BBM 418 - Computer Vision Laboratory

Assignment 1 Report

Ömer Kayra Çetin
2210356060

Methods

Two images are chosen as examples to further help explaining the methods. One is a good working example (Curved Image) and the other one is a bad working example (Random Image).

Edge Detection

I used Canny Edge Detection algorithm for edge detection. The images were blurred first with Gaussian Blur. I chose a 7x7 kernel for blurring process because it was just enough to get rid of the noise in the images but not to lose document lines. An adaptive thresholding approach is used for lower and upper thresholds for Canny Edge Detection. In this way, the thresholds were chosen differently for each image. The idea was to make the edge detection more robust across various lighting conditions and image contrasts by calculating the thresholds based on the median intensity of each image.

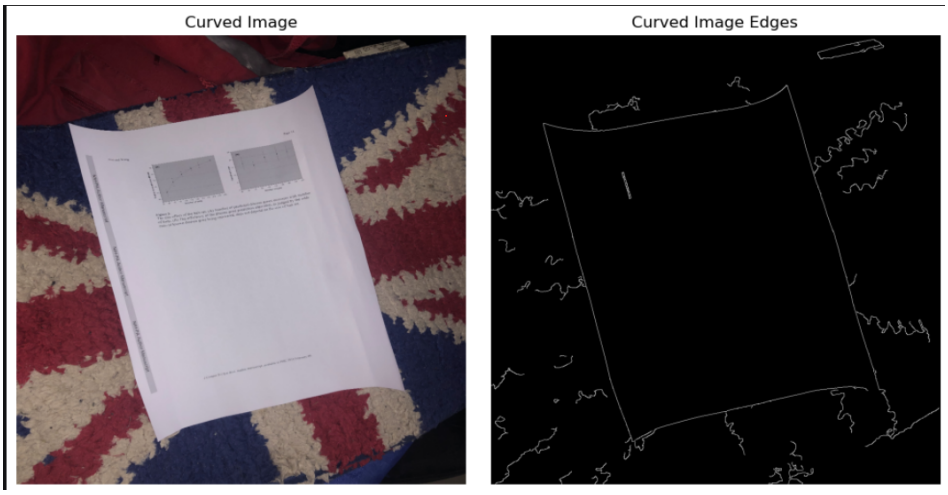

Figure 1: Example curved image edges

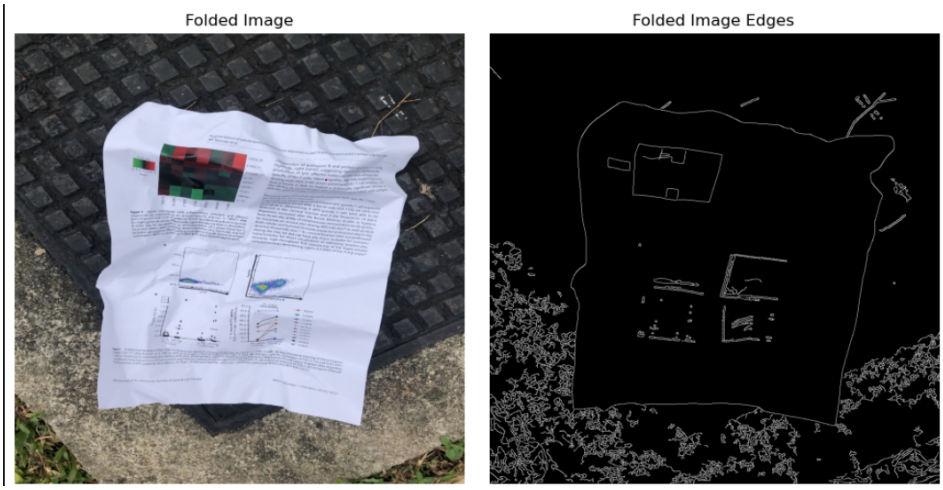

Figure 2: Example random image edges

Line Detection

Hough Transform

Hough Transform method is used to find the lines in the document. Each edge point in the image space puts a vote in the parameter space. If votes are bigger than some chosen threshold that line is chosen. I used an adaptive thresholding approach to limit the lines that I have. I have a base threshold parameter and a threshold factor. Threshold factor is multiplied by the number of edge pixels and then added to the base threshold to finalize the threshold value. I chose this approach to find less lines in the images that have more edge pixels.

