Mobile Camera Based Text Detection and Translation

Derek Ma
Department of Electrical Engineering
Stanford University
Email: derekxm@stanford.edu

Qiuhau Lin
Department of Electrical Engineering
Stanford University
Email: qhlin@stanford.edu

Tong Zhang
Department of Mechanical Engineering
Stanford University
Email: tongzhang@stanford.edu

***Abstract*—Inspired by the well-know iPhone app "Word Lens", we developed an Android-platform based text translation application that is able to recognize the text captured by a mobile phone camera, translate the text, and display the translation result back onto the screen of the mobile phone. Our text extraction and recognition algorithm has a correct-recognition rate that is greater than 85% on character level. In this report, we demonstrate the system flow, the text detection algorithm and detailed experiment result.**

***Index Terms*—Text Translation, Android, OCR, google translate**

I. Introduction

The motivation of a real time text translation mobile application is to help tourists navigate in a foreign language environment. The application we developed enables the users to get text translate as ease as a button click. The camera captures the text and returns the translated result in real time.

The system we developed includes automatic text detection, OCR (optical character recognition), text correction, and text translation. Although the current version of our application is limited to translation from English to Chinese, it can be easily extended into a much wider range of language sets.

A. Prior and Related Work

1) Text Extraction: Text extraction techniques are widely studied because text embedded in images and videos provides important information. Many characteristics of text regions have been summarized and characterized effectively by several features, e.g. text pixels have near-homogenous color, character strokes form distinct texture, etc. Y. Hasan and J. Karam developed a text extraction algorithm that utilized morphological edge/gradient detection [1]; Algorithm by Epshtein et al tackled the problem from another approach using text stroke transform [9].

2) OpenCV: OpenCV stands for Open Source Computer Vision. It is a library of programming functions for real time computer vision. The library has more than 2000 optimized algorithms and has been widely used around the world. Befitting from this, android programmers are able to implement many digital image-processing algorithms in Android phone platform.

3) Optical Character Recognition: OCR, Optical Character Recognition, is developed to translate scanned images of handwritten, typewritten or printed text into machine-encoded text. A lot of OCR software have been developed to accomplish this mission. Tesseract, originally developed as proprietary software at Hwelett-Packard between 1985 and 1995, now sponsored by Google, is considered to be one of the most accurate open source OCR engine currently available. It is capable of recognizing text in variety of languages in a binary image format.

4) Text Correction: The text correction is a necessary step after OCR text recognition, since the result returned by the OCR engine is not be always correct due to image imperfections. This type of errors can be categorized into so called non-word error - which means that the text string returned by OCR does not correspond to any valid word in a given word set. Existing robust text correction algorithms have a good performance in correcting this type of non-word error. Such text correction systems include Aspell, and also the well-known Peter Norvig's text correction algorithm.

II. System Flow

In this paper, we propose a text detection / recognition / translation algorithm that consists of following steps:

1) Morphological edge detection
2) Text feature filtering
3) Text region binarization

4) Optical character recognition
5) Text correction
6) Text translation
7) Display of the translation

A. Step 1 - Morphological Edge Detection

To perform the edge detection algorithm, we first convert the input RGB color image to a gray-scale intensity image Y using (1), where R, G, and B represent red, green and blue components of the input image.

$$Y = 0.299R + 0.587G + 0.114B \quad (1)$$

The gray-scale image is then blurred using open-close and close-open filters to reduce false edge noise and over-segmentation. Structuring element used for this operation is a 3 by 3 8-connected element. Next, a morphological gradient detection operation is performed on the blurred image Y_{bl}, as shown in (2).

$$Y_2 = MG(Y_{bl}, B) = dilation(Y_{bl}) - erosion(Y_{bl}) \quad (2)$$

In order to get the threshold level of Y2, we use a global nonhistogram-based thresholding technique. The threshold level is determined by (3), where s is an edge detector obtained by applying central difference edge detection filter to Y_2. [1]

$$\gamma = \frac{\sum Y_2 \cdot s}{\sum s} \quad (3)$$

B. Step 2 - Text Feature Filtering

In order to reduce the number of connected components that have to be analyzed, a close operation with a 5 by 5 structuring element is performed to the binary edge image obtained from Step 1.

