Novel Algorithm for Finger Tip Blob Detection Using Image Processing

Prashanth C Ravoor
Student, Dept of Information Science and Engineering, BMS College of Engineering, Bengaluru, India
Email: prash.ravoor125@gmail.com

Ranjani S and Sudhir Rao Rupanagudi
Worldserve Education, Bengaluru, India
Email: sudhir@worldserve.in

***Abstract*—Touch screens are immensely popular in the modern world. They have reached the pinnacle of their success in mobile phones. Touch screens, along with the Android OS, have completely revolutionized the cell phone and have led to the creation of the smart-phone as we know it today. These touch screens are generally resistive or capacitive. A third type exists, which is the image processing based touch screens. The image processing touch screen has lost ground over its competitors due its large size, and have been limited to certain fields. One notable disadvantage of image processing touch screens is that its use is hindered by lighting: it is simply unusable in poor or unsuitable lighting conditions. This paper presents a solution to this problem by introducing a novel and cost effective approach to detect a fingertip blob on an image processing based touch screen under any surrounding light conditions. The setup uses basic LEDs and a normal visible light camera as opposed to the conventional use of IR cameras. The algorithm was implemented using the Java Development Kit, version 1.5 (update 22). A very high accuracy for detecting blob co-ordinates at any location on the screen under different lighting conditions was obtained.**

***Index Terms*—image processing, finger-tip blob, web-cam based touch-screens, touch-screens, algorithm for finger-tip detection, blob detection**

I. Introduction

Touch screens have been yet another feather in the cap for the booming field of technology. Along with the multi-touch features, touch screens provide an interface that have been difficult to compete with, for other interfaces.

The history of touch screens began in the year 1965, when E. A. Johnson from the Royal Radar Establishment, Malvern, UK, published an article describing a capacitive touch screen [1] and [2]. Later, in the 1980s, the Fairlight CMI [3](and Fairlight CMI IIx) was a high-end musical sampling and re-synthesis workstation that utilized light pen technology, with which the user could allocate and manipulate sample and synthesis data, as well as access different menus within its OS by touching the screen with the light pen. The later Fairlight series IIT models used a graphics tablet in place of the light pen. The HP-150 from 1983 was one of the world's earliest commercial touch screen computers [4], [5]. Similar to the PLATO IV system [6], the touch technology used employed infrared transmitters and receivers mounted around the bezel of its 9" Cathode Ray Tube (CRT), which detected the position of any non-transparent object on the screen. Today, touch screens are ever present, from ATMs to PDAs to smart-phones.

The touch screens existent today use differing technologies to achieve their objectives [7]. Some of the technologies prevalent today include capacitive, resistive, surface acoustic and optical.

In optical touch screens, light is used as the medium of touch detection. In these, the light used could be IR or visible spectra. When IR is used, the method generally employed is the Frustrated Total Internal Reflection (FTIR) technique [8] - [12]. This uses the phenomenon of total internal reflection. It consists of an acrylic sheet placed just below the glass plane which acts as the touch surface. IR LEDs are placed sideways and perpendicular to the direction of touch to be used by the user. The light from the IR LEDs is totally internally reflected at all points along the glass pane (acrylic sheet). However at the point of contact of the finger, these IR waves are said to be 'frustrated' as they are trapped by the acrylic sheet and the reflection at that point is no longer complete; these points are registered by an IR camera, and appear lighter than their surroundings.

In the image processing touch screen that uses visible spectrum, shadows are used to distinguish the points of contact. Most of the work is performed through image processing (software), which includes segmentation, shadow (noise) elimination and determining the final result - a single pixel on the screen.

The setup used for the visible spectrum image processing touch screen is described in III. A novel algorithm to find the point of contact is described in the later sections.

II. General Working

Manuscript received Feb. 22, 2013; revised March 15, 2013

doi: 10.12720/ijsps.1.1.11-16

All image processing touch screens utilize the following four steps [13] and [14].

