

A MULTI TOUCH WALL AND ITS APPLICATIONS

PURU GUPTA 081261

KUMAR NISHANT 081293

PRATIK SHARMA 081451

SUPERVISOR- MRS SHIPRA SHARMA (SR. LECTURER, DEPT. OF CSE & IT, JUIT)



May, 2012



SUBMITTED IN THE PARTIAL FULLFILLMENT OF THE DEGREE OF
BACHELOR OF TECHNOLOGY


DEPARTMENT OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY,
WAKNAGHAT

TABLE OF CONTENTS

CHAPTER NO.	TOPIC	PAGE NO.
	CERTIFICATE FROM THE SUPERVISOR	II
	ACKNOWLEDGEMENT	III
	SUMMARY	IV
	LIST OF FIGURES	V
	LIST OF SYMBOLS AND ACRONYMS	VII
1	INTRODUCTION	1
2	LITERATURE SURVEY	9
3	THE DESIGN AND CONSTRUCTION OF MULTI TOUCH DEVICE	12
4	MULTI TOUCH DETECTION AND PROCESSING	18
5	MULTIPLEXER MODULE	35
6	PYMT- OPEN SOURCE MULTI TOUCH DEVELOPMENT LIBRARY	38
7	TUIO- TANGIBLE USER INTERFACE OBJECT	47
8	APPLICATION WALKTHROUGH	51
9	SOFTWARE DEVELOPMENT LIFE CYCLE	76
10	CONCLUSION	80
11	BIBLIOGRAPHY	81

CERTIFICATE

This is to certify that the work titled "**A Multi Touch Wall and Its Applications**" submitted by "**Puru Gupta, Kumar Nishant and Pratik Sharma**" in partial fulfillment for the award of degree of "**Bachelor of Technology**" at the Department of Computer Science and IT, Jaypee University of Information Technology, Waknaghat, has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor 

Name of Supervisor **Mrs. Shipra Sharma**

Designation **Sr. Lecturer, Dept. of CSE & IT**

Date 

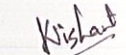
ACKNOWLEDGEMENT

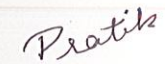
We take this opportunity to express our gratitude to the people who have been supportive in the successful completion of our project.

We want to express our sincere thanks to Mrs. Shipra Sharma, our project mentor for giving her valuable time and guidance. Constant motivation provided by her kept us focused throughout the project.

We would also like to convey our appreciation to our project coordinator, Dr. Nitin and Head of Department, Brig (Retd) SP Ghrera for giving us support as and when required.


Puru Gupta


Kumar Nishant


Pratik Sharma

Date:- 31/5/2012

SUMMARY

The primary objective of our project is to develop a multi touch screen (wall). The user can directly interact with it using his/her hands and thus eliminating the use of any intermediary input device like a mouse or keyboard. User's hand can be detected by the wall to generate actions which control the processing of the connected computer. The movements made by user are provided as input to the system using optical multi touch technology; this technology uses a combination of infrared lasers and an infrared detecting web camera to locate the position of touch on the screen. The camera's video feed is passed to the image processing pipeline for removal of noise. The detected blobs are tracked and their information is sent to the various multi touch application (developed on PYMT) using TUIO protocol. The project is further extended by developing a multiplexer so that we have a distributed framework. To conclude, observing both the market and research in this field, touch technology is the future of Human Computer Interaction.

LIST OF FIGURES

FIGURE NUMBER	FIGURE DESCRIPTION
1.1	Capacitive Multi Touch Technology
1.2	Resistive Multi Touch Technology
1.3	Frustrated Total Internal Reflection
1.4	Diffused Illumination
1.5	Diffuses Surface illumination
1.6	Laser Light PLane
1.7	LED- Light Plane
3.1	IR Lasers
3.2	Line Generator Lens
3.3	Adapter
3.4	Modification of Web camera
3.5	Web camera
4.1	Software Pipeline
4.2	Greyscale Conversion
4.3(a)	Narrow Gaussian filter
4.3(b)	Wider Gaussian filter
4.3(c)	Resultant Band pass Gaussian filter
4.4	Input image for filter comparison
4.5	Output of Lowpass butterworth filter
4.6	Output of Highpass butterworth filter
4.7	Output of Lowpass ideal filter
4.8	Output of Highpass ideal filter
4.9	Output of Lowpass gaussian filter
4.10	Output of Highpass gaussian filter
4.11	Mesh of Lowpass Butterworth Filter
4.12	Image of the Lowpass butterworth filter kernel in frequency domain
4.13	Mesh of Highpass Butterworth Filter
4.14	Image of the Highpass butterworth filter kernel in frequency domain
4.15	Mesh of Lowpass ideal Filter
4.16	Image of the Lowpass Idea; filter kernel in frequency domain
4.17	Mesh of Highpass IdealFilter
4.18	Image of the Highpass Ideal filter kernel in frequency domain
4.19	Plot of Gaussian 1-D function
4.20	Plot of Gaussian 2-D function
4.21	Integer valued kernel
4.22	1-D component of kernel
4.23	Mesh of Lowpass Gaussian Filter
4.24	Image of the Lowpass Gaussian filter kernel in frequency domain
4.25	Mesh of Highpass Gaussian Filter
4.26	Image of the Highpass Gaussian filter kernel in frequency domain
4.27	Barycentric Coordinate system
5.1	Schematic Diagram for Distributed system

5.2	Global and local coordinates
5.3	Tuio value distribution
5.4	Need for tracking in multiplexers
6.1-6.4	Pymt paint application
6.5-6.8	Pymt picture viewer application
6.9-6.18	Pymt calculator application
7.1	Working of TUIO protocol
8.1-8.24	Application Walkthrough
9.1	Evolutionary Model
9.2	Spiral model
9.3	Use Case Diagram

LIST OF ACRONYMS

FTIR- Frustrated Total Internal Reflection

DI- Diffused Illumination

LCD- Liquid Crystal Display

DSI- Diffused Surface Illumination

LLP- Laser Light Plane

LED- Light Emitting Diode

LED-LP- LED Light Plane

IR- Infrared

GUI- Graphical User Interface

NIR- Near IR region

USB- Universal Serial Bus

VGA- Video Graphics Array

JPEG- Joint Photographic Expert Group

OLED- Organic LED

BSD- Berkeley Software Distribution

IDE- Integrated Development Environment

PYMT- Python for Multi touch

TUIO- Tangible User Interface Object

OSC- Open Sound Control

UDP- User Datagram Protocol

TCP- Transmission Control Protocol

RGB- Red Green Blue

PSNR- Peak Signal to Noise Ratio

MSE- Mean Squared Error

1. INTRODUCTION

1.1 Introduction

Multi-touch displays are interactive devices that combine camera and image processing technologies for direct on-screen input. Multi-touch technology is not a very new technology. Since 1970s it has been around in different forms. Due to the improvement of processing power of modern desktop computers, multi-touch technology's no longer requires expensive equipment. Modern computer interaction mostly consists of a monitor, keyboard and a mouse. Limited by the operating system, the traditional approach allows the user to control a single pointer on the screen. With multi-touch, multiple input pointers are introduced to the system which can be controlled independently.

1.2 Existing multi-touch technologies

1.2.1 Capacitive touch

A Capacitive touch screen consists of an insulator such as a glass, coated with a conductor like Tin oxide. As human body is a conductor, on touching the screen the electrostatic field gets distorted which is registered as a touch and then the position is sent to the controller for processing. Unlike a resistive touch screen, one cannot use a capacitive touch screen with electrically insulating material, such as gloves; one requires a special capacitive stylus, or a special-application glove with an embroidered patch of conductive thread passing through it and contacting the user's fingertip.

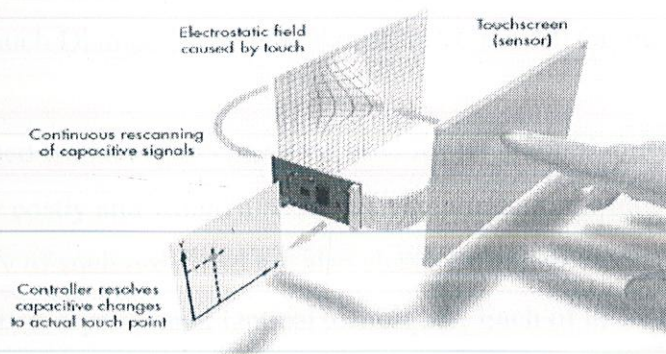
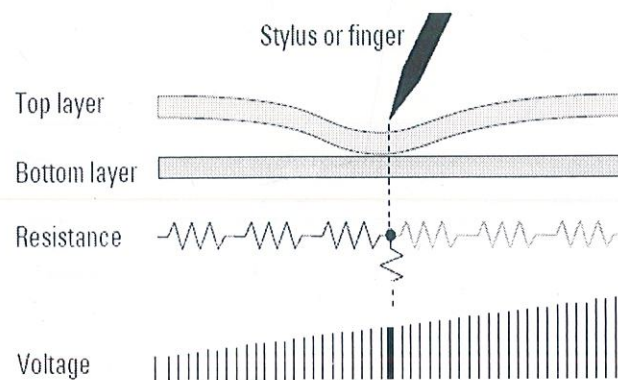


Figure 1.1

Example:- Products from Apple® are based on capacitive touch.

1.2.2 Resistive Touch

The screen is made up several layers, most important of which are two layers that are very thin and have very little gap between them. The top screen has a resistive layer which has a coating on the underside. Just beneath it is a similar resistive layer on top of its substrate. One layer has conductive connections along its sides, the other along top and bottom. A voltage is passed through one layer, and sensed on the other. When an object, such as a fingertip or stylus tip, presses down on the outer surface, the two layers touch to become connected at that point: The panel then behaves as a pair of voltage dividers, one axis at a time. By rapidly switching between each layer, the position of a pressure on the screen can be read. Resistive touch is used in restaurants, factories and hospitals due to its high resistance to liquids and contaminants. A major advantage of resistive touch technology is its low cost. Disadvantages include the need to press down and a risk of damage by sharp objects. Resistive touch screens also suffer from poor contrast. This is because they have additional reflections from the extra layer of material placed over the screen.



Example:- HTC Touch Diamond® and the Samsung SGH-i900 Omnia® use resistive touch screens.

The above mentioned technologies cannot be used for large multi touch table top surfaces as they would be very costly and would also have a size limitation. Multi-touch technologies that offer scalability of such order and are also cheap are called Optical Multi-Touch technologies. We have 5 prominent Optical techniques. Each of these techniques consists of an optical sensor (typically a camera), infrared light source, and visual feedback in the form of projection or LCD. Brief description of the 5 techniques is given below:

1.2.3 Frustrated Total Internal Reflection (FTIR)

FTIR is a name used by the multi-touch community to describe an optical multi-touch methodology developed by Jeff Han. It is based on a well known optical phenomenon called total internal reflection. Total Internal Reflection describes a condition, present in certain materials. When light enters one material from another material with a higher refractive index, at an angle of incidence greater than a specific angle. Han's method uses this to great effect, flooding the inside of a piece of acrylic with infrared light by trapping the light rays within the acrylic using the principle of Total Internal Reflection. When the user comes into contact with the surface, the light rays are said to be frustrated, since they can now pass through and into the contact material (usually skin), also the reflection is no longer total at that point. [Fig. 2] This frustrated light is scattered downwards towards an infrared webcam, capable of picking these 'blobs' up, and relaying them to the tracking software.

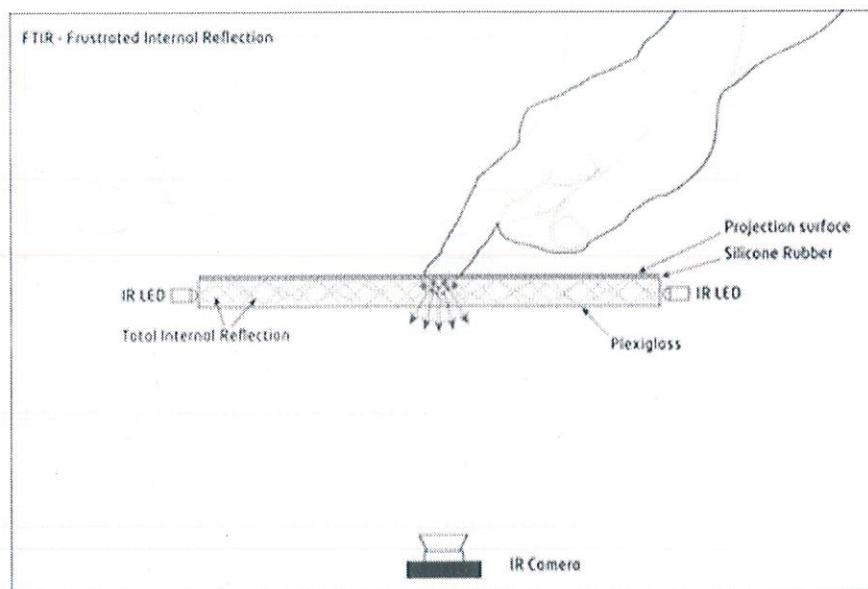


Figure 1.3

1.2.4 Diffused Illumination (DI)

Diffused Illumination (DI) is possible in two forms: Front Diffused Illumination and Rear Diffused Illumination. Both techniques rely on the same basic principles - the contrast between the silent image and the finger that touches the surface.

Front Diffused Illumination: Visible light (often from the ambient surroundings) is shone at the screen from above the touch surface. A diffuser is placed on top or on bottom of the

touch surface. When an object touches the surface, a shadow is created in the position of the object. The camera senses this shadow.

Rear Diffused Illumination: Infrared light is focused at the screen from below the touch surface. A diffuser is placed on top or on bottom of the touch surface. When an object touches the surface it reflects more light than the diffuser or objects in the background; the extra light is sensed by a camera. Depending on the diffuser, this method can also detect hover and objects placed on the surface.

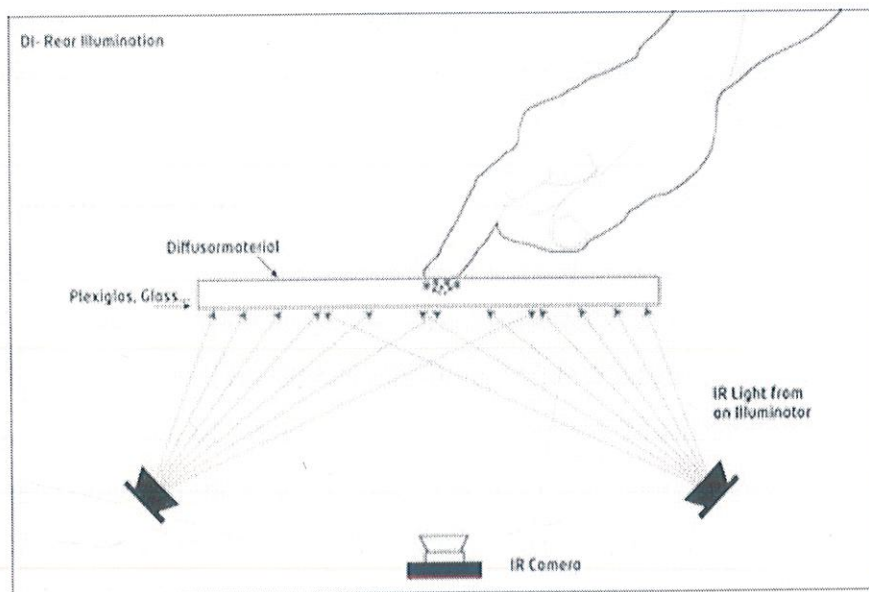


Figure 1.4

1.2.5 Diffused Surface Illumination (DSI)

DSI uses a special acrylic to distribute the IR evenly across the surface. Basically it uses standard FTIR setup with an LED Frame (no compliant silicone surface needed), and a special acrylic. This acrylic uses small particles that are inside the material, acting like thousands of small mirrors. When you shine IR light into the edges of this material, the light gets redirected and spread to the surface of the acrylic.

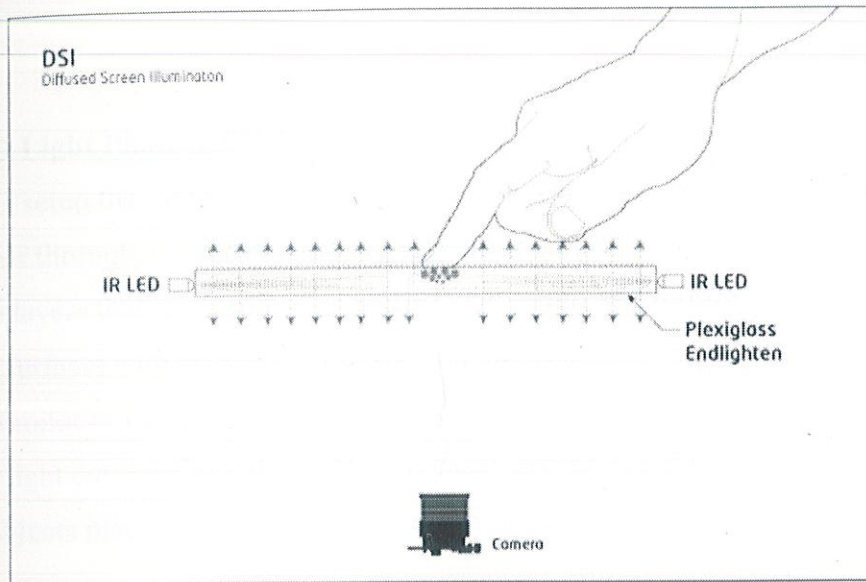


Figure 1.5

1.2.6 Laser Light Plane (LLP)

Infrared light from a laser(s) is passed just above the surface. The laser plane of light is about 1mm thick and is positioned right above the surface, when the finger just touches it, it will hit the tip of the finger which will register as a IR blob. Infrared lasers are an easy and usually inexpensive way to create a MT setup using the LLP method. Most setups go with 2-4 lasers, positioned on the corners of the touch surface. The laser wattage power rating (mW, W) is related to the brightness of the laser, so the more power the brighter the IR plane will be. The common light wavelengths used are 780nm and 940nm as those are the wavelengths available on the Aixiz.com® website where most people buy their laser modules. Laser modules need to have line lenses on them to create a light plane. The 120 degree line lens is most commonly used, so as to reduce the number of lasers necessary to cover the entire touch surface.

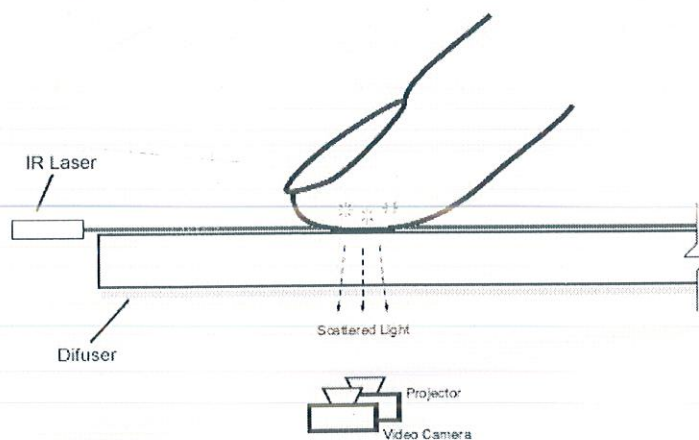


Figure 1.6

1.2.7 LED Light Plane (LED-LP)

LED-LP is setup the same way as an FTIR setup except that the thick acrylic that the infrared light travels through is removed and the light travels over the touch surface. This figure shows the layers that are common in an LED-LP setup. The infrared LEDs are placed around the touch surface; with all sides being surrounding preferred to get a more even distribution of light. Similar to LLP, LED-LP creates a plane of IR light that lays over the touch surface. Since the light coming from the LEDs is conical instead of a flat laser plane, the light will light up objects placed above the touch surface instead of touching it.

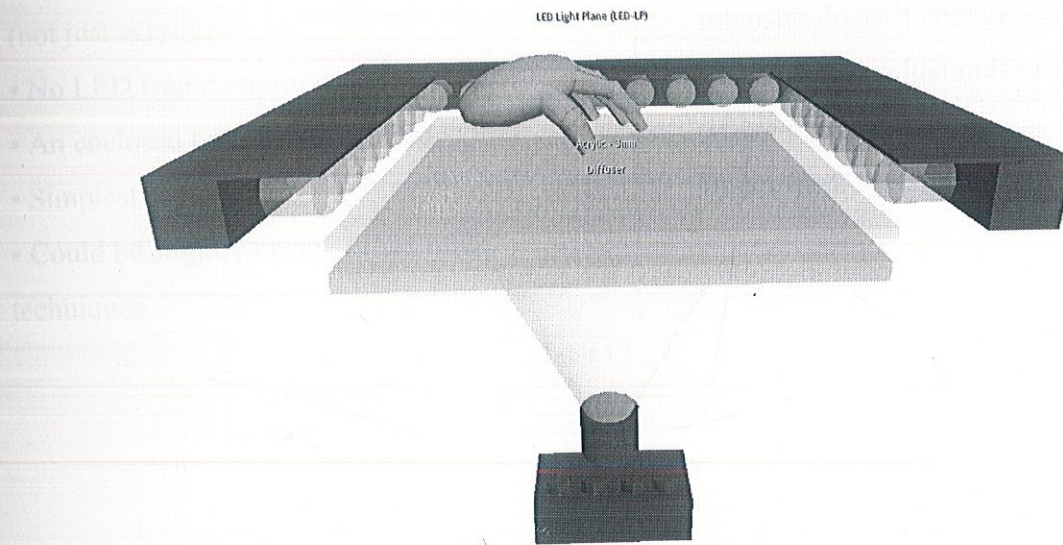


Figure 1.7

1.3 Comparison between various techniques

Every technique described above offers advantages and disadvantages over the other techniques. Therefore there is no single best technique. The answer is less about what is best and more about what is best for the user. For this project we have chosen LLP technique which has the following advantages and disadvantages:-

Advantages	Disadvantages
<ul style="list-style-type: none">• No compliant surface (silicone)• Can use any transparent material like glass (not just acrylic)• No LED frame required• An enclosed box is not required• Simplest setup• Could be slightly cheaper than other techniques	<ul style="list-style-type: none">• Cannot track traditional objects and fiducials• Not truly pressure sensitive (since light intensity doesn't change with pressure).• Can cause occlusion if only using 1 or 2 lasers where light hitting one finger blocks another finger from receiving light.

Table 1.1

1.4 Overview of this work

Recent publications have shown that multi-touch displays have a potential to revolutionize human-computer interaction in the way that they allow intuitive interaction with applications through direct manipulation of graphical constructs. Several research groups have demonstrated that the performance of simple tasks through multi-touch displays show great improvement over conventional methods. Unfortunately, most interactive graphical applications that exist today are not able to exploit this potential because they are designed for a single pointer device such as the mouse.

1.5 Objective and Scope

1.5.1 Objective

The main objective of the project was to construct a multi-touch wall which would have the capability to recognize human touches as specific actions. First the camera captures an image from the wall through which the touch details are extracted. Then these touch details are transmitted to the GUI based applications which create actions specific to the touch events.

1.5.2 Scope

The touch wall has varied scope with its implementations in following fields:

1. **Markets:** Touch walls can be implemented in markets to provide users with interactive menu and video description of products. User can also order products from these walls and thus reduce rush over the counters.
2. **Teaching Halls:** Touch walls can be used for teaching students in an interactive way. By installing touch walls teachers can use the computer functions on the touch-screen directly. For example- performing calculations, using applications etc.
3. **Table top surfaces:** Touch screens can also be used as table top surfaces at board rooms for interactive file transferring and sharing purposes. These tables would allow all the members to analyze the scenario more appropriately.

2. LITERATURE SURVEY

2.1 Literature

- Seeing through the Fog: An Algorithm for Fast and Accurate Touch Detection in Optical Tabletop Surfaces Christopher Wolfe, T.C. Nicholas Graham, and Joseph A. Pape School of Computing, Queen's University, Kingston, Ontario, Canada fwolfe, graham, papeg@cs.queensu.ca.

This research paper explains working of a touch detection algorithm, the authors propose a novel algorithm for applying a bandpass filter on an image in $O(p)$ i.e. linear complexity. They compare their algorithm with other noise reduction approaches based on the computation costs. Their algorithm basically focuses on noise reduction particularly in noisy environments without imposing high computational cost.

- Interfacing the TCM8230MD CMOS Camera With an ARM7 Microcontroller by Justin L. Shumaker

This project made us aware about how to capture and process 128 x 96 pixel images from the camera and vary the capturing frames per second, the frames per second rate helps in stabilizing the attitude of the robotic platform during stabilization. It also explains the interfacing of CMOS camera with microcontroller.

- Edge Detection in Digital Images Using Fuzzy Logic Technique Abdallah A. Ishennawy, and Ayman A. Aly

The authors of this paper explain the fuzzy logic technique and introducing of fuzzy operators on hand of a computer vision application. They have proposed a method for edge detection based on fuzzy logic technique which can be used in digital images without determining the threshold value.

- International Journal of Computer Science & Communication Vol. 1, No. 2, July-December 2010, pp. 387-390 Fuzzy Edge Filter-Edge Detection and Feature Extraction Technique for Any JPEG Image Neha Agrawal¹, Rishabh Agrawal² & Sushil Kumar¹. This paper highlights the ways to detect edges at five different levels which will help in pattern recognition and feature extraction with the use of fuzzy logic a method well known in Artificial intelligence. Their fuzzy edge filters detect different classes of image pixels corresponding to grey level variations in various directions.

- Robust Real-Time Tracking of Non-rigid Objects Richard Y. D. Xu, John G. Allen, Jesse S. Jin School of Information Technologies University of Sydney, NSW 2006, Australia {richardx,jallen,jesse}@it.usyd.edu.au

This research document gives a novel real time video -object tracking algorithm. The authors explain that after the detection of any object how can it be tracked for its movements on the basis of its color and the color of its surrounding objects. The tracking is performed on the objects in a sequence of video frames based on user selected initial frame.

- Adaptive background mixture models for real-time tracking Chris Stauffer W.E.L. Grimson The Artificial Intelligence Laboratory Massachusetts Institute of Technology Cambridge,MA 02139

The authors explain that for real time segmentation of moving regions in image sequences involves background subtraction, there are numerous approaches to this problem and the authors discusses about modeling of each pixel as a mixture of Gaussians and using an online approximation to upgrade the model. Each pixel is classified based on the Gaussian distribution whether it belongs to foreground or not.

- In-place Algorithm for Connected Components Labeling Tetsuo Asano tsano@jaist.ac.jp School of Information Science, JAIST, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan Hiroshi Tanaka gja164@jaist.ac.jp School of Information Science, JAIST, 1-1 Asahidai, Nomi, Ishikawa 923-1292, Japan Journal of Pattern Recognition Research 1 (2010) 10-22 Received January 18, 2010. Accepted July 1, 2010.

The authors present two linear time algorithm on connected component labeling of a binary image. The first method reports the number of connected components and the second one is for labeling every element of value 1 (or white pixel) by its component index. The algorithm has been implemented without using any extra array other than the input array.

- Applied Mathematical Sciences, Vol. 5, 2011, no. 35, 1727 – 173 Neighbourhoods, Classes and Near Sets Christopher J. Henry Department of Applied Computer Science University of Winnipeg Winnipeg, Manitoba R3B 2E9, Canada ch.henry@uwinnipeg.ca The article calls attention to the relationship between neighbourhoods and tolerance classes in the foundations of tolerance near sets. A particular form of tolerance relation is

given by way of introduction to descriptively ϵ -near sets. Neighbourhoods and tolerance classes have practical applications in digital image analysis.

- Kaltenbrunner, M., Bovermann, T., Bencina, R., Costanza, E.: "TUIO - A Protocol for Table-Top Tangible User Interfaces". Proceedings of the 6th International Workshop on Gesture in Human-Computer Interaction and Simulation (GW 2005), Vannes, France, 2005

In this article the authors present TUIO, a simple protocol designed to meet the requirements of a table top tangible user interface. The authors explain the tracking of finger-tips for gesture control within the table interface.

- Kaltenbrunner, M., Bencina, R.: "reactIVision: A Computer-Vision Framework for Table-Based Tangible Interaction". Proceedings of the first international conference on "Tangible and Embedded Interaction" (TEI07). Baton Rouge, Louisiana, 2007

The article gives an overview of the reactIVision framework- an open source cross platform computer vision framework primarily designed for the construction of table based tangible user interfaces. The article also provides key points relevant to the construction of table hardware.

- Jorda, S., Geiger, G., Alonso, A., Kaltenbrunner, M.: "The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces". Proceedings of the first international conference on "Tangible and Embedded Interaction" (TEI07). Baton Rouge, Louisiana, 2007

The author first discusses the reasons for which live music performance and Human computer Interaction, in general, and musical instruments and table top surfaces in particular. The authors present the reacTable, a musical instrument based on a table top interface.

- Wright, M., Freed, A., Momeni A.: "OpenSound Control: State of the Art 2003". Proceedings of the 3rd Conference on New Instruments for Musical Expression (NIME 03), Montreal, Canada, 2003.

The authors of this paper present Open sound control (OSC) is a protocol for communication among computers, sound synthesizers and other devices, the protocol is optimized for modern network technology.

3. THE DESIGN AND CONSTRUCTION OF A MULTI-TOUCH DEVICE

3.1 Hardware Requirements

These requirements are based on the LLