Document Scanner and Perspective Correction

Members:

Name	Enrolment No.
Jainik Patel	AU2340042
Dhruvik Shah	AU2340113
Ayush Makwana	AU2340138
Divyam Thakkar	AU2340227

***Abstract*— The aim of this project is to convert a normal camera photograph of a document into an image that looks like it was produced by a hardware scanner, and to correct the perspective from which the document is captured. To do this, we use basic image processing methods such as edge detection, contour-based document localisation, perspective transformation and image enhancement. The final pipeline takes a smartphone image as input and produces a neat, high–contrast, scanner-like output that is easy to read and store.**

Keywords—Image processing, document scanning, perspective correction, edge detection, adaptive thresholding.

I. INTRODUCTION

Digitising paper documents is very common today. Flatbed scanners can do this job well, but they are not portable and are not always available. On the other hand, almost everyone has a smartphone with a camera. Taking a photo of a document is easy, but the result often has problems such as background clutter, uneven lighting, shadows and perspective distortion because the picture is not taken exactly from the top.

The goal of this project is to build a simple document scanner using digital image processing so that a basic mobile photo can be turned into a clean document image. The system should automatically detect the document area, correct the perspective and enhance the final look so that it looks closer to a scanned page.

We implement the project in Python using the OpenCV library. The pipeline is divided into smaller steps: image acquisition, preprocessing, document (paper) detection, perspective correction and final enhancement. Each step uses the output from the previous step, and together they form a complete scanning flow.

II. METHODOLOGY

The overall project is implemented in Python with OpenCV functions. In this section we briefly explain each step of the final pipeline.

A. Image Acquisition

To test the system in realistic situations, we captured images under different conditions: normal classroom photos, exam papers on a table, different backgrounds and different viewing angles. In all cases, the document is present in the scene, but not always at the centre and not always flat.

B. Preprocessing

First, the input image is converted from colour (BGR/RGB) to grayscale. This reduces the amount of data to process and keeps only brightness information, which is enough for edge detection.

Next, we apply Gaussian blur with a 7×7 kernel. This step smooths the image and reduces small noise, which helps in getting cleaner edges later.

After smoothing, we use Canny edge detection to highlight the edges in the image. The thresholds are chosen experimentally. Then, we dilate the edge image slightly so that broken edges become more continuous and suitable for contour detection.

C. Paper Detection and Mask Creation

From the edge image, we find contours in the scene. The idea is that the document will usually form one of the largest contours in the image and will roughly look like a quadrilateral.

We sort contours by area and search for a contour that can represent the paper. Using the best matching contour, we create a binary mask that separates the document region from the background. In some failure cases (for example, when no clear contour is found), we simply treat the full image as the paper region so that the code still runs and produces an output.

Once we have the paper mask, we also create a "background darkened" version where everything outside the document area is coloured with a darker tone. This gives a visual effect similar to mobile scanner apps where the paper is bright and the surroundings are dark.

D. Corner Extraction

The contour of the paper is then approximated with a polygon. From this polygon, we select four main corner points that correspond to the four corners of the document. We order these points as top-left, top-right, bottom-right and bottom-left using simple sum and

difference based rules. This ordering is important for the perspective transformation step.

E. Perspective Transformation

Using the ordered four corner points, we calculate a perspective transformation matrix. We decide a target width and height for the output document based on the distances between the corner points. With the help of `cv2.getPerspectiveTransform` and `cv2.warpPerspective`, we map the document region from the original image into a new image where the document appears straight and front-facing. This step fixes the problem of tilted or slanted captures.

F. Final Enhancement and Scanner Effect

The warped document image already looks better, but it still has natural lighting variations and colours. To make it look more like a scanner output, we convert it again to grayscale and apply adaptive thresholding (Gaussian method). This gives a high–contrast black-and-white image where text is dark and the background is almost white.

This type of output is easier to read and is similar to what tools like CamScanner produce. We save both versions: the perspective corrected colour image and the enhanced scan-like image.

III. RESULTS

We tested the final pipeline on different examples. Here we show two main cases: a classroom slide and a printed exam paper. For each case, we show the original capture, the perspective corrected image and the final enhanced scan.

A. Classroom Slide Example

Figure 1 shows the original image of a classroom slide captured from the back of the room. The slide is small in the frame, and the viewing angle is not frontal.

