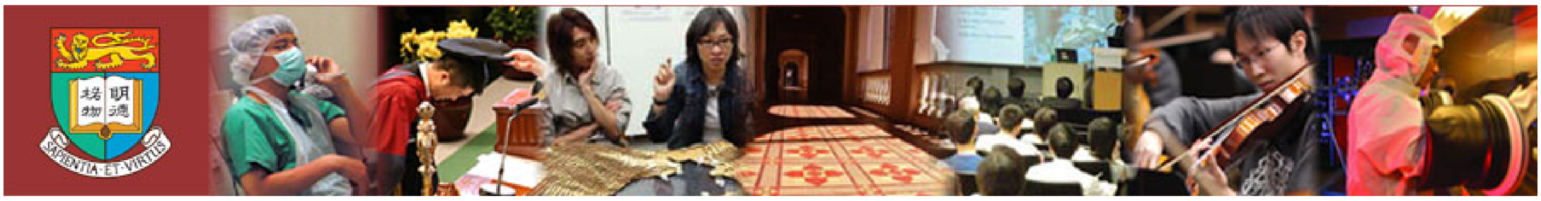

Building Smart Cameras on Mobile Tablets for Hand Gesture Recognition

Xiang Meng, Chung-Ming Cheung, Ka-Lok Ho, King-Shan Lui, Edmund Y. Lam and Vincent Tam
Department of Electrical and Electronic Engineering
The University of Hong Kong, Hong Kong

***Abstract*—Mobile tablets have become very popular due to their portability and the wide diversity of the applications available. The touch screens and built-in accelerometers facilitate different forms of input instead of keyboards and mice. Nowadays, high-resolution cameras become a standard feature in a mobile device. Nevertheless, this camera is seldom considered to serve as a form of user inputs to the application, although similar technology is realized in some home entertainment systems. This paper describes our experience on making a smart camera on an iPad that can recognize pre-defined hand gestures. We study the time performance of performing image processing on an iPad. We find that due to the limited computational power of the mobile device, recognition results may not be available fast enough for a real-time application. We explore applying cloud computing to solve the problem. To the best of our knowledge, this is the first study on recognizing hand gestures on an iPad. Our results facilitate the development of a brand new type of applications that require smart cameras.**

I. Introduction

In recent few years, mobile tablets has become more and more popular. With the help of the intelligent operation systems (iOS [1] & Android [2]) and millions of applications developed based on these operation systems, the mobile tablets can almost fulfill all the common functions of PC which people use on a daily basis. Being supported by the widely deployed wireless network and mobile network, mobile tablets gradually becomes the most popular terminal. With mobile tablets being known and used by more and more people, question raised: how should human interact with the mobile tablets?

Being a mobile device, mobile tablets share some common features:

- They have very few ports to external devices (only one for iPad);
- Multi-touch screen, accelerometer and camera are installed in most mobile tablets.

The first feature makes traditional input methods, such as keyboards and mice, not suitable for mobile tablets. The majority of instructions for mobile tablets are coming from the multi-touch screen. The accelerometer is also used and studied for some special operations [3]. Although being a default device equipped in mobile tablets, camera is seldom used as an input medium of instructions.

It is not a brand new idea for using the camera as the input device for instructions. Real product such as Kinect from Microsoft [4] has already been released to the market. In general, the camera control system works as follows:

1) The camera captures the users' gestures and movements;
2) The camera control system can interpret and translate those gestures and movements into some well formatted information;
3) Developers will utilize that information to implement corresponding functions.

Studies about how to implement and improve the camera control system have been conducted. In Ref. [5], the authors proposed a new algorithm to realize multi-touch on surface by utilizing normal cameras placed overhead. The new algorithm relies on a machine learning method and a geometric finger model to achieve a high precision of a few millimeters. [6] is a work focusing on capturing images of the object of interest. The authors developed a system that can track the position of an object using different pictures provided by two or more cameras.

The camera control system has many advantages. It does not require physical contact and can serve as a mean of remote control. Many industries have already applied the camera control system in practice. Thus, it is fair to say camera control is promising. However, current applications with a camera control system are all computer based. As far as we know, applications and implementations of camera control are rare on mobile tablets.

In this work, we study hand gesture recognition on iPads. We extract the image of the hand from the picture captured using the iPad and recognize the gesture using a novel recognition algorithm. Although the recognition is feasible to be carried out on an iPad, due to the limited computational capacity, the response time is slow. This is not favorable to a real-time application that requires instant response. We thus explore applying cloud computing to reduce the processing time needed.