A. Colour-conversion

The source image, obtained by the camera is colour converted whenever required. It is possible to work with colour images, but that would mean dealing with at least 3 values i.e. red, green and blue in case of RGB or cyan, magenta, yellow and black in the case of CMYK. In either case converting a color image to gray scale leaves us with an intensity (luminance) value of only 1 byte, ranging from 0 (black) to 255 (white). Also, since we are dealing with touch screens, colour is not needed. Thus processing time is effectively reduced almost by a factor of 3.

B. Segmentation

The color converted source image is narrowed down only to the required portion. The rest of the image, which is not needed, is ignored. This means that in the current scenario, all portions of the image which do not contain the finger-tip blobs are eliminated.

C. Noise Reduction

In the segmentation process, generally, the image is segmented with respect to the background, i.e. the background in which the image was captured is separated. However, noise is also retained. If left as it is, noise would interfere with the working of the device, giving rise to faulty output. Thus noise reduction techniques are used to remove any foreground noise. In this case, the shadow of the hand forms the noise, along with any dust particles on the surface of the screen.

D. Result: Obtaining the Co-ordinate of Contact

The final step of the process is to calculate the result and take actions with due accord. However, differing finger sizes result in ambiguity in finding the point of contact of the finger, as the fingertip blob is to be reduced to a single pixel. Most algorithms go about this by taking a mean of all the pixels that fall under the blob region, or a variation of the same.

Calculating the mean of a set of values, may cause a delay, however slight, especially since the time taken increases exponentially with the area to be covered (time complexity is $O(n^2)$). This combined with the fact that there may have to be one or several divisions and subtractions could lead to a significant increase in the time taken for the response of the device. This is also one of the factors this paper aims to address.

In this paper, a simple algorithm is discussed for noise removal and to obtain the point of contact. The required apparatus has been explored in III. The algorithm itself has been described in IV.

III. Setup

A web camera with the following specifications is used in our experiments:

- Lens Resolution: 25 Megapixel interpolated
- Feature: 4 LEDs placed around the lens provided with a switch to turn them on/off
- Capture Resolution: up to 640 x 480
- Capture rate: up to 30 fps
- USB 2.0 interface

The web-cam is placed in a box made of white thermocole with the top end of the box open. Its dimensions are approximately 8 x 7.5 x 10 inches. An ordinary glass plane around 0.25 inches thick is placed on the open side of the box providing the touch surface. Instead of an acrylic sheet, tracing paper (size A4) is used. The tracing paper is placed between the surface of the box and the glass pane. LED's are provided around the lens of the web-cam to assist in case of inadequate lighting in the area. The setup is as shown in Fig 1.

As seen in Fig 1, the setup is very simple to implement and the apparatus used are also easily available.

The main purpose of the tracing paper is to provide a uniform white background. This way, when the finger is placed on the glass, it appears dark against a dark background.

In case of poor light, the LEDs provided are switched on. Now, the background provided by the tracing paper is not so bright. When the finger is touching the glass, light from the LEDs reflects off it and the finger appears to be brighter than the background.

