RESEARCH ARTICLE OPEN ACCESS

Virtual Keyboard Using BLOB Analysis

Antara Kanade [1], Ayesha Shaikh [2], Mabel Fernandes [3]
Shubhangi Chikhalthane [4]
Department of Computer Science and Engineering
Modern Education Society's College of Engineering
Pune - India

ABSTRACT

To develop an Application to visualize the keyboard of computer with the concept of image processing. The virtual keyboard should be accessible and functioning. The keyboard must give input to computer. With the help of camera, image of keyboard will be fetched. The typing will be captured by camera, as we type on cardboard simply drawn on paper. Camera will capture finger movement while typing. So basically this is giving the virtual keyboard.

As the technology advances, more and more systems are introduced which will look after the users comfort. Few years before hard switches were used as keys. Traditional QWERTY keyboards are bulky and offer very little in terms of enhancements. Now-a-days soft touch keypads are much popular in the market. These keypads give an elegant look and a better feel. Currently keyboards are static and their interactivity and usability would increase if they were made dynamic and adaptable. Various on-screen virtual keyboards are available but it is difficult to accommodate full sized keyboard on the screen as it creates hindrance to see the documents being typed. Virtual Keyboard has no physical appearance. Although other forms of Virtual Keyboards exist; they provide solutions using specialized devices such as 3D cameras. Due to this, a practical implementation of such keyboards is not feasible. The Virtual Keyboard that we propose uses only a standard web camera, with no additional hardware. Thus we see that the new technology always has more Benefits and is more user-friendly.

Keywords:- Image Display, Image Processing software, Digitization and Image Capture, Camera Calibration, Edge and feature detection, Pixel classification, Colour, Motion, Shape.

I. INTRODUCTION

As the demand for computing environment evolves, new human-computer interfaces have been implemented to provide multiform interactions between users and machines. Nevertheless, the basis for most human-to-computer interactions remains the binomial keyboard/mouse. We are presenting here a next generation technology, which is the Virtual Keypad. As the name suggests the virtual keypad has no physical appearance. Virtual keyboard is an application which virtualizes hardware keyboard with different layouts hence allowing user to change the layout based on application. E.g. user can select different language for editor or select a specialized layout for gaming applications. User can even design his own layout in hardware version.

1.1 Problem definition:

Recent years have marked a sharp increase in the number of ways in which people interact with computers. Where the keyboard and mouse were once the primary interfaces for controlling a computer, users now utilize touch screens, infrared cameras (like Microsoft's Kinect), and accelerometers (for example, within the iPhone) to interact with their technology. In light of these changes and the proliferation of small cameras in many phones and tablets, human computer interface researchers have investigated the possibility of implementing a keyboard-style interface using a camera as a substitute for actual keyboard hardware. Broadly speaking, these researchers envision the following scenario: A camera observes the user's hands, which rest on a flat surface. The camera may observe the hands from above the surface, or at an angle. The virtual keyboard's software analyses those images in real-time to determine the sequence of keystrokes chosen by the user. These researchers envision several applications for this technology: in some countries (for example, India), users speak many different languages, which makes producing physical keyboards for many different orthographies expensive. A camera-

based keyboard can easily support many languages, Smart-phone and tablet users may occasionally want to use a full-sized keyboard with their device, but are unwilling to carry a physical keyboard. Since most mobile devices are equipped with a camera, a camera-based keyboard could provide a software-based solution for this problem.

. In the system we implemented, a single low-quality camera captures RGB images of a user's. Hands, which touch a patterned surface, or keyboard-mat, in order to select keystrokes. Based upon this information and an initial calibration image of the keyboard-mat, the system produces a discrete sequence of keystrokes that can be used to control other software. We examine the performance of each of the three main phases of image analysis and compare the efficacy of two techniques for locating the user's fingertips in an image. We also analyse the degree to which our system is sensitive to changes in lighting and frame rate.

