# MAGNIFY

Haluk Can Gavas

[#]*Department of Information Systems and Engineering*
*Istanbul Technical University*
*Istanbul, Turkey*
[1]gavas17@itu.edu.tr

***Abstract*— Taking an image of a document is the easiest way of obtaining a digital copy of it. However you can't edit, copy or paste the text from an image. In order to deal with these issues I have created an python program called magnify.**

***Keywords*— Computer Vision, Opencv, Pytesseract. Pillow, OCR**

## I. Project Summary

Magnify is a python application that analyses document images and then extracts their text into a raw text file by using optical character recognition.

## II. Project Requirements

This project requires the use of Python as its core language for image processing algorithm with openCV and PIL library. For the optical character recognition component I have used pytesseract. Pytesseract is a wrapper for Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries.

## III. Project Structure

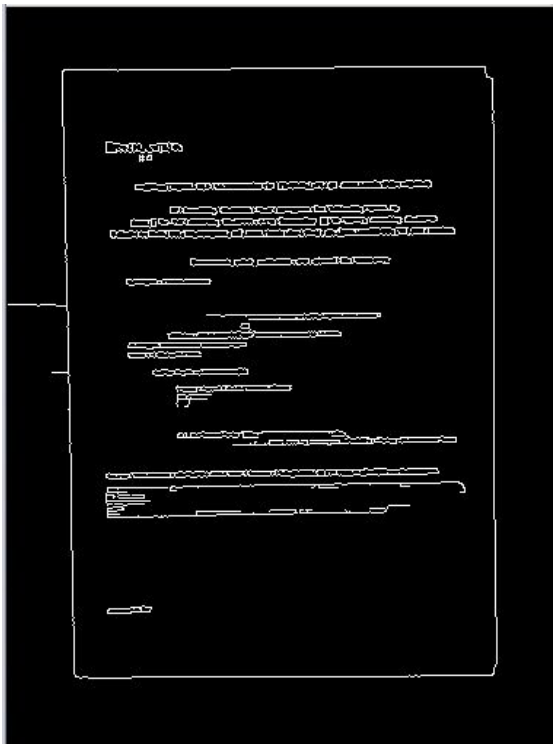
The program converts an image to text by going through 4 primary steps:

1. Edge Detection

The initial step is to perform edge detection. So as to speedup image processing, just as make edge detection step more precise, I resized my scanned picture to have a height of 500 pixels.

I additionally had to take careful consideration to monitor the proportion of the first height of the picture to the new height this allows me to perform the scan on the original picture as opposed to the resized picture.

From that point, I changed over the picture from RGB to grayscale, perform Gaussian blurring to remove high frequency noise, and perform Canny edge detection. After these actions the layout can be seen.



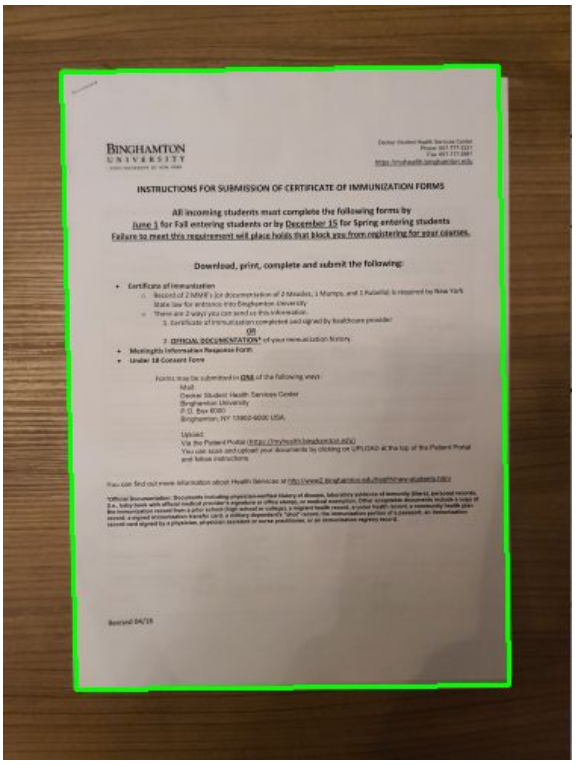
2. Finding Contours

While finding the contours I assume that the biggest contour in the picture with precisely four points is our bit of paper to be filtered. This is a logical assumption the scanner program essentially expect that the document you need to scan is the main focal point of my picture.

What's more, it's additionally safe to expect that the bit of paper has four edges. I began by finding the contours in our edged picture. A little change I made is to sort the forms by region and keep just the biggest ones. This allows me to just look at the biggest of the contours, discarding the rest. I at that point begin looping over the contours once again and estimated the quantity of edges.

