User defined custom virtual keyboard

Rishikesh Kumar
Android Platform
Paytm
Bangalore, India
rishi.sit@gmail.com

Poonam Chaudhary
Android Platform
Samsung Research India
Bangalore, India
poonam.c@samsung.com

***Abstract*— At present, the smart phones are mostly touch based, and they don't have an input hardware (keyboard) associated with them. Also, the software keyboard present in these smart devices, are very small as compared to desktop/laptop. This results in slow typing and also, causing inconvenience for the disabled people. In the smart devices (mobile, tablets, TV etc.), a hardware keyboard is missing. There are few solutions for making virtual keyboard but those solutions are not customizable and accurate enough with existing phone's hardware. Some solutions like laser keyboard which can be used as an alternative to hardware keyboard are there, but it requires an extra hardware to carry along with your device. Also, using specialized 3D cameras can be an alternative to the above problem. This paper defines new way of developing a virtual and custom keyboard, using 2D camera and other sensors available in phone, without the use of any extra input hardware. This implementation enables users to create his keyboard on any homogenous surface like table or paper.**

Keywords-Image Processing; Mobile keyboard; Accessibility

I. INTRODUCTION

The objective of this paper is to develop a Virtual keyboard using existing hardware. This paper is based on image processing of captured frames of virtually created keyboard. This paper is based on dynamic learning of letters which is defined by user. This can be extended not only for disabled people but also for TV like devices which doesn't have generic inbuilt keyboard.

Our implementation is based on image processing of keyboard for finding letters and fingertip/color band. In this paper, a new dynamic learning algorithm for finding letters has been used with the custom paper/user made keyboard. User can create his keyboard on any homogenous surface like table or paper with his custom letters.

We have developed 2 kinds of keyboards:-

1) Self-created keyboard on white homogenous table and letters written in black ink.
2) Custom keyboard on white paper and letters written in black ink.

This experiment has been done with android phone which has front camera of 2MP to 5MP. This can be extended to any resolution of camera. We have got 96% of accuracy with different typing speed. As typing speed varies from person to person, so we kept threshold as dynamic and it updates according to speed of typing.

Flow Diagram

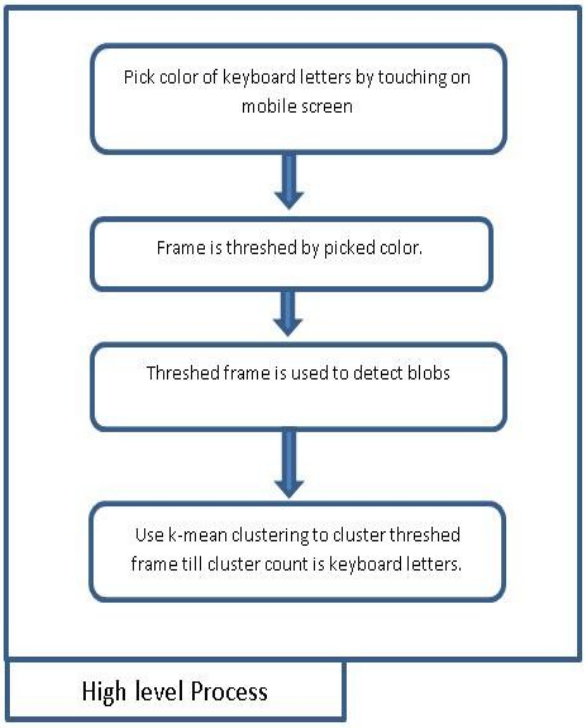

Figure 1: High level diagram

II. RELATED WORK

There have been some of the proposed solutions for android phone for virtual keyboard but it requires extra hardware either the 3D camera or laser hardware. There is no proposed solution for android phones without any extra hardware. We have implemented our own algorithm for creation of center for individual letters and finding the co-ordinates of letters. [1] Defines a way for detecting the keyboard with the help of camera on desktop version in which camera vision has clear keyboard image. However, this solution doesn't yield satisfactory result when the camera is in tilted position. Most successful result has been obtained with custom laser hardware. [5] Defines character reorganization using shadow mechanism. All the existing solutions that are available at present, which provide the given functionality without the use of any external hardware has 92% accuracy. We have achieved more than 96% accuracy with different light and camera conditions along

978-1-5090-1987-8/16/$31.00 ©2016 IEEE

different typing speed.[6][7] Tesseract provides a way to detect characters but we haven't got satisfactory result with tilted camera.

