Book Cover Recognition

Linfeng Yang(linfeng@stanford.edu)
Xinyu Shen(xinyus@stanford.edu)

***Abstract*—Here we developed a MATLAB based Graphical User Interface for people to check the information of desired books in real-time. The GUI allows user to take photos of the book cover. Then it will automatically detect features of the input image based on MSER algorithm, then it will filter out non-text features based on morphological difference between text and non-text regions. In order to further improve the detection result, we implemented a text character alignment algorithm which significantly improves the accuracy of the original text detection. We also compared the built in MATLAB OCR recognition algorithm and a commonly used open source OCR. To perform better detection results,we implemented post detection algorithm and natural language processing to perform false detection inhibition and word correction. Finally, we linked the detection result to internet to perform online matching. With some tolerance, more than 86% accuracy can be achieved by our algorithm.**

keywords-Text Recognition, MSER Feature Extraction, Text Box Alignment, Automatic Word Correction.

I. Introduction

With the development of online shopping, more and more people prefer to shop online for their favourite books. However sometimes we still want to visit the real store to get more details about the books. When visiting bookstores, people always want to find more details about the book they are interested. Generally, we want to know more comments on that book, we may also want to compare the prices for the same book from different stores. In this way, we aimed at developing a GUI that can take the advantages of both online shopping and real store shopping. Therefore, the goal of our project is to provide readers with more book information (comments,prices). In this way, people can make better decisions on taking books in the bookstore. In the future, this project may be implemented on mobile device and better help readers.

II. Related Work

During the last few decades, there are a lot of research and development related to this topic. Smith, R. help us reviewed the development of Tesseract OCR engine[1], their inspiring work lead us to think about using OCR combined with necessary pre-processing steps. Patel.C also lead us to a case study using OCR in the detection of natural texts[2]. In their work, they compared the Tesseract OCR with Transym, and concluded that Tesseract OCR is more stable when analyzing large image database, and they analyzed the advantages of Tesseract OCR. Their work inspired us to develop more techniques to increase the accuracy of Tesseract OCR. There are several ways to extract features from images for text recognition, WT Ho et. al used SIFT Algorithm to extract important features from license plate for text recognitions[3], whereas Chen H et.al firstly introduced the idea of using Maximally Stable Extremal Region (MSER)[4]. Using mobile document database their experimental results demonstrate the excellent performance of the proposed method. After some online research, we also found MATLAB R2016b documentation provided us a detailed example on extracting texts from natural images. Based on all the literature research, we decided to use MSER combined with geometric and stroke width filtering to design our algorithm.

III. Methodologies and Algorithms

The following processing pipeline (Fig.1) shows how we extract the book information from the book cover and get related price information from the Internet. First,we capture the image from the webcam as the input. Next, some preprocessing methods are conducted in order to make the book cover horizontally. Then, with MSER method help finding the bounding box, we can extract text information such as book title, author name in the book cover by OCR method. Finally, the prices of the book will be showed by searching the Internet using the information we obtained as keywords. Later we will discuss each part in detail in the following sections.

A. Image Preprocessing

1) Image Capture: Considering the sources of the images in our daily lives, we implement two image capture ways. These are loading an image in your