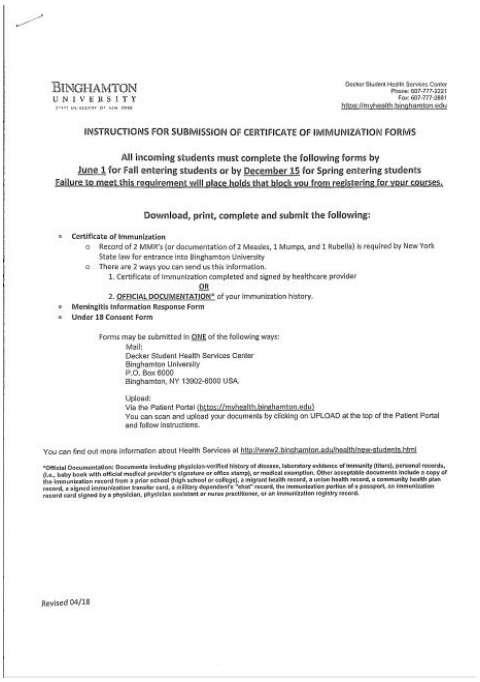
In the event that the approximated contour has four edges, I expected that I have discovered the document in the picture. I believe this to be a fairly logical assumption. The scanner program will accept that the document to be examined

is the primary focal point of the picture and the document is rectangular, and in this way will have four unmistakable edges. From that point, the program shows the contours of the document I went to check. As you can observe, the program have effectively used the edge recognized picture to discover the contours of the document, outlined by the green square shape seen around my document.




3. Applying perspective transform

The third step is to take the outline of the document and apply a perspective transform to get a top-down angle of the picture. four_point_transform function performs the warping transformation. I have passed two arguments into four_point_transform. the main argument is my original picture I loaded during runtime (not the resized one), and the subsequent argument is the contour of the document, multiplied by the resized proportion. I multiply by the resized proportion since I performed edge detection and discovered contours on the resized picture of height=500 pixels. However, I need to perform the scan on the first picture, not the resized picture, in this way I multiplied the contour points by the resized proportion. To get the highly contrasting feel to the picture, I at that point take the warped picture, convert it to grayscale and apply adaptive thresholding. Because of the adaptive thresholding, the new picture likewise have a decent, clean high contrast feel to the document.


BINGHAMTON
UNIVERSITY

Decker Student Health Services Center
Phone: 607-777-2221
Fax: 607-777-2881
https://myhealth.binghamton.edu

INSTRUCTIONS FOR SUBMISSION OF CERTIFICATE OF IMMUNIZATION FORMS

All incoming students must complete the following forms by
June 1 for Fall entering students or by December 15 for Spring entering students
Failure to meet this requirement will place holds that block you from registering for your courses.

Download, print, complete and submit the following:

- Certificate of Immunization
  - Record of 2 MMR's (or documentation of 2 Measles, 1 Mumps, and 1 Rubella) is required by New York State law for entrance into Binghamton University
  - There are 2 ways you can send us this information.
    1. Certificate of Immunization completed and signed by healthcare provider
    OR
    2. OFFICIAL DOCUMENTATION* of your immunization history.
- Meningitis Information Response Form
- Under 18 Consent Form

Forms may be submitted in ONE of the following ways:

Mail:
Decker Student Health Services Center
Binghamton University
P.O. Box 6000
Binghamton, NY 13902-6000 USA.

Upload:
Via the Patient Portal (https://myhealth.binghamton.edu)
You can scan and upload your documents by clicking on UPLOAD at the top of the Patient Portal and follow instructions.

You can find out more information about Health Services at http://www2.binghamton.edu/health/new-students.html

*Official Documentation: Documents including physician-verified history of disease, laboratory evidence of immunity (titers), personal records, (i.e., baby book with official medical provider's signature or office stamp), or medical exemption. Other acceptable documents include a copy of the immunization record from a prior school (high school or college), a migrant health record, a union health record, a community health plan record, a signed immunization transfer card, a military dependent's "shot" record, the immunization portion of a passport, an immunization record card signed by a physician, physician assistant or nurse practitioner, or an immunization registry record.

