Computational Photography

Final Project – Paper Homography

Introduction

In this final assignment we will combine several concepts from earlier in the course with new concepts of line parameterization and image rectification. In particular, you will be developing a "scanner app".

Throughout this you will use the image you took of a piece of paper. **This must be a picture different than the one shown throughout this example, although you can use that for verifying your work.**

You **may** work with one other person if you like.

Although not a definitive/comprehensive list, some Matlab functions that you cannot use include:

- `hough`
- `imregionalmax`
- `gradient`
- `bwlabel`
- `regionprops`
- `houghpeaks`
- `imtransform`
- `imwarp`

Grading Scheme:

1. Image Prep and Edge Detection (5pts)
2. Hough Transform (15pts)
3. Line Selection (20pts)
4. Line Intersections (10pts)
5. Rectification (20pts)
6. Works on more than one sample image (10pts)
7. Additional tests (performed by faculty) (20pts)

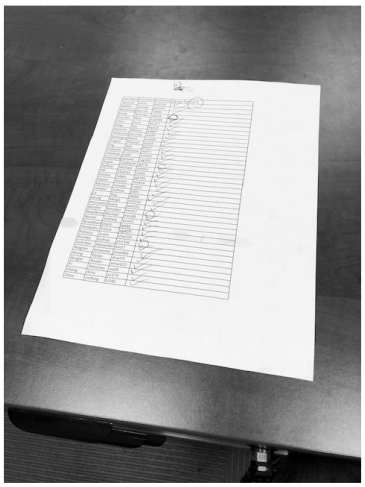
Part 1: Image Prep and Edge Detection

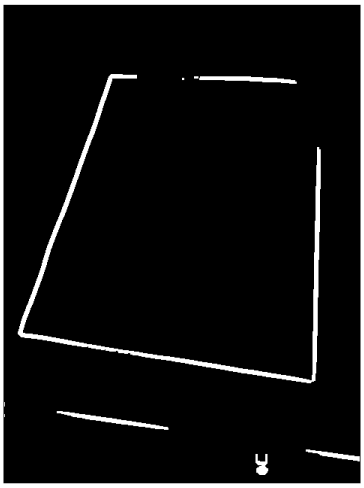
First take your image and apply edge detection to it. Hopefully you did this successfully in HW3. If not, now's the time to make it work! **Resize the image to have a height of 512 pixels (maintain aspect ratio).** This way things aren't too computationally expensive.

Notes:

- You **may** use Matlab functions to reshape the image.
- You **may** use Matlab functions to extract edge pixels.

For your report show the original image and the edge image.





Part 2: Hough Transform for Line Detection

Next use a Hough Transform to find candidate lines, like you did in HW5. Display this as an image.

Part 3: Relevant Line Identification.

From the Hough transform you will likely be able to identify locations of local maxima. The difficulty is grabbing the correct subset to form the four lines of your piece of paper.

I'm going to let you think about how to do this yourself. Keep in mind that this should be automated such that it could be applied to another test image. To get you started, here's some ideas:

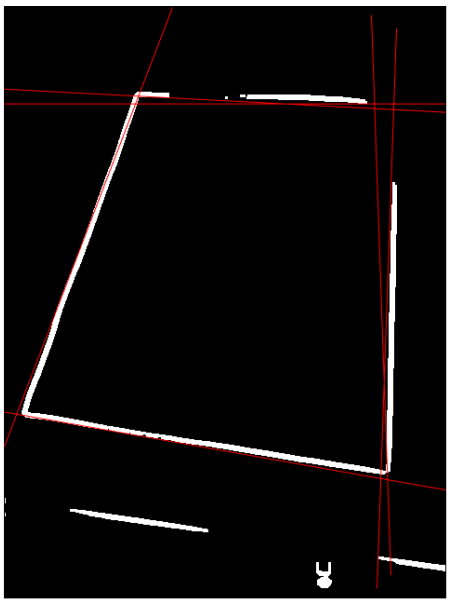
1. Set a threshold.
2. Find local maxima over some fixed window size.
3. Enforce some constraints based on the nature of the problem (in this case, finding the four edges of a plane/paper.

Again, you **MAY NOT** use functions like `imregionalmax` or `regionprops` to help you with this.

In your writeup explain how you selected potential lines (including citations, if necessary).

Finally, superimpose on your Hough transform these locations and draw the lines on your edge image.

NOTE: If you can't do this automatically, just let us know and "manually" select them. That way you can move on to the last part.