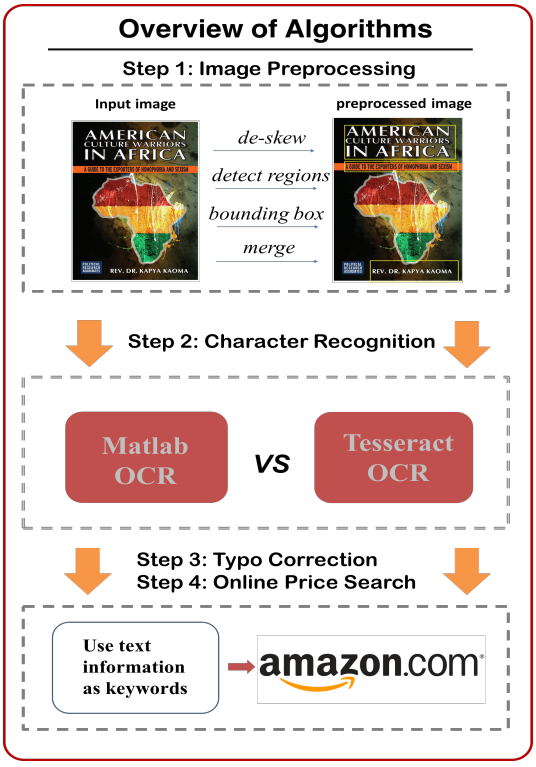

Fig. 1: Algorithm Pipeline

database or taking a real time image by webcam. We use MATLAB GUI to automatically preview and take snapshot of book cover images in natural environment as input. After the input, we let user to select parameters to perform better detection result (the default parameters can already provide decent detection results).

2) Skew Correction: Some limitations of the captured images will influence the performances of character recognition, such as the geometrical distortions caused by the digital camera or the rotations when people took the photo. Thus we need to correct the rotations problem. Since we can use the edge detection algorithms like sobel edge detection to find the book cover's edges, and edges always appear as direct lines, then we use Hough transform to do skew correction. In order to avoid false correction resulted from skews lines that appear in the book cover, we only take top 3 hough peaks.

3) MSER: Before we extract text information from the book cover, we need to recognize which parts of the book contain text. MSER is a method that shows how we detect useful regions in an image that contains text. Basically, MSER will extract the regions where we observed large image gradient.

4) Morphological filtering: After the extraction of MSER, the next step is to filter out non-text regions based on the geometric difference between text characters and non-text regions. The properties weve chosen are: Aspect ratio, Eccentricity, Euler number. Extent and Solidity. Another common parameter used to discriminate between text and non-text region is stroke width. Stroke width is a measure of the width of the curves and lines that make up a character. Text regions tend to have little stroke width variation, whereas non-text regions tend to have larger variations. We use standard deviation to represent the level of stroke width variations.

5) Positional filtering: Generally, the characters are well aligned in the text region on the book cover !!!!!, therefore the bounding boxes should be well aligned if they are real character boxes. We used histogram to group aligned bounding boxes based on their vertical position. Because the non-text bounding boxes will have more random vertical positions, they tend to be low peaks or individual values on the histogram, therefore we can set up group threshold to further filter out non-text features.

6) Bounding Boxes Merging: After the detection of individual character bounding boxes, we need to merge individual characters into words to perform better Optical Character Recognition. Matlab has already implemented an algorithm for merging bounding boxes. However, the algorithm does not consider the individual text lines inside the merging boxes, and it will decrease the total recognition accuracy. We developed a new merging method which considers the single lines inside merged bounding boxes, which significantly improves the recognition accuracy.

B. Optical Character Recognition(OCR)

In order to get better optical character recognition result, we used two different OCR functions. The first one is the built in OCR function in MATLAB, and the second one is an open source OCR from Google[5]. Both of them are generated from tesseract engine but their performances are not identical across different book covers. We will have a detailed comparison between these OCR functions in the next chapter.

C. False detection inhibition and words auto correction

Because the diversity of font sizes and styles, both OCR functions we implemented will have false detections. After the OCR, we first filtered out false detections by setting a word confidence cutoff, then we implement automatic word(typo) correction by downloading the English word database and compare the words detected with the words in the dictionary.

