Using a Generic Camera as a Scanner

Colin Schmidt, Nicholas Sunjaya, Yi Hong To
UC Berkeley
{colins, nicholas.sunjaya, craigyihongto}@berkeley.edu

Abstract

In this paper we describe a method to turn ordinary pictures of books and papers into scanner-like images. This transforms and aligns the image such that the page takes up the entire viewing area. This makes the image more readable and enables further processing such as character recognition. We present the results of running our algorithm on a hand-annotated set of 50 images. We discuss the successes failures and future work for our algorithm.

1. Introduction

Scanning documents has been a common practice for a long time. Even before the advent of digital scanners faxes were a popular of sending documents to others. But as smartphones have become more ubiquitous and fully featured they have begun to absorb many features of other electronics.

In this paper we present an algorithm for turning a simple smartphone camera, or any other camera, into a scanner. The properties of a scanner we will replicate are: the page scanned fills the image, it is orientated with the top of the page at the top of the image, and that the ratio of the image is the same as that of the paper.

We first describe our motivation for developing this tool in section 2. Next, we introduce our multi-step algorithm with illustrations in section 3. Following this is a description of our experimental methodology and the results of applying the algorithm to a set of images in section 4. Then, we discuss which images are problematic for our algorithm and potential solutions in section 5. Finally, in section 6 we conclude with an exploration of future work and highlight out successes.

2. Motivation

Sending documents to others digitally has huge advantages over physical mail. It is nearly instaneous, the recipient can easily share it with others, and you can often search through the document for particular words and phrases. However, some things are still printed and sent physically including books, notes, and bills. Therefore there is a need for the ability to turn physical papers into digital files. For several decades this was accomplished with a special machine, the scanner, also sometimes integrated into high-end printers.

Recently cell-phone cameras have become much higher resolution. This has enabled a new opportunity, to allow these camera's people have with them at all times to be turned into scanners. The added convience will allow for the convience of digital transmission for all documents even without access to a potentially expensive and/or large scanner.

A cell-phone application can also offer an easy interface to the user. Printers and scanners can be notoriously fussy so moving to an easy to update and change UI/UX could greatly improve the usability.

3. Algorithm

Our algorithm is a multi-step process that identifies the bounding box of the item and then use homography transform to produce the final image. We will now walk through the algorithm with a sample image seen in figure 1.

The first step of the algorithm is to identify edges in the image. We converted the images to gray scale. Then we smooth the image with a gaussian filter to help remove some background noise. Then we use a canny edge detector to give us the binary image of edges. Because of the non-maximal supression the canny edge detector, it gives us the best chance of getting fine lines. Fine lines enable the next steps in our algorithm to be more precise. The output of the edge dection can be seen in figure 2.

Once we have the binary image of edges we need to detect which edges correspond to straight lines. To accomplish this we use the Hough transform [1]. Hough transform indentifies lines by keeping track of the pixels in terms of their slope. If a bunch of pixel belongs to a line, they should have the similar slope with respect to a starting point. The Hough transform algorithm keep track and score the collection of pixels. We set this transformation such that gives us up to 10 line candidates. These prospective lines can be

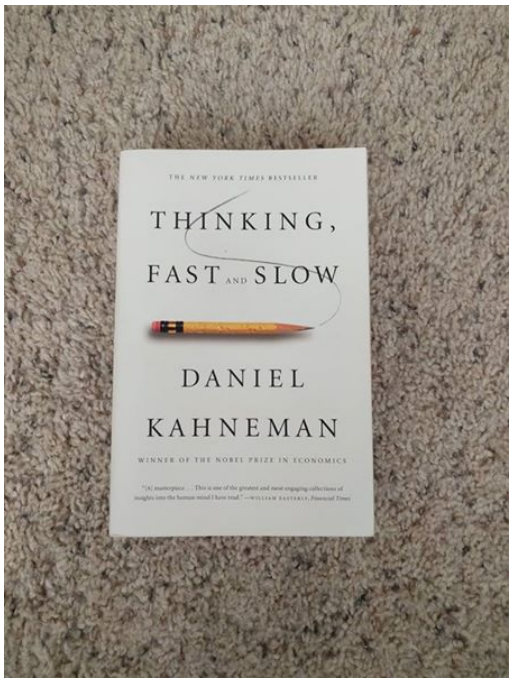

Figure 1. What the image looks like before any modification.

