# Mobile Chinese Translator

## Using Maximally Stable Extremal Regions (MSER)

Ray Chen
Department of Electrical Engineering
Stanford University
rchen2@stanford.edu

Sabrina Liao
Department of Electrical Engineering
Stanford University
liaos@stanford.edu

***Abstract*— In this paper, we describe a system for Android smartphones which detects and extracts Chinese text, and translates it into English. Most of the processing is done on a server; the user simply takes a picture with no other input. In this project we are mainly concerned about the text detection part, for which we use the MSER algorithm; the Optical Character Recognition (OCR) is based on Tesseract OCR, and translation is done through Google Translate.**

***Keywords: Mobile; Text detection; Chinese; MSER; OCR.***

### I. INTRODUCTION

With smartphones developing an ever increasing amount of computing power, more and more image processing algorithms can be used for mobile augmented reality applications. As such, we have seen a growing number of text detection and translation applications over the past few years, most notably Google Goggles and World Lens. With these applications, frequent travelers can worry about one less thing when abroad.

However, there are still limitations, especially with Chinese. Most of these applications available either do not support Chinese characters and/or require significant user input. This is because unlike Latin-based languages, Chinese characters pose a number of unique problems. For example, the classical Latin alphabet contains only 26 characters; even when considering lower and upper case versions in addition to punctuation, we end up with a much smaller set than the Chinese character set which consists of over 20,000 cases. Furthermore, Chinese characters can be composed of a disconnected set of other valid characters, unlike in English where there are no letters with disconnected strokes (with the exception of "i" and "j"). In our implementation, we consider all these cases as well as instances of vertically written text.

### II. SYSTEM OVERVIEW

Our initial goal was to have a standalone application running completely on the mobile device. However, we quickly found that Tesseract's OCR engine [1] for Chinese often took over 15 seconds for just OCR alone (with almost zero correct results on unfiltered images!). And even if we optimized the OCR, we still had to add our MSER text detection algorithm. So, we decided to offload the heavy work to a server; this solution provided a much better user experience. A flowchart of our system is shown in Figure 1.

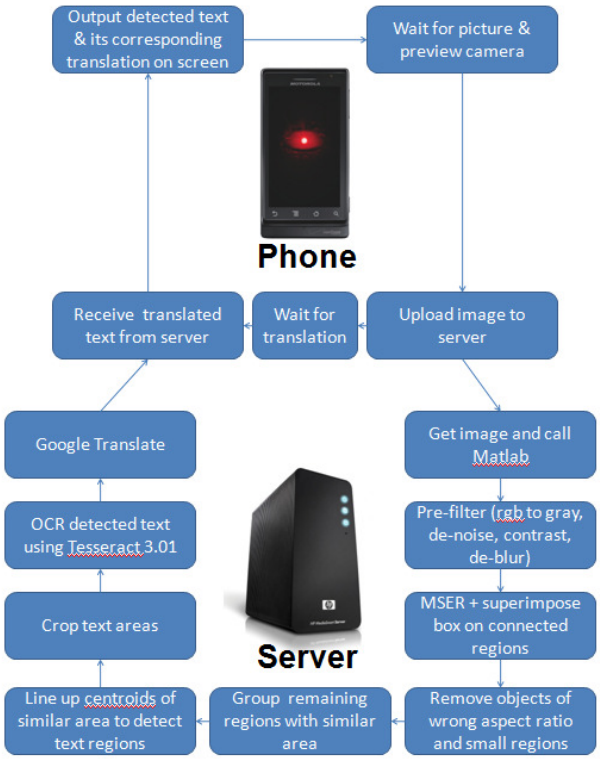



Figure 1. System flow

### III. TEXT DETECTION ALGORITHM

The text detection process is divided into 4 stages as depicted in Fig. 2. The pre-processing stage prepares the image for efficient MSER extraction in the region segmentation stage. The extracted MSER regions are then filtered for probable text features and the remaining features are overlaid with their respective bounding box to aid further removal of clutter and noise in the post-processing stage. Lastly, the text region extraction stage searches for horizontally and vertically aligned regions in the remaining connected regions to isolate areas where text is present.