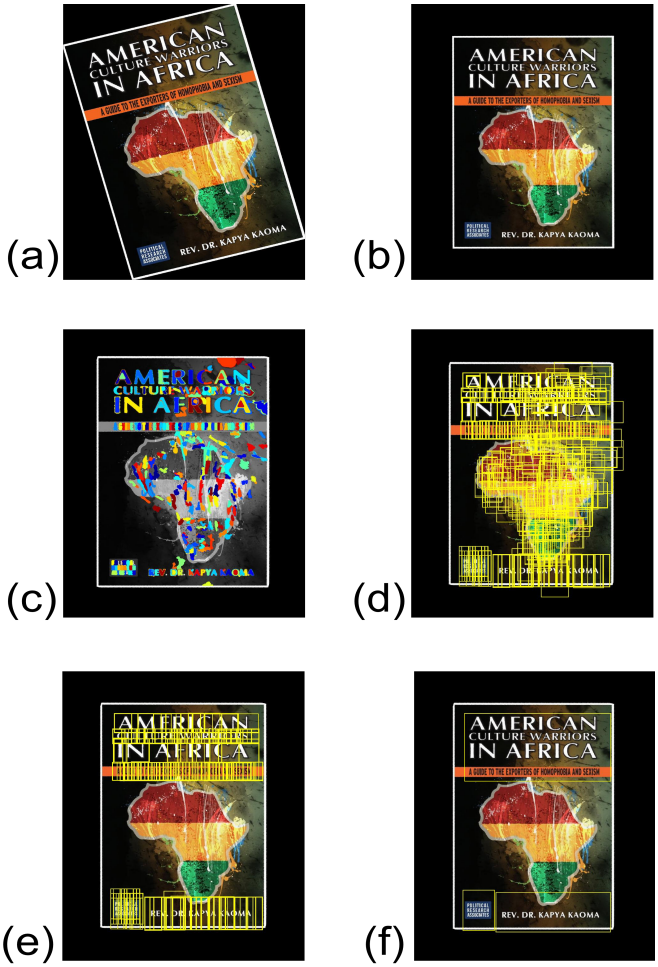

Fig. 2: Results for preprocess
(a): input image (b) de-skewed image (c) detect regions with MSER (d): find bounding box (e): remove non-aligned bounding box (f): merge bounding box

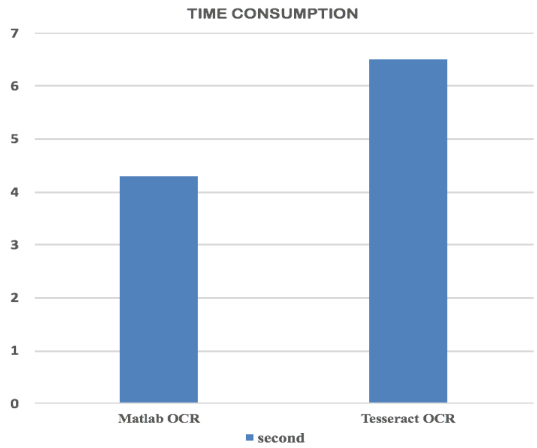

Fig. 3: time cost on MacBook Pro (Retina, 13-inch, Late 2012)

IV. Results Analysis

A. Matlab OCR vs Tesseract OCR

Tesseract is an optical character recognition engine and it is considered as one of the most accurate open source OCR engines. Here, I made comparisons of these two methods. We downloaded 100 test images from google. In order to test the robustness of the system, the front book covers should be of different design styles.

TABLE I: Accuracy comparison

Accuracy	text accuracy	online searching matching
Matlab OCR	57.3%	78%
Tesseract OCR	78.2%	86%

Overall, we can see from the tables (TABLE 1 and Fig.3)that Matlab OCR lacks of the accuracy compared with Tesseract OCR, it saves some time during the processing by the way. And the time cost stands for all the processing procedure after we load an image. Besides, even if we cannot extract exactly the same book name, we can still find the book information online by other information provided and robustness of online searching. Thus, if the user cares more about the time consumption, Matlab OCR would be a good choice.

B. Robustness to rotation

In order to test two OCR algorithms robustness against rotation, we generate a dataset consists of 10 book cover images with different rotation angles (from -30 to +30).Since we want to know more about the rotation effects rather than the OCR detection effects,the dataset we chose has good OCR detecting qualities. We calculated the accuracy by counting the correct characters that are recognized by both algorithms divided by the total number of all characters on the book cover, and the result is shown below. It is suggested from the figure above that both MATLAB built-in OCR and Tesseract OCR perform relatively robust against rotation. It is also shown that the accuracy of open source Tesseract OCR is more correlated with the rotation angle, but the MATLAB built-in OCR is more rotation-invariant. Although as shown in the previous session that the accuracy of Tesseract OCR is slightly better, MATLAB OCR is more robust against rotation. Therefore it is worth putting future effort into integrate the de-skew algorithm in MATLAB OCR and text recognition algorithm in Tesseract OCR to improve the accuracy of OCR algorithm.

