VIRTUAL SCREEN IMPLEMENTATION USING IMAGE PROCESSING

by
Deepika Chaudhary(d25,50248725) & Manpreet Dhanjal(dhanjal,50248990)

Abstract

With the advancements in the field of artificial intelligence, efforts are going on to make the hardware as portable as possible. Computer has an indispensable part of human life, but the portability issue still remains with it. Through this project we attempt to develop a Virtual Keyboard that can be used to input data into any kind of device. This keyboard requires no external hardware and is quite easy to calibrate and use.

1. Introduction

A virtual keyboard is basically a grid of characters drawn on any plane surface that can be inputted using a keyboard. The demand of virtual keyboard is increasing day by day because of the inherent issues with the traditional keyboard i.e. it is bulky and not portable and the cannot be used with different devices.

For this project we have developed an application (GUI) that takes input from a virtual keyboard and displays the output in a textbox. The most demanding part of this project was to get input from the virtual keyboard. Basically, a continuous video feed is provided to the application using the web camera of the laptop and frames are extracted from the video every few seconds. Computer vision and Image processing techniques are then applied on these frames to detect the key touched by the user.

Through this project we were able to employ many of the image processing techniques learnt in this semester like morphological operations, image transformations, edge detection, color formats, image filters etc as well as learn new techniques like image segmentation, image bounding etc.

2. Literature Review

Various efforts have been made to reduce the size of keyboard, from minimizing the hardware to including image processing techniques. The appearance of Laser Projection Virtual Keyboard presents a latest solution for input on personal computers and portable device too.

In their research paper, Turk, M and Kölsch [1] described the Virtual Keyboard in various forms out of which, mostly virtual keyboards depend on CCD cameras and 3D optical ranging and significantly depends on primarily on image processing. They did a research which focused on various forms like rings, gloves, projection and devices based on hand gestures. A special type of 3D camera or two 2D cameras was used.

In addition to this, pattern projector has been used in this project for projecting the keyboard. Single CCD camera will be used while making the Virtual Keyboard design using high power infrared light source and a camera with the help an infrared filter.

Sarcar S., Ghosh S., Saha P. K., Samanta D [2] developed a drag-type mode based on touch screen techniques to take the location of the keystroke in smart mobile phones.

Hagara M, Pucik J. [3] had proposed detecting fingertips in form of 2D images using a pair of cameras and then the fingertip positioning 3D space would be determined using triangulation method.

For this project, we researched on different methods to develop a virtual keyboard using image processing techniques and make it as robust and as easy to use as possible. We developed techniques to detect a keyboard on a sheet of paper and detect finger touch on a virtual keyboard in 2D space.

3. Method

The whole implementation of this application can be divided into the steps shown in the flow chart below:

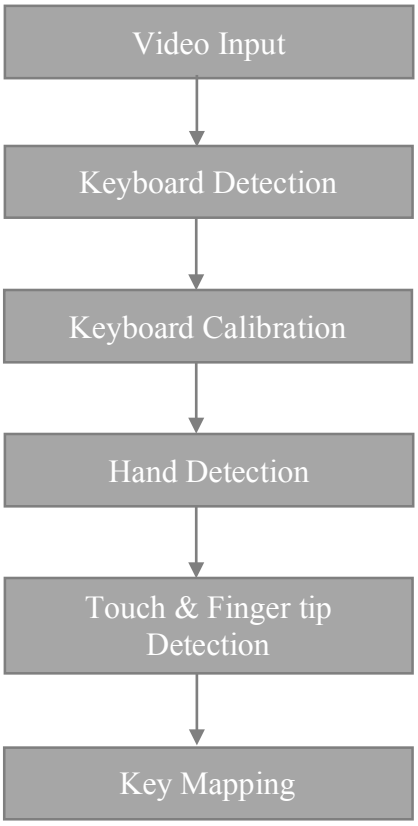


Figure 1. *Flow diagram of the virtual keyboard detection using image processing.*

3.1 Video Input

The web camera of the laptop was used to take continuous feed of video for the virtual keyboard GUI. Image frames are extracted from this video feed after every 1 second. These image frames are further processed to detect the virtual keyboard and location of finger tip on the virtual keyboard.