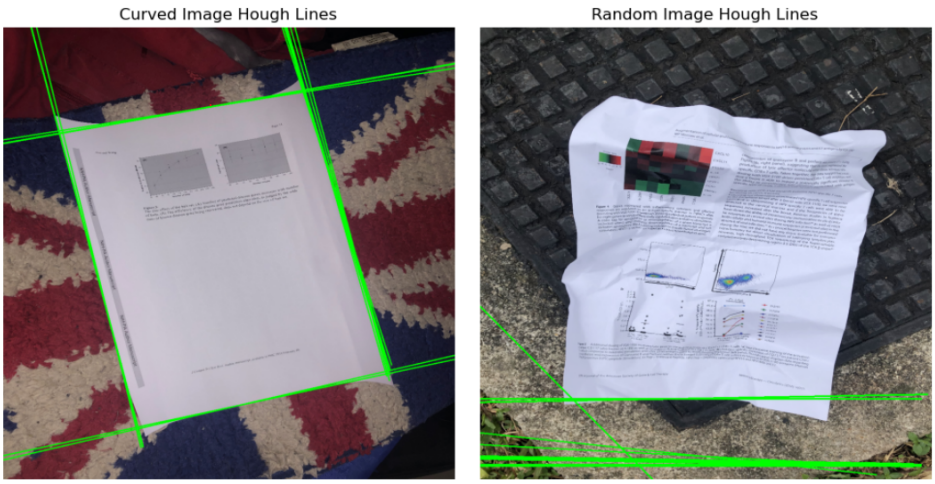

Figure 3: Example hough transform lines

As it can be seen on Figure 3, in first image, hough lines algorithm works well because there are dominant lines. In the second image, since the document is wrinkled, hough lines algorithm can't find lines that match the document well.

RANSAC

RANSAC method is used to find the lines in the document. It is used to enhance the lines that I get from Hough Transform. RANSAC algorithm simply works like this: Iteratively two points are chosen, then the inlier points that are supporting that line is determined, finally if number of inlier points are larger than some threshold, that line is chosen. I chose two thresholds for the RANSAC algorithm to better control the lines that are chosen. One is a basic threshold that limits the minimum number of inlier points and the other one is the minimum ratio of inlier points to all points.

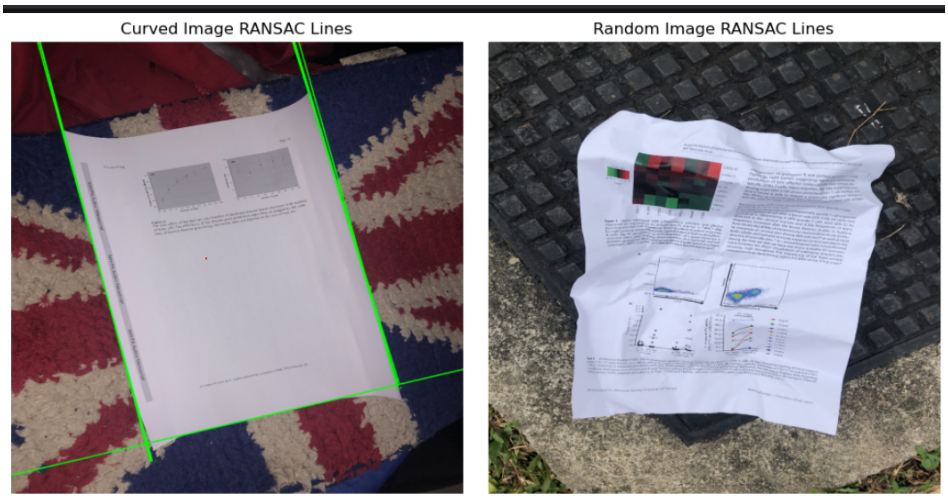

Figure 4: Example RANSAC lines

As it can be seen on Figure 4, there are fewer lines chosen comparing to the Hough Transform. The reason for that is I chose harsher parameters for RANSAC algorithm as Hough Transform were finding enough lines. That way, I

got only the very dominant lines to combine it with Hough lines. In first image, it chose 3 dominant lines and it couldn't find any in the second image.

Combine and Filter Lines

The lines that I get from Hough Transform and RANSAC are combined. Then the similar lines are filtered based on rho and theta thresholds (lines are in polar coordinates). With this approach, I have less lines to work with. That way, the processes that will be done from now on work more efficiently.

