Hello,

This is Lakshman Kumar from University of Maryland. In this video, I will be talking about “Mobile Scanner”, which was my final project for the course , Digital Image and Video Processing .

As you all might know, the usage of cell phones is increasing day by day. They have started to replace various traditional devices that have been used for a variety of functions . One such device is the document scanner. Several popular applications like CamScanner and EverNote , which are available in the app stores of various mobile phones, are being used to scan documents nowadays. This project attempts to reverse engineer the methodology used by such apps to extract and register documents.

So the goal of the project will be to extract a document from a mobile phone image, and then to register a reference blank page, and finally to extract content from the filled version of the reference document.

Before I talk about the techniques being used, some assumptions are made to facilitate easier processing. The reference document is considered to be captured in a well lit and contrasting environment . It is also expected to be rectangular and convex in shape. The boundaries of the document are assumed to be clearly visible . The reference document is supposed to occupy the most area in the captured image, and also the filled document is presumed to have sufficient features.

Now let’s look at the methodology for Document Extraction. Firstly, we need to resize the image if its greater than a certain size to reduce computational cost. Then we enhance the image and extract the edges in it. The largest rectangular convex contour in the image is extracted and it is considered to be the document. The corners of this rectangle is then used to compute the perspective transform that is required to extract the document.

Now lets go through an example for document extraction.

Consider this image which contains the reference document. In order to make this image suitable for document extraction, we first convert it to grayscale and then boost its edges by sharpening. We also smoothen the image to reduce the noise but without distorting the edges using the Bilateral Filter.

This would be the resulting image.

In order to extract the edges in this image, we use the sobel filter to compute the gradients and the direction. At that point we do Non Maximum Suppression to make the edges thinner. Then we classify the pixels to Strong or weak edges by double thresholding, after which we do hysteresis to identify whether the weak edge is a part of the strong edge or not.

The extracted edges would look like this.

There might be some holes and discontinuities in the extracted edges. To rectify that we do morphological closing by dilating the edges first and then eroding it. Also there might be some noise in the image, to remove that, morphological opening is done by doing erosion followed by dilation.

The output image appears to have clear boundaries.

Now, we need to find the largest rectangular and convex polygon in the image. For doing this we use Open see vees find contour function which basically implements the border following method proposed by Suzuki yet all.

As can be seen, the extracted contour has clearly identified the boundaries of the reference document .

In order to extract the corners of this contours, we will be using Harris Corner Detection method. We compute the gradients of the contour using a Sobel Filter. Based on these gradients we construct the Harris Matrix and compute its response. If we threshold the Harris response properly and apply non maxima suppression, we will be able to extract the 4 corners of the contour.

The algorithm has seamlessly extracted the four corners.

We then , order these corners in a clockwise manner and create a reference rectangle based on this. At that moment we compute the perspective transform between the corners and the reference rectangle using the equations shown in the flowchart. Using this perspective transform, we warp the image to obtain the reference document and apply bi linear interpolation to fill in the vacant pixels.

The resulting image appears to have properly extracted the reference document

In order to obtain the binary version of this document, we will be applying Adaptive Thresholding using the Gaussian Minus C method. Compared to global thresholding which has a constant threshold, adaptive thresholding gives good results as the threshold is dynamic.

And, here is the output of adaptive thresholding.

Now, consider the filled version of the reference document that is partially visible. We need to extract the content of this document and overlay it on top of the reference document in-order to remove the obstruction.

To do that, we should first align the scanned document with the reference document and then perform background subtraction to extract the content. This extracted foreground is added to the reference image .

To align the filled document with the reference document we compute ORB features of both the images using Open see vees feature detection API. ORB features are computationally less intensive than SIFT and gives similar performance. The API finds the corresponding features in both the images using the K Nearest Neighbor technique, and these features are then filtered by ratio test and symmetry test. At least 500 such features are computed. Then we extract the four best corresponding features using ran sack and hence we can obtain the perspective transform that aligns the filled document with the reference document.

Here are the 500 corresponding ORB features that passed the ratio test and the symmetry test.

And here are the 4 best features obtained from the 500 corresponding features in 50,000 iterations using ran sack.

The result of warping the filled document and doing bi linear interpolation is seen here.

Now that we have aligned the scanned document, we need to extract the content. In order to do that, we convert both the reference document and scanned document into its binary form using adaptive thresholding. Using combinational logic based on the values of the pixels in thresholded images, the additional content in the filled document is extracted which is then filtered a bit by dilation and median filters.

Here are the binary forms of the reference and filled documents.

The extracted foreground seen here has covered all of the additional text in the filled document but it also has some additional residue due to imperfect thresholding and slight differences in alignment caused by either Bi linear Interpolation or lesser number of iterations in ran sack.

Now the extracted foreground can be overlaid on top of the reference document by replacing the corresponding pixels of the reference image with that of the scanned image.

Hence we have finally obtained the restored version of the filled document.

And here is the binary version of the same.

So, based on these results , we can sort of observe a few things.

The techniques hold good only as long as the assumptions made are satisfied.

The algorithm takes about 35 seconds to give out the final output.

Also, sometimes image specific tuning is required.

And since the assumptions made do not require any sort of markings or specific ink color for the filled content, the output is not perfect.

Background subtraction works really well in-spite of partial visibility of the filled document.

With regards to the future works, I would like to optimize the code and make the algorithm take less than 2 seconds to give the final output. Also the number of assumptions made can be reduced and additional filtering can be done to reduce artifacts. Likewise, automated tuning of certain parameters should be done based on the nature of the image. And finally Optical Character Recognition can be integrated to enhance the mobile scanner.

The source code of this project can be found on my git hub profile, which can be accessed through this link.

And here are some references that I have used as a part of this project.

Thanks a lot for watching my video