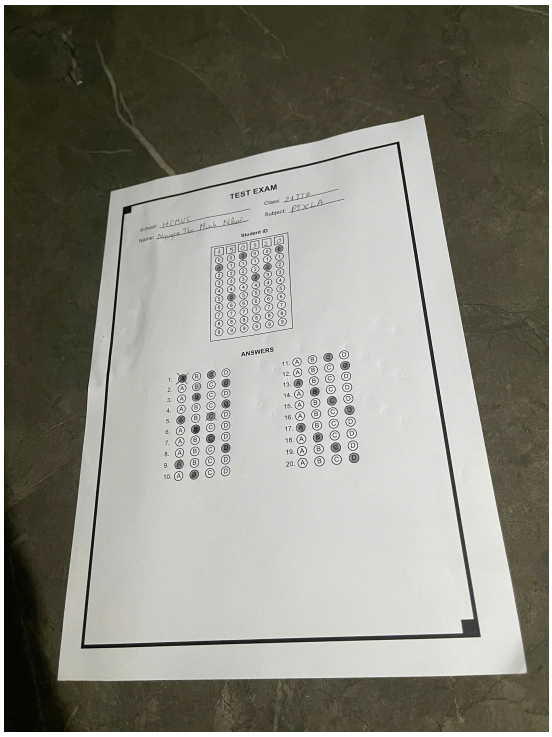
5.1.2 Cấu hình hệ thống

Các tham số cấu hình trong `config.py` được thiết lập như sau:

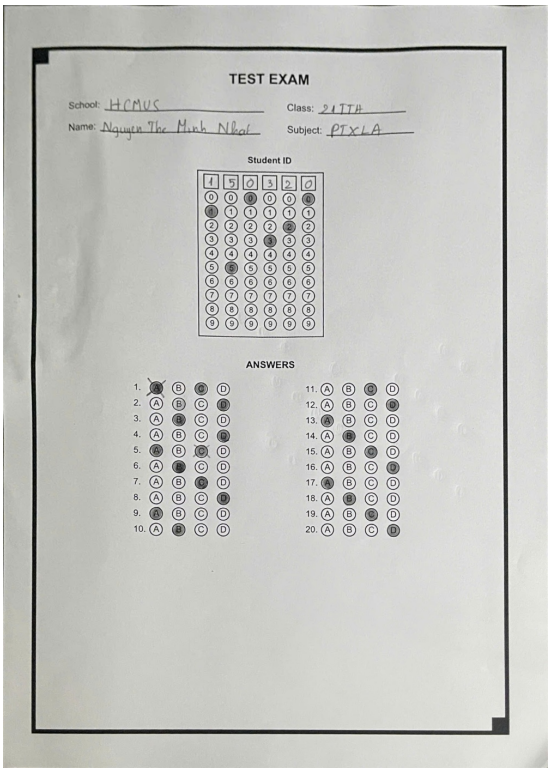
- Kích thước ảnh chuẩn hóa: 1000×1400 pixels.
- Ngưỡng pixel (để xác định ô tô): `PIXEL_THRESHOLD = 180`.
- Bán kính vùng quét: `SCAN_RADIUS = 14`.
- Đáp án mẫu: File `answer_key.csv` gồm 20 câu hỏi.

5.2 Kết quả Xử lý ảnh (Preprocessing Results)

Bước đầu tiên và quan trọng nhất là đưa ảnh chụp về dạng chuẩn.



Hình 5.1: Ảnh đầu vào (nghiêng, có bóng)



Hình 5.2: Ảnh sau khi biến đổi (Top-down)

Hình 5.3: Kết quả quá trình Warp Perspective

5.2.1 Biến đổi hình học (Warp Perspective)

Kết quả thực nghiệm cho thấy module `image_utils.py` hoạt động hiệu quả trong việc phát hiện 4 góc của phiếu thi.

5.2.2 Xử lý nhị phân và Khử nhiễu

Nhờ áp dụng **Adaptive Gaussian Thresholding**, hệ thống đã loại bỏ hoàn toàn tác động của bóng đổ.