III. Proposed Solution

We have proposed two solutions specific to android platform but can be extended to any platform. First method is related to process the letter color and fingertip color both and the second is related to processing the fingertip while position of letters will be determined intelligently.

A. Learn Letters using Color

Keyboard detection (Fig. 5) explains the complete flow of determining the color of letters. Typed letter detection (Fig. 6) explains how to detect the fingertip using the color tip and deciding the touched letter. There are below methods to achieve this result.

1) Keyboard mapping using Color detection of letters

This will detect letters using color detection. The letters will be mapped based on the positions of letters. The algorithm used here will make more than one contour to one blob for one letter. This algorithm determines different blobs for the same letter and combines all the detected blobs intelligently to create one blob for each letter. In order to detect color of each letter, touch gesture on phone screen is needed. It has been assumed that all letters are of same color.

We select the frame with color detected. This will give us a set of frames that can be used to detect boundary [10] of the object in consideration. After this processing we will get some blobs having boundaries and area. Only the blobs, whose area is greater than threshold area, will be selected. This process will be repeated, until we get blob of each letters. After all blobs are detected we make fixed circle on blob's position (actually on letter's position).

a) Fingertip/color band detection

We have implemented algorithm to detect fingertip color and its co-ordinate in different frames. Whenever fingertip co-ordinates and co-ordinates of letters coincide, the algorithm will assume that a letter has been typed.

Depending upon light condition, it has been observed that fingertip may not be detected accurately. To remove this problem to some extent the color band on fingertip can be used. When we type (touch letter in paper keyboard), capture that frame, capture just next frame as well. See the difference of previous and current frames.

b) Mapping of position of letters

We will map all letters with its co-ordinates. Our algorithm compares the co-ordinates of fingertip in each frame and compares with mapped letter co-ordinates. If difference is less than some threshold than we process current frame and detect position of fingertip using threshing and boundary detection. We calculate Euclidian distance between fingertip and all letters (fixed blob positions) of paper keyboard.

Thus we find letter having minimum distance from fingertip. That letter (fixed blob position) is only our printed letter (fixed blob positions) of paper keyboard. Thus we find letter having minimum distance from fingertip. That letter (fixed blob position) is only our printed letter.

B. Learn Letters using Position of letter

This is the second method of finding the letters position and detecting the exact overlapping of fingertip and letter position.

1) Dynamic learning of co-ordinates of letters by placing fingertip on suggested letters

Dynamic learning refers to detecting positions of all letters present on paper keyboard. We have used method of manual learning and automatic mapping.

To start with, users can choose any color band to wear on fingertip. Phone's front camera should be placed at a fixed position so as to cover the drawn keyboard in the camera's viewport. To learn a letter we have to keep fingertip on that letter ensuring that at least till 10 frames on that position, has been captured. Meanwhile all the 10 frames are stored and average positions of fingertip in each frame is taken. Average position calculated is mapped to letter's (X, Y) co-ordinate. This process has to be repeated until all letters has been trained. [9]After detecting all the letters, keyboard layout along with its coordinates and the coordinates of center of each letters are stored.

2) Key Board Boundary Detection

All letters are learned and mapped. When application is opened where user wants to type, he will be shown the rectangle that we saved at the time of learning. User has to adjust the custom keyboard so that it will fit in rectangle shown. Now user can start typing using custom keyboard.

Note: Mapping of letters and fingertip while typing in this method is same as previous method. Here we map on previously trained data.

IV. Experiment

In our implementation, we have used Samsung galaxy android phone (S4) with 2MP camera and tested till 5MP [Figure 2].We have tested in low light, moderate and high light. We are changing the UI with a border line in camera UI as soon as device detects the keyboard after which user can start typing. This paper addresses problems with current Virtual Keyboard implementations and describes a novel technique with more accuracy, using existing hardware of any normal android phone to solve current problems. It can be extended to any devices which has front camera and image processing.