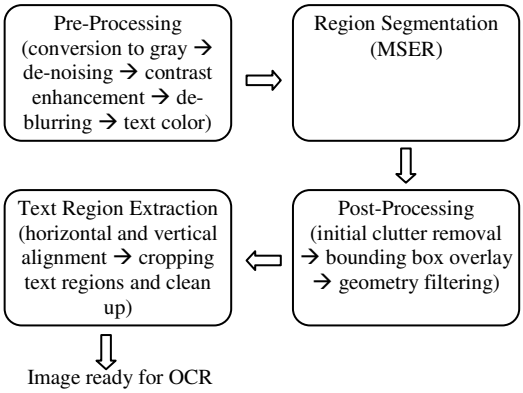



Figure 2. Text extraction flow chart

### A. *Pre-processing*

1) Conversion to gray scale: The transformation of color to gray scale is performed upon the reception of an image from the Android phone. This step significantly reduces the computational complexity as well as memory requirements for all subsequent processing steps. The conversion is done using MATLAB's rgb2gray function.
2) De-noising: Salt-and-pepper type noise is suppressed by a median filter that replaces the pixel with the median value of itself and its neighbors. A median filter is chosen here since it is edge-preserving and will not blur the image too much as a result of noise removal.
3) Contrast enhancement: More contrast in an image leads to improved MSER results; hence, contrast enhancement is done on the image to mitigate undesirable effects of poor lighting conditions and similar background and text color. A simplified enhancement algorithm is proposed in [2] and applied here:

$$G(x,y) = \frac{\left((S(x,y) - \min(S)) \times 255\right)}{\max(S) - \min(S)} \quad (1)$$

where S(x,y) is the original pixel value, min(S) and max(S) are the minimum and maximum of the original image, and G(x,y) is the contrast enhanced pixel value. This algorithm guarantees that the resulting image has a full range of 0 to 255.
4) De-blurring: Images captured by mobile devices are often blurry and while the MSER technique is known to be a good choice for text detection [3], it is also sensitive to image blur. Therefore, the last step of the pre-processing stage de-blurs the image using the "unsharp" filter provided by MATLAB.
5) Determining the color of text: finding the color of text (light text on dark background, or dark text on white background) proved to be an important input for the MSER algorithm. For this step, we use a rather simple but effective method - we use Otsu's method to find the threshold for the center region of the image (as we assume there is text there). We then count the number of pixels within the central region that are either above or below the threshold.

### B. *Region Segmentation*

1) MSER: As suggested by [3], the contrast of text to background is usually significant and a uniform color within every character can be assumed. With these assumptions, MSER is a good candidate method for separating out the text from the background. The vl_mser function from the VLFeat open source library is used to perform the operation. After MSER extraction, the image becomes binarized with connected regions that could be potential Chinese characters.

### C. *Post-processing*

1) Preliminary clutter removal: MSER regions that touch the boundary of the image are first removed. We can safely assume that any text a user is intending to translate is not cut-off on the image. In addition, borders or frames surrounding the text are likely to cause low solidity regions after MSER extraction; hence low solidity regions are removed. Typically at this point, most text detection algorithms will perform certain geometry filtering (aspect ratio, size, etc.); however, due to the nature of Chinese characters, we perform a "bounding box overlay" step – described below – before proceeding further with removing possibly non-text maximally stable regions.
2) Bounding box overlay: Chinese characters are very different from English alphabets. Most alphabets are represented by a single connected region; Chinese characters, on the other hand, may have one or more (possibly 8 or more) connected regions that need to be grouped together to form a character. This property makes the clutter removal process difficult.