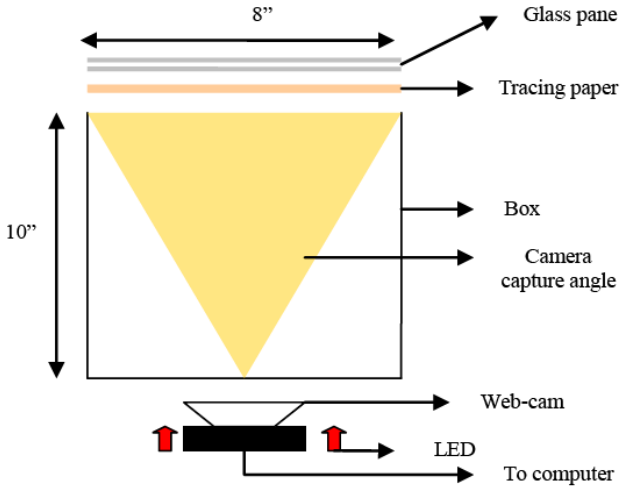


Figure 1. Setup used for our experiments

This means, in spite of the lighting being good or bad, we have a contrasting image. The point of contact appears dark when the background is white and vice-versa. This leads to the thinking that the finger i.e. the area of touch can be detected by focusing on this contrast. Difference in pixel intensities is brought into the picture. When there is large difference in the pixel intensity between 2 pixels in the same image, we can deduce that there is fingertip blob at one of the points. This is further elaborated in IV.

IV. Fingertip Detection

Obtaining the complete result involves 4 steps, as seen in II.

The source image is obtained by the web camera placed below the glass (touch) surface as shown in fig 1. The camera continuously captures images at 640 x 480 resolutions and submits it for processing. This particular resolution is chosen because it helps in reducing processing time whilst preserving sufficient information

to determine the end result. Also, it does not occupy much space in the RAM or secondary memory.

The image obtained is then converted into gray scale. This results in the pixel intensity values ranging from 0 to 255. The image is converted using the formula shown in (1) [15].

$$Y = (0.299 * R) + (0.587 * G) + (.114 * B) \quad (1)$$

where R,G,B are the red, green and blue values of the pixel respectively.

The *Y* value in the equation corresponds to the luminance value of the pixel. This calculation is done for each pixel in the source image. The image thus obtained is the color-converted monochrome equivalent.

The next step is to ensure the device i.e. the web-cam is receiving sufficient light to function. If the device exterior is dim, the background and the finger-tip blob, the finger-tip blob and the shadow of the finger become indistinct, and their differentiation is difficult. Hence under this a circumstance, a prompt is displayed telling the user to switch on the LEDs of the web-cam, which illuminates the glass from below, through the tracing paper.

The segmentation and noise reduction steps are combined to a single step in our experiment. One algorithm performs both these steps in one go. The algorithm works on the principle that, any gradual increase or decrease in intensity or near constant intensity spread across a region of the color converted source image can be ignored to be either background or noise or a combination of both. Wherever there is a sudden change in pixel intensity, there has to be an edge of a finger or the object that is touching the glass.

First, a blank white image is considered with the same size as the source image. The algorithm will work on this image based on pixel values obtained from the source image. Let this image be λ.

The source image is now scanned from top to bottom, rather than the more conventional left to right scan. The reason for the same is explained a little later. The scan does not scan at the origin. Rather, it is started a few pixels directly below the origin at say (0, θ). θ determined through experimentation, is found to be 5. In general, consider 2 pixels α at (x, y) and β at (x, y - θ) in the source image. Equation (2) is applied for each pixel in the picture under consideration.

$$Y_{new(x,\,y)} = Y_{\alpha} - Y_{\beta} \quad (2)$$

where x is the row co-ordinate of the present pixel and x $\in$ (0, 639), y is the column co-ordinate of the present pixel and y $\in$ (δ, 479), Y_{new} is the resultant intensity difference between the intensity values of the pixels α and β.

If the resultant values of Y_{new} obtained above are greater than a threshold δ, this value then is a part of the fingertip edge and is assigned black (*Y* value 0). If the Y_{new} value is less than δ, the pixel belongs to the background and is assigned white (*Y* value 255). This is guarded by (3). In this way, the fingertip is segmented from the rest of the image.

$$Y_{(x,y)} = \begin{cases} 0 & Y_{new(x,y)} > \delta \\ 255 & Y_{new(x,y)} \leq \delta \end{cases} \quad (3)$$

For the sake of illustrations, a simplified table is presented in Fig. 2. The table is assumed to be a 2D matrix A. In the matrix A, each cell is considered to be a pixel. The Y value shown is the pixel intensity of the color converted source image. The output of each cell in the digital image is shown below it. 0 corresponds to black, and 1 corresponds to white.