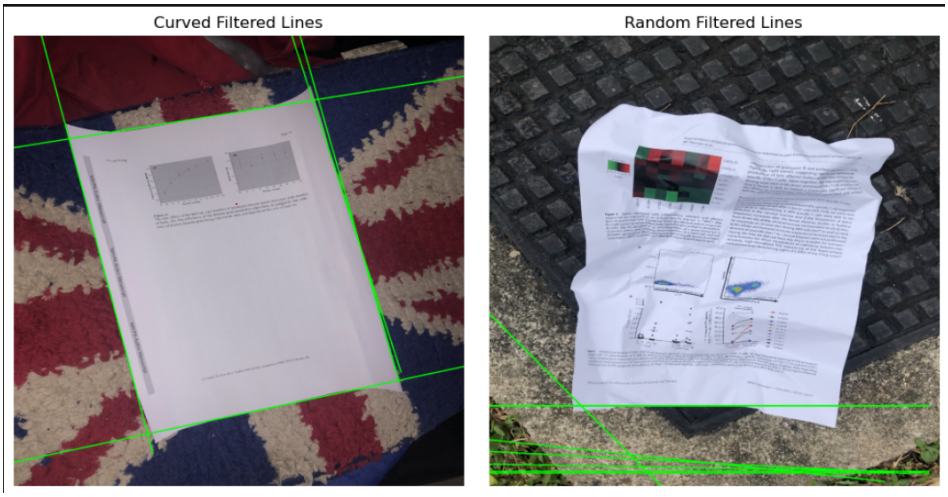

Figure 5: Example combined and filtered lines

Line Intersection

Line intersection points are found as they are the candidates of being the corner points of the documents. They are found by intersecting the lines and filtering the points that fall outside the bounds of the images.

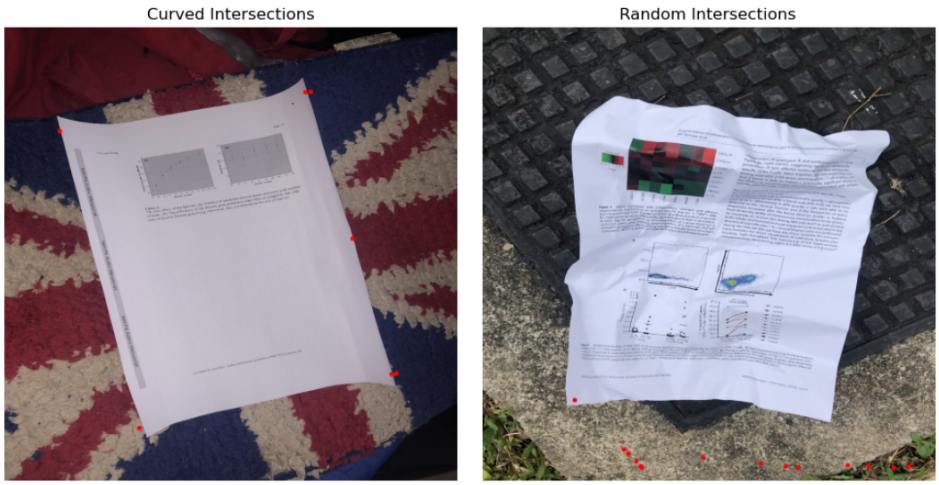

Figure 6: Example line intersection points

Corner Points Detection

In this method, corner points of the document is chosen from the line intersection points. The idea behind the choosing process is to select 4 points from the intersection points, then form a quadrilateral. Then by calculating the area of each quadrilateral, the points that form the biggest area is chosen. They make up the corner points of the document. The reason for choosing the biggest one is to maximize the chances that the document is within this quadrilateral.