1.2 Fingertips detection:

In this step, we estimate the locations of the user's fingertips (in image-space) based on geometrical features of the contours and regions obtained from the last step. We represent a contour ˜ γ as a sequence of positions {p1 = (x1, y1), p2 =(x2, y2)} in image-space. Given a contour, one can derive several other sequences of geometrical significance and use these to locate the fingertips. The previous processing step gives us a contour ˜ γ consisting of pixel locations pi such that the Euclidean distance between pi and pi+1 is 1 for all i, and the angle in between the displacement vectors pi+1 − pi and pi−1 − pi is in {−π/2,0,π/2,π}.

This angular information does not give a very good idea of how the contour bends around the hand, so we consider a subsequence γ in which the points of the contour are spaced further apart. In experiments, we took γ to be every tenth point of ˜ γ. We then define a sequence of angles in (in (−π, π]) from γ as we did with ˜ γ, by considering the angle between displacement vectors. A second important property we derive from γ is curvature. Curvature can most easily be understood in the situation where γ is parameterized by arc length. In this situation, the acceleration vector γ00 (t) is always perpendicular to the contour. The curvature K (t) is simply the magnitude of γ00 (t), with larger values indicating that the curve is bending more severely. The data we use to represent γ does not parameterize the contour by arc length, but the authors of report there is still a formula for calculating K (t)

$$K(t) = \frac{\det(\gamma'(t), \gamma''(t))^2}{\|\gamma'(t)\|^3}$$

Given this geometrical data, we have two approaches for locating the fingertips in an image. The first approach is to calculate the sequence of angles in, and associate fingertips with points where in is maximized. The second approach is to calculate the curvature values Ki and associate fingertips with positions where Ki is maximized. Naturally, both approaches require non-maximum suppression or other post-processing to be effective. Both of these approaches are described in and we compare their performance.

1.3 Touch Detection:

In this phase of processing, we are given as input the estimated positions of the user's fingertips and must output which of those tips are estimated to be in contact with the keyboard-mat. We used a technique called shadow analysis, described in, to solve this problem. The first step in shadow analysis is to extract the shadow of the user's hands from the scene. We accomplish this by thresholding the image in 8-bit HLS colorspace coordinates.

From examining many test images, we found that a threshold of L < 0.30·255 produced a binary image S that clearly depicts the shadows in the image.4 For each fingertip position p, we examine square 20×20 neighbourhood of p in the binary image S. If the percentage of shadow pixels in the neighbourhood is smaller than a fixed percentage s, we conclude that the fingertip is in contact with the table. We refer to s as the shadow threshold. In the experimental section, we investigate the effect of varying this threshold

1.4 Mapping Touch Points to Keystrokes:

The previous phase of processing yields a list of image-space coordinates that estimate where the user has touched the keyboard. In this section, we describe how we translate those image-space coordinates to rectilinear keyboard-space coordinates, and then to actual keystrokes, correcting for perspective. The perspective

correction technique we used did not appear in any reference we read, instead we derived it from ideas in the lecture on stereo vision. An image captured by the camera in our system is a perspective projection of a three dimensional scene. Our system assumes the layout of the keyboard is known at compile time and that the keyboard-mat given to the user has several control points printed on it, arranged in a rectangle of known height and width. For test purposes, our keyboard consisted of a square-shaped grid with 25 keys, as depicted in Figure 3. We designate the four control point's c0, c1, c2, c3. Each point ci = (xi, yi, zi) is a point in the plane of the keyboard mat. We assume our camera is a pinhole camera of focal length d, and carry out all of our calculations in "camera coordinates", so that the camera's aperture is at the origin with its optical axis pointing in the −z direction, with no rotation about the z-axis.

For simplicity, we assume all of our length-valued variables have the same units. To obtain the positions of the control-points in image-space, we begin by thresholding an 8-bit RGB test image captured at the beginning of processing with the inequality B < 100. The resulting binary image has four connected components, we assume that the keyboard-mat is aligned so that the vectors c1 − c0and c2 − c3 are parallel to the x-axis with length w and the vectors c3 − c0and c2 − c1 are perpendicular to the x-axis, with length h. Thus, under the perspective projection, the keyboard-mat looks like the image in Figure.