Below figure, displays the experimental setup. The device is tiled in such a way that complete keyboard should come in the front camera view. In our experiment it is tilted about 10 degree.

Figure 2 : Experimental Set-up

We have made the experiment for 4, 9,12,16,24 letters and 20 iterations for each case [Fig 3] and [Fig 4]. The letters has been written on white surface with black marker with visible intensity. The result was same with printed keyboard as well. We have constantly typed, keeping finger on each letter for 1 second. No. of skipped frames define the speed of typing letters.

Results shown below.

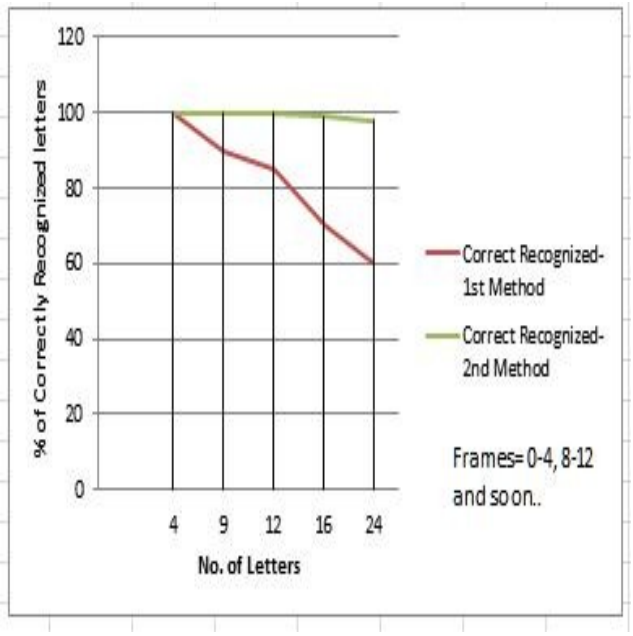

Figure 3. Total no. of letters Vs Correctly recognized with 4 frame skip

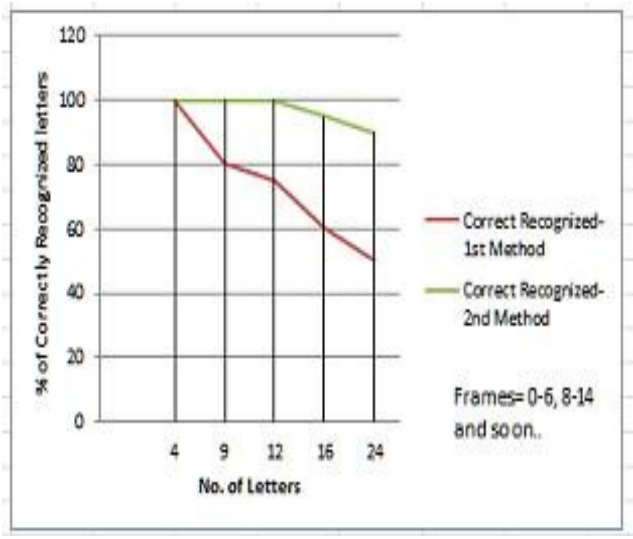

Figure 4. Total no. of letters Vs correctly recognized with 2 frame skip

V. CONCLUSION

In this paper we have implemented virtual keyboard with two different and new methods on android phone. Out of these two methods, second method is more generic in our day to day life as it is based on dynamic learning. Dynamic learning keeps the letter's co-ordinates at fixed place irrespective of other factors. If we are using a printed keyboard then next time we have to keep the keyboard only with matched border (as kept previously). In the first method, there is no need of training again but performance wise second method is better as shown in graph. In the second implementation we have removed the color detection of letter. It enhances the performance as it detects only the fingertip/color band and fixes the co-ordinates of letters respectively. [8]In first case, blob detection and making the nearby blobs into one blob, degrades the performance and accuracy; as we increase the number of letters, number of blobs increases and diameter of blobs decreases. Above experiment has taken with total 150 users, and out of which 50 users 0.5sec per letter, 30 users are with 1sec per letter speed, 30 users 2sec per letter and 40 users with 3sec per letter. We have to keep the entire letters set in front of camera and it should not be overlapped. Reference implementation as well as feasibility tests were conducted and successfully concluded by us.

