ID Cards Processing using OCR

Milestone 1 of 3: setting pipelines

Max Ptasiewicz, Sabin Dahal, A S M Shahadat Hossain
University of Missouri, Columbia, MO, USA
{mpcpr,sddk7,ahr8v}@missouri.edu

I. Introduction

The goal of this project is to classify ID cards by country with the help of Optical Character Recognition (OCR). This report is based on the first milestone of the project, where we configured a few pipelines. Each pipeline consists of three steps:

1) Image enhancement
2) Channel selection and
3) Optical character recognition

We developed several assertions to measure the accuracy of the pipelines built. We also report the latency observed.

II. Pipeline Architecture and Configurations

A. Dataset

For this project, we used the `lansinuote/ocr_id_card` dataset, which is publicly available on Hugging Face. The dataset consists of 20,000 images of Chinese resident identity cards paired with corresponding OCR ground-truth labels. Each sample in the dataset includes the card image and associated metadata such as text content. Each image typically includes the cardholder's name, gender, ethnicity, date of birth, residential address, and citizen identification number, together with a portrait placeholder region. One significant challenge associated with the dataset is that the orientation of the ID card images is not uniform.

B. Pipelines

We configured a total of four pipelines, each containing image enhancement, channel selection, and OCR steps. We varied the image enhancement and channel selection mechanisms across the configurations while using Tesseract as the OCR tool.

Pipeline 1: Deskewed Red-Channel OCR

In this pipeline, we implemented a sequence of preprocessing and recognition steps designed to improve the robustness of text extraction from ID card images. First, we converted each image to grayscale and applied Canny edge detection, followed by the Hough line transform. We used these methods to detect strong structural lines on the ID card, which allowed us to estimate the card's orientation. We then applied an affine warp to deskew the image. This step is important because OCR engines are sensitive to text rotation, and correcting the skew helps align text horizontally, improving recognition accuracy.

Next, we performed channel selection, retaining only the red channel while setting the green and blue channels to zero. We made this choice because many ID cards contain red seals and stamps that may interfere with recognition when combined with the full RGB image. By isolating the red channel, we reduced background noise and enhanced the contrast of text regions.

After preprocessing, we passed the processed images to Tesseract OCR configured with the Chinese Simplified language model. Tesseract was chosen for its ability to handle multilingual text, including Chinese characters, and for being a well-established open-source OCR engine. Finally, we cleaned and tokenized the raw OCR output using regular expressions to separate numbers, Chinese characters, and English letters. This postprocessing step ensured that the recognized text could be evaluated consistently and compared with ground-truth annotations.

Pipeline 2: Deskewed Grayscale OCR

This pipeline follows the same preprocessing approach as the Deskewed Red-Channel OCR: we first convert the image to grayscale, apply Canny edge detection and the Hough line transform to detect structural lines, and then perform an affine warp to deskew the card so that text is horizontally aligned.

Unlike the red-channel pipeline, we retain the image in grayscale rather than isolating a color channel. This choice reduces color-related noise and emphasizes character strokes without relying on specific color cues. The preprocessed images are then passed to Tesseract OCR with the Chinese Simplified language model, and the outputs are cleaned and tokenized with regular expressions to separate numbers, Chinese characters, and English letters for evaluation.

Pipeline 3: Deskewed Red-Channel with Post-Denoising OCR

This variant follows the same deskew preprocessing as before (grayscale, Canny, Hough lines, and affine warp) to align text horizontally. After rectification, we isolate the red channel (zeroing green and blue) for the same reason as in the red-channel pipeline, reducing background tints and emphasizing stamp/text contrast. The key difference is that we then apply non-local means denoising after warping. We tried this to suppress speckle and compression noise that can confuse OCR in low-contrast regions. The denoised image is finally passed to Tesseract, and the output is cleaned/tokenized as before.

Pipeline 4: Pre-Denoise Red-Channel OCR

In this pipeline, we again apply deskewing through grayscale conversion, Canny edge detection, Hough line transform, and affine warping to align the text horizontally. The main difference is that we introduce non-local means denoising before the warp step, aiming to suppress background noise so that distortions are not amplified during geometric correction. After deskewing, we isolate the red channel to reduce interference from colored backgrounds and highlight text contrast. Finally, we run Tesseract OCR with the Chinese Simplified model and perform regex-based tokenization to separate numbers, Chinese characters, and English letters.