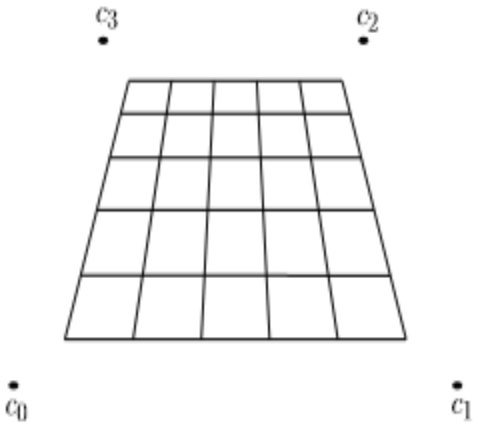
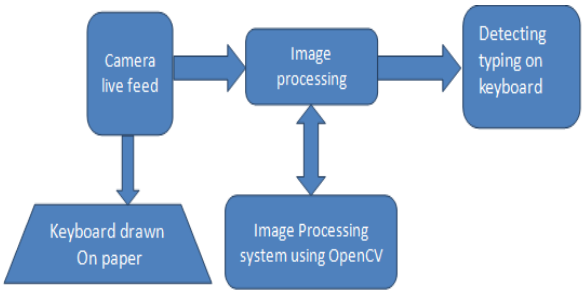


Figure: An example of the keyboard mat, under perspective projection

II. SYSTEM ARCHITECTURE DESIGN



1. We describe the image analysis techniques used to convert the user's surface-touches into keystrokes. Most state-of-the-art camera-based virtual keyboard schemes, including, are implemented using the following sequence of image analysis techniques:

2. The system's camera captures an image I for processing. By performing skin segmentation on I, we obtain a binary image with a region H representing the hand(s) in the scene.

3. H is analysed to determine the locations of the user's fingertips. The references listed above determine contours that parameterize the boundary of H. They associate fingertips with the points on the contours that optimize certain geometric quantities (for example, curvature). We consider two of these geometric approaches to finding the fingertips and compare their performance.

4. Given the estimated fingertip positions, one must calculate which fingertips are touching the keyboard's keys. Propose a technique called shadow analysis to solve this problem.

5. To map touch points to key presses, we assume the keys of the virtual keyboard are described as rectangles in R2. We assume the layout of the keyboard is known at compile time and the keyboard-mat has control points from which we can derive a perspective correction transformation. We then apply a simple formula to convert the keyboard-space coordinates of the touch points to key presses.

III. LITERATURE

[1]Contain the following Points:

1. Basics Idea of Touch Detection-

We use shape characteristics like click or touch in touch detection. The shape of projected interactive element is distorted by the fingers when we touch it. Detecting the distortion of the interactive element is an important way to monitor the touch.

2. Touch Detection in Virtual Keyboard-

Detect the average offset Hoffset and select candidate keys with button's distortion width K. Then determine which key distortion are caused by fingertip from candidates. Judge whether the fingertip touch the key or not.

3. Touch Detection with the help of Camera

We used Paper for Virtual Keyboard Implementation.

In [2] we learn an image processing filter is applied the input image to improve it for further processing. Here we either blur the image in case it's too sharp. Else we sharpen.

In [3]This paper describes that How to Map Touch Point to Keystrokes

In [5] This IEEE Paper describes that how current Virtual Keyboard implementation done and describes shadow analysis, edge detection, keyboard detection, tip detection to solve these problems.

From [11]we got idea about How Drag and type method used for secure and accurate password entry on the small touch screen for security purpose.

[13] Represent a novel mono-vision virtual keyboard design for consumers of mobile and portable computing devices such as PDA's, mobile phones

IV. ALGORITHMS

Blob Analysis

Blob Analysis: Blob Analysis is a fundamental technique of machine vision based on analysis of consistent image regions. As such it is a tool of choice for applications in which the objects being inspected are clearly discernible from the background HSVMODEL HSL stands for hue, saturation, and lightness, and is often also called HLS. HSV stands for hue, saturation, and value, and is also often called HSB (B for brightness). A third model, common in computer vision applications, is HSI, for hue, saturation, and intensity.

4.1Algorithm1/Pseudo Code (BLOB)

The image is blurred with increasing standard deviations. The difference between two successively images from video feed using back- ground subtraction method and the results are then stacked up. The movement of object will be captured and that object will be detected by the BLOB.