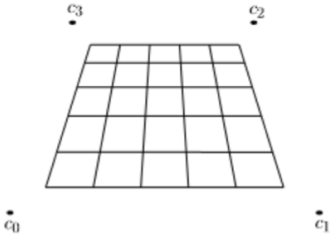
3.2 Keyboard Detection

For keyboard detection we implemented and compared 2 methods. In first method, the corners of the keyboard were marked 4 black circles. The image of this keyboard is converted into a binary image using appropriate threshold. Filtering is done to remove noise from the image and erosion is done to finally remove the remaining noise. After all of the preprocessing, the four corner circles are detected which gives us the coordinates of the virtual keyboard relative to the video frame.
The second method, also included similar preprocessing but instead of using the 4 corner circle technique, it required manual calibration by the user. The GUI has a button named 'Calibrate KeyBoard' which enables user to first adjust the keyboard properly in order to proceed. The snapshot of the virtual keyboard is taken and the mage is transformed in the 2D plane. The Keyboard may be kept at any angle from the camera. The challenge was to transform the image taken by camera so that it looks straight to the view point.

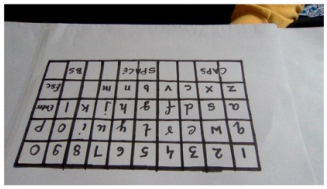
3.3 Keyboard Calibration

One of the main requirements of virtual keyboard is a plane background. However, in this project we tried to make the keyboard robust to the background changes. Thus comes the need of keyboard calibration.
After thresholding the image of the virtual keyboard is converted to a binary image. Then we perform the sobel technique for edge detection and the keyboard is extracted from the image by getting the bounding boxes on the image. The number of bounding

boxes can vary from 1 to 4 depending on the noise in the background. Certain elements in the background environment may get detected through edge detection. Our application tries to find the best match for the keyboard but might detect a wrong match and hence calibration needs to be done again and again until correct match is found. After detecting the appropriate bounding box, the image is cropped to appropriate size. Note that the image is still at an angle and needs to be transformed so that it appears straight to the camera. As shown in the figure 2b, the perspective projection of the keyboard mat as viewed by the camera is at an angle. There are 4 coordinate points c0, c1, c2, c3 of the figure in shape of a trapezoid. This trapezoid should be converted to a rectangular image in order to map the coordinates correctly. This can be done by rotating the image in the –z direction. But the constraint here is that the rotation angle is unknown and it can be different for different settings. The solution to this problem is to find the mid point lines diving the image into four coordinates and then performing the transformation. The horizontal mid point line is calculated by averaging c0, c3 and c1, c2. Whereas the vertical mid line is calculated by averaging c0, c1 and c2, c3. Now our original keyboard mat image is transformed with respect to these coordinates in the –z direction into a rectangular shape using 'imtransform'. Now the image is straightened from the top view perspective and now the grid lines are drawn on it thus appropriately dividing the keyboard 6 x 10 matrix where each cell represents a corresponding ascii character.



(a)



(b)

(c)

(d)

(e)

(f)

Figure 2. *Images for keyboard calibration. a). keyboard mat orientation with 4 points of trapezoid b) original keyboard mat image used c) output of edge detection d) area markings for detected trapezoid e) transformed image from trapezoid to rectangle f) 6 x 10 matrix for character detection.*

3.4 Hand Detection

Once the keyboard calibration and detection are done, we can start typing onto the virtual keyboard. Now, in order to detect the key that the user intends to press, we need to detect the user's hand first. For this, color based image segmentation was used.
After a lot of experimentation with different images of skin the range of skin color in yCbCr color format was found out to be

77 <= Cb <= 127; 133 <=Cr <=173

Finally, after segmentation, a binary image is created where the hand is represented in white color and the background is black.

3.5 Touch & finger tip Detection

There are a large number of ways to detect the touch of the finger on the keyboard once we have found the image of hand using segmentation like edge detection, convex hull creation etc and finding the coordinates of the tip of the finger. We used morphological operations and found the coordinates of the tip or the image by find the max coordinate of the white pixel appearing in the image. The tip of the finger touching the keyboard will definitely be on the max point in the direction of our hand. After finding the coordinates of the tip, we transform the point into the coordinates of the transformed keyboard image by again performing imtransform on the image. In this way, we can finally map the coordinates of the tip of hand to the grid that we created earlier on the transformed keyboard image.

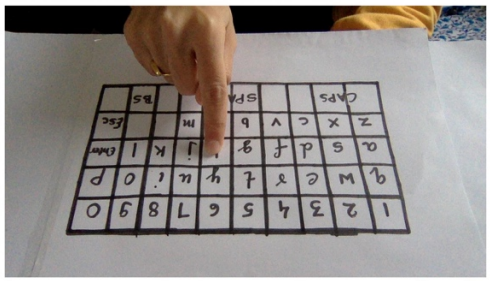
(a)

(b)

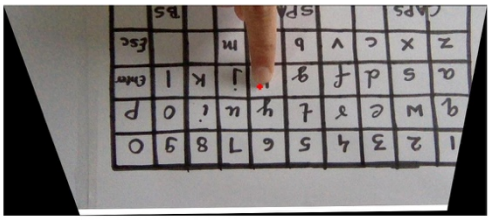
Figure 3. *a) Segmented Image of hand generated via edge detection b) image after transformation in the plane of keyboard coordinates.*

3.6 Key Mapping

The mapping of ASCII values of various alphabets and special keys like enter, escape, space etc are stored in a 2 dimensional array according to the grid coordinates. We obtain the characters represented by the keys by mapping the grid points obtained from the previous steps to the 2 dimensional array already created to store the corresponding values. These characters can be viewed in the textbox in the virtual keyboard GUI.



