Instant Camera Translation and Voicing for Signs

Kaifeng Chen
Department of Applied Physics
Stanford University
Stanford, California 94305
Email:kfchen@stanford.edu

Yao Zhou
Department of Material Science Engineering
Stanford University
Stanford, California 94305
Email:yaozhou@stanford.edu

Shanshan Xu
Department of Physics
Stanford University
Stanford, California 94305
Email:xuss@stanford.edu

***Abstract*—Optical Character Recognition (OCR) is a class of problems that converts images or handwritten texts to machine-encoded text. In this project, we focus on one aspect of the general OCR problems and apply image processing techniques to sign image recognition for people who are not familiar with English. We incorporate four different kinds of algorithms that either handle the input image at text level or image level, and build a neat and useful Android application to facilitate instant English-to-Chinese translation and voicing. Such application has the advantage of in-phone computation, and thus saves the trouble of the communication with the servers. In addition, we perform detailed analysis on the algorithms and compare them under different cases. The statistical aspects concerning the accuracy, robustness and speed of each algorithm are presented.**

1. Introduction

In this project, we develop an Android application that can translate signs from English to Chinese or other foreign languages with autovoicing. In our application, the translation of the signs can be immediately performed by first placing the sign in the camera and then pressing the search button. This application can be used for sign recognition at airports or on road, which helps foreigners to guide and enjoy themselves when traveling in the states without the hassle of opening a digital dictionary and typing in the text, and thus saving the trouble of making mistakes such as input typos. These functionalities can also be generalized potentially to cover guide translation at tourist attractions, menu translation at a restaurant, and text translation when reading a book. In addition, we implement all the computation in the cell phone using different commercial packages and saves the trouble of server communication.

Our task is in the traditional domain of optical character recognition (OCR) which converts "images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast)". A survey of this area can be found in [1] and a very classical example is the handwritten digits recognition [2]. As a mature area, there has been a lot of available OCR software packages such as the Tesseract OCR engine [3]. On the other hand, compared to the general side of OCR, our task is in a relatively narrow subfield. Typical OCR processes the text either character by character or word by word. In our case, since the signs are pretty uniform and have limited kinds, we can process signs at the image level. Our task is more specifically related to the problem of traffic sign recognition [4] or the sign road recognition [1]. There are great interests and efforts in studying relevant algorithms and techniques for such problems due to the recent rise of self-driving car research.

The report is organized as follows. In Section 2 and Section 3, we briefly review the four algorithms used in our application and their implementations, respectively. Our results are summarized in Section 4. We test the performances of the four algorithm both in accuracy and speed. Finally, we point out the future work in Section 5.

2. Algorithm

We provide the following four algorithms for sign recognition: Google Mobile Services (GMS), Maximally stable extremal regions (MSER) and GMS, Sirovich and Kirby Algorithm, Features from Accelerated Segment Test (FAST) detector and Oriented FAST and Rotated Binary Robust Independent Elementary Features (BRIEF) (ORB) matching. The final translation is done by using the database that contains a hash map between English text and the corresponding Chinese translation.

2.1. GMS

GMS Text Recognition [5] is an algorithm designed for OCR detection problems. It is generally utilized to detect text in images or video streams and recognize the text contained in them. Once the text block is detected, the recognizer then determines the actual text in each block and segments it into lines and words, as shown in Fig. 1 [5]. The Text API detects text in Latin based languages (French, German, English, etc.), in real-time, on device.

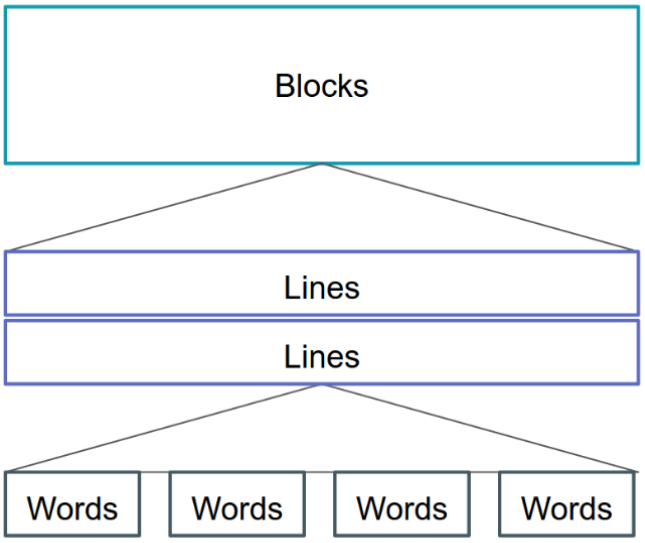

Figure 1. GMS: Text Structure

2.2. MSER and GMS

MSER and GMS Text Recognition is also an algorithm designed for OCR detection problems. MSER [6] are used as a method of blob detection in images. The key idea of MSER is to choose thresholds such that the resulting bright (or dark) extremal regions are nearly constant when these thresholds are perturbed by a small amount. In this algorithm, we first use MSER to detect text blocks and then use GMS to recognize the text contained in the blocks.