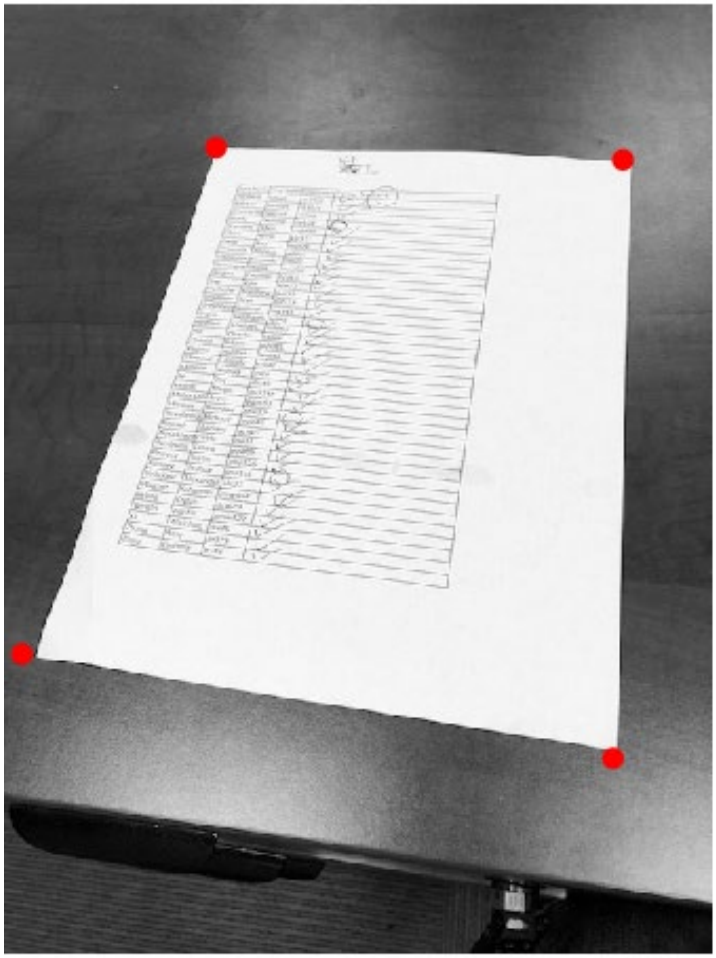
Part 4: Line Intersections

Next, you'll want to find the intersections of your lines to establish the four corners of your paper.

From the polar form of the line, $\cos(\theta)\, x + \sin(\theta)\, y = r$, you should be able to get the slope-intercept form of a line, $y = mx + b$.

Go through your potential lines and compute their intersections. Use this set of intersection points to determine the four corner points of your paper.

Superimpose on your image the four corners that were found.



Part 5: Image Rectification

Now we should be able to rectify our paper! We will assume a standard 8.5×11 letter paper size. Based on this aspect ratio create a "blank" image (make it large enough to show details, likely similar size to the original image).

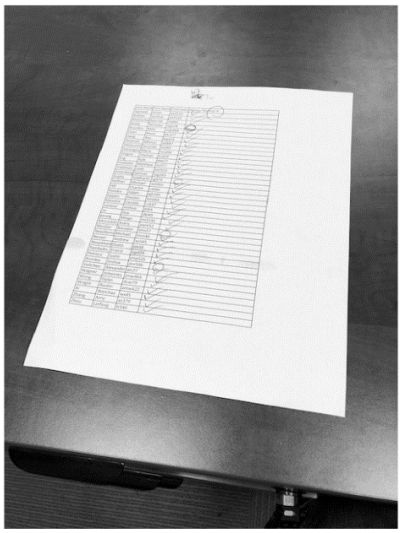
The four corners of this blank image should correspond to the four intersection points you discovered. Using these four correspondences, compute the *homography matrix.*

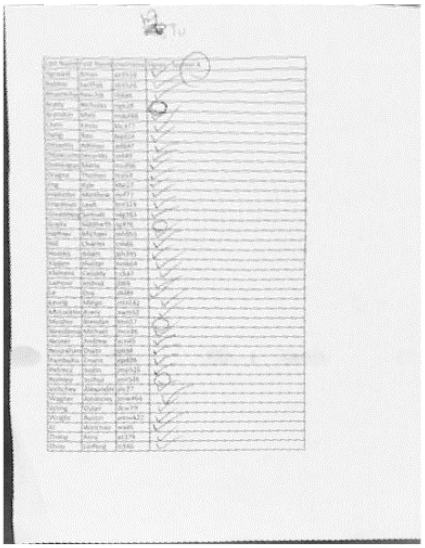
Once you have your homography matrix, go through each location in your new image and compute its corresponding location in the original image. Copy the pixel at that location to your new image.

Show your newly rectified image!

You **may not** use any Matlab functions to do this for you. This includes (but is not comprehensively limited to):

- `imtransform`
- `imwarp`





Part 6: Another Example!

Finally, we want to make sure that your implementation works on another example without having to change anything in your code.

Provide a second input and output image showing this. Again, this cannot be the sample image I've been showing to illustrate the parts of the project.

Submission

1. Assignments must be submitted via Bd Learn
2. Submit a single compressed file (zip, tar, etc..) containing:
    a. A PDF file containing:
        i. Your input and edge image for Part 1
        ii. Your 2D Hough Transform histogram image for Part 2
        iii. A description of how you extracted points from your Hough Transform and the selected lines superimposed on your edge image, for Part 3
        iv. The intersections, superimposed on your original image, for Part 4.
        v. Your rectified image for Part 5
        vi. Your extra example, for Part 6
    b. A README text file (**not** Word or PDF) that explains
        i. Any additional features of your program
        ii. Any instructions on how to run your program to reproduce your results.
    c. Your source files
    d. The chosen images that you are processing.