Again, to illustrate the algorithm, θ value, is assumed to be 2 (difference in intensity is checked every 2 pixels). δ value (the threshold value for difference in pixel intensities) is assumed to be 10.

Y = 190 Digital – 1	Y = 193 Digital – 1	Y = 191 Digital - 1	Y = 188 Digital - 1
Y = 186 Digital - 1	Y = 189 Digital - 1	Y = 183 Digital - 1	Y = 180 Digital - 1
Y = 181 Digital - 1	Y = 130 Digital – 0	Y = 120 Digital - 0	Y = 125 Digital - 0
Y = 132 Digital - 0	Y = 125 Digital - 0	Y = 126 Digital - 0	Y = 119 Digital - 0
Y = 140 Digital - 0	Y = 135 Digital - 1	Y = 129 Digital - 1	Y = 113 Digital - 0

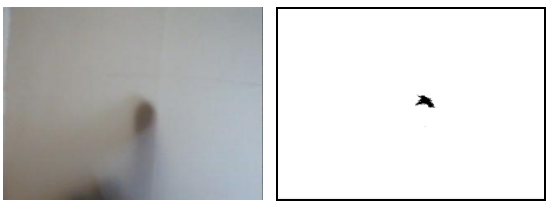
Figure 2. Grid containing pixel intensity values and resultant digital values

Since the algorithm starts from y = θ, the first 2 rows of the matrix *A* are left as it is: thus the first two rows are assigned white in the resultant digital image.

Starting from A_{31}, each pixel's intensity is subtracted with the pixel intensity 2 rows above it in the same column. Thus, A_{31} is compared with A_{11}. $A_{31} - A_{11}$ results in (181 - 190) = -9. Only the magnitude is considered. 9 is less than δ (δ = 10), thus A_{31} is left white.

Continuing in similar fashion, A_{32} is compared with A_{12}. The difference turns out to be -63, whose magnitude is greater than δ. Hence in the resultant image λ, pixel (3, 2) is assigned black. This process is carried on for all the pixels with y >= 3. Thus the resulting digital image is obtained as a single black blob against a white background.

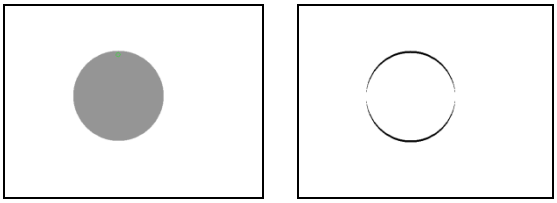
A sample result of the algorithm is shown in Fig 3. It is to be noticed that the shadow of the finger, the finger that is not touching the surface and any other noise has been almost completely eliminated.



(a) (b)

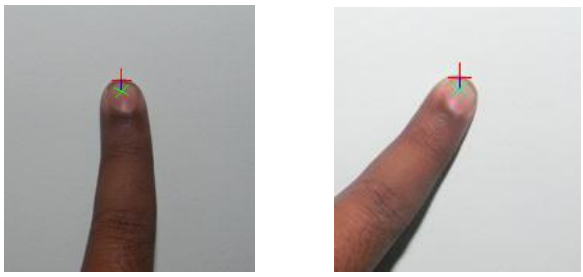
Figure 3. A sample result: a) Image of the finger on the screen; b) Resultant digital image after applying the algorithm.

It can be seen that the algorithm detects only upper or lower circular arcs, since the scanning takes place top to bottom. The sideways arcs are ignored. This effect has been illustrated in Fig. 4. A circle is drawn using the MS paint application and is processed using the aforementioned algorithm. The output of the algorithm is seen in Fig 4b.