(a)



(b)

Figure 4. *a) image when key is pressed b) locating the edge of finger tip as shown by red marker.*

4. Experimental Analysis & Results

We experimented with different techniques while working on the project. Some of them are mentioned below:

4.1 Graphical user Interface

A GUI was developed for the project in order to capture all the images of keyboard. It is used to calibrate keyboard, start capturing the live video feed in order to take user input. End the live video feed to terminate the connection to camera and closing the application. And lastly the save button to capture the user input data and save to a file. The GUI is shown below:

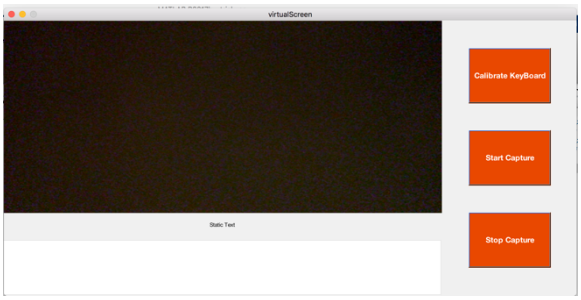


Figure 5. *GUI interface for Application.*

4.2 Keyboard calibration

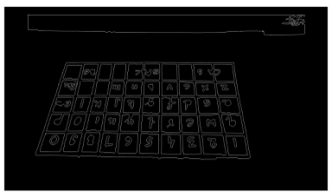
Calibrating the keyboard is the first step to out project. There is some background constraint with the calibration part. Certain part of the background in the plane of keyboard mat should be white. Otherwise it takes long time to calibrate the keyboard. For efficiency and consistency, we used a white background for keyboard detection. For the ease of proper calibration, the keyboard mat is kept at an angle from camera. We were able to get quick results by this technique. Another way of keyboard detection is to perform a color based segmentation of the four corners of keyboard but for this we need to assign for example a blue color to the corners of keyboard mat. To avoid such type of constraints we proceeded with non color based segmentation. Another constraint of keyboard that we faced was that the keyboard should have equal spacing for each cell and it should be in the form of grid shown in the figure 4. We tried working with different keyboards but the challenge was unequal divisions of keys. This can be considered as the constraint of our project.

4.3 Circle Detection

Another technique used to detect the virtual keyboard was to put circles on all of its corners. To detect those circles, we experimented by using simple blob detector that was developed in one of the homework assignments on it as well as detecting circles using hough transform method in matlab. The hough transform method provided more accuracy to this step.

4.4 Edge Detection Techniques

We found that Sobel techniques was best for analysis in our case. We worked with other edge detection techniques like Prewitt, Cross Entropy and canny. Sobel technique gives more appropriate results in better ways. The first step is to detect the keyboard. Next the hand is detected and then the fingertips are detected using this technique. When the edge is obtained, Sobel technique will thicken it to remove any discontinuities and finally result is completely traversed into appropriate objects.



(a)

(b)

Figure 6. *output of edge detect objects.*

4.5 Bounding box detection

In order to detect the keyboard mat from the scene we use bounding box option from region properties of an image. Region properties gives us all the properties like Area, Perimeter, Centroid, Bounding box etc. Bounding boxes are drawn around every object that's was detected in the scene. There were multiple objects on which bounding box was detected. The challenge was to detect keyboard image amongst all the objects. For this, every bounding box generated should be traversed and the one whose dimensions are maximum is detected to be a keyboard.

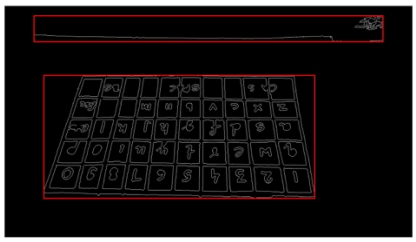


Figure 7*. Bounding box for different objects detected in image.*

4.6 Image Dilation

Dilation method was used to detect the hand in the image. The dilation operator takes two pieces of data as inputs. The first is the image which is to be dilated. The second is a (usually small) set of coordinate points known as a structuring element (also known as a kernel). It is this structuring element that determines the precise effect of the dilation on the input image. The binary image was superimposed by a diamond structuring filter to bring forward the image of the hand.