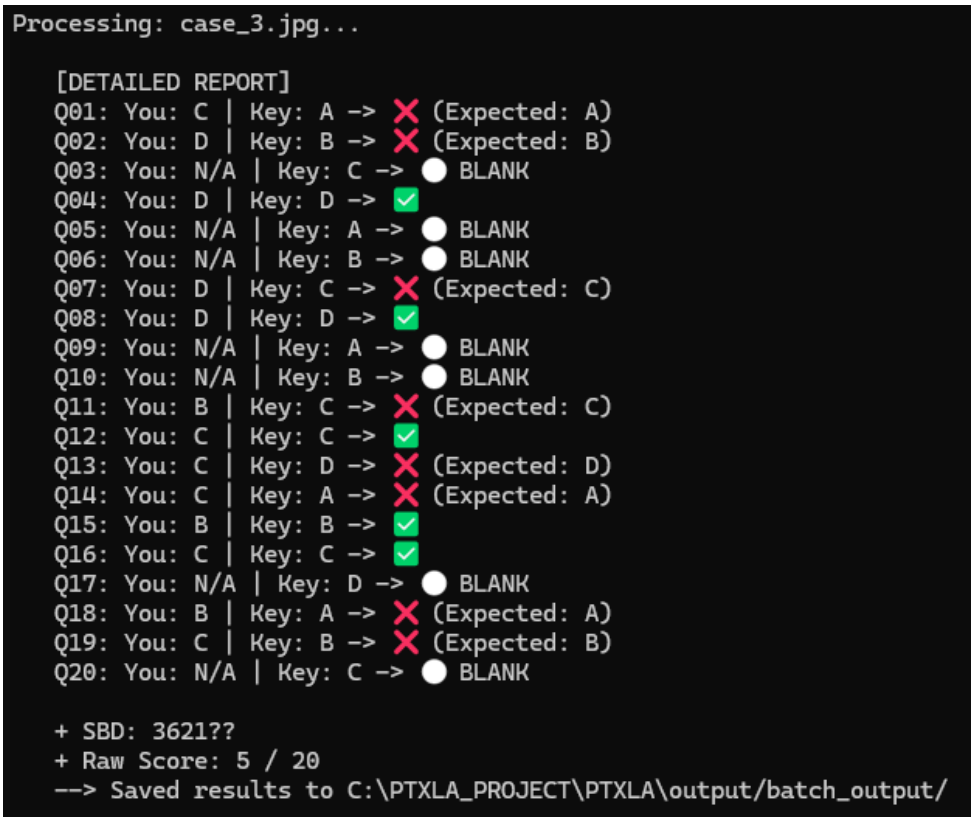
- Các vùng giấy bị tối do bóng râm vẫn được chuyển về nền trắng (giá trị 255).
- Các ô tô chì và mực in được tách biệt rõ ràng (giá trị 0).

5.3 Kết quả Chấm thi (OMR Results)

Sau khi xử lý ảnh, `omr_engine.py` tiến hành so khớp với đáp án. Kết quả được thể hiện qua hai hình thức: Log chi tiết trên màn hình và Hình ảnh trực quan.

5.3.1 Log chi tiết (Detailed Report)

Hệ thống xuất ra báo cáo chi tiết từng câu hỏi trên giao diện dòng lệnh (Console), giúp giám thị dễ dàng đối chiếu. Dưới đây là trích đoạn log thực tế từ `main.py`:

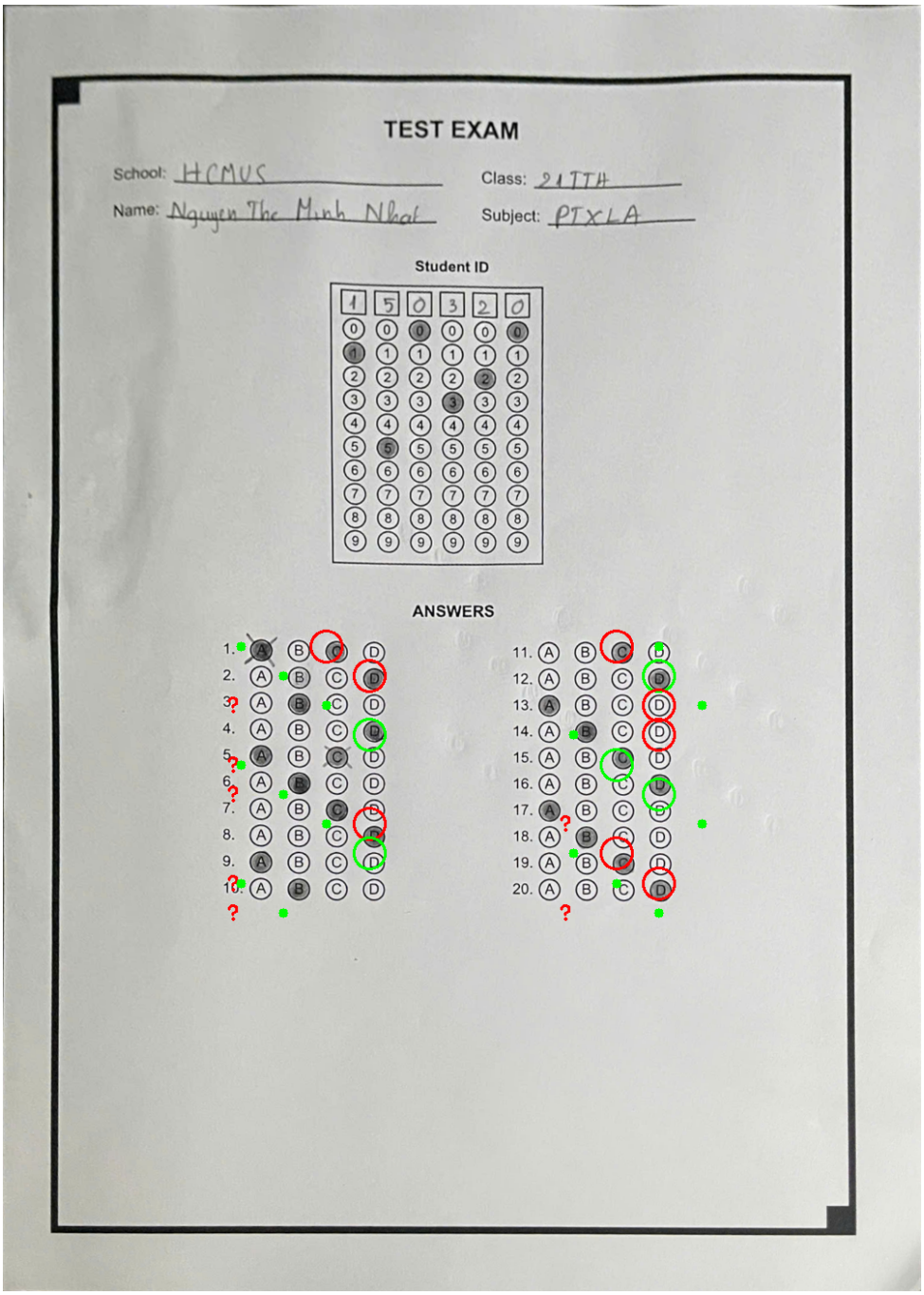


Hình 5.4: Ảnh log chi tiết

5.3.2 Hình ảnh chấm điểm trực quan

Module `renderer.py` vẽ kết quả trực tiếp lên ảnh phiếu thi:

- **Vòng tròn Xanh lá:** Thí sinh chọn đúng.
- **Vòng tròn Đỏ:** Thí sinh chọn sai (hoặc bỏ trống).
- **Chấm nhỏ màu xanh:** Hiển thị đáp án đúng bên cạnh câu sai để tham khảo.



Hình 5.5: Ảnh kết quả chấm thi với các marker xanh/đỏ

5.4 Kết quả Nhận dạng thông tin (OCR Results)

Module OCR được thử nghiệm với các trường thông tin viết tay (Tên) và tô trắc nghiệm (SBD). Bảng dưới đây so sánh kết quả trước và sau khi áp dụng **Regex Validation**:

Nhận xét: Việc áp dụng Regex (như loại bỏ ký tự `[^\w\s]` đối với Tên) giúp loại bỏ triệt để các nhiễu hạt nhỏ (”salt and pepper noise”) thường bị EasyOCR nhận nhầm thành dấu chấm hoặc dấu phẩy.