TABLE I
ACCURACY AND LATENCY OF THE PIPELINES.

Pipeline	Accuracy	Latency (seconds)
Deskewed Red-Channel OCR	0.562	0.974
Deskewed Grayscale OCR	0.452	0.912
Deskewed Red-Channel with Post-Denoising OCR	0.414	1.124
Pre-Denoise Red-Channel OCR	0.456	0.974

III. RESULTS

In our experiments, we reported two metrics: accuracy and latency on 100 images. We developed the following assertions to measure the accuracy of the pipelines:

- **Name**: we check whether the name extracted by using OCR matches the name present in the ground truth.
- **Year**: we validate if the extracted year is between 1900 and 2025.
- **Month**: we validate the month by checking whether it ranges from 1 to 12.
- **Date**: we validate the date by checking whether it ranges from 1 to 31.
- **ID**: we check if the extracted ID matches the ground truth ID of 18 digits.

For each image, the accuracy score was computed by awarding one point for every assertion satisfied, with a maximum of five points. The score was then normalized by dividing by five, and pipeline accuracy was reported as the average of these normalized scores across the evaluation set. Latency was measured by recording the elapsed wall-clock time for the OCR step alone, using timestamps immediately before and after the Tesseract call. The reported latency values, therefore, reflect the recognition stage rather than the full preprocessing pipeline.

Table I presents the accuracy and latency observed for the pipelines configured. It shows that the Deskewed Red-Channel OCR pipeline achieved the highest accuracy (0.562) while maintaining a moderate latency of 0.974 seconds, making it the most effective configuration among those tested. The Deskewed Grayscale OCR pipeline performed slightly worse in terms of accuracy (0.452) but had the lowest latency (0.912 seconds), suggesting that grayscale preprocessing can improve speed but sacrifices recognition quality. Introducing denoising after warping in the Post-Denoising Red-Channel OCR pipeline led to a significant drop in accuracy (0.414) and the highest latency (1.124 seconds), indicating that this additional step blurred important text features rather than enhancing them. Applying denoising before warping in the Pre-Denoise Red-Channel OCR pipeline yielded a marginal improvement over the post-denoising variant (0.456 accuracy, 0.974 seconds latency), but it still fell short of the baseline red-channel pipeline. Overall, the results suggest that deskewing combined with red-channel selection is effective, while additional denoising steps reduce both efficiency and accuracy.

IV. TEAM ORGANIZATION AND ACTIVITIES

Our team consists of three members with diverse backgrounds and experience. Max is a Mechanical Engineer currently working on a PhD in the same area. Besides mastering computing technologies going beyond the traditional mechanical engineering domain, he is pretty comfortable with the mathematics behind the machine learning optimizations. Sabin is doing an MS in Computer Science after working for several years in the industry. He has a solid understanding of machine learning. Notably, he brings his prior experience of using Docker. Shahadat is doing a PhD in Computer Science, after spending several years in academia while completing a MS as well as a few years in the industry. He has previously worked on several academic projects on machine learning.

We are using a Microsoft Teams group to maintain communication among the team members. We share our understandings, plans, and individual progress in the group. We briefly meet after the classes on Wednesday. When needed, we also attend instant Zoom meetings. After working for a few days individually, we meet to share our understanding of the requirements, prepare the execution plan, and divide the work among the team members. During the final submission of Milestone 1, Max created the initial pipeline and then designed the assertions as well as the other pipelines with Shahadat and implemented them. Sabin created the Dockerfile. Finally, Shahadat prepared the final report.

So far, we have not faced any teamwork problems. We have a good understanding of each other. The best thing is the diversity of the team, which fosters a great learning environment among the team members. One thing we would like to do in the future is to start working earlier.

V. Reproducibility

The notebook, Dockerfile, and other required files to reproduce the pipelines are available on GitHub: https://github.com/sabindahal/ICP_ML/.