EN.600.461 Computer Vision
Final Project
Recognizing and Translating Text from Images

Joon Hyuck (James) Choi
(Senior Undergraduate)
The Johns Hopkins University
3400 N Charles Street
Baltimore, MD 21218, USA
jchoi100@jhu.edu

Abstract

This work explores optical character recognition (OCR) in photos of printed and handwritten documents. It first explores basic preprocessing of photos of documents using the OpenCV [1] library such as thresholding and denoising. It next discusses the use of the tesseract-ocr library [2] to perform OCR. It finally discusses the incorporation of the Google Translate API [3] to translate the output text from tesseract-ocr into multiple different languages and template matching using the Johns Hopkins University (JHU) logo.

1 Introduction

This work achieved the following main goals from the original project proposal: 1) Given a photo of a document, convert it into a clean scanned version; 2) Take the scanned version and perform OCR. Optionally, this work took the OCR output and translated the text into different languages using the Google Translate API. Additionally, this work experimented with template matching using the JHU logo in order to determine whether the input document is an official JHU document or not.

2 Methods

In this section, we discuss the methods we took and external libraries used for each stage of our work.

[1] http://opencv.org/
[2] https://github.com/tesseract-ocr
[3] https://cloud.google.com/translate/

2.1 Image Preprocessing

We implemented our program in Python using the OpenCV library. In order to feed the OCR algorithm clean input to achieve best performance, we used `cv2.medianBlur` to smoothen the input image. We used an aperture size of 5. Then, we passed the smoothened image through `cv2.adaptiveThreshold` with *adaptiveMethod*= `ADAPTIVE_THRESH_GAUSSIAN_C`, *threshold = binary*, *blockSize=5x5*, and *C=2*. The adaptive method we chose uses the weighted sum of the *blockSize* x *blockSize neighborhood of pixel (x, y) - C* as its threshold value.

Finally, we performed denoising on the thresholded image to remove noise and make the output clean. We used `cv2.fastNlMeansDenoising` with *templateWindowSize=7*, *searchWindowSize = 21*, and *h=7*. Numbers were chosen empirically. The motivation behind denoising in our work was that document scans are prone to have a lot of specs and noise such as the salt and pepper noise. Such noise can greatly reduce OCR accuracy because an OCR algorithm may confuse a noisy spec with punctuation marks or associate a spec with an actual character near by (e.g. confuse an l with an i).

2.2 OCR

In order to perform OCR on the cleaned inputs, we used the `tesseract-ocr` library and via python wrapper `pytesseract`. We used the raw output from `pytesseract`'s `image_to_string` method to pass into python's file writer and Google Translate API.

As a side experiment, we used the `Keras` [4] library to train the MNIST [5] dataset of 70,000 handwritten digits on three different convolutional neural networks (CNN). Results will be discussed in section 3.X. We referred to code for building the three CNNs from an online source [6].

2.3 Translation

We made use of Google's Translate API in order to translate the output OCR processed document into several different languages based on user input. We referred to code posted online [7] and modified details for our purposes.

2.4 Template Matching

Aside from OCR, the main focus of this project, we experimented with template matching in images. Specifically, we used the Johns Hopkins University (JHU) logo to determine whether the input document was an official JHU document or not. We naively assumed that official JHU documents would contain the JHU logo for experiment's sake. We used `cv2.matchTemplate` with our input document as the source image, multiple JHU logo images (each different scale-`cv2.matchTemplate` is scale sensitive), and matching method `cv2.TM_CCOEFF_NORMED`. If the normazlied `TM_CCOEFF` value came out to be greater than 0.51 for any of the input JHU logo templates, we determined that the input document contained the JHU logo in it. The matching method and determinant value of 0.51 were chosen empirically.

3 Results

4 Discussion

5 How to Run the Code

Assuming that the user has Python 2.5+, pytesseract, and tesseract installed, run the following command on the command line in the same directory as `driver.py`.

[4]https://keras.io/

[5]http://yann.lecun.com/exdb/mnist/

[6]http://machinelearningmastery.com/handwritten-digit-recognition-using-convolutional-neural-networks-python-keras/

[7]http://github.com/mouuff/mtranslate

Type of Text	Font Size	Style
paper title	15 pt	bold
author names	12 pt	bold
author affiliation	12 pt	
the word "Abstract"	12 pt	bold
section titles	12 pt	bold
document text	11 pt	
abstract text	10 pt	
captions	10 pt	
bibliography	10 pt	
footnotes	9 pt	

Table 1: Font guide.

```
$python driver.py {path-to-image-file} {list of target languages separated by single space}
```

Sample usage:
`$python driver.py doc7.png es fr`
performs OCR on `doc7.png`, outputs text in the original input language, and outputs the text translated in Spanish and French.

- Left and right margins: 1in
- Top margin:1in
- Bottom margin: 1in
- Column width: 3.15in
- Column height: 9in
- Gap between columns: 0.2in

Captions: Provide a caption for every illustration; number each one sequentially in the form: "Figure 1. Caption of the Figure." "Table 1. Caption of the Table." Type the captions of the figures and tables below the body, using 10 point text.

Acknowledgments

We thank the python libraries we used for image processing and OCR, the code we referred to for constructing CNNs using `Keras` and making use of the Google Translate API, and MNIST for the handwritten digit dataset.