TEXT DETECTION AND MARKER BASED FINGER TRACKING IN BUILDING A LANGUAGE ASSISTANT FOR WEARABLE GLASSES

Ting Kwok Chan, Ying Kin Yu and Kin Hong Wong
Dept. of Computer Science and Engineering
The Chinese University of Hong Kong
Shatin, Hong Kong
khwong@cse.cuhk.edu.hk

ABSTRACT

Mobile phone with a touch screen display and high quality cameras is common the modern society. With a special piece of equipment, a smartphone can be turned into a pair of wearable glasses for augmented reality applications. The user can see virtual objects overlaid on the real scene where the user is looking at. In our project, we have developed an interface to let users to interact with the computer system with smart glasses. An application of our system is that the user can use his hand to point to a word that he does not understand. Then the system can look up the meaning and display the results to the user through the glasses. In the proposed system, we have applied a series of image processing techniques, which include text region detection, marker detection, fingertip detection and optical character recognition. Moreover, we have also used the ARuco 3-D marker to enable a better fingertip tracking performance. The work described in this paper is a part of a larger project that targets to develop a user friendly interface for wearable computing.

Keyword: Language Assistant, Mobile system, Wearable Computer, Computer Vision, Human Computer Interaction

1. INTRODUCTION

The invention of wearable smart glasses makes mobile internet computing easier. With such glasses, the user can see the world with virtual objects augmented into the real space. Usually, a camera is embedded into the glasses to obtain the images of the real world. With such a setup, many interesting applications can be implemented. In our project, we have developed a user interface for wearable glasses. One can use his hand to give commands to and interact with the system. More specifically, when the user is reading a book written in a foreign language, he can use his finger to point to a word that he cannot understand. Our system is able to locate, recognize the characters, and find out the meaning of the word using an electronic dictionary. The results can be displayed on the intelligent wearable glasses, such as the Google glasses. In this paper, we concentrate on the description of the finger tracking process and illustrate how the results can be used by an optical character recognition system. The results of

our project are useful for many real-life augmented reality and mobile computing applications [10].

The contents of the paper are arranged as follows. Section 2 discusses the background of the work. Section 3 describes the theory and methods used in our approach. Section 4 shows the experimental results of our system. Sections 5 and 6 are the discussions and conclusions of the proposed approach, respectively.

2. BACKGROUND

Smart wearable computers are becoming popular because it is easy to use and carry. Wearable cameras are mobile devices that can be taken outdoors for various usages. For example, Looxcie [1] is a general-purpose wearable camera that enables hand free video recording. It is mainly used for entertainment. For more serious applications, the Hong Kong Police Force introduced the body worn video camera for capturing evidence when a police constable is on-duty [2]. There are other wearable cameras that are designed for special purposes. The one by Recon Instrument [3] is used by athletes to record their activities during exercises. Some wearable cameras have features like Global Positioning System (GPS), High Density camera, Head-up-display (HUD) and smartphone connectivity. Some other products even have communication capabilities. For example, Golden-i [4] provides functions for users to send and receive email, and browse webpages. The Google Glass is designed to have a head mounted display together with a camera. It is versatile and can be used for various intelligent applications. Some researchers applied such devices for real-life applications [5]. For instance, a system in [6] translates the word pointed by the user's finger based on an optical character recognition approach known as Tesseract [7]. Our approach proposed here were built on top of [6] and the ArUco marker [8] were adopted to improve the fingertip detection performance.

3. DESIGN AND THEORY

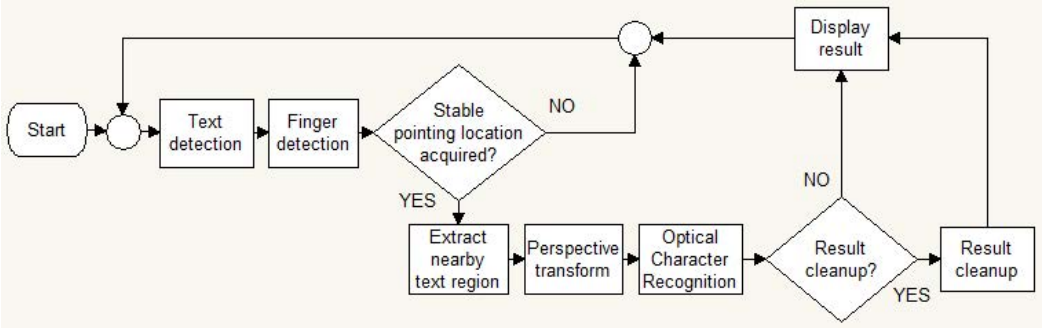

Figure 1. A flowchart showing the overall design of our system.