4.2Algorithm2/PseudoCode (HSV Model)

HSV stands for Hue, Saturation, Value. The HSV colour model, also called HSB (Hue, Saturation, Brightness) It defines a colour space in terms of three constituent components:

• Hue is the colour type (such as red, magenta, blue, cyan, green or yellow). Hue ranges from 0-360 deg.

• Saturation refers to the intensity of specific hue. Saturation ranges are from 0 to 100%. In this work saturation is presenting in range 0-255.

• Value refers to the brightness of the colour. Saturation ranges are from 0 to 100%. Value ranges are from 0-100%.

In this work saturation and value are presenting in range 0-255.

V. CONCLUSIONS

In this project, we implemented a virtual keyboard by using Camera and Image Processing.

Implementing a virtual keyboard system gave us a better understanding of the trade one must consider when choosing image analysis techniques in the context of a larger system.

We are try to characterize what typing tasks users currently perform on their touch screens and then examine whether those tasks can be performed effectively with a virtual keyboard.

ACKNOWLEDGMENT

It gives us great pleasure in presenting the preliminary project report on 'VIRTUAL KEYBOARD'.

I would like to take this opportunity to thank my internal guide Prof. S. S. Raskar for giving me all the help and guidance I needed. I am really grateful to them

for their kind support. Their valuable suggestions were very helpful.

I am also grateful to Dr. N. F. Shaikh, Head of Computer Engineering Department, M. E. S. College of Engineering for her indispensable support, suggestions.

Last but not least, our special thanks to the college for providing various resources such as laboratory with all needed software platforms, continuous Internet connection, for Our Project.

REFERENCES

[1] Bare-fingers Touch Detection by the Button's Distortion in a Projector–Camera System Jun Hu, Guolin Li, Xiang Xie, Zhong Lv, and Zhihua Wang, Senior Member, IEEE APRIL 2014.

[2] Int. Journal of Engineering Research Paper and Applications www.ijera.com ISSN: 2248-9622, Vol. 4, Issue 4 (Version 4), pp.129-130, April 2014.

[3] A Camera Based Virtual Keyboard with Touch Detection by Shadow Analysis, Joseph Thomas, December 10, 2013.

[4] E.Psner, N.Starzicki, & E.Katz. A single camera based floating virtual keyboard with improved touch detection. In Electrical Electronics Engineers in Israel (IEEEI), 2012 IEEE 27th Convention of, Page 1 to 5, 2012.

[5] Y. Adajania, J. Gosalia, A. Kanade, H. Mehta, Prof. N. Shekokar, "Virtual Keyboard Using Shadow Analysis",Third International Conference on Emerging Trends in Engineering and Technology, 2010.

[6] H. Du, T. Oggier, F. Lustenberger and E. Charbon, "A virtual keyboard based on true3D optical ranging," Proc. British Machine Vision Conference (BMVC), Oxford, pp. 220-229, Sept. 2013

[7] M. Hagara and J. Pucik. Fingertip detection for virtual keyboard based on camera. In Radioelektronika (RADIOELEKTRONIKA), 2013 23rd International Conference, page 356 to 360, 2013.

[8] Sumit Srivastava and Ramesh Chandra Tripathi. Real time mono-vision based customizable virtual keyboard using finger tip speed analysis. In Masaaki Kurosu, editor, Human-Computer Interaction. Interaction Modalities and Techniques, volume 8007 of Lecture Notes in Computer Science, page 497 to 505. Springer Berlin Heidelberg, 2013.

[9] Wijewantha N. S.",Vista Key: A Keyboard Without A Keyboard – A Type Of Virtual Keyboard" Final year project hesis, Informatics Institute of Technology, Wellawatta, Sri Lanka, April 20012.

[10] YouTube Videos For Virtual Keyboard using Image Processing.

[11] Drag-and-Type: A New Method for Typing with Virtual Keyboards on Small Touchscreens

[12] Taekyoung Kwon, Member, IEEE, Sarang Na, Student Member, IEEE, and Sang-ho Park, Non-member, IEEE

[13] ''Real Time Mono Vision Gesture Based Virtual Keyboard System" Contributed Paper Manuscript received October 15, 2006 IEEE Hafiz Adnan Habib, Muid Mufti, Member.