(a) (b)

Figure 4. A filled circle submitted to the algorithm; (a) The color-converted circle; (b) After application of the algorithm



(a) (b)

Figure 5. The '+' crosshair is the point identified by the algorithm. After applying the correction factor δ, the point where the segments of the 'X' meet is returned as the absolute point of contact; (a) The finger is placed parallel to the glass surface and touching it; (b) The finger is tilted right

The resultant digital image is then scanned left to right to find the point of contact. The first instance of black pixel is returned. However, this point would be the edge of the finger or the topmost point of the fingertip. To account for this error in blob resolution, a value γ is added to the y coordinate as a correction factor. Since the image is at a relatively low resolution, the value of γ is roughly constant for almost any finger size. A value of γ is set such that it works tolerably for both oblique and straight touches. This can be seen in Fig 5. The straight line drawn from the point of contact that would be identified has a length equal to γ. Thus for this value, the error rate in touch would be minimum.

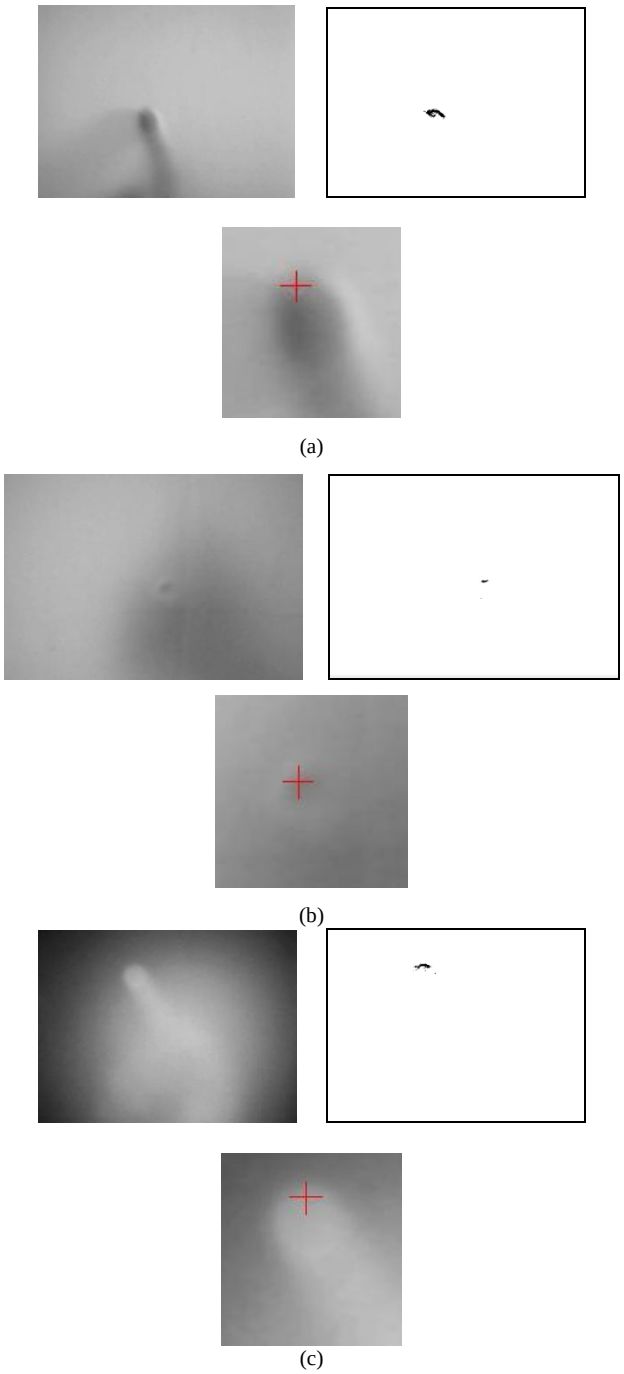
V. Results

The algorithm was implemented in Java programming language. It is possible to combine colour conversion, segmentation and digitalisation steps such that they are executed in a single scan of the entire source image. However, there has to be a separate loop to treat the resultant digital image as shown in the previous section.

The results for various experiments conducted have been shown in the next section.

The output of the tip detection algorithm, along with the points of contact identified (circled in green) is shown in Fig. 6.