The proposed approach consists of three major components. One is the text detection module. It detects and returns the locations of the words in the image obtained by the camera. Our approach is similar to the one described in [6]. The second module is finger detection. It finds the presence of a finger and gives the image coordinates that the user is pointing at. Most of the existing techniques make use of the HSV color space to find the contour of a hand and apply convex hull to estimate the number fingers appeared in the image. In our approach, we investigate the use of a marker

based method known as ArUco marker [8] to locate the finger position. The result is more robust and accurate. The third component is the optical character recognition engine. It converts the selected word into a text string. Our project adopts Tesseract in [7] for such purpose. Apart from performing optical character recognition, it also provides the facilities for training the recognition model and the library for working with multiple languages. Our project only focuses on using the pre-trained English model. The OpenCV library [9] is adopted to carry out most the image processing procedures throughout the project.

Figure 1 gives an overview of the program flow. Firstly, the program finds the positions of all the words in the image captured by the camera. It then detects the location of the user's fingertip. Next, it calculates the stable coordinates of the fingertip. With the coordinates, the program extracts an image region in the area around the nearest detected word. Then, a perspective transform is performed on the image extracted, which is in turn fed into Tesseract for character recognition. Finally, a cleaning up procedure is performed on the results and the word selected by the user is returned as a string.

3.1 Text Detection and Extraction

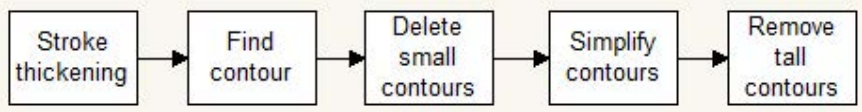

Figure 2. The procedure for text detection.

From the original image captured by the camera, our system detects text regions and represents them using a list of 4 corners. Figure 2 shows the procedure. Our method is similar to that in [6]. It uses a series of image processing functions such as Sobel, erosion and dilation to thicken the text strokes. In this way, nearby characters can be merged together and represented by a block of white pixels in a binary image as shown in Figure 3.b.

The contour detection function in OpenCV is then applied to group the pixels such that each word can be fit perfectly into the resulting contour. It is then tidied up by the polygon approximation function in OpenCV if it its height is smaller than 1.2 times of the width or the area is not larger than a threshold. The result is a list of rectangles represented by 4 coordinates and each one bounds a single word.

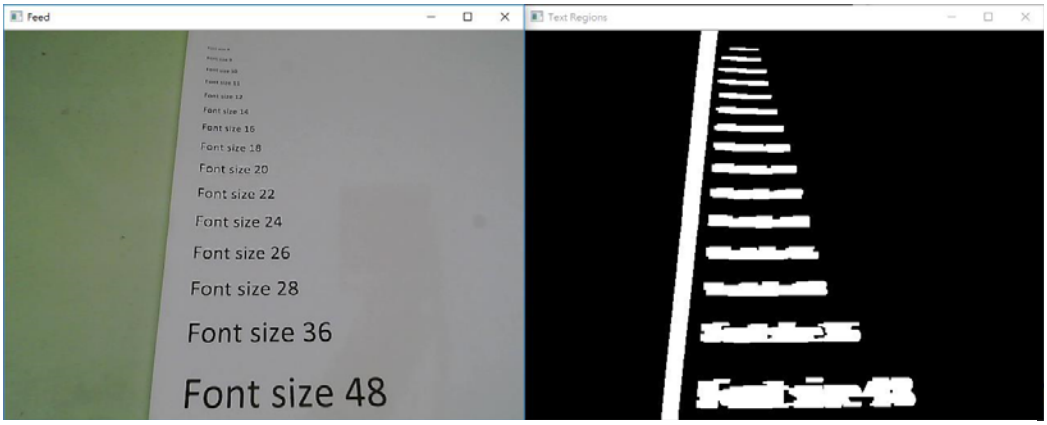

Figure 3.a. The original image from the camera. **Figure 3.b**. The potential text regions.