2.3. Sirovich and Kirby algorithm

Sirovich and Kirby algorithm is also an algorithm designed for image detection problems. The use of Sirovich and Kirby algorithm requires a database. The approach of using eigenfaces for recognition was developed by Sirovich and Kirby and was used in face classifications [7]. Here, we apply the Sirovich and Kirby algorithm to text image recognition. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of the database images. The eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the database images. Classification can be achieved by comparing images represented by the basis set. We use cosine similarity to match images.

2.4. FAST Feature Detector

FAST detector and ORB matching is also an algorithm designed for image detection problems. The use of FAST detector and ORB matching also requires a database. There

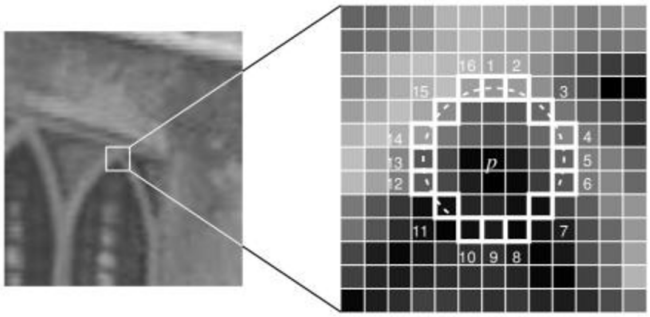

Figure 2. FAST detector

are several feature detectors, including SIFT, SURF. Although they perform very well in terms of feature detection, they are not fast enough in real-time application for our need. FAST was proposed as a solution to this in 2006 [8]: Compare "nucleus" p to circle of sixteen pixels, as shown in Fig. 2 [9]. Nucleus is feature point, if at least $n = 9$ contiguous circle pixels are either all brighter, or all darker, by t. Optimize pixel comparisons to reject non-corners early. We use OpenCV functionalities for FAST algorithm [10].

3. Implementation

Algorithm 1 which utilizes GMS directly is implemented in the following way shown in Fig. 3. In fact, to facilitate

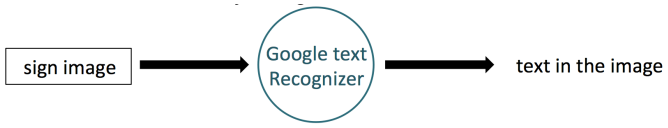

Figure 3. Pipeline of the implementation of algorithm 1: GMS.

the in-phone calculation, we resize the input image with less pixels before directly using GMS.

The algorithm 2 is implemented as shown in Fig. 4. Here again we resize the input image. The additional MSER

Figure 4. Pipeline of algorithm 2: MSER and GMS

feature detector can help text block detection, and thus filter out some of the undesired unrelated texts.

The algorithm 3 is implemented as shown in Fig. 5. Here, the Sirovich and Kirby algorithm of algorithm 3 is implemented in detail as the following:

1. Vectorize the database images $\Gamma_1, \Gamma_2, ...\Gamma_L$. Each column vector represents one image. (Here, $L = 13$)
2. Define the database image matrix $S = (\Gamma_1, \Gamma_2, ..., \Gamma_L)$.
3. Calculate the eigenvectors $\boldsymbol{v}_i$ of $S^H S$.