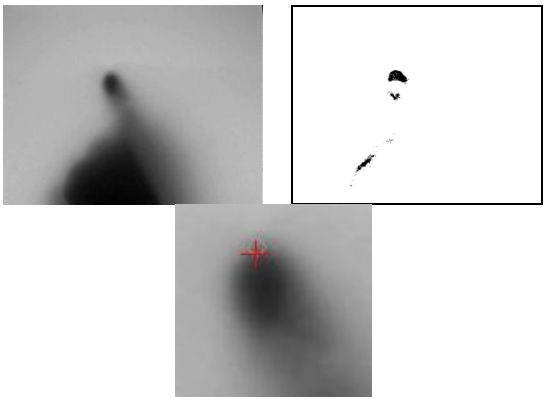
Since the fingertip always forms a lower circular arc, it is detected by the algorithm. This means that if the finger were to be placed sideways on the screen, it would detect 2 separate finger tips, rather than one single tip. This is because the actual fingertip is not detected at all. The sides of the finger are treated as the tips and these portions are registered in the digital image. This is shown in Fig. 7.



(a)

(b)

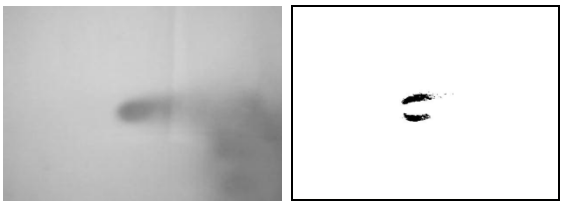
(c)



(d)

Figure 6. The source image, the digitalized image and the zoomed image with the point of contact as computed by the presented algorithm as seen in (a) Daylight, oblique touch; (b) Daylight, vertical touch (fingernail); (c) Poor lighting with LEDs switched on; (d) Artificial lighting (tube light)

Another point to be noted is that since the algorithm is capable of detecting finger tips of any size, at any angle, it can also effectively detect multiple-fingertip blobs in the same frame. This means that the touch screen is capable of responding to multi-touch. However, the steps to treat a multi-touch input sequence are not described in this paper.



(a) (b)

Figure 7. Finger placed sideways on the screen; (a) The color converted image obtained from the camera; (b) The resultant digital image

VI. Conclusion and Future Work

The success rate of the algorithm is very high, and was tried and tested for several user inputs. The point of contact is detected accurately in several lighting scenarios for all user inputs. A slight reduction in accuracy was observed when the external light source was placed very close to the box. However this can be corrected with another algorithm in future versions, without the actual need for additional hardware.

The image processing touch screen has a wide scope in application. Due to its bulk, it perhaps will not replace the mouse or keyboard as the primary input device, nor could it be installed in mobile phones. However, it can be introduced as an advanced, easy-to-use gaming console, an auxiliary to a personal computer.

Multi touch features can also be added to the touch screen without making any changes to the setup. A new algorithm would be required to look out for sequences of input spread across several frames captured, in order to increase the functionalities of this prototype. Also, applications such as a calculator, MS Paint, virtual keyboard can be developed specifically for the touch screen to introduce new features to these existing applications.

Apart from being an auxiliary device, it can be expanded to table-top computer, with a projector placed next to the web-cam. Now the touch surface will act as both the input and output device, which holds more promise. This is explored in [16]. Already, table-top computers are making their way to the computer and business industries. Image processing touch-screens will make a huge difference to their progress.

Instead of the wired web-cam that is being used in our current experiment, a wireless web-cam can be introduced. This would mean that the device can be moved around freely and provides greater flexibility.

References

[1] E. A. Johnson, "Johnson touch display," U.S. patent 3,482,241, Dec. 2, 1969.

[2] E. A. Johnson, "Touch displays: A programmed man-machine interface," *Ergonomics*, vol. 10, no. 2, pp. 271–277, 1967.