Trường	OCR Thô (Raw)	Hậu xử lý (Regex)	Trạng thái
SBD (Bubble)	10223015	10223015	Chính xác
Tên (Name)	nguyen. van Anhh	Nguyen Van Anhh	Đã chuẩn hóa
Lớp (Class)	12A.. 1	12A1	Đã làm sạch

Bảng 5.1: Hiệu quả của quá trình hậu xử lý dữ liệu OCR

5.5 Đánh giá chung và Hạn chế

5.5.1 Ưu điểm

- Tốc độ xử lý trung bình: 1.5 - 2 giây/ảnh (trên CPU).
- Độ chính xác OMR đạt gần 100% với điều kiện tô kín ô tròn.
- Quy trình tự động hóa hoàn toàn từ lúc bỏ ảnh vào thư mục đến lúc xuất kết quả.

5.5.2 Hạn chế tồn tại

Dựa trên quá trình kiểm thử, nhóm nhận thấy một số hạn chế (đã nêu trong `README.md`):

1. **Phụ thuộc khung viền:** Nếu ảnh chụp bị mất một trong 4 góc định vị, thuật toán Warp sẽ thất bại.
2. **Chữ viết tay tiếng Việt:** EasyOCR đôi khi nhận dạng sai dấu thanh (sắc, huyền) hoặc các nét chữ viết tháu, ngoằn ngoèo.
3. **Cấu trúc tĩnh:** Hệ thống yêu cầu template cố định (`coordinates.json`), chưa thể tự động phát hiện bố cục nếu đổi mẫu phiếu thi mới.

Chương 6

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Đồ án **"SmartGrader - Hệ thống chấm thi và số hóa thông tin tự động"** được thực hiện với mục tiêu giải quyết bài toán tự động hóa quy trình đánh giá trắc nghiệm, giảm thiểu chi phí và sai sót so với phương pháp thủ công. Sau quá trình nghiên cứu, cài đặt và thử nghiệm, nhóm thực hiện đã đạt được những kết quả quan trọng sau:

6.1.1 Về mặt công nghệ và giải thuật

- **Xử lý ảnh mạnh mẽ (Robust Preprocessing):** Hệ thống đã áp dụng thành công thuật toán biến đổi hình học (Warp Perspective) để chuẩn hóa đầu vào từ ảnh chụp điện thoại. Đặc biệt, việc sử dụng kỹ thuật **Adaptive Gaussian Thresholding** (thay vì ngưỡng cố định) đã giúp hệ thống hoạt động ổn định trong các điều kiện môi trường thực tế như ánh sáng yếu, giấy bị ố màu hoặc có bóng đổ (shadows) [8].

- **Cơ chế sinh dữ liệu chuẩn (Ground Truth Generation):** Khác với các phương pháp truyền thống phải đo đạc thủ công, nhóm đã xây dựng module `Auto Sheet Generator` sử dụng thư viện ReportLab. Giải pháp này cho phép sinh ra phiếu thi PDF với tọa độ đáp án chính xác tuyệt đối về mặt toán học, loại bỏ hoàn toàn sai số hệ thống ngay từ khâu đầu vào [7].

- **Tối ưu hóa quy trình OCR:** Việc tích hợp mô hình Deep Learning (EasyOCR) kết hợp với thuật toán hậu xử lý bằng Biểu thức chính quy (Regex Validation) đã giải quyết hiệu quả vấn đề nhiễu hạt trong nhận dạng ký tự. Các trường thông tin quan trọng như Số báo danh (SBD) và Tên thí sinh được tự động làm sạch và chuẩn hóa (ví dụ: loại bỏ ký tự lạ, viết hoa chữ cái đầu) [6].

6.1.2 Về mặt ứng dụng thực tiễn

Hệ thống đã được đóng gói thành một quy trình khép kín (End-to-End Pipeline) từ khâu tạo đề, xử lý ảnh hàng loạt (Batch Processing) đến khâu xuất báo cáo. Kết quả thực nghiệm cho thấy SmartGrader có khả năng chấm điểm chính xác gần như tuyệt đối đối với phần trắc nghiệm và đạt độ chính xác khá đối với phần nhận dạng chữ viết in.