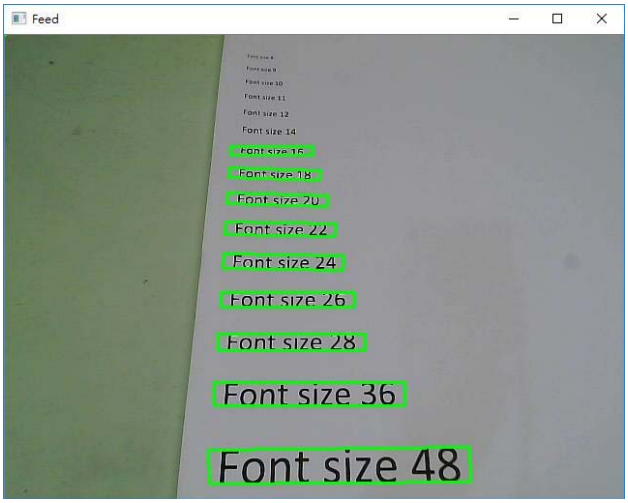

Figure 4. The detected text regions.

3.2 Finger Detection Using Colors

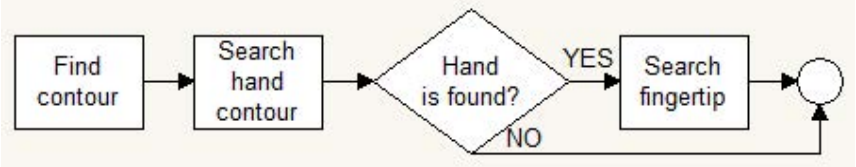

Figure 5. The procedure for finger detection based on colors.

To find out the fingertip by the color properties, the program first converts the image into the HSV color space. By filtering out the pixels that does not fall inside the HSV range specified, a binary image that shows the outline of a human hand is produced. Erosion and dilation are performed to remove noise from the binary image. The contour finding function is applied to find the shapes in the image. The program selects the contour with the biggest area. It then looks for the fingertip from the contour. The point with the smallest y-coordinate, which is the closest point to the top of the image, is regarded as the fingertip.

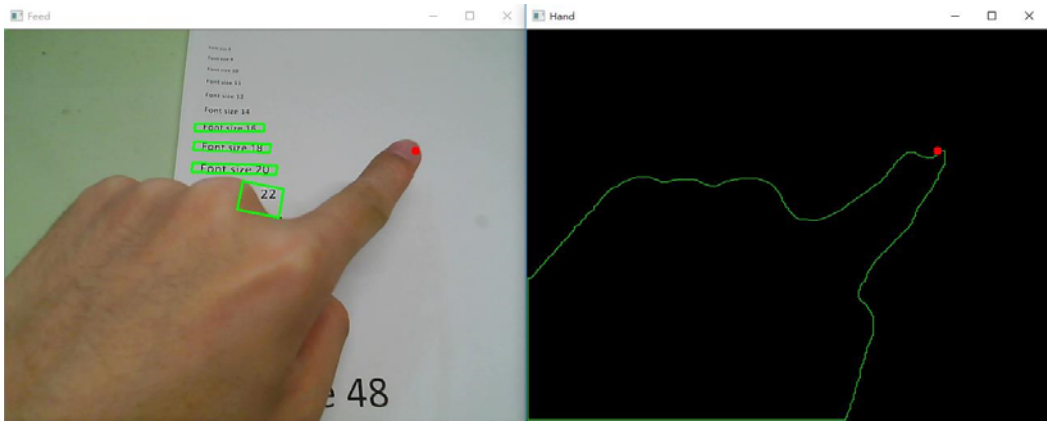

Figure 6.a. Some of the detected texts are occluded by the user's hand.

Figure 6.b. The contour of user's hand obtained.

3.3 Finger Detection Using ArUco Markers

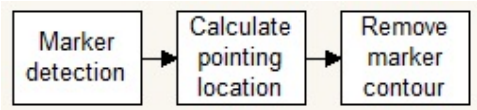

Figure 7. The steps for fingertip detection with ArUco Markers.