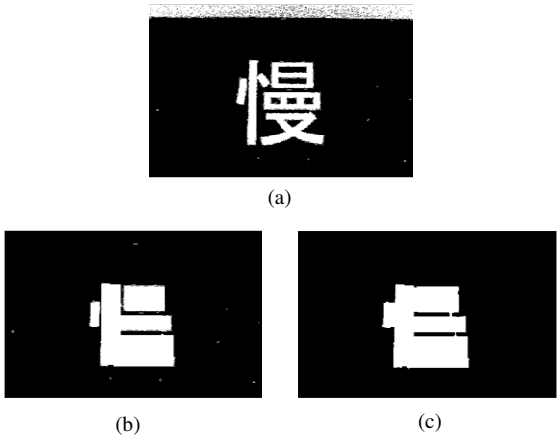



Fig. 3 a). MSER result b). Bounding box overlaid c). Small holes closed - 1 character now corresponds to 1 connected region

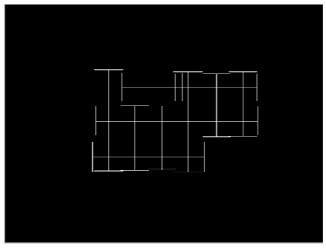
For example, the left-most dot in the character for "slow" (Fig. 3) could be accidentally removed because it is small and difficult to distinguish from general noise that might have been picked up by the MSER extraction. One feature of the Chinese character that can be leveraged, however, is that each character regardless of the number of components resides in a rectangular block. The idea behind bounding box overlay is to replace each connected region with its bounding box in hopes that the bounding boxes of each component will merge and mark out a more block-like region than the separated regions did. As figure 3 shows, after bounding box overlay, the originally separated regions have now formed a single square region that appears more like a text character than possible noise and clutter. In some cases, overlaying the bounding boxes still leaves some parts of a character disjoint. These holes are closed by slightly dilating and then eroding the binarized image. Now the image is ready for further clutter removal.

3) Geometry Filtering: With characters marked as rectangular connected regions, we can assume that text should have an aspect ratio close to 1 and therefore remove clutter that has the incorrect aspect ratio (for example > 3 and < 0.3). Lastly, out of the remaining features, the ones that have small area are removed (de-noising). Note that this last step cannot be performed without overlaying the bounding boxes on the connected regions first as parts of a character could be lost.

### D. *Text Region Extraction*

1) Horizontal and vertical alignment: It is possible that not all clutter and noise have been removed by the previous processing steps. Luckily though, a string of characters will most likely have good horizontal and/or vertical alignment (Chinese could be written horizontally or vertically). We use this heuristic to perform that last text extraction steps. The centroids of the remaining connected regions are calculated using MATLAB's regionprops function and an algorithm is written to loop through all the regions and group them together according to similar y coordinate (horizontal text) or x coordinate (vertical text). Some slant or rotation is allowed in the grouping. Area is also considered here. If the size ratio of the smallest and largest remaining connected regions is less than 2.5, then the horizontal and vertical alignment is done on all of the regions together; otherwise, the regions are first clustered according to size and then the alignment is done for each cluster. Any alignment found (vertical or horizontal) is indicated by drawing a line through the centroids of the connected regions segmented out, with the addition of a line at each end to indicate either the height (horizontal text) or width (vertical text) of the string of text (see Fig. 4 for demonstration).


(a)



(b)

Fig. 4 a). MSER result b). horizontal and vertical text aligned

2) Cropping text regions and final cleanup: Through the alignment step above, any text regions would have been marked out by a connected region of horizontal and vertical lines. In this very last step, we determine the boundary of each connected region resulting from the alignment algorithm and crop the MSER result before bounding box overlay according to each text region found. Each cropped image is then opened and average-filtered with a small structuring element in order to get rid of potential noise introduced by previous operations. Now the image is ready to be sent to the OCR engine for text recognition.

## IV. RESULTS & DISCUSSION

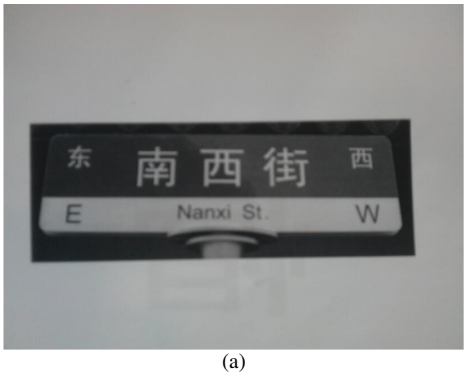
For testing, we printed out sample images of signs, some of which had cluttered backgrounds to simulate real-world pictures. We then took pictures of these signs using our phone camera. Printed out signs had a slightly better detection rate than a computer display due to auto-zoom issues and vertical lines caused by refresh rate mismatch.