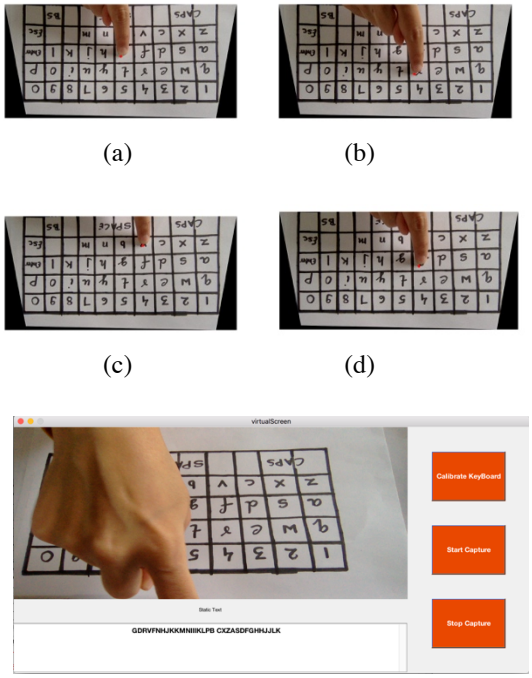
4.7 HSV vs ycbcr for hand detection

We tried the hand detection by converting the rgb image to HSV and then to ycbcr. We got better results in ycbcr method.

4.8 Effect of lightning

The lighting conditions also affect the accuracy of application. In little dark areas the accuracy decreases.

We were able to successfully identify the characters pressed on our virtual keyboard and the results were around 80% accurate. The results depend on how accurately you calibrate the keyboard in the initial phase.



(a) (b) (c) (d) (e)

Figure 8. *Sample results of virtual keyboard*

5. Discussion & Conclusion

The project illustrates the practical implementation of virtual keyboard and how it can be useful in future technology. The user doesn't have to carry bulky keyboards around anymore. All they need is a camera and a projection area. With the increasing demand of compact technology, conventional data entry is required which is flexible and easy to use without affecting the portability of the mobile devices. This app can be reconfigured for different kind of multi lingual keyboards. Thus increasing the user experience and making it flexible for users to customize their keyboard. The idea of virtual touch screen provides easy and simple touch style of writing using a webcam and a paper layout for keyboard mat. Our application is not that robust, it is just a basic design supporting the idea for virtual keypads. But if this idea is reconfigured properly, it would be a great achievement in many technical areas. This idea would be great for the specially abled people who can't see. They don't have to worry about finding out the ways to work on laptops any more. All they will be needing is a paper layout with their recognizable keyboard drawn on it.

6. Future Work

The current implementation of the virtual keyboard is not robust and accuracy might vary as the background changes. Effort is required to make the virtual keyboard implementation invariant to the environment and also requiring minimum manual calibration.

Machine learning techniques like convolutional neural networks can be trained to detect the virtual keyboard.

We also plan to add special characters and keyboard shortcuts like copy, paste etc. to the virtual keyboard.

Further, this work can be extended to implement multilingual keyboards as well keyboards can be developed for blind people.

References

[1] Kolsch, M. and Turk, M. (2002) Keyboards without Keyboards: A Survey of Virtual Keyboards. UCSB Technical Report 2002-21.

[2] Sarcar S., Ghosh S., Saha P. K., Samanta D. "Virtual keyboard design:State of the arts and research issues," . 2010 IEEE Students' Technology Symposium (TechSym), Apr. 2010,pp. 289-299.

[3] Hagara M, Pucik J. "Accuracy of 3D camera based virtual keyboard".2014 24th International Conference on Radioelektronika (RADIOELEKTRONIKA), IEEE, 2014:14.

[4] http://www.ijetmas.com/admin/resources/project/paper/f201605031462271964.pdf

[5] http://www.ijera.com/papers/Vol4_issue4/Version%204/U04404129130.pdf

[6] https://jsthomas.github.io/docs/vkeyboard/vkeyboard.pdf

[7] https://electronicsforu.com/electronics-projects/software-projects-ideas/virtual-paper-keyboard-using-opencv-visual-studio

[8] http://dc.org/files/asciitable.pdf

[9] https://evilporthacker.blogspot.com/2017/10/gesture-driven-virtual-keyboard-using.html

[10] http://www.ijetmas.com/admin/resources/project/paper/f201605031462271964.pdf

[11] https://www.mathworks.com/help/matlab/creating_guis/adding-components-to-the-gui.html

[12] https://help.adobe.com/en_US/AS2LCR/Flash_10.0/help.html?content=00000520.html

[13] https://www.mathworks.com/videos/edge-detection-with-matlab-119353.html