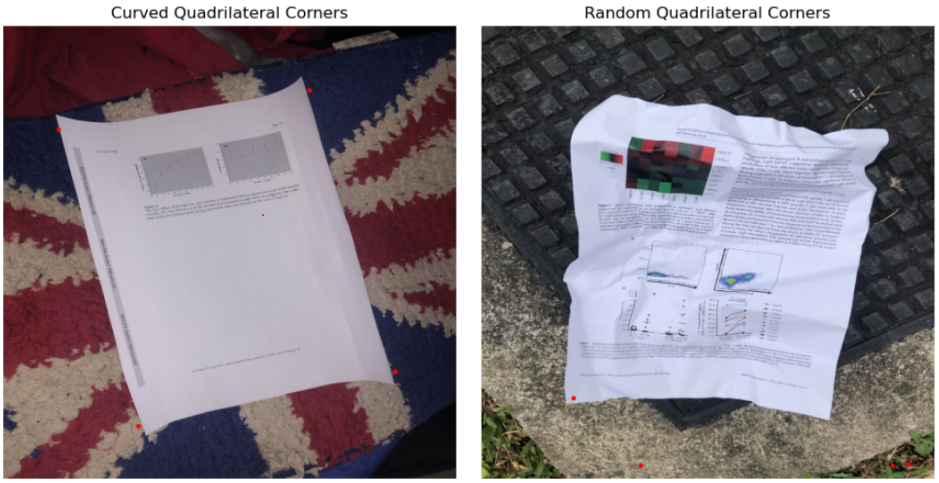

Figure 7: Example corner points

Geometric Transformations

Geometric transformations are used to warp the document into a format that is more readable. I used the four corner points to compute a transformation matrix using cv2.getPerspectiveTransform. This matrix was then applied to distorted image using cv2.warpPerspective to correct perspective distortions and align the document properly.

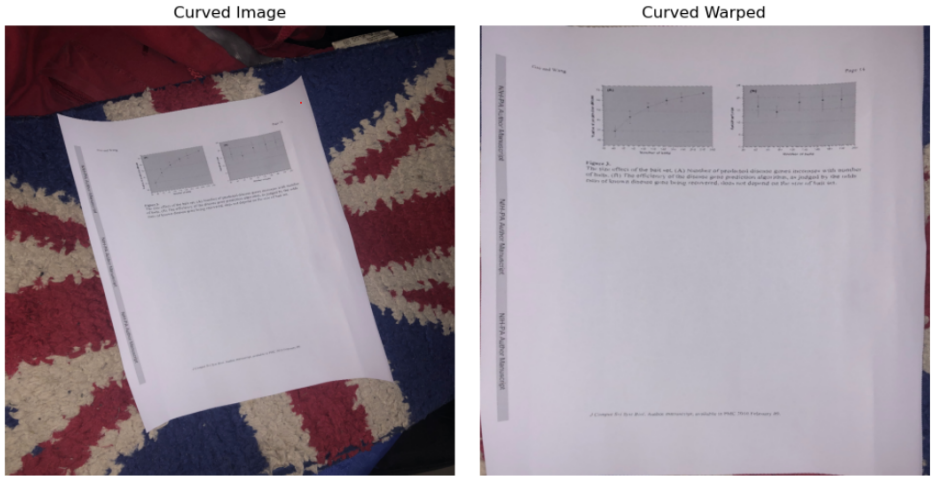

Figure 8: Example corrected document

Figure 9: Example corrected document

As it can be seen on 8, the document is very close to ground truth with the SSIM value of 0.82. The reason for that is that it had very dominant lines and less noise in the image. If we look at 9, it can be seen that the document is not even captured. Its SSIM value is 0.18. This document had more noise and less dominant lines leading it to picking the corner points wrongly and warping the image incorrectly.

Results and Findings

SSIM Results

Looking at SSIM results at Table 1, we can see that results are not great and they do not differ much from each other. I found the reasons why that is the case.

First of all, since the images vary from each other vastly, I wasn't able to find parameters for my algorithms that work well in each case. Some of them have background noise that can lead to incorrect line findings. Suppressing some of this by changing the parameters of edge detection leads to loss of document lines in some other image. It is the same with parameters of Hough Transform and RANSAC. Fine-tuning the parameters for some images leads poor performance in other ones. That is why I chose the parameters that give the overall best performance for both edge detection and line finding algorithms. I chose the parameters by trial and error.

Furthermore, I discovered that this method of line findings is not great with document types such as curved and wrinkled ones in the random folder. Because there may be some dominant lines are missing from the document and the algorithms can miss it. That leads to the image not being correctly warped.

Distortion Type	**Average SSIM**
Curved	0.3579
Fold	0.3673
Incomplete	0.3703
Perspective	0.3485
Random	0.3497
Rotate	0.3680

Table 1: Average SSIM scores for each distortion type.

In some cases I couldn't find even one quadrilateral. In that cases, I passed the raw image to SSIM as the warped image.

Example Images

Here are some examples from each class. In each class, there are two examples, a good working one and a bad working one.

Curved

In Figure 11, warping couldn't be done because there were not a meaningful quadrilateral.