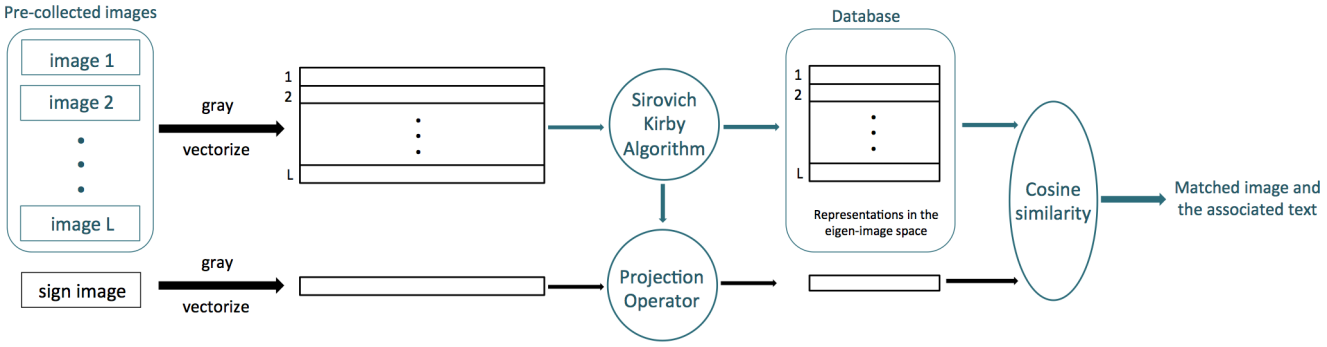

Figure 5. Pipeline of algorithm 3: Sirovich and Kirby Algorithm

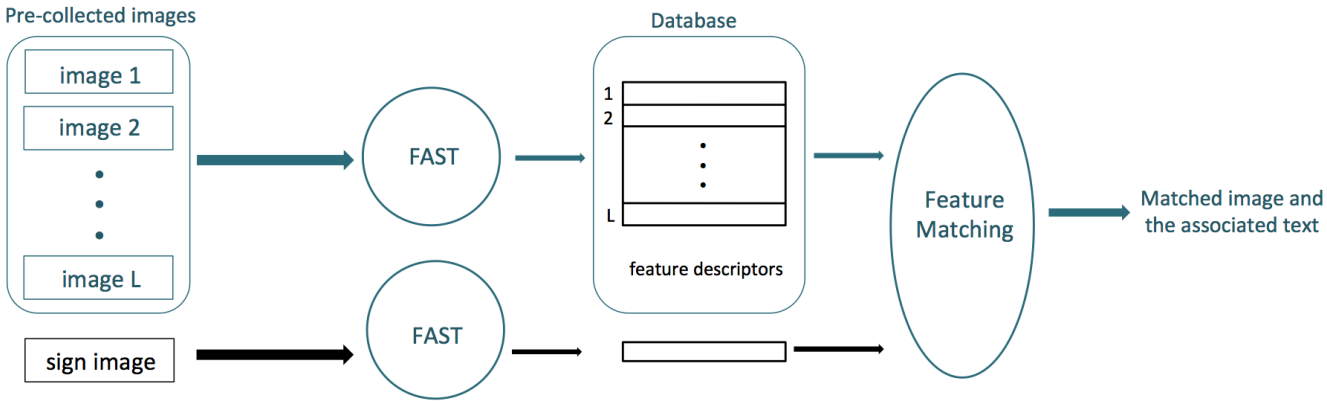

Figure 6. Pipeline of algorithm 4: FAST detector and ORB matching

4. Calculate eigenimages as $\boldsymbol{w}_i = S\boldsymbol{v}_i$. (normalized)
5. For an input camera frame T, compute the projection on the eigenvectors $\boldsymbol{u}_i = T\boldsymbol{v}_i$.
6. Compute the cosine similarity between $\boldsymbol{u}_i$ and all $\boldsymbol{w}_i$ and output the result associated with the largest similarity. The cosine similarity here is defined as

$$\cos\langle \boldsymbol{u}_i, \boldsymbol{v}_i \rangle = \frac{\boldsymbol{u}_i^T \boldsymbol{v}_i}{||\boldsymbol{u}_i||_2 ||\boldsymbol{v}_i||_2} \tag{1}$$

The algorithm 4 is implemented as shown in Fig. 6. The choice of using FAST feature detector is to improve the speed of feature extraction, and thus meets the needs of in-phone high-speed computation.

4. Results

In this section, we implemented the four algorithms introduced above using the procedures explained in the last section. The following paragraphs are organized as follows: in section 4.1, we discuss the data based we used for algorithms 3 and 4. In section 4.2, by using the application on different sizes of images, we compare the accuracy of the four algorithms. In section 4.3, we give an example showing the robustness of the application. In section 4.4, we evaluate the speed of the four algorithms.

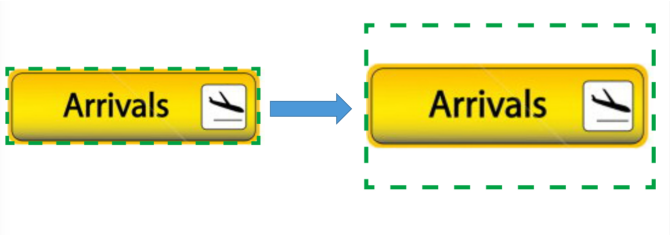

Figure 7. (left) the raw image downloaded from the web and (right) the images after adding white padding. The outlines for the images are labeled using green dashed lines. The final image has the ratio of 16 : 9, which is the same ratio as the cell phone screen.