After the close operation, all connected components of the edge image are screened with their position, size, and area information. A candidate of letter should meet a set of constraints in size and shape. In our algorithm, we select connected components as letter candidates if the following requirements are met:

1) Width of the bounding box $<$ 0.5 image width
2) Height of the bounding box $<$ 0.3 image height
3) 0.1 $<$ center width of the bounding box $<$ 0.9
4) 0.3 $<$ center height of the bounding box $<$ 0.7
5) Width vs. height ratio $<$ 10
6) Width of the bounding box $>$ 10 pixels
7) 0.1 $<$ Connected component filled area over (width height of the bounding box) $<$ 0.95
8) Width of the bounding box $>$ 10 pixels
9) 0.1 $<$ Connected component filled area over (width height of the bounding box) $<$ 0.95

After the first round filtering, it is expected that most of the non-letter components would be removed. So the majority of the remaining candidates should be letters with the same font and size. Based on this condition, we calculate the mean height h_m of the bounding box of the remaining components, and remove any connected component with its height smaller than $0.6h_m$ or greater than $1.8h_m$.

C. Step 3 - Text Region Binarization

Each remaining bounding box is used as a mask to the original gray-scale image. Otsu's method [2] is used to obtain the threshold of the masked gray-scale image for binarization. Since each bounding box is relatively small compared to the size of the entire image, no further adaptive thresholding method is implemented. Theoretically after this step, only stroked letters are left as the foreground, 1, and the rest of the image would go to background, 0.

D. Step 4-6 - Text Recognition, Correction, and Translation

Since the project is focused on implementing text extraction on a mobile phone, we implemented the following three steps - text recognition, correction and translation on a server with open source software for simplicity's sake. Google's open source OCR - Tesseract [3] is used as the optical text recognition engine. Peter Norvig's algorithm [4] is added to the routine to perform text correction. Then Google translator [5] is used to translate the text into Chinese.

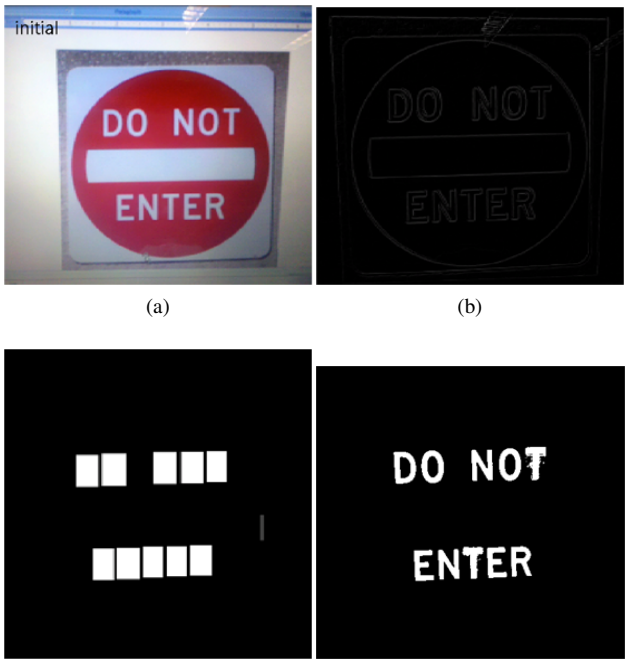
E. Step 7- Display of the Translation

The translated text string from Step 6 is sent back to the mobile device (Andriod phone) from the server, and then displayed at the top center region of the screen. A sample text extraction process flow is shown in Figure 3 below. The final result frame display after step Figure 4-7 is shown in Figure 2.

III. Test and Results

The performance of our system is evaluated by the rate of successful recognition. We decide to use recognition rate rather than successful translation rate as the criterion of performance, because the recognition rate more directly measures the successfulness of the text identification algorithm, whereas the measure of translation rate can be influenced by the Google translation engine, over which we have no control. The recognition rate is defined as,

- the ratio between the number of successfully recognized letters and the total number of letters in a test image.



(a) (b) (c) (d)

Fig. 1: : (a) Original captured image frame; (b) Edge detection (after Step 1); (c) Text region filtering (after Step 2); (d) Edge image binarization (after Step 3).

Fig. 2: Final result display

- the ratio between the number of successfully recognized words and the total number of words in a test image.

We conducted experiments to evaluate the performance of our system under different scenarios. The font size, font and means of display are varied in order to test their effect on the performance of the system. Two phrases, "Digital Image Processing" and "Visual Information Plays An Important Role" are used for the test. The two phrases contain 44 and 72 letters respectively. We counted the number of letters recognized from the output of the OCR engine and calculated the recognition rate.

A. Font

The test phrases were displayed in Arial, BlairMdITC TT, Times New Roman and Arial Bold, and their recognition rates were measured. We did not observe significant difference of the recognition rate among the four fonts.