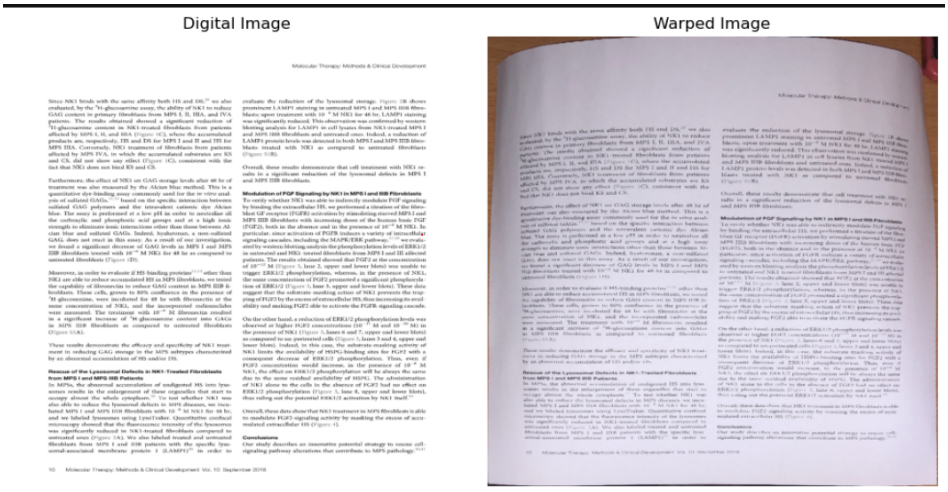

Figure 10: Curved good example

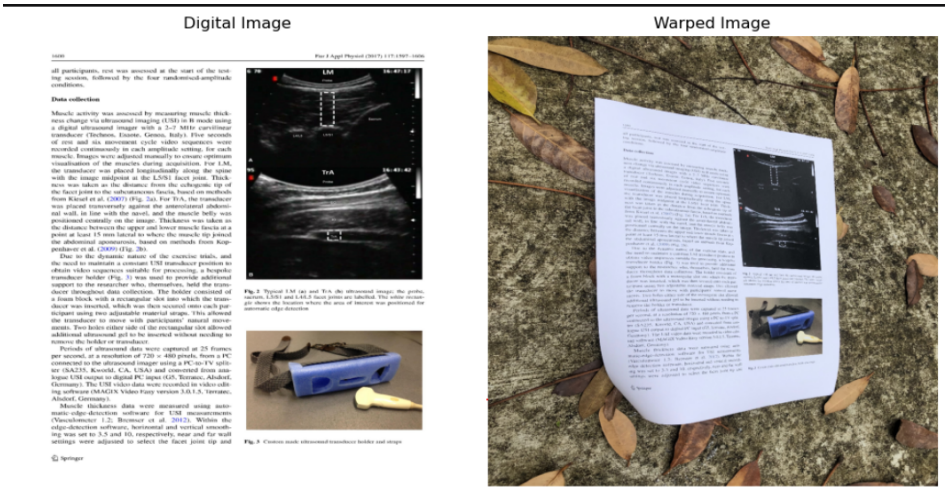

Figure 11: Curved bad example

Fold

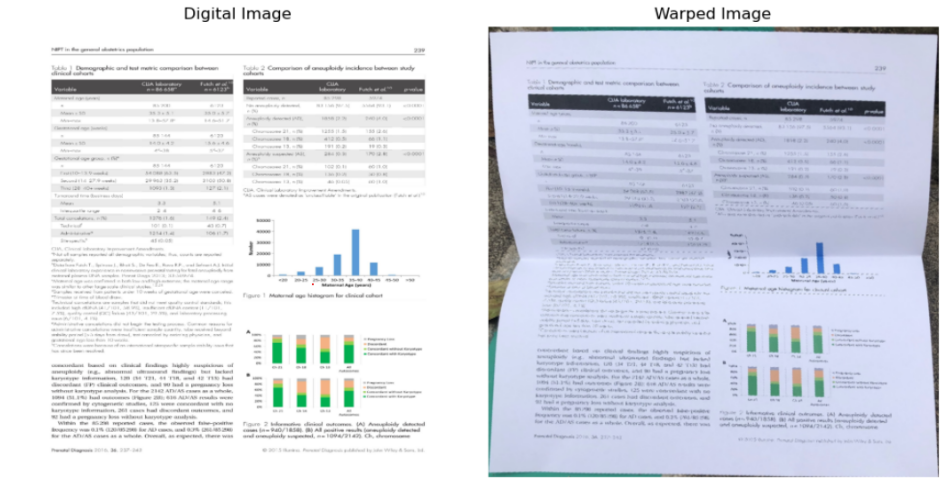

Figure 12: Fold good example