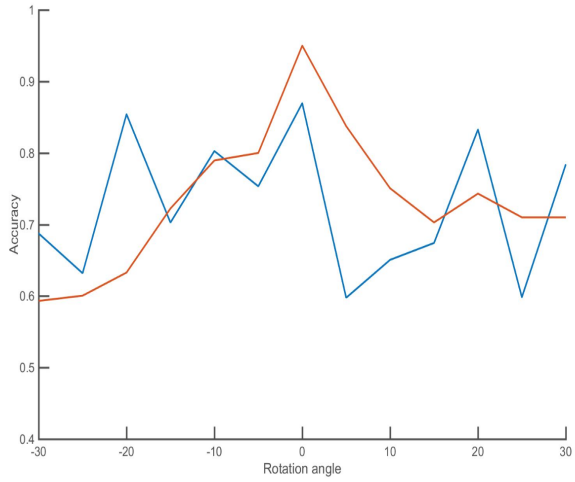

Fig. 4: The comparison of performance against rotation angle for MATLAB OCR(blue line) and Open Source Tesseract OCR(red line)

C. Online price searching

In our project, we just linked to the amazon online website to search for the book price, this can go further by linking to different online stores and find reasonable price by price sorting. The two examples in Figure and Figure show that if we could extract correct book information, we can get the online searching results successfully.(Fig.5 and Fig.6)

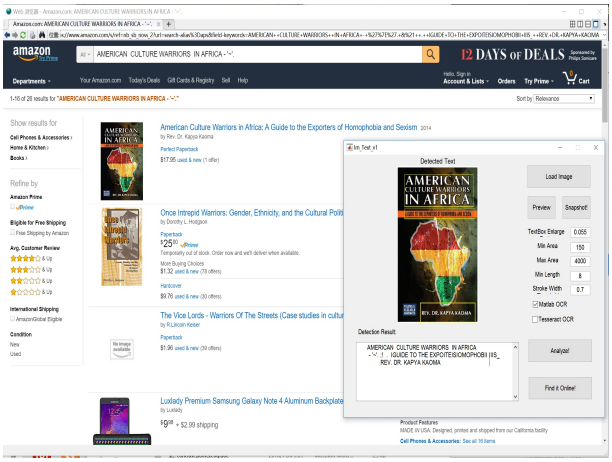

Fig. 5: successful online search examples-1

D. some unsuccessful test cases analysis

This example (Fig.7) shows an input image that our system cannot detect the text information and thus can

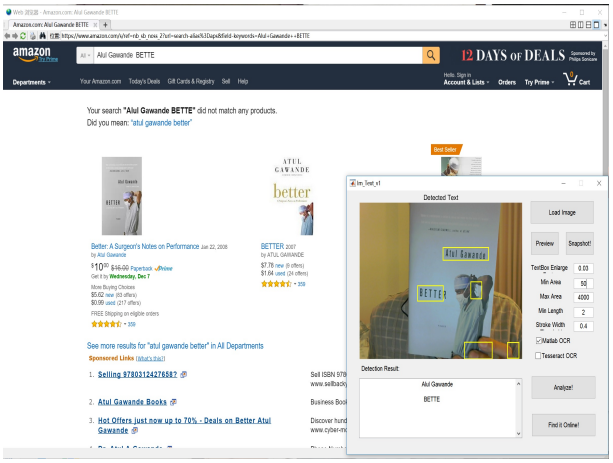

Fig. 6: successful online search examples-2

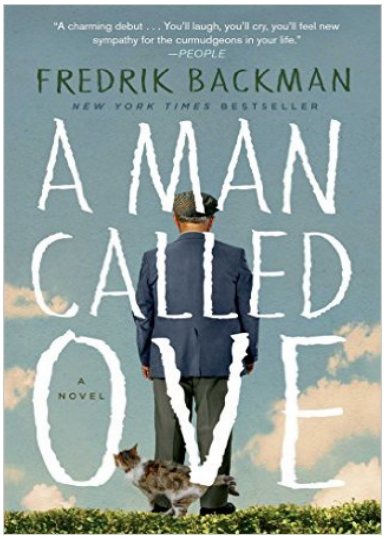

Fig. 7: image cannot be recognized

not find it online. Even if it is obvious to recognize the book name, it is hard for the OCR system to separate out the book information. The main reason may be although the book cover has good visual quality, after preprocessing, the image will have poor contrast because of the man's back picture in the background. OCR has difficulty differentiating documents that have both images and text. Besides, owing to the separate regions of the characters, OCR will detect some single character instead of the words.