Revised 04/18

4. Converting scanned image to a text file

The final step for my program is to detect the characters present on my newly generated scanned document. To accomplish this I have used pytesseract. As I discussed before, is a wrapper for Tesseract-OCR Engine. It is also useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries. For this application I used the text_to_string function from pytesseract library for my program. In order for this to work more accurately I resize the image by %200 percent and implemented a thresholding function in order to increase the accuracy of the OCR. However those were not enough to make the function useful. The outputs were still messy and unorganized. In order to fix this I have Implemented a simple config file to the function data by using the panda library. After these fixes are over my end result looked like this.

```
NY Decker Student Health Services Center
BINGHAMTON Phone: 607.777.2221

UNIVERSITY Fax: 607-777-2881
BATE WNOVERSITY OF NEW YORK https://myhealth.binghamton.edu

INSTRUCTIONS FOR SUBMISSION OF CERTIFICATE OF IMMUNIZATION FORMS

All incoming students must complete the following forms by
June 1 for Fall entering students or by December 15 for Spring entering students

Failure to meet this requirement will place holds that block you from registering for your courses.

Download, print, complete and submit the following:

* Certificate of Immunization
o Record of 2 MMR's (or documentation of 2 Measles, 1 Mumps, and 1 Rubella) is required by New York
State law for entrance into Binghamton University
o There are 2 ways you can send us this information.
1. Certificate of Immunization completed and signed by healthcare provider
OR
2. OFFICIAL DOCUMENTATION* of your immunization history.
o WMeningitis Information Response Form
o Under 18 Consent Form

Forms may be submitted in ONE of the following ways:
Mail:
Decker Student Health Services Center
Binghamton University
P.O. Box 6000
Binghamton, NY 13902-6000 USA.

Upload:

Via the Patient Portal (https://myhealth.binghamton.edu

You can scan and upload your documents by clicking on UPLOAD at the top of the Patient Portal
and follow instructions.

You can find out more information about Health Services at http://www2.binghamton.edu/health/new-students.html

*Official Documentation: Documents Including physIcian-verified history of disease, laboratory evidence of immunity (titers), personal records,
(i.e., baby book with official medical provider's signature or office stamp), or medical exemption. Other acceptable documents include a copy of
the immunization record from a prior school (high school or college), a migrant health record, a union health record, a community health plan
record, a signed immunization transfer card, a military dependent's "shot" record, the immunization portion of a passport, an immunization
record card signed by a physician, physician assistant or nurse practitioner, or an immunization registry record.

Revised 04/18
```

```
def get_string(image):

        #thresh = 255 - cv2.threshold(image, 0, 255, cv2.THRESH_BINARY_INV + cv2.THRESH_OTSU)[1]
        #thresh = cv2.GaussianBlur(thresh, (3,3), 0)
        custom_config = r'-c preserve_interword_spaces=1 --oem 1 --psm 1 -l eng+ita'
        d = pytesseract.image_to_data(image, config=custom_config, output_type=Output.DICT)
        df = pd.DataFrame(d)

        # clean up blanks
        df1 = df[(df.conf!='-1')&(df.text!=' ')&(df.text!='')]
        # sort blocks vertically
        sorted_blocks = df1.groupby('block_num').first().sort_values('top').index.tolist()
        for block in sorted_blocks:
                curr = df1[df1['block_num']==block]
                sel = curr[curr.text.str.len()>3]
                char_w = (sel.width/sel.text.str.len()).mean()
                prev_par, prev_line, prev_left = 0, 0, 0
                text = ''
                for ix, ln in curr.iterrows():
                        # add new line when necessary
                        if prev_par != ln['par_num']:
                                text += '\n'
                                prev_par = ln['par_num']
                                prev_line = ln['line_num']
                                prev_left = 0
                        elif prev_line != ln['line_num']:
                                text += '\n'
                                prev_line = ln['line_num']
                                prev_left = 0

                        added = 0  # num of spaces that should be added
                        if ln['left']/char_w > prev_left + 1:
                                added = int((ln['left'])/char_w) - prev_left
                                text += ' ' * added
                        text += ln['text'] + ' '
                        prev_left += len(ln['text']) + added + 1
                text += '\n'
                return text
```

## IV. State of the art & Novel contributions

Similar works and solutions for this problem are mostly scanner programs that scan a document and provide a digital copy of the scanned document. The key difference of my solution is unlike scanning a whole page I am detecting all the words one by one and create a pure text document which will be easier to manipulate and make changes unlike a classic scanned document, in addition to that it will also allow to copy and paste from an image. Aside from this another contribution I tried to add onto the existing solutions are indents. The library in and on itself doesn't provide a particularly accurate indent detection. Meaning it doesn't detect the indents well. What I did was to create a custom config for the library so the indents turned out to be slightly better.

## V. Shortcomings

Due to lack of proper library support I couldn't made this into an android application. Even though I already made a GUI I couldn't make the code work inside the application. The second shortcoming is handwriting recognition. The example ocr handwritings found online mostly works, however unlike them my own handwriting is pretty bad, due to this the algorithm can't recognise the characters inside the image of my handwriting.

```
TT Aic docum eny is Made tor Jesbing
```

Here is how my program read my own handwriting. It was supposed to read "This file is made for testing".

In summary the code needs to be refactored in order to be usable inside an android application and handwriting/indent recognition can also be improved to exactly match to the document.

## VI.Conclusion

While creating this project I learned and revisited a lot of topics. I learned how to apply edge detection more accurately using. I learned how optical character recognition worked. I learned how to customise config dataset files in order to modify the output of my ocr function.While I am learning all this I also had the opportunity to brush up on my knowledge on filters, convolution etc.

Possible improvements I am planning on making on my project:

- Refactoring the code so it runs on android
- Improving handwriting recognition
- Improving indent recognition
- Improving edge detection so the edges could be detected even on non flat surfaces
- Improving edge detection so only the edges of the document are recognised instead of everything in the image.

All in all creating this project thought me a great deal regarding image processing and helped me brush up on my knowledge I gained throughout the semester

*References*


[1] OpenCV python documentation
https://docs.opencv.org/master/d6/d00/tutorial_py_root.html
[2] Pillow python documentation
https://pillow.readthedocs.io/en/stable/
[3] Scipy python documentation
https://docs.scipy.org/doc/
[4] Pytesseract python documentation
https://pypi.org/project/pytesseract/