The fingertip detection algorithm described previously are not reliable in some occasions, another detection approach is adopted to tackle the problem. We make use of the ArUco marker in [8]. The user needs to wear a paper ring with the ArUco marker printed on it when using the application.

The marker in the image is detected and stored in a vector, which contains the coordinates of the 4 corners and its identification number. In order to achieve the highest detection speed, we use the simplest 4-by-4 ArUco marker set. The coordinates of the fingertip are obtained by extrapolating the line joining the center and the mid-point of the top edge of the marker.

In some circumstances, the markers may be wrongly detected as a text region in the image. To avoid the situation, bitwise operation is applied to the text and the marker region.

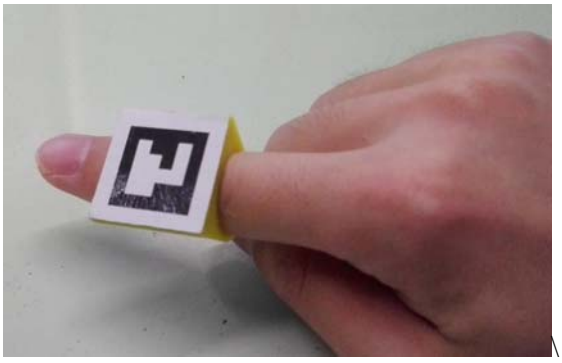

Figure 8. The paper ring with the ArUco marker printed on it.

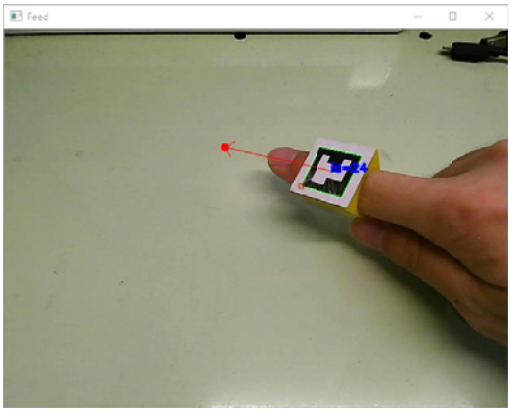

Figure 9. The ArUco cursor that we have implemented. The red point in the above image is the position where the user is targeted at.

3.4 Estimating the Stable Cursor Coordinates

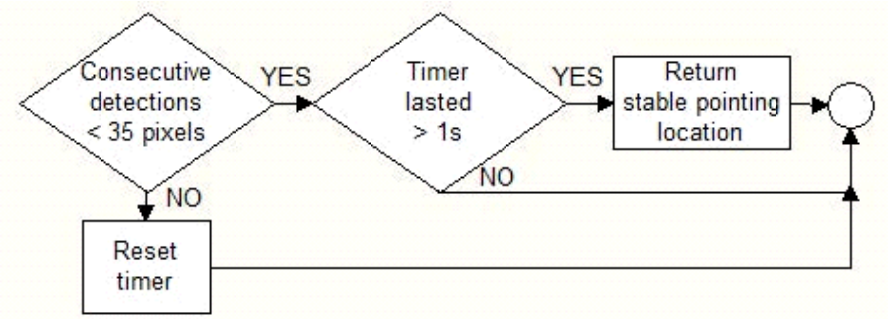

Figure 10. The procedure for finding the stable cursor coordinates.

The program is required to determine if the finger is moving. If so, the program will not extract the nearest text region for character recognition. If the distance between the previous and the current detected fingertip is less than a threshold within 1 second, the fingertip position retrieved is regarded as stable.

3.5 Finding the Nearest Detected Word Region

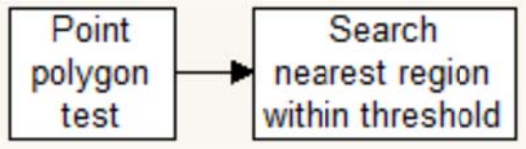

Figure 11. The steps to find the nearest detected word region.

We use the point polygon test function in OpenCV to obtain the text region that is closest to the stable pointing coordinates.

3.6 Character Recognition with Tesseract

Figure 12. The procedure for character recognition.