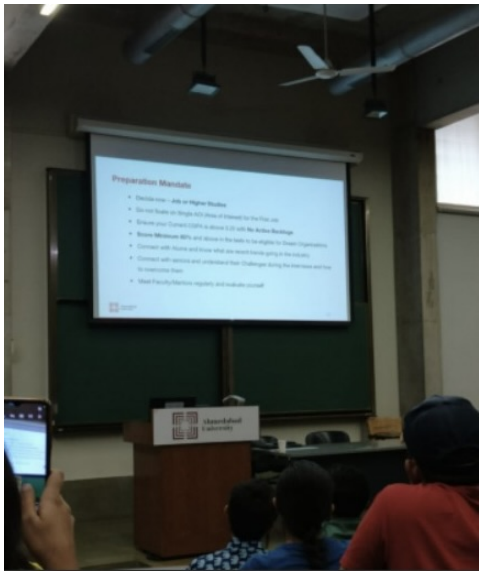

Fig. 1. Original classroom slide image captured from the audience.

In Figure 2, the same slide after perspective correction is shown. The slide area is now properly aligned and appears as a rectangle.

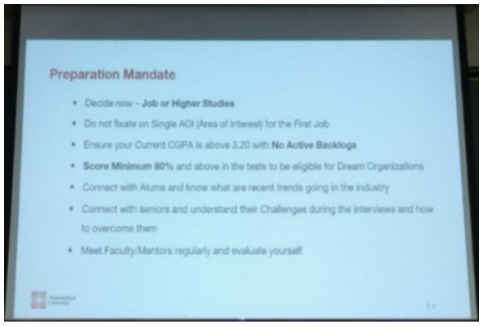

Fig. 2. Slide after document detection and perspective correction.

Finally, Figure 3 shows the enhanced scan-like version. Here, adaptive thresholding has been applied to create a strong black-and-white effect similar to CamScanner.

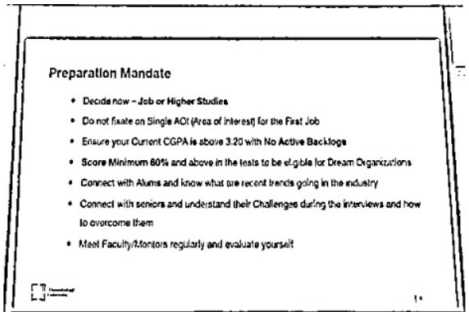

Fig. 3. Enhanced scan of the slide (CamScanner-like output).

B. Exam Paper Example

Figure 4 shows the original photo of an exam paper kept on a textured surface. Different lighting and shadows are visible.

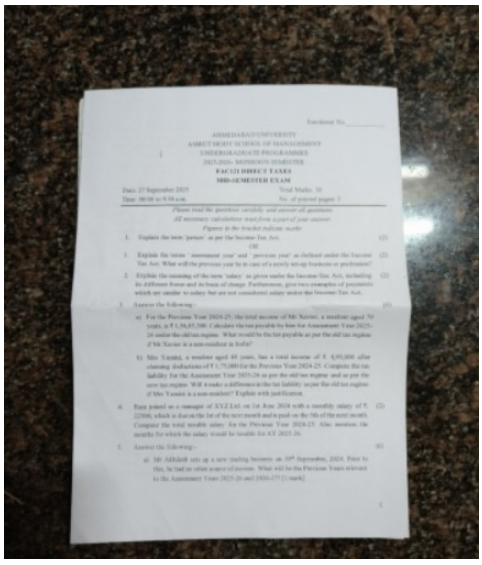

Fig. 4. Original exam paper image captured with a phone.

Figure 5 shows the perspective corrected version where the paper appears flat. The boundaries are aligned and more suitable for reading.

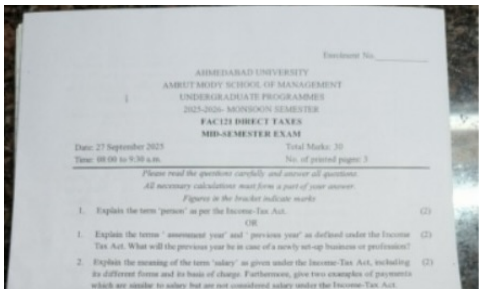

Fig. 5. Exam paper after perspective correction.