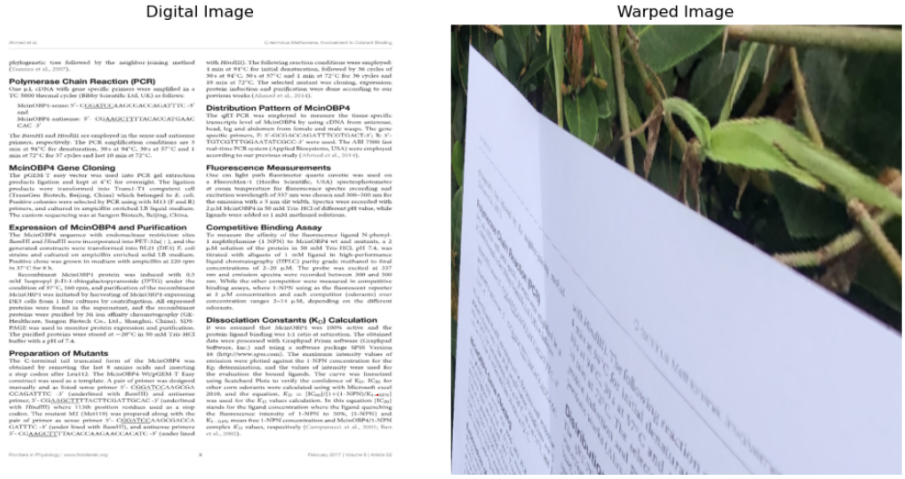

Figure 13: Fold bad example

Incomplete

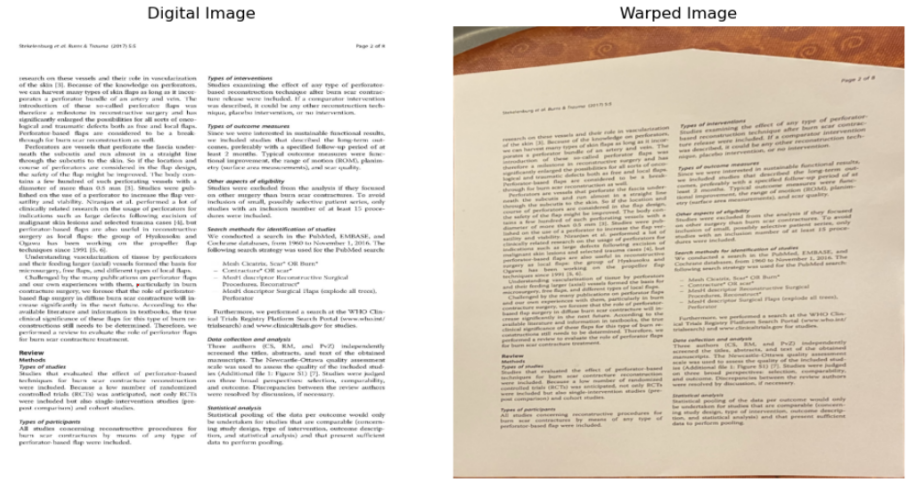

Figure 14: Incomplete good example

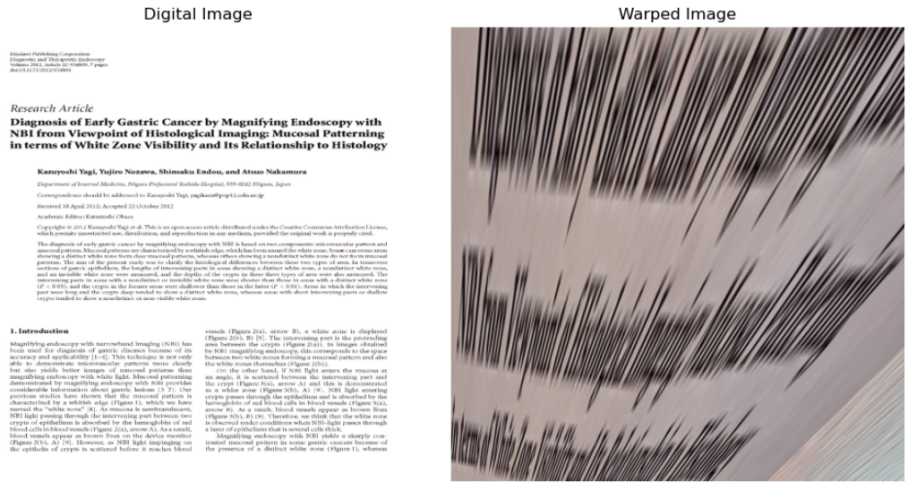

Figure 15: Incomplete bad example

Perspective