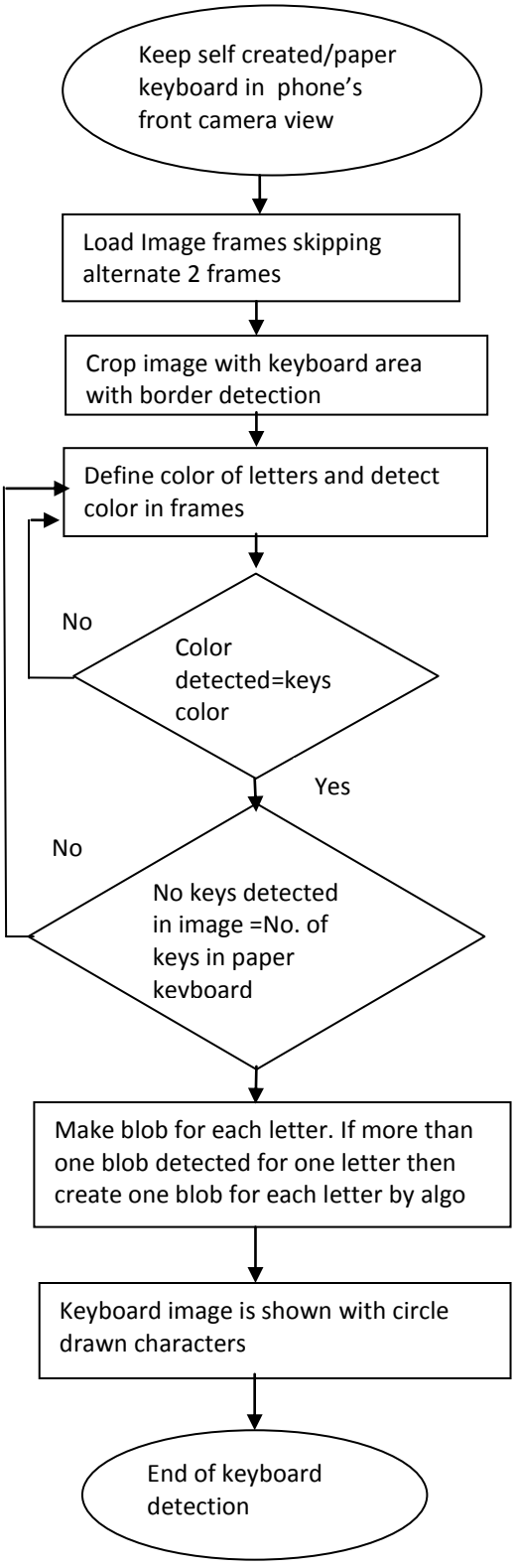

Figure 5. Keyboard detection

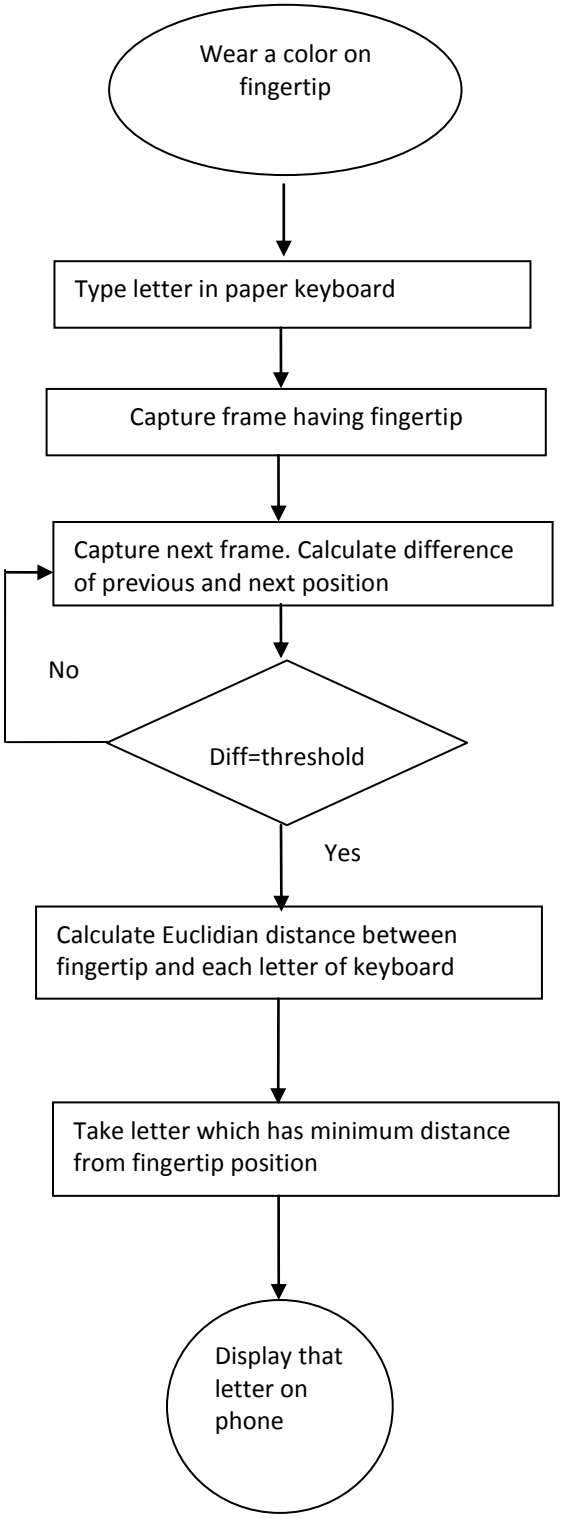

Figure 6. Typed Letters detection

VI. Future Scope

These are the 2 major areas where improvements can be done:-

1) Fast typing
2) Fingertip in air View

Fast typing defined as per letter per second.When the user typing speed is fast then comparison of fingertip and letter co-ordinates becomes difficult to map. Also one more use case needs to be addressed: if the fingertip is in air and co-incident with letter position. In this case, it may give wrong value. [2] Defines the method of detecting the fingertip with the help of shadow mechanism which can be adopted for more accuracy.

REFERENCES

[1] Matsui, N. & Yamamoto, Y. "A New Input Method of Computers with One CCD Camera" *Virtual Keyboard. Keio University*, 2000.

[2] Kölsch, M. and Turk, "M. Keyboards without Keyboards: A Survey ofVirtual Keyboards", *Workshop on Sensing and Input for Media-centric Systems, Santa Barbara, CA*, June 20-21, 2002

[3] Segen, J., and S. Kumar, Thad Starner, "Keyboards Redux: Fast Mobile Text Entry," *IEEE Pervasive Computing*, vol. 3, no. 3, pp. 97-101, July-Sept.2004.

[4] B. Martin, P. Isokoski, F. Jayet, T. Schang, "Performance of finger-operated soft keyboard with and without offset zoom on the pressed key," *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, Nice, France, 2009.

[5] Segen, J., and S. Kumar, Shadow Gestures: 3D Hand Pose Estimation Using a Single Camera in *Proceedings of Computer Vision and Pattern Recognition*, (1999), 479-485.

[6] http://sourceforge.net/project/showfiles.php?group-id=158301&package-id=215908. Last accessed: May 12, 2009.

[7] Ray Smith, "An Overview of the Tesseract OCR Engine", *Proc. of ICDAR 2007*, Volume 2, Page(s):629 - 633, 2007.

[8] Kimme C, Ballard D, Sklansky J. "Finding circles by an array of accumulators". *Commun ACM*, pp. 120-122, 1975.

[9] V.F. Leavers, Shape Detection in Computer Vision Using the Hough Transform. *New York: Springer-Verlag*, 1992.

[10] Jain, S.; Bhattacharya, S.; "Virtual keyboard layout optimization," *Students' Technology Symposium (TechSym), IEEE* vol., no., pp.312-317, 3-4 April 2010 doi: 10.1109/TECHSYM.2010.5469164, 2010