4.1. Database

We use a preselected database for the implementation of PCA algorithm (algorithm 3) and FAST feature detector algorithm (algorithm 4). The original images are downloaded from the web. To ensure a fair comparison between the database images and the input frame from the Camera, we first process the raw images so that the ratio of the image is exactly the same as the camera screen. For Nexus 6P, its screen ratio is 16 : 9. Therefore, we pad the images with white pixels such that the database that is finally employed has the same width-height ratio. Fig. 7 is an example of how

to do the while pixel padding for the sign "arrival".

Here for simplicity, we pick 13 most representative sign images as the database. These signs include "arrival", "departure", "exit","entrance", "stop", "baggage", "check in", "information", "lost & found", "money exchange", "restroom", "security" and "custom". We apply the steps described above on the images and store them as the database. In initializing the application, the computations that are related to the database are then conducted and stored so that the eigenvalues and descriptors of the database images do not need to computed again. This significantly improves the speed of the application.

4.2. Accuracy

In this section, we focus on the discussion of the accuracy of these algorithms, and the working range of the algorithms. To conduct the comparison, we use the application to recognize the given image sign when the camera of the cell phone and the image are placed at three different distances. The closest distance, which corresponds to the case that the sign fills the whole camera screen, is denoted as $1\times$. Then as the camera is moving away from the sign, the sign becomes smaller in the camera frame and in those cases we denote it as $n\times$ where $n > 1$ and represents the approximated ratio between the camera frame and the captured image size. Here we only conducted experiments for three different sets of ratios $1\times$, $3\times$ and $5\times$.

The results for algorithm $1-4$ for the cases of $1\times$ and $3\times$ are shown as follows in Fig. 8. We can clearly see that, for the case of $1\times$, where the input images almost exactly matches the screen size, the algorithms 3 and 4 work the best. Where for algorithms 1 and 2, no text is recognized. This is because for PCA and FAST feature detector algorithm, the accuracy of them is significantly improved when the input image is close to the stored image in the database, whereas algorithms 1 and 2, which are based on Google Mobile Service, are not application to signs with large fonts that are comparable to the size of the screen.

On this other hand, when the images are far from the cell phone camera, for example $3\times$, the algorithms 1 and 2 outperform algorithms 3 and 4. As can be seen from the four figures, algorithms 1 and 2 give the right results, while the outputs from algorithms 3 and 4 are incorrect. This agrees with the observation in the case of $1\times$ since correct recognition from algorithms 3 and 4 really require an input image that resembles one of the database image.

In addition to the experiment for the case of $3\times$, we also compare the algorithms for $5\times$ and the results are summarized in Fig. 9. From the table, we conclude that algorithms 1 and 2 are more limited to the case when the sign has small font (or the sign image itself is small), while image can exactly fit the screen, the algorithms 3 and 4 performs much better. In summary, here the four algorithms actually cover most of the cases in real applications where the view angle of the user actually affects the output of the text recognition, and having four algorithms built in a single application avoid wrong detections in the two extreme cases and thus can be used in a broad range of scenarios.

Figure 8. (left) the result at $1\times$ and (right) the result at $3\times$. The ith row represents the result from the ith algorithm.

Factor	1x	3x	5x
Algo 1	✘	✔	✔
Algo 2	✘	✔	✔
Algo 3	✔	✘	✘
Algo 4	✔	✘	✘

Figure 9. The correctness of the four algorithms for the cases of $1\times$, $3\times$ and $5\times$, respectively

4.3. Robustness

In terms of robustness for the four algorithms, the algorithms 1, 2, and 4 are more robust to angle rotation due to the nature of the underlying algorithms. To get accurate result from algorithm 3, one needs to use signs with different styles as the database such that the eigenvectors of the database images are totally different. In the case of an input image that does not even exist in the database, algorithm 3 and 4 will give the wrong answer. To avoid that problem, one can set a threshold cosine similarity for algorithm 3 and minimum number of matched features for algorithm 4. Then

Figure 10. the outputs of algorithm 1 in different view angles.

when the image has cosine similarity or the matched features less than the threshold, the algorithms can output empty translation.

Fig. 10 is an example of algorithm 1 on different view angles. We can see that algorithm 1 is pretty robust against x axis rotation, but not stable in y axis rotation. However in general, the robustness improved by preprocessing the input frame using Hough transform.

4.4. Speed

speed	time (ms)
algorithm 1	355
algorithm 2	4935
algorithm 3	29
algorithm 4	1656

TABLE 1. SPEED OF EACH ALGORITHM.