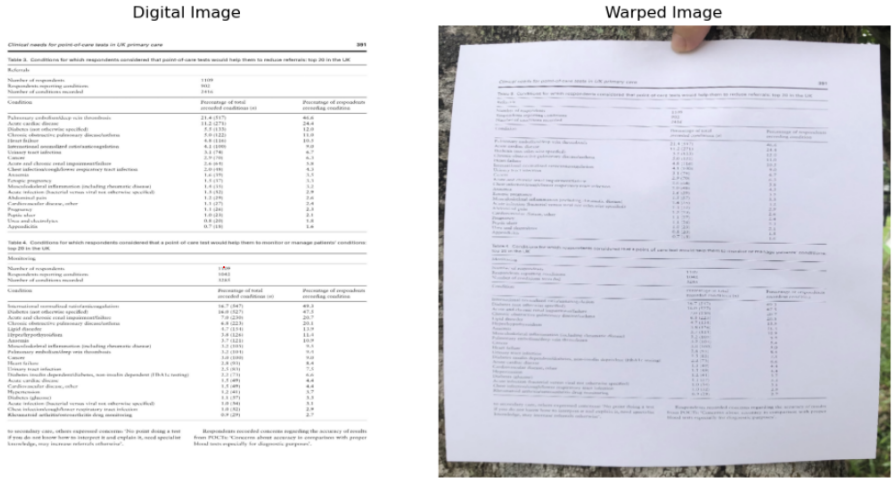

Figure 16: Perspective good example

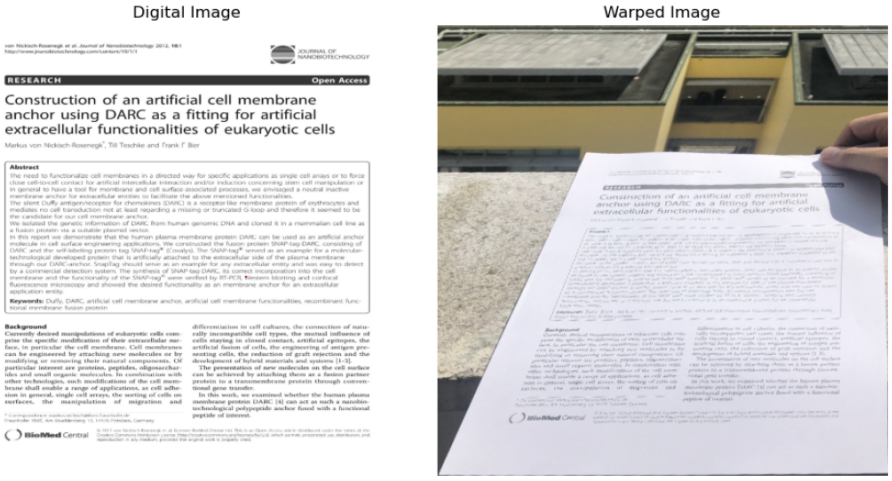

Figure 17: Perspective bad example

Random

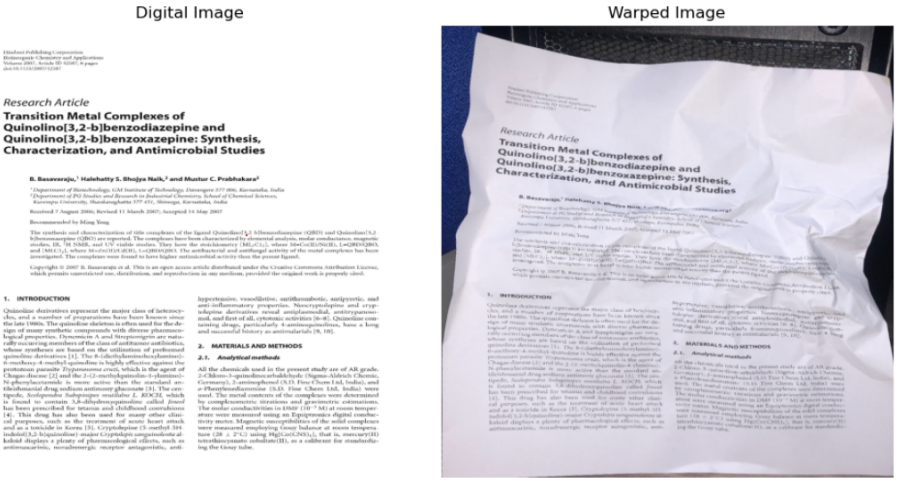

Figure 18: Random good example