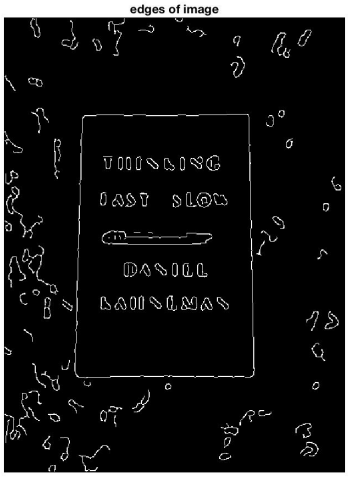

Figure 2. After edge-detection.

seen in figure 3

Next we use a random sampling method inspired by RANSAC to pick the lines that best represent the item. In this step we use some assumptions about the images to develop a heuristic for which lines should be best. Hough transform returns the coordinates of the endpoints of the lines. From there, we can compute the vector that parametrizes the line. To determine the corner, which is

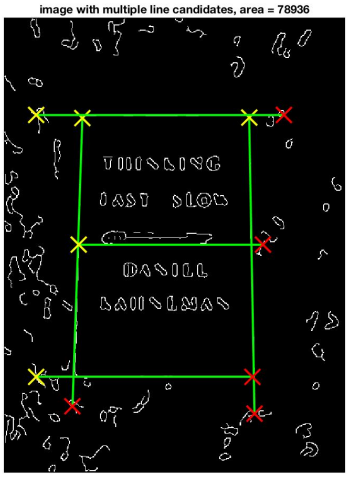

Figure 3. The second step

the intersection of a horizontal and vertical line, we perform cross product on the vectors of the lines. We assume that the largest quadrilateral in the image is the item, such that maximizing the area of our polygon will be the correct bounding box for our item. We also assume that it will be approximately rectangular, such that lines forming near right angles are preferred over acute or obtuse angles. Using this criteria we then randomly sample four lines from the set keeping track of which best meets our heuristic. At the end of this process we have four lines we believe to lie on the bounding box of the item. With these lines we then find their intersections and mark these as the corners of the item. The result of this step can be seen in figure 4.

Finally we use this set of corners as the input to a homography transformation. Once we have computed the transform we then transform all pixels in the bounding box to get the final image as seen in figure 5.

4. Results

We tested our algorithm on a set of 50 manually annotated images of books, and papers. The image set had a variety of backgrounds, colors, and lighting. We tuned the parameters of our algorithm across all the images such that for the final test run we used the same parameters for all images.

We categorized images as successes or failures based on the detected corners compared to the annotated corners. Results that were within a small margin of the annotated corners counted as successes and otherwise were failures. This resulted in an accuracy of 71% in our image set. In the following section we will discuss the failure modes of our

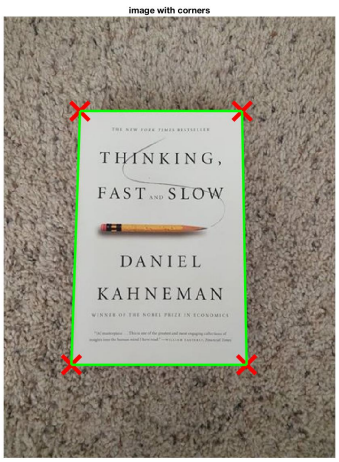

Figure 4. The third step

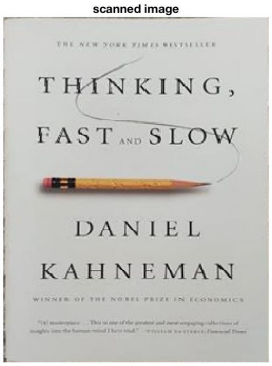

Figure 5. The fourth step

algorithm.

5. Discussion

Even though we have a decent success rate for most images there are some particular failure modes that are worht talking about.

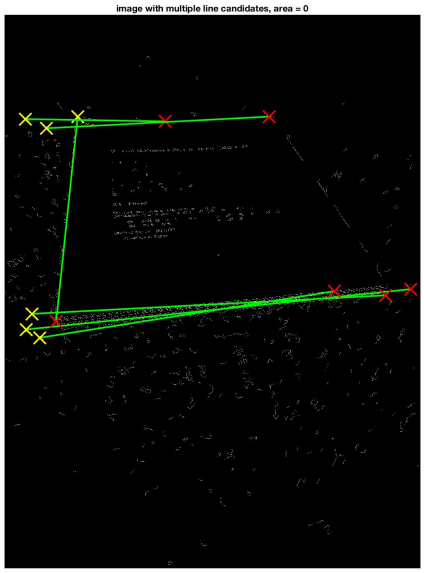

Figure 6. An image which has an undetectable edge.