In this section, we compare the speed of the four algorithms when each of them is operated at the best working range, as is shown in the last section. Here we pick "departure" sign as the example. The run time for each algorithm is averaged. The results are summarized in table. 1.

The above table is visualized in Fig. 11. We can clearly see that algorithm 3 perform the best in terms of speed because of the significant deduction of image dimension. As a contrast, algorithm 2 perform the slowest because of the MSER detector. It turns out that algorithms 1 and 4 actually have large variations of run time depending on the complexity of the input frame. This might come from from the fact that the computation of the text blocks (algorithm 1) or feature descriptors (algorithm 4) scales up when the input image becomes more complicated, i.e. has more details or more other texts.

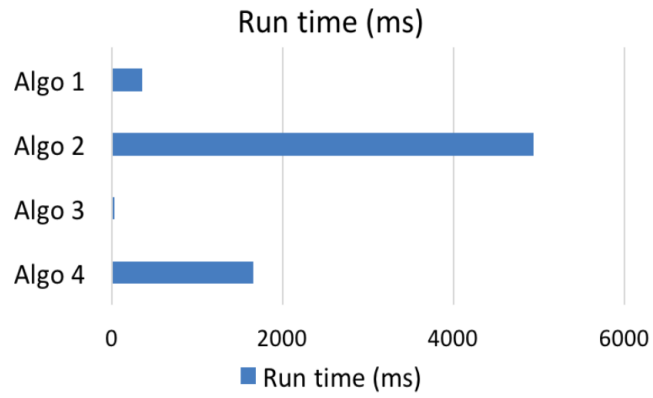

Figure 11. The run time (ms) for each algorithm at the best working range.

5. Future work

In additional to directly using Google mobile service for optical character recognition, we provide two ways to extract image features for image matching. One is the Sirovich and Kirby algorithm, where we decompose pre-collected images into eigen-images and represent each image as the projection coefficients in the eigen-image space. The other one is the FAST algorithm which outputs the feature descriptors directly.

A future extension is to apply the deep learning techniques such as the convolutional neural networks (CNN) [11] for image or text recognition. In our problem, since the number of distinct signs are limited and their fonts are

almost fixed, we can collect all the kinds of signs in advance. The image to be matched is then expected to be similar to one of the collected signs but may differ in size, rotation and homography. As a result, to train a neural network, we can generate massive artificial sign images by applying homography transformations to the collected sign images. A more aggressive way is to train a neural network to recognize the English characters in the sign images directly, which can recognize signs that are beyond the collected signs. For this purpose, a more advanced application with built-in deep learning algorithms needs to be implemented for mobile devices. Due to the high computation requirements of training a neural net, one can also build severs that interact with the mobile application. We believe that with these approaches, the results will be further improved.

Acknowledgments

The authors would like to thank the EE 368 TA Jean-Baptiste Boin and professor Gordon Wetzstein for the guidance of the project. The source code of the project can be found in this Github repository [12].

References

[1] G. Piccioli, E. De Micheli, P. Parodi, M. Campani, *Robust method for road sign detection and recognition*, Image and Vision Computing, 14 (3) 209-223, 1996.

[2] archive.ics.uci.edu/ml/datasets/Optical+Recognition+of+Handwritten+Digits

[3] R. Smith. *An overview of the Tesseract OCR engine*. 2007.

[4] A. de la Escalera, J.M Armingol, M. Mata, *Traffic sign recognition and analysis for intelligent vehicles*, Image and Vision Computing, 21 (3) 247-258, 2003.

[5] https://developers.google.com/vision/text-overview

[6] J. Matas, O. Chum, M. Urban, and T. Pajdla, *Robust wide baseline stereo from maximally stable extremal regions*, Proc. of British Machine Vision Conference, 384-396, 2002.

[7] L. Sirovich; M. Kirby, *Low-dimensional procedure for the characterization of human faces*, Journal of the Optical Society of America A, 4 (3) 519-524, 1987.

[8] E. Rosten and T. Drummond, *Machine learning for high speed corner detection* in 9th European Conference on Computer Vision, (1), 430-443, 2006.

[9] E. Rublee, V. Rabaud, K. Konolige, G. R. Bradski, *ORB: An efficient alternative to SIFT or SURF*, ICCV 2564-2571, 2011.

[10] OpenCV FAST tutorial

[11] Alex Krizhevsky and Sutskever, Ilya and Hinton, Geoffrey E, *ImageNet Classification with Deep Convolutional Neural Networks*, Advances in Neural Information Processing Systems 25, 1097-1105, 2012.

[12] https://github.com/kfrancischen/cameraTranslation