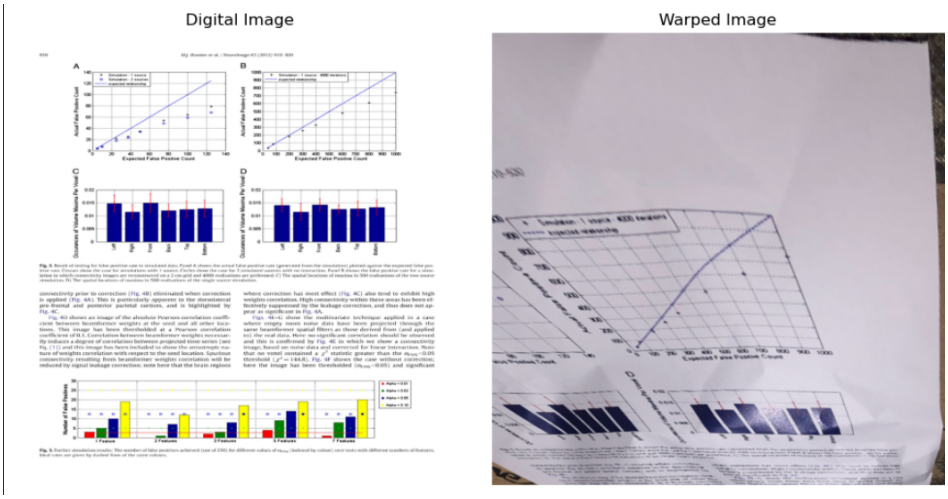

Figure 19: Random bad example

Rotate

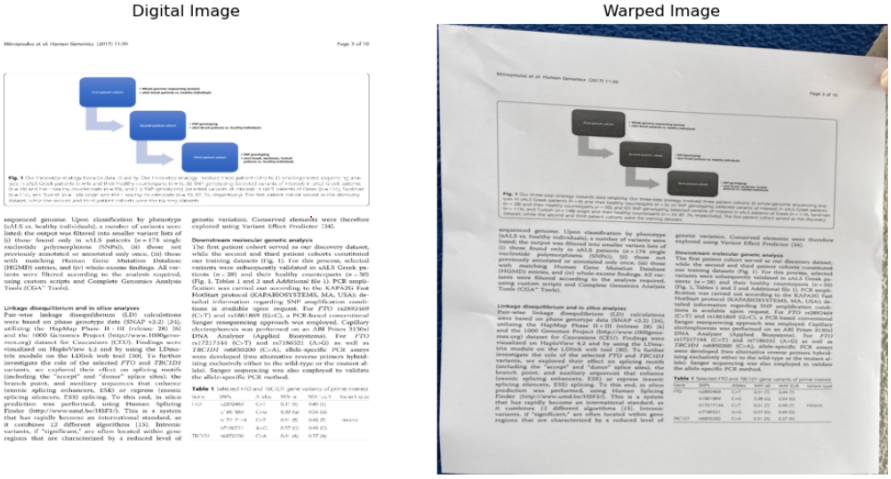

Figure 20: Rotate good example

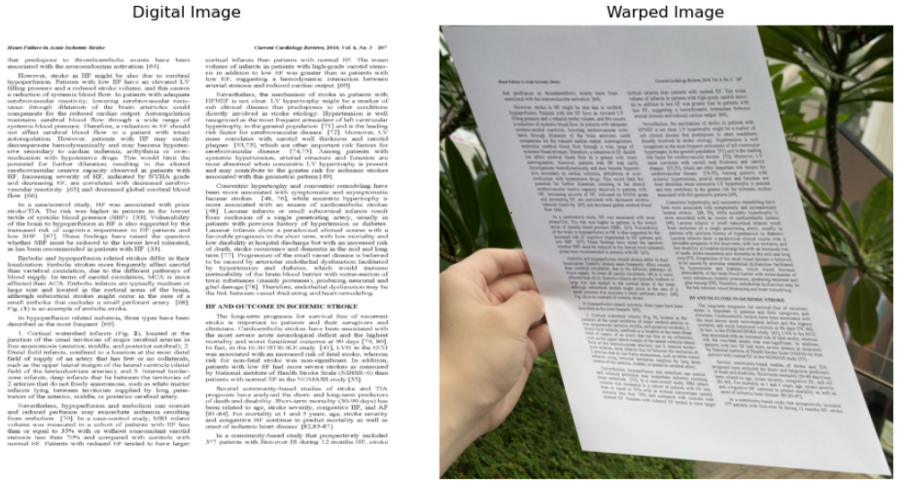

Figure 21: Rotate bad example