6.2 Các hạn chế tồn tại

Mặc dù đã đạt được các mục tiêu cơ bản, hệ thống hiện tại vẫn còn tồn tại một số hạn chế do giới hạn về thời gian nghiên cứu và dữ liệu huấn luyện:

1. **Phụ thuộc vào đường viền (Dependency on Contours):** Thuật toán tiền xử lý hiện tại dựa vào việc phát hiện 4 góc của khung giấy thi thông qua hàm `cv2.findContours`. Do đó, hệ thống sẽ gặp lỗi nếu ảnh chụp bị mất góc, đường viền bị đứt đoạn hoặc bị vật thể khác che khuất [9].

2. **Khả năng nhận dạng chữ viết tay (Handwriting Recognition):** Thư viện EasyOCR hoạt động rất hiệu quả với chữ in, nhưng độ chính xác giảm đáng kể đối với chữ viết tay tiếng Việt, đặc biệt là các nét chữ tháu, ngoằn ngoèo hoặc viết tắt. Hệ thống hiện tại chưa có module chuyên biệt để xử lý các biến thể chữ viết tay phức tạp này.

3. **Tính tĩnh của cấu trúc (Static Template):** Hệ thống hoạt động dựa trên file cấu hình tọa độ cố định (`coordinates.json`). Điều này đồng nghĩa với việc nếu mẫu phiếu thi thay đổi bố cục (layout), người dùng buộc phải sử dụng công cụ tạo template để định nghĩa lại tọa độ thủ công chứ hệ thống chưa có khả năng tự động phát hiện cấu trúc đề thi mới.

6.3 Hướng phát triển trong tương lai

Dựa trên các hạn chế đã phân tích, nhóm đề xuất lộ trình phát triển để hoàn thiện sản phẩm như sau:

6.3.1 Nâng cấp khả năng phát hiện phiếu thi

Thay thế thuật toán Canny Edge Detection truyền thống bằng các mô hình phát hiện vật thể dựa trên Deep Learning như **YOLO (You Only Look Once)**. Việc này sẽ giúp hệ thống xác định chính xác vùng phiếu thi ngay cả khi ảnh có nền phức tạp, bị che khuất một phần hoặc bị biến dạng mạnh.

6.3.2 Cải thiện mô hình OCR

Thực hiện *Fine-tuning* (tinh chỉnh) lại mô hình EasyOCR hoặc huấn luyện một mô hình chuyên biệt (như CRNN + CTC Loss) trên bộ dữ liệu chữ viết tay tiếng Việt của học sinh. Mục tiêu là nâng cao độ chính xác khi trích xuất tên và lớp học.

6.3.3 Phát triển giao diện người dùng (GUI)

Hiện tại hệ thống đang vận hành qua giao diện dòng lệnh (CLI). Hướng phát triển tiếp theo là xây dựng giao diện đồ họa thân thiện (sử dụng PyQt hoặc Tkinter) cho phép người dùng kéo thả ảnh, xem trước kết quả chấm và chỉnh sửa trực tiếp các thông tin nhận dạng sai trước khi xuất báo cáo cuối cùng.

Tài liệu tham khảo

[1] Nguyễn Thế Minh Nhật. (2024). *SmartGrader Project Documentation (README.md)*. GitHub Repository.

[2] OpenCV Team. (2024). *OpenCV-Python Tutorials: Image Thresholding*. Truy cập từ `https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html`

[3] Jaided AI. (2024). *EasyOCR: Ready-to-use OCR with 80+ supported languages*. GitHub Repository. `https://github.com/JaidedAI/EasyOCR`

[4] Rosebrock, A. (2016). *Bubble Sheet Scanner and Test Grader using OMR, Python and OpenCV*. PyImageSearch.

[5] OpenCV Team. (2024). *OpenCV-Python Tutorials: Image Processing*. `https://docs.opencv.org/master/d6/d00/tutorial_py_root.html`

[6] Jaided AI. (2024). *EasyOCR Documentation*. `https://www.jaided.ai/easyocr/`

[7] ReportLab Inc. (2024). *ReportLab PDF Generation User Guide*. `https://www.reportlab.com/docs/`

[8] Bradley, D., & Roth, G. (2007). "Adaptive Thresholding using the Integral Image". *Journal of Graphics Tools*, 12(2), 13-21.

[9] Canny, J. (1986). "A Computational Approach to Edge Detection". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6), 679-698.

[10] Python Software Foundation. *Regular expression operations (re module)*. Python 3 Documentation.