The perspective transform is applied to the selected text region to warp the image view such that the shapes of the characters are not distorted. A small boundary region with white pixels is added to the extracted image to optimize the recognition performance of Tesseract. Finally, the resulting image patches are fed to Tesseract for character recognition.

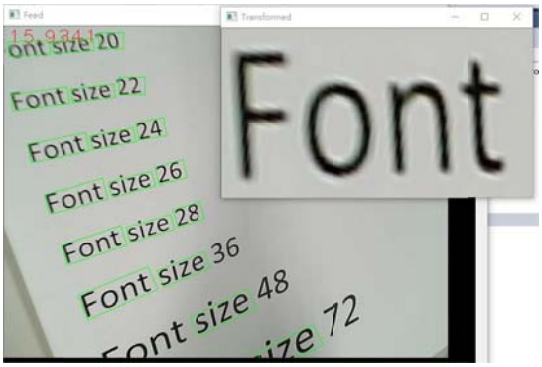

Figure 13. An example of perspective transformation. The word "Font" is transformed.

3.7 Tidying up the Word Extraction Results

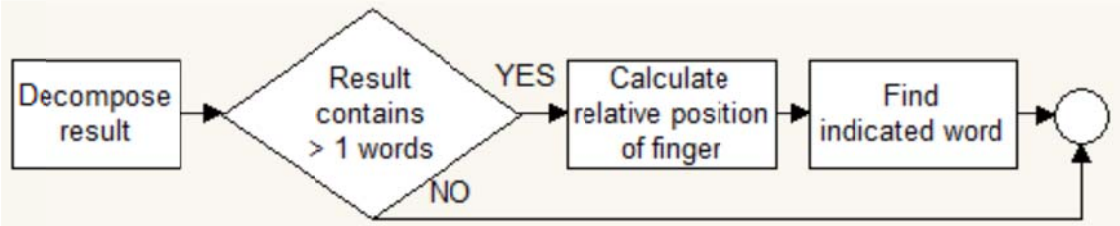

Figure 14. The procedure to tidy up word extraction results.

The text detection procedure described in the previous section may not be able to separate words from a sentence when the words are small or the words are too close together. Our program further processes the results so as to give a single word selected by the user. If the string returned by Tesseract contains one or more spaces, we regard the resulting string is a sentence. Then this string is decomposed into individual words according to the presence of the space characters. The relative position of the selected word in the extracted area is calculated in terms of the number of characters. The width of each character is estimated by dividing the length of the extracted area by the number of characters recognized. In this way, the actual word selected by the user can be determined.

4. EXPERIMENTAL SETUP AND RESULTS

To make the setup similar to the scenario of a user reading a book, a webcam was mounted on a vertical stand having a length of 20cm. The camera was pointed downward and positioned as shown in Figure 15.

In the following experiments, we are going to evaluate the design of each part as described in the previous Sections one by one.

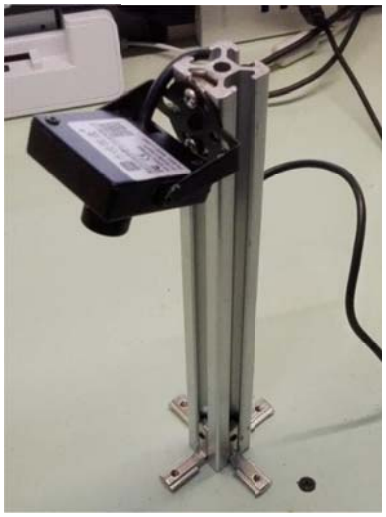

Figure 15. The setup of the experiment.

4.1 Text Detection and Extraction

We first evaluated the range of text height and width that our system can detect. To find out the valid text sizes, a page of text with different font sizes was printed and a picture of the page was taken. The distance between the camera and paper was 20cm. Figure 16 shows the results. A detected word was highlighted with a green box. It was found that the program could reliably find out words with font size larger than or equal to 14, i.e. 3mm high. It could be noticed that the program was unable to separate a sentence of words when font size was below 72.

Some languages contain words of fixed length, for example the Chinese language, while some have variable length like English. The second test was designed to find out the range of word lengths that the program was able to detect. Another page of text was printed and all characters on this page had font size of 14, which was the minimum valid font size of our system. It was found that the shortest word length was 1.1cm.

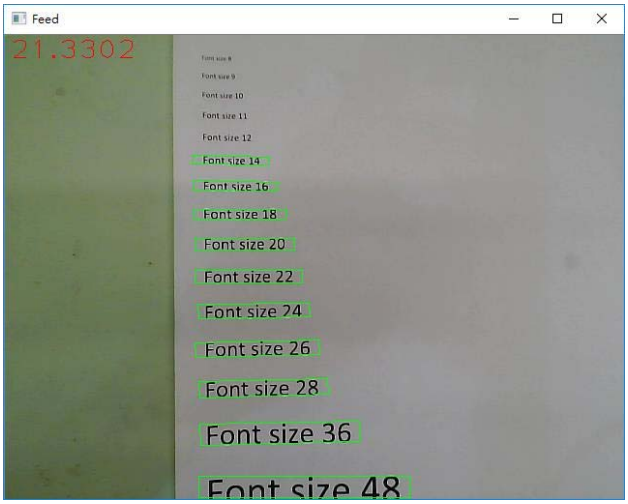

Figure 16. The results of detecting text with common font sizes.

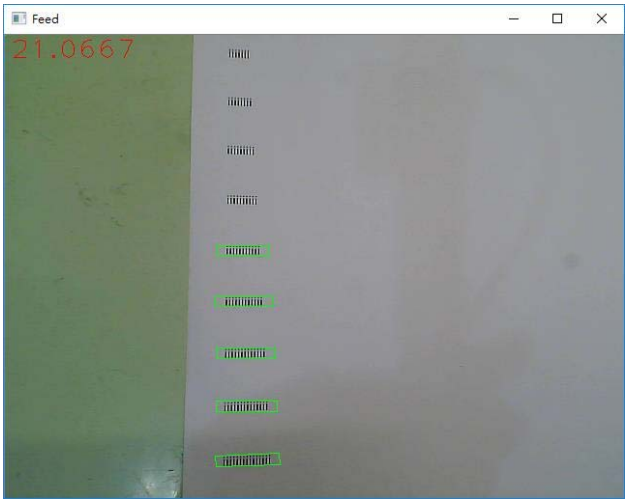

Figure 17. The results of detecting words of different lengths.

4.2 Finger Detection

Two different approaches were developed to locate the user finger position. The first approach is based on color and regard the highest point as the user's fingertip. It could detect the user's hand, given that the color filter was tuned appropriately. The result was satisfactory. The accuracy was within 30 pixels in Euclidean distance.

Another more reliable approach was based on the ArUco marker. This method achieved an accuracy of less than 10 pixels in Euclidean distance without prior calibration.

4.3 Optical Character Recognition

If the input image contained texts with perfect quality, the Tesseract OCR engine had a very high accuracy. The processing was in real-time. The algorithm could accurately determine a single word if the user was pointing at the center of the word. However, in case that the user was pointing at the first few characters near both ends, the program might mistakenly grasp the word before or after.

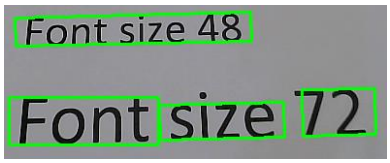

Figure 18. Tidying up the results was required for the string "Font size 48" in the above example as the original text detection algorithm failed to separate a single selected word from a sentence.

5. DISCUSSIONS

It was found that the program could not detect words that are too small and the program might not be able to separate the words from sentences. The cause of the problem was due to the use of erosion and dilation during image pre-processing.

The current system used a camera of fixed focus. The software had neither mechanism for detecting out of focus camera images nor ways of communicating with the camera to adjust focus, resulting in blurred images.

The use of ArUco marker was better than finger detection by colors. Although it requires the user to wear the marker while using the system, our program can exactly identify the user's finger and the corresponding positions.

The accuracy and speed of the Tesseract OCR engine were high in general. However, it is worth mentioning that it took noticeably longer time to process long sentences as well as blurred characters. This is because the engine is unable to separate words from sentences and the images captured are out of focus.

To let the proposed system to be installed on a mobile device, it should not be too demanding on the computational power and memory requirement. Our program was tested on a normal computer, which had a 2.4GHz Core 2 Duo processor, 8 GB RAM and no dedicated graphics card. It could achieve a frame rate ranged from 14fps to 17fps, which was quite good considering that there was no noticeable lag. The test results indicate that the program can probably run well on mobile devices with relatively low computation power such as smartphones and the Raspberry Pi board. Two demo videos can be viewed at https://youtu.be/I3bowdnqcHM and https://www.youtube.com/watch?v=j7t7a8maBv0.

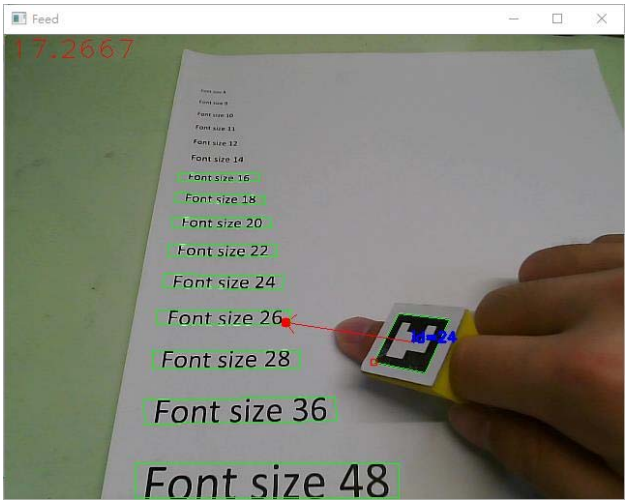

Figure 19. A screen capture of our working system. The frame rate of our program is shown in the top left corner.

6. CONCLUSION

An application for smart wearable glasses was developed. It can recognize a word pointed by the user on a page of text document. In the project, the user wears a paper ring with a marker printed on it. Our system finds out the position of the ring from the camera images. From the marker position, the word that the user is pointing at can be determined. The image region containing the word is then passed to an optical character recognition engine to transform the image pattern into a character string. Considering the speed performance of the current implementation, the proposed system can be run in an embedded system of a wearable device. Our solution is useful for a great variety of mobile applications, for example an interactive electronic dictionary.

7. Acknowledgement

This work is supported by a direct grant (Project Code: 4055045) from the Faculty of Engineering of the Chinese University of Hong Kong.

8. REFERENCES

[1] Boland, Justin, and Romulus Pereira, “Wireless headset camera lens”, U.S. Patent No. D643,867. 23 Aug. 2011.

[2] Hong Kong Police Force, Body Worn Video Camera Field Trial, http://www.police.gov.hk/ppp_en/11_useful_info/bwvc.html, accessed on 16 Nov 2017.

[3] Recon Instrument, https://www.reconinstruments.com/, accessed on 15 Nov. 2017.

[4] Golden-I,http://www.kopin.com/offerings/headset-solutions/default.aspx accessed on 15 Nov. 2017.

[5] McNaney, Rísin, et al., "Exploring the acceptability of google glass as an everyday assistive device for people with parkinson's" In *Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems*, ACM, 2014.

[6] Shi Fan Zhang, Kin Hong Wong, "A language assistant system for smart glasses", in *Proceedings of the 3rd International Conference on Next Generation Computing*, Kaohsiung, Taiwan, December 21~24, 2017.

[7] Smith, Ray. "An overview of the Tesseract OCR engine" In *Proceedings of the IEEE Ninth International Conference on Document Analysis and Recognition 2007*, vol. 2, 2007.

[8] Garrido-Jurado, Sergio, et al., "Automatic generation and detection of highly reliable fiducial markers under occlusion", *Pattern Recognition*, vol. 47, no. 6, pp. 2280-2292, 2014.

[9] Bradski, Gary and Adrian Kaehler, "Learning OpenCV: Computer vision with the OpenCV library", O'Reilly Media, Inc., 2008.

[10] Kai Ki Lee, Michael Ming Yuen Chang, Kin Hong Wong and Ying Kin Yu, "A hand-held augmented reality projection system using trifocal tensors and kalman filter", in *Proceedings of the IEEE 7th International Conference on Intelligent Computer Communication and Processing 2011* (ICCP 2011), pp. 253-260, Romania, August 25 - 27, 2011.