To assess these results, we look at our output that we feed into Tesseract OCR; errors caused by the OCR engine and Google Translate are outside the scope of this project. We calculate our error rate as any character not detected or any noise blob not filtered out. Using this definition, our error rate was 17.86%. Please bear in mind that we are working with a limited set of data (a dozen images containing over 50 characters of different fonts, sizes, orientation, and backgrounds, with two test trials per image).

We can take a more detailed look by analyzing a few sample results shown in figures 5, 6, 7. From these, we can see that our method is fairly robust. In figure 5, we have a street sign with some clutter present, including valid Chinese characters that are not part of the main street name. We are able

to filter those out as they are not within the proper alignment. In figure 6, we test vertical alignment, and we see that it works fine (while not shown, the output of the OCR is combined into one phrase before translating, giving us a coherent sentence in this case). Finally, we can also see that off-center, slanted text with some noise (Fig. 7) also gets properly filtered out – in our current implementation, the bounding box of the aligned text is used and this causes the English letters to be cropped out as well; however, we can easily change the algorithm to use the convex hull of the aligned text to crop out only the Chinese characters.

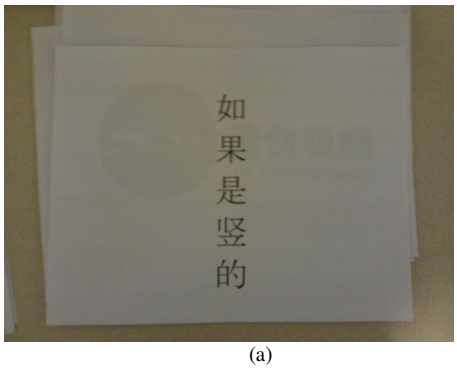
From our tests, one cycle of our application (from picture taken to translated text returned) takes about 6 seconds to run when connected to Wi-Fi.




(a)


(b)

Fig. 5 a). Input image, with some clutter b). Detected text output, automatically cropped.





(a) (b)

Fig. 6 a). Input image, with vertical text b). Detected text output, automatically cropped.


(a)


(b)

Fig. 7 a). Input image, with off-center slanted text and clutter b). Detected text output, automatically cropped.

## V. CONCLUSION

While our method performed fairly well, there can still be a lot of improvements. A few of the recommended potential next steps include:

- Speeding up MSER and optimizing OCR in order to run completely on a smartphone, possibly by taking a subset of the most frequent characters or combining the OCR process with our text detection & filtering.
- Adding capability for rotating slanted text. This should be fairly simple as we are already checking the alignment of text.
- Add an algorithm to clean up the translated English text, thus filtering out and/or correcting incoherent phrases or words.
- Integrating a dictionary into the mobile device so that an internet connection is not required.
- Expanding to other languages so that users could choose from a list, or even detecting the language.

## ACKNOWLEDGMENT

We would like to thank Professor Bernd Girod and the EE368 TAs (David Chen, Derek Pang) for educating us on image processing methods and for all their support over the course of this project.

## WORK BREAKDOWN

Ray - Android app, server setup, integration & testing.

Sabrina - Text detection algorithm in Matlab.

Both - Troubleshooting & tweaking the text detection algorithm, results analysis, poster, report.

## REFERENCES


[1] Tesseract OCR Engine. http://code.google.com/p/tesseract-ocr/f

[2] J. Yuan, Y. Zhang, K.K. Tan and T.H. Lee, “Text Extraction from Images Captured via Mobile and Digital Devices,” International Conference on Advanced Intellligent Mechatronics, Singapore, July 2009.

[3] Huizhong Chen, Sam Tsai, et al., "Robust Text Detection in Natural Images with Edge-enhanced Maximally Stable Extremal Regions", Proc. IEEE International Conference on Image Processing, ICIP-11, Brussels, Belgium, September 2011.