5.1. Fewer Edges

The first such scenario is where the edges of the item are simply difficult to detect. In figure 6 the Canny edge detector is unable to find the right edge of the paper. This kind of error could be adressed potentially with more preprocessing of the image. There is a tradeoff with this type of error and some of the future errors which are results of having too many edges. If we have the edge detector identify more edges it becomes difficult to pick the right set.

5.2. Extra Edges

The remaining types of errors fall into a different category where we have detected edges that do not correspond to the actual edges of the book cover.

The first failure mode we see is where the book has significant depth that also has a sharp edge with the background as in figure 7. This causes the largest bounding box to include an additional edge of the book that is not the true edge of the cover. One heuristic to possibly detect the difference between the cover and the side of the book would be to expect that side of the book will often be a consistent cover, e.g. white. Then if we find that we have two quadrilaterals to pick from and one of them is essentially the other but with a large monochrome segement attached we can guess that this is the side of the book.

A similar failure mode appears in figure 8. However in this failure rather than the side of the book it is the shadow of the book that creates the larger false rectangle. This failure seems similar enought to the previous mode that the same heuristic may work well again. If the shadow is solid

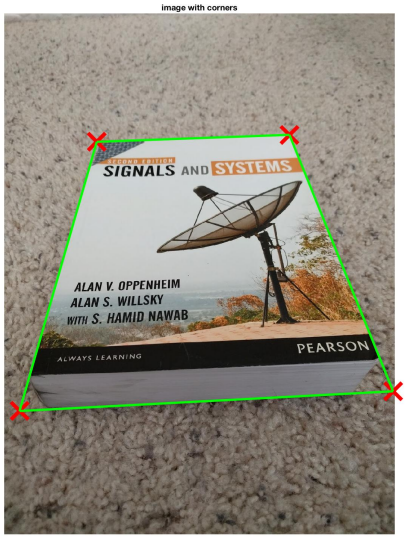

Figure 7. A book which is deep and thus has a larger rectangle than the correct one.

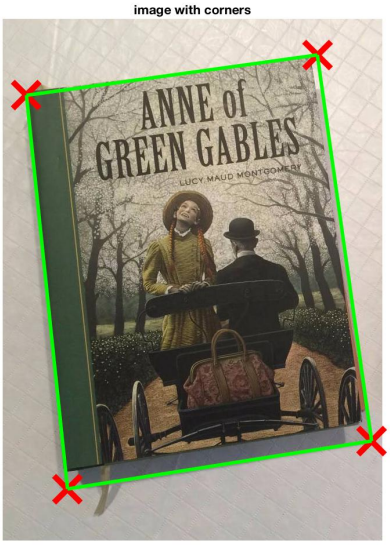

Figure 8. A book which has a well defined shadow and thus a larger rectangle than the correct one.

enough to be detected as an additional line then it is probably monochrome just like the side of the book. In our example image this turns out to be true and so we hypothesize that using the same monochrome attachement heurstic from before could correct this image.

Finally the last failure mode we discuss is that of extra lines on the cover of the book or page. These are actual lines that should be detected by any sensible edge detector but cause problems with our bounding box selection as seen in

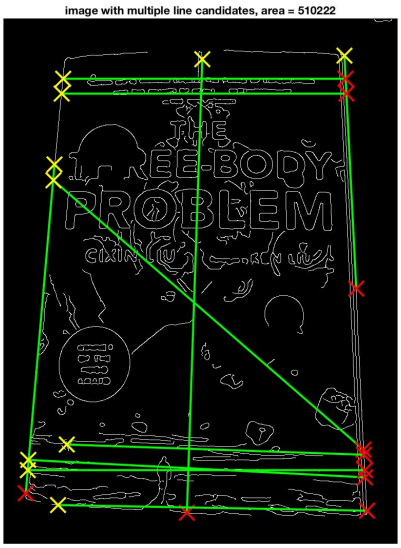

Figure 9. A book which has a lot of extra edges on the cover making it hard to find the right set of four.

figure 9. This is certainly our most difficult to handle failure mode, and has no clear solution of mitigation technique.

6. Conclusion

Overall our algorithm is able to acheieve a 71% accuracy rating on a diverse set of book and papers with interesting and varied backgrounds. There are several particular failure modes that could be addressed with additional some heuristics. We believe the algorithm remains useful as a cammer to scanner conversion despite these hiccups.

References

[1] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer vision, graphics, and image processing*, 44(1):87–116, 1988.