The second unsuccessful test case (Fig.8) shows in natural environment, we can see in the Figure that the some of detected bounding boxes outreach the edge of the book cover. Since OCR process text information in the bounding box, the false bounding box will significantly affect the detecting results. This kind of

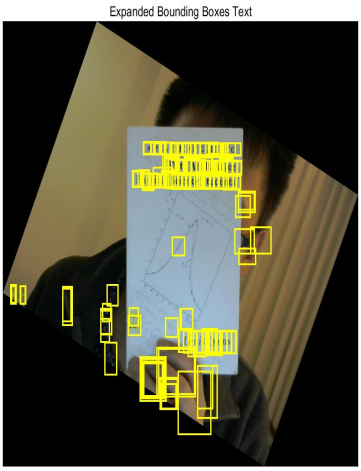

Fig. 8: image cannot be recognized

problem occurs when we use Matlab 2014b. It is because we cannot use graph function in 2014b.The goal of using graph function is merging all the overlapping bounding box and the bounding boxes which are located outside the book edge can be removed by non-aligned bounding box removal. For 2014b, we took another idea. The idea is counting bounding box's size according to their vertical positions and remove them if the vertical bounding box's size is less than the defined threshold.

V. Discussion and Future Work

In this project, we have successfully developed a MATLAB GUI with automatic book cover detection and recognition to help people take the advantages of both online shopping and real book store shopping, with emphasis on adding pre-process and post-process steps to improve the performance of OCR functions, and finally increase the rate of correct recognition. The GUI we developed is platform-invariant, robust, fast and accurate in providing good user interaction experience. Although the initial motivation for this project is to provide better experience for readers, but this system can also be used to detect other text based items such as CD covers and newspapers. With good flexibility, the system can be further extended to be functional for multiple languages and implemented on mobile devices. Although our text detection system can have a reasonably good accuracy, there are still a lot of ways to further improve this platform. The first version of the software is not functional on MATLAB versions earlier than 2016A, because there are several built-in functions that are not implemented in former MATLAB versions. In order to make the GUI text detector more robust on different MATLAB platforms, we developed our own MSER detection and bounding boxes merging algorithms. However, the new algorithm we developed is not as robust as the newest MATLAB built-in functions. Some future effort is to tune the algorithm so that the text detector is more user friendly across all MATLAB versions. Our presented approach brought up a number of novel ideas, including using hough rotation, positional bounding boxes filtering and auto word correction. We also compared the accuracy of matlab OCR function and open source OCR function, as shown previously, their performances are different across different datasets. Therefore another future angle is to carefully investigate the implementation of the two methods and try to take the advantages of all them. Also as shown above, the text detector will also misrecognize some non-text features as text, and it cannot be filtered by all methods discussed. Hence another possible option is to implement Machine Learning and Deep Learning to better characterize text region compared with non-text feature.

VI. Acknowledgment

We would like to thank Professor Gordon Wetzstein for teaching this great class and offering the opportunity of working on a project as a team. We also want to thank our tutor Jayant Thatte and all the TAs, who have always been helpful in both the project and homeworks. Last but not least, we thank all our classmates who we have sought help from.

References

[1] Smith, R. (2007). An overview of the Tesseract OCR engine.

[2] Patel, C., Patel, A., & Patel, D. (2012). Optical character recognition by open source OCR tool tesseract: A case study. International Journal of Computer Applications, 55(10).

[3] Ho, W. T., Lim, H. W., & Tay, Y. H. (2009, April). Two-stage license plate detection using gentle Adaboost and SIFT-SVM. In Intelligent Information and Database Systems, 2009. ACIIDS 2009. First Asian Conference on (pp. 109-114). IEEE.

[4] Chen, H., Tsai, S. S., Schroth, G., Chen, D. M., Grzeszczuk, R., & Girod, B. (2011, September). Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In 2011 18th IEEE International Conference on Image Processing (pp. 2609-2612). IEEE.

[5] Google open source Tesseract OCR https://code.google.com/archive/p/matlab-tesseract-ocr/downloads