The rest of this paper is organized as follows: we will present our methodology Section II. Comprehensive simulation results will be presented in Section III. In Section IV, we will discuss simulation results and future directions.

II. Methodology

In this section, we describe our gesture recognition and cloud processing in details. The recognition consists of two steps: retrieve binary image of the hand from the picture captured by the camera and recognize the hand gesture.

A. *Retrieving the Binary Image of the Hand*

We identify the hand by detecting pixels of skin color on the picture. We detect skin color through building a global skin detector first. We only consider the hue value of the HSV color space [7] when segmenting skin color, because this can minimize the effects of illumination in affecting the accuracy of our algorithm. In the global skin detector, we used several sample images of hands to set a loose threshold for the hue value of skin color.

Then, we need to build a more accurate adapted skin color model for skin detection according to the surroundings' conditions. To do so, pixels belonging to the foreground (anything not belonging to the background) need to be extracted. We have considered two different approaches to segment the foreground.

1) Static Image Approach: In this approach, an image of solely the background is required. This image is acquired by considering the average of 3 photos of the background taken to reduce noise. Afterwards, 3 photos with the presence of skin are taken to acquire some samples of the skin color. By subtracting the background from the sample photo and taking the pixels with a difference larger than a certain threshold, the foreground can be obtained.

The advantages of this approach are coding is simple and results are accurate. However, background images have to be retaken whenever the environment changes.

2) Video Approach: A mixture of Gaussians (MoG) model [8] is used to model the background continuously (Fig. 1). It segments the foreground and background by considering objects in motion to be in the foreground and static ones to be in the background.

When compared with the static image approach, this method can better adapt to changes in background. Nevertheless, as more data have to be processed, more time is needed to process the data or the resolution of the images has to be tuned down.

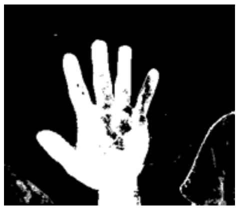

Fig. 1. Foreground extracted by MoG model

Once the foreground is obtained, the following steps are performed to determine new thresholds for the hue values (Fig. 2).

1) Filter obvious non-skin pixels in the foreground (e.g. shirts) by the global skin detector;
2) Plot a histogram of the hue values of the pixels in filtered foreground;
3) This histogram (H_2) is added to the histogram used in the previous frame (H_1) to produce the histogram (H_{new}) to be used for adaptively filtering skin color pixels in the current frame, according to the following equation.
$$H_{new} = H_1 \times (1 - A) + H_2 \times A$$
where A is the weight of H_2;
4) Take the lower and upper thresholds such that it only includes a certain percentage of the histogram with the smallest range as possible.

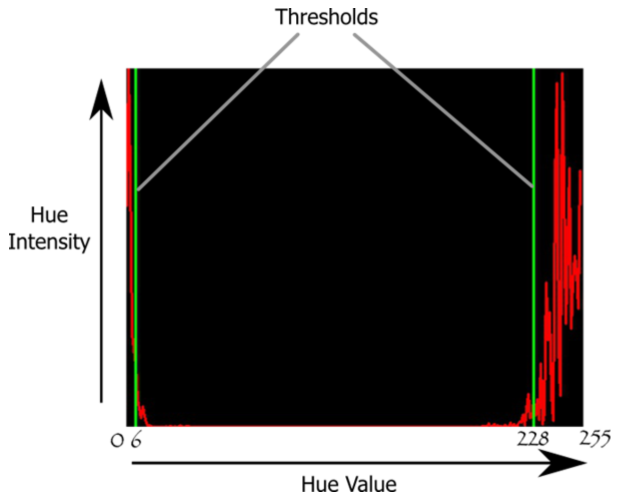

Fig. 2. Thresholds for the hue values

In both approaches, hue value thresholds for detecting skin color are adaptively determined. If the static image approach is used, the background can be subtracted from the image using a low threshold value to give more accurate results.

We mainly use the static image approach in this application although it needs more time for taking background images before processing starts, because it can guarantee more accurate results. The process is illustrated in Fig. 3.

B. *Hand Gesture Recognition*

After extracting the hand from the background, we determine the gesture. We count the number of fingers holding up and use the angle formed between fingers to differentiate the different gestures.

1) Acquiring the contour of the hand: We make the assumption that it is highly likely the hand is the main feature in the binary image. Therefore, after finding all contours in the image with the *findContour* function provided by the OpenCV library, it is assumed that the contour with the largest area is the hand.