B. Font Size

We used large, medium and small sizes of letters for testing. The font size of large letters was 40, the size of medium letters was between 20 and 25, and the size of small letters was between 10 and 15. We fixed the camera lens approximately 30cm away from the letters, so that small letters would appear small in the camera frame. We found significant effect of the font size on the recognition rate. As shown in the table above, large text achieve much higher recognition rate than medium and small size texts do.

TABLE I: The Recognition Rate for Text with Large, Medium and Small Font Sizes

Font Size	Large	Medium	Small
Recg. Rate (letter)	0.94	0.83	0.88
Recg. Rate (word)	0.83	0.71	0.70

C. Means of Display

The text was displayed on a computer screen and on a piece of paper. We tested the recognition performance with both display methods. As we had expected, the recognition rate of the text displayed on a computer screen was slightly higher than the result of the text printed on a piece of paper. This is because computer screen has higher contrast than paper.

TABLE II: The Recognition Rate for Text on a Computer Screen and on Paper

Display	Computer Screen	Paper
Recg. Rate (letter)	0.90	0.87
Recg. Rate (word)	0.82	0.74

We repeated the above experiment after text correction is applied. The text correction improves the performance by 5% if we measure how many words are successfully recognized. We summarized our experimental results in Figure 3.

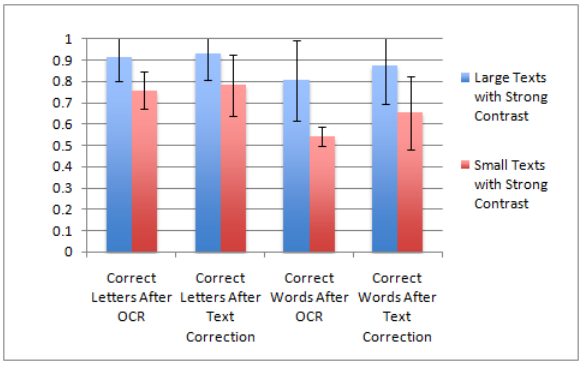


Fig. 3: Bar Plot of Test Results

D. Additional Tests

By placing an object in front of the camera and time how long it will appear in the view finder, we measured the system lag to be 2 second. The server lag largely depends on the strength of the wireless connection. With good wireless connection, the server delay is approximately 1 second.

IV. Conclusion

We have achieved an Android based application for real-time text extraction, recognition and translation. The average correct character-recognition rate is above 85%. From the performance evaluation of our system, we concluded that our application is very robust for large text such as road signs, etc. Following work needs to be done in order to drive our application into a commercial product:

- Further optimize text extraction algorithm to increase the processing speed to real-time (multi-thread processing structure in Android Java code);
- Adapt text extraction algorithm to smaller and denser text;
- Local Otsu's method to adapt non-uniform background of the scene;
- Localize the OCR, text correction and text translation algorithms onto the mobile device;
- More language translation selections for the user;

Acknowledgment

We would like to thank Prof. Bernd Girod for giving so wonderful lecture and teaching us many advanced digital image processing methods. We also would like to thank David Chen and Derek Pang, and our mentors Sam Tsai and Frank Chen for their patience and help.

References

[1] Yassin M.Y.Hasan and Lina J.Karam, *Morphological Text Extraction from Images*. IEEE Transaction on Image Processing Vol.9 No.11, Nov 2000
[2] Nobuyuki Otsu, *A threshold selection method from gray-level histograms*. IEEE Trans.Sys.,Man., Cyber 9(1):62-66
[3] http://code.google.com/p/tesseract-ocr/
[4] http://norvig.com/spell-correct.html
[5] http://austingulati.com/2009/07/google-translate-php-api/
[6] Farshad Ghazizadeh, *Optical Character Recognition*. US Patent: 5,007,809.
[7] Huiping Li, David Doermann and Omid Kia, *Automatic Text Detection and Tracking in Digital Video*. IEEE Transaction on Image Processing Vol. 9 No. 1, Jan 2000
[8] Celine Mancas-Thillou, Bernard Gosselin, *Color text extraction with selective metric based clustering*. Computer Vision and Image Understanding 2007
[9] B. Epshtein, *Detecting Text in Natural Scenes with Stroke Width Transform*. Image Rochester NY, pp. 1-8.

APPENDEX – Individual Work Breakdown

Derek Ma:
Implemented the text detection algorithm and the augmented reality of the UI on Android.

Qiuhua Lin:
Wrote the PHP code for the server and the http upload class in Android

Tong Zhang:
Developed the text detection algorithm in MATLAB, implemented spell correction algorithm in Python, and tested the performance of the system