In Figure 6, we show the final enhanced scan-like output. The background becomes almost white, and the text becomes darker and sharper, making it look very similar to a real scanned page.

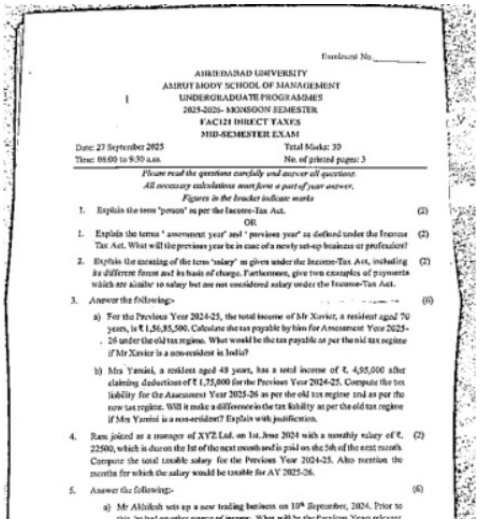

Fig. 6. Enhanced scan of the exam paper (CamScanner-like output).

Challenges Faced

While building and testing the full pipeline, we faced several practical issues:

- In low contrast scenes or with very bright backgrounds, the edges of the paper were weak and sometimes incomplete, which made contour detection harder.
- When the document and background had similar colours, the algorithm sometimes picked a wrong contour as the paper.
- Strong perspective distortion required careful tuning of contour area limits and Canny thresholds to still detect the correct quadrilateral.
- To avoid full failure cases, we had to design a simple fallback where the whole image is treated as paper if a proper contour is not found.

After tuning the parameters and testing on multiple images, the system started giving consistent results for most normal cases.

IV. DISCUSSION

The final results show that it is possible to get scanner-like outputs from normal mobile photos using only classic image processing methods. Grayscale conversion and Gaussian blur form a simple but effective preprocessing block, which prepares the image for edge detection.

Canny edge detection combined with dilation is able to produce strong edges around the document in most situations. Using these edges, contours can be detected and analysed to estimate the document shape and position.

Perspective correction using the four corner points is an important step. It changes the document from a skewed

or tilted view to a flat rectangular view. This makes the document easier to read and more suitable for further processing.

Adaptive thresholding on the warped document gives a strong improvement in clarity. It automatically adjusts to local lighting changes and gives a good black-and-white effect. This is very close to what commercial scanner apps do and is enough for many basic use cases like reading, printing or sharing.

However, the method is still sensitive to extreme lighting, very cluttered backgrounds or very low contrast between the paper and the surface. In such cases, more advanced methods such as learning-based document detection or automatic parameter tuning could help.

V. CONCLUSIONS

In this project, we built a simple but complete document scanning pipeline using Python and OpenCV. Starting from a normal smartphone capture, the system is able to detect the document, correct its perspective and produce a clean scan-like output.

The main steps are preprocessing (grayscale and blur), edge and contour based paper detection, perspective transformation and adaptive threshold based enhancement. Experiments on classroom slides and exam papers show that the final results are clear and usable in most normal conditions.

The project also highlighted the importance of parameter tuning and careful handling of failure cases in real-world images. Even with classic methods, we can get good performance if the pipeline is designed step by step and tested properly.

As future work, the system can be extended with automatic parameter selection, better document detection in cluttered scenes and a user-friendly GUI so that non-technical users can also use it easily as a simple desktop or mobile tool.

REFERENCES

[1] J. Canny, “A Computational Approach to Edge Detection,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-8, no. 6, pp. 679–698, Nov. 1986.

[2] S. Suzuki and K. Abe, “Topological Structural Analysis of Digitized Binary Images by Border Following,” Computer Vision, Graphics, and Image Processing, vol. 30, no. 1, pp. 32–46, 1985.

[3] X. Li, B. Zhang, P. C. Yuen, R. W. H. Lau, “Smartphone-based Document Image Rectification,” IEEE Access, vol. 7, pp. 127654–127666, 2019.

[4] OpenCV Documentation, “OpenCV: Open Source Computer Vision Library,” [Online]. Available: https://docs.opencv.org/.

[5] “Document Scanner using OpenCV,” YouTube Video Tutorial, [Online]. Available: https://youtu.be/q-gWjCgigTg.