COL783 Assignment 4 Report

The first part of the pipeline was to convert the image to monochrome, as working with three channels would've been complicated and unnecessary.

Alignment

It was clear that I'd have to get the angles of slope of the major lines in the photo in order to straighten it. Hough transform was the most obvious way to go about doing it. Here are the steps in my pipeline —

- **Canny edge detection** — Before applying Hugh transform, I'd need a binary version of the image with the lines in it. So I applied Canny edge detection on the image.
- **Hough transform for lines** — Applying Hugh transform, I got a list of lines.
- **Getting the average slope of the 10 longest lines** — Sorting the lines by the length in descending order, I grabbed the first 10 lines. After adjusting their angles of slope properly, I took the average of them.
- **Rotating the image** — Then I rotated the image by the angle acquired in the previous step. I used Affine transformation instead of normal rotation in order to preserve the image dimensions.

Form field segmentation

In this section of the assignment, I had to use thresholding, morphological operations and connected component analysis to get a decent enough result. Here are the steps in the pipeline —

- **Straightening** — Using the function of the last step to straighten the image.
- **Blurring the image** — Blurring the using a gaussian kernel so that the operation in the next step—the thresholding—gives smoother results.
- **Adaptive thresholding** — The image has different illumination and lighting conditions in different parts of the image, so global thresholding was not giving a good enough result. That's why I went with adaptive thresholding.

Collecting rectangular segments

In the following steps of the pipeline, I tried different transformations to the image, and some of them work well for some form fields, others work for others. So I decided to do all of them and collect the segments I got from each step in a single place.

I created a mask image, i.e. an image with all the pixels set to black. And I made a copy of the original image too, let's call that canvas. Now, in each of the steps, as I keep finding new rectangles, I draw the borders of them on the canvas image, and set the corresponding pixels to white in the mask image. While drawing on the mask image, I fill the rectangles, i.e. not only the borders, but the whole rectangle.

The following are the steps —

- **Threshold** — From the thresholded image, find all the contours. Then filter the list of contours and keep the ones with the correct shape and size.
- **Morphological opening** — Then I applied morphological opening to remove the small white noise blobs in the image. After that I grabbed all the contours with the correct shape and size.
- **Morphological gradient** — I found out that morphological gradient helps detect some more additional fields, so I applied that and grabbed the correct contours too.
- **Connected component analysis** — Finally, I applied connected component analysis to the image from the last step. From all the connected components, grabbed the ones with the right shape and size.

Character detection

For character detection, I used the result from the last step, i.e. the mask image. Combined that with thresholding and connected component analysis, I was able to detect most characters.

The steps in the pipeline being —

- **Blurring and thresholding** — Applying a small gaussian blur to remove the noise, followed by adaptive thresholding to detect the darker lines. The thresholding used was an inverse one, so the darker regions got thresholded to white and the brighter ones to black.
- **Acquiring the ROI** — Using the segmentation function implemented in the last step, I got the mask image containing the form fields. Bitwise ANDing it with the thresholded image resulted in a thresholded image, but only the form fields parts. All the other parts were set to black.

- **Morphological dilation** — Applying a morphological dilation helped to close any gaps between a single written character.
- **Connected component analysis** — Finally, applying connected component analysis followed by filtering the components having a bounding rectangle of the correct shape and size, I was able to detect most of the characters.