[3] G. Roma and A. Xambo, "A tabletop waveform editor for live performance," In *Proc 8th International Conference, New Interfaces for Musical Expression*, Genoa, Italy 2008.

[4] M. Haas, "The HP 150 computer," Nov. 1984, pp. 262 – 275.

[5] W. A Clough, D. Ouelette, and S. D. L. Sablonniere, "Portable computer with touch-screen and computing system employing same", U.S. Patent 5,675,362, Oct. 7, 1997.

[6] D. L Bitzer and R. L. Johnson, "PLATO: a computer-based system used in the engineering of education," In *Proc. IEEE*, vol. 59, 1971, pp. 960 – 968.

[7] R. Chang, F. Wang, and P. F. You, "A survey on the development of multi-touch technology," *Wearable Computing Systems, Asia-Pacific Conference*, pp. 363 – 366, 2010.

[8] D. Yeliussizov, A. Tuyakbayev, and A. Akshabayev, "Multi-touch sensing using frustrated total internal reflection," *Application of Information and Communication Technologies International Conference*, 2009, pp. 1-4

[9] Ahsanullah, A. K. B. Mahmood, and S. Sulaiman, "Investigation of fingertip blobs on optical multitouch screen," *Information Technology International Symposium*, 2010, pp. 1-6.

[10] Ahsanullah, A. K. B. Mahmood, and S. Sulaiman,; "Design and implementation of multi-touch system using FTIR technique for optimization of finger touch detection", *Information Technology International Symposium*, 2010, pp. 1 – 6

[11] F. Wang, X. S. Ren, and Z. Liu*;* "A robust blob recognition and tracking method in vision-based multi-touch technique", in *Proc. International Symposium Parallel and Distributed Processing with Applications*, 2008, pp. 1-5

[12] S. S. A. Sheikh, S. M. Hanana, Y. Al-Hosany, and B. Soudan, "Design and implementation of an FTIR camera-based multi-touch display," *GCC Conference & Exhibition*, 2009 5th IEEE, pp. 1-6.

[13] K S Srinivas, B S Usha, S Sandya, and Sudhir Rao R, "Modelling of edge detection and segmentation algorithm for pest control in plants," in *Proc. International Conference in Emerging Trends in Engineering*, Karnataka, India, 2011

[14] R. C Gonzalez and R. E Woods, "Digital image processing", 2nd ed., Prentice Hall, ch. 9–12.

[15] C. Saravanan, "Color image to grayscale image conversion," *in Proc. Second International Conference Computer Engineering and Applications* (ICCEA), 2010, pp. 196 –199.

[16] Z. Y. Ang, C. T. Yuen, T. Y. NG, Y. L. Ng, and J. H. Ho, "Development of a multi-touch table for natural user interface," in Proc. IEEE Conference Sustainable Utilization and Development in Engineering and Technology (STUDENT), 2011, pp. 201-206.

Prashanth C. Ravoor is currently pursuing his final year B.E in Information Science and Engineering in BMS College of Engineering, Bengaluru. His areas of interest include Java programming, C programming, image processing, game design and animation.

Sudhir Rao Rupanagudi is the founder and Managing Director of WorldServe Education, Bangalore–A non-profit Research and Development organization. He has guided over 400 undergraduate students in the fields of Digital FPGA and VLSI design. His major research interests include Digital Signal Processing and Image Processing. He has over 5 years of experience in Digital FPGA design. He received his B.E in Electronics and Communication from Atria Institute of Technology in 2006, his MS in System on Chip from Lunds Tekniska Hogskola (LTH), Sweden in 2008. He has also associated himself with the Indian Institute of Science (IISc), Bangalore as a research associate in Electronics and Communication. He has authored over 10 journals for various international conferences across the world.

Ranjani S is a senior e-commerce producer in a top MNC in Bengaluru. She has 5 years of industry experience as a web development engineer in companies such as INEP and Honeywell. She is well versed in web technologies, providing technical training and also conducting technical interviews