However, we have discovered that the head often appears in the image as well. It is possible that it creates a contour larger than the hand and thus the algorithm will consider the wrong contour for hand gesture recognition. To solve this problem, face detection is used to find faces and regions with faces are ignored during the find contour step.

2) Finding the fingertips in the contour: To find the fingertips, well-defined characteristics that can be used to identify

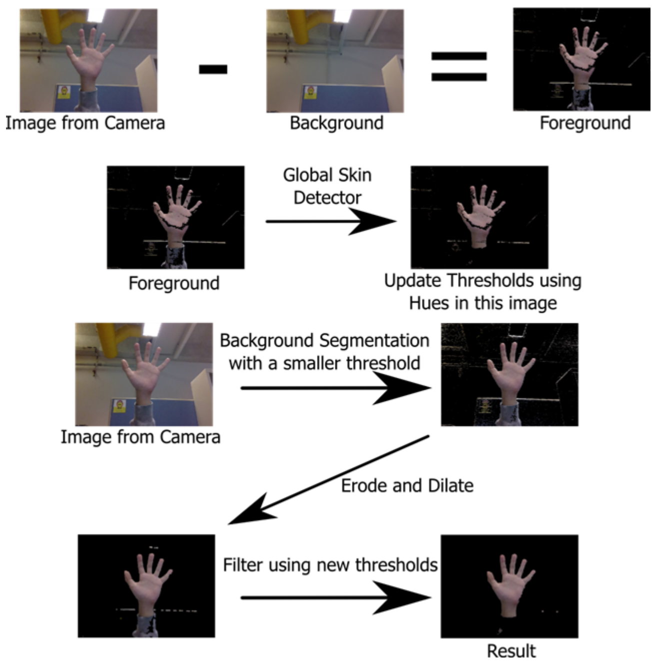
Image from Camera
Background
Foreground
Foreground
Global Skin Detector
Update Thresholds using Hues in this image
Image from Camera
Background Segmentation with a smaller threshold
Erode and Dilate
Filter using new thresholds
Result

Fig. 3. Summary of static image approach

fingertips are needed. In our algorithm, a point on the contour is identified as a fingertip if it satisfies the following three conditions:

- The angle that this point forms between the lines formed by joining this point with the points a certain distance in front of it and behind it on the contour must be smaller than 60°;
- This angle must also be smaller than the angles calculated from its neighbor points, that is, it should be a local minimum;
- The cross products of the two lines formed from this point must have a positive z-direction vector to ensure the point is a fingertip, not the valley-like shape between two fingers (marked by purple circles in Fig.4)

3) Finding the center of the palm of the hand: We observe that in the contour of a hand, the fingers and the wrist are narrower than the palm. Therefore, the largest circle that can be fit into the inside of the contour represents the location of the palm. Then, the center of this circle will be a good estimation of the center of the palm.

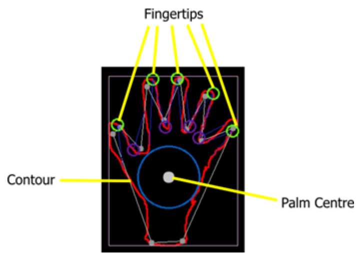

Fig. 4. Acquiring data from image

4) Identifying hand gestures: Gestures with different number of fingers raised can simply be distinguished by counting the number of fingertips detected in the hand contour.

For gestures with the same number of fingers raised, further information needs to be calculated from the contour to distinguish them.

The method used in our algorithm is to find the angle difference between raised fingers and compare it with results from experiment to determine which gesture it is.

First, angles from each fingertip to the palm's center are calculated, then the angle differences between these angles are found. This additional information will be used to identify hand gestures.

For example, in our application, there are two hand gestures with two fingers raised(Fig. 5). The angle difference between

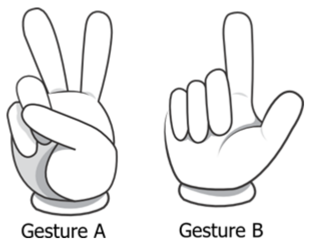

Fig. 5. Acquiring data from image

the two fingers are calculated and it is defined to be gesture A if the difference is smaller than $43°$, or gesture B if the difference is larger than $43°$ (Fig. 6).

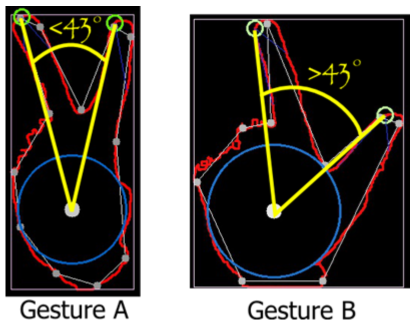

Fig. 6. Difference between the fingers' angles of Gesture A & B

We implemented the recognition program on an iPad and most gestures can be recognized successfully. Nevertheless, we found that after a gesture was captured by the camera, it took a long time for the recognition result to be produced. The response time of the program is not desirable for a real-time application. We thus explore using cloud computing to reduce the processing time.

C. Applying Cloud Computing

Due to the limited processing capability of iPad, we found it is quite time-consuming to perform the program solely on iPad. To enhance the user experience, we borrow the idea of mobile cloud computing to speed up the process [9]. Part of the computation is still conducted on the iPad, while some data are sent to a server through the network to obtain result.

We divide the program into three parts:

1) Initialization and Presentation
2) Transmission
3) Processing

The iPad will still be in charge of initializing the request and collecting necessary data. The processing job will be sent to a more powerful server and wireless connection is used as the media of communication between iPad and the server. With this approach, we can save a large amount of processing time, which greatly enhances the responding speed and user experience.

First, we combined all pictures into one single JPEG file. The JPEG file is converted to NSData on iPad. It is then sent to the server through PHP. The data is uploaded to a temporary folder in the server, and is converted back to a JPEG file for further process. After the processing has finished, the server returns a string containing the results and the iPad terminal will interpret and present the result. The logic flow is presented in Fig. 7.

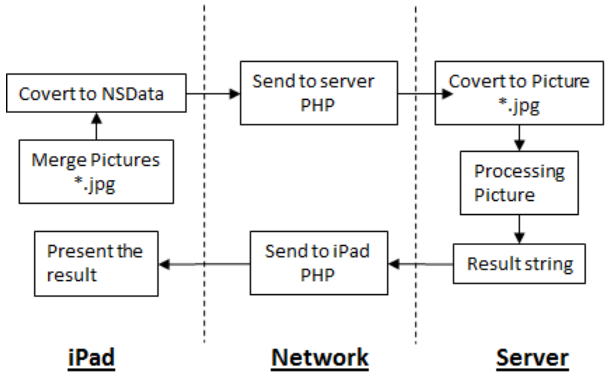

Fig. 7. Logic flow of the server model

III. Results

In this section, we present our experiments on the response time performance of our hand gesture recognition program.

We conducted our experiments with iOS and Windows as the platform for mobile device and the cloud server, respectively. The specifications of the devices we used are shown in Table I.

	CPU	RAM
Server	Intel Core i5 3.30 GHz	4 GB
iPad	Apple A5 1 GHz	512 MB

TABLE I
Device Specifications

We measure the response times of using iPad only and using the iPad with cloud server to process different numbers of gestures. As stated before, we use static image approach to retrieve the binary image of the hand in the experiments. The result is presented in Fig. 8. Every point in the figure is the average of 50 trials.

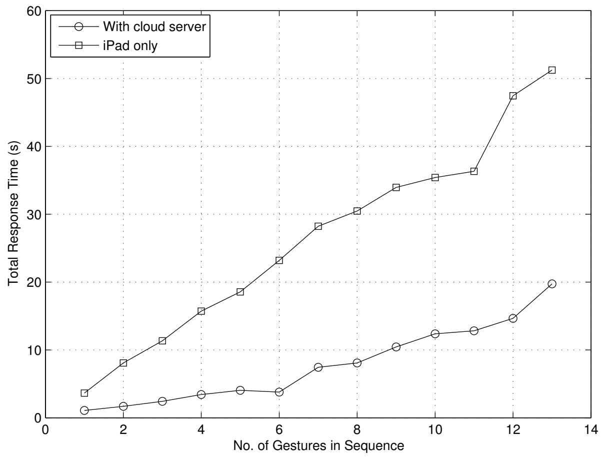

Fig. 8. Response time of the iPad-only processing and server-assisted processing

From the result, we can see that by applying the cloud computing concept and using a server to do the major processing, we can save approximate $\frac{3}{5}$ of the processing time. The more complex the gesture sequence is, the more significant the improvement made by applying cloud computing.

The total response time of the cloud server actually consists of three portion: iPad processing time, network delay for sending/receiving the data, and the server processing time. Fig. 9 shows the time needed for each portion.

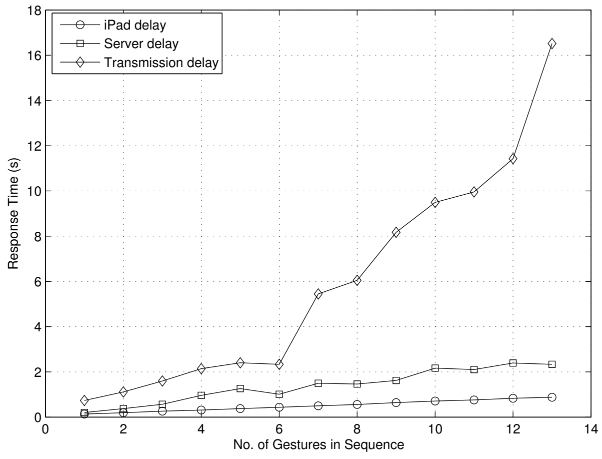

Fig. 9. Breakdown of the response time of the server-assisted processing

Delay on iPad is due to the pre-processing of the collected pictures, which is pixel-wise combining all the jpeg files into a single file for transmission. Delay on server is the real processing time for the gesture recognition. Network transmission delay is the time consumed for the transmission. From Fig. 9, we can see that by applying cloud computing, we actually reduce the processing job of iPad significantly. The complex processing is carried by the much more powerful server and the processing time is also quite small. It can also be observed that with more gestures to recognized at a time, the processing times for iPad and server only experience a small increase while the major delay is caused by network transmission.

IV. Discussion and Conclusion

Even though our camera control algorithm achieves the fundamental function, there are a still a lot of improvements we can consider in the future.

A. Network Transmission

Fig. 9 shows that network delay is the major source of response delay. Besides, the delay grows exponentially when the volume of data becomes larger. Therefore, to recognize a fast and long sequence of gestures, we have to explore strategies to reduce the network delay in sending the data. One possible direction is to send only important data to reduce the volume. That requires the iPad to locally process the data and may increase processing time. Another direction is to pipeline the image transmission process to send images one by one. This, on the other hand, introduces extra overheads on network transmission. The tradeoff needs to be studied further.

B. Hand Recognition Restriction

Currently, we distinguish different gestures through angles between fingers. However, this criterion may not be sufficient to distinguish gestures where angles formed between fingers are similar. To increase accuracy, some other criteria should be considered.

In this paper, we present our experience on developing the first hand gesture application on a mobile device. We found that gesture recognition on iPad is feasible but the response time may not be good enough for a real-time application. We proposed to use cloud computing to reduce the response time. Our experimental results show that the response time can be reduced significantly by using cloud computing. Our results demonstrate that it is possible to adopt the camera on a mobile device as an input device, which facilitates a whole new type of applications to be developed.

Acknowledgements

The project has been funded in part by the HKU 81 Inclusion Fund. The authors would also like to thank Kristen Au, Anji Ren, and Ami Wang for their valuable discussions.

References

[1] iOS Developer Site, "https://developer.apple.com," 2012.

[2] Android Developer Site, "http://www.android.com," 2012.

[3] Christian Metzger, Matt Anderson, and Thad Starner, "Freedigiter: A contact free device for gesture control," in *IEEE Proceedings of the Eighth International Symposium on Wearable Computers*, 2004.

[4] Kinect Official Site, "http://www.microsoft.com/enus/kinectforwindows," 2012.

[5] A. Agarwal, S. Izadi, M. Chandraker, and A. Blake, "High precision multi-touch sensing on surfaces using overhead cameras," in *Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, 2007.

[6] E. Hildreth and F. Macdougall, "Multiple camera control system," in *US Patent no. 7058204*, June 2006.

[7] Shamik Sural, Gang Qian, and Sakti Pramanik, "Segmentation and histogram generation using the hsv color space for image retrieval," in *IEEE Proceedings of International Conference on Image Processing*, 2002.

[8] Kedar A. Patwardhan, Guillermo Sapiro, and Vassilios Morellas, "Robust foreground detection in video using pixel layers," *IEEE Pattern Analysis and Machine Intelligence*, vol. 30, no. 4, pp. 746–751, 2008.

[9] Andreas Klein, Christian Mannweiler, Joerg Schneider, and Hans D. Schotten, "Access schemes for mobile cloud computing," in *Eleventh International Conference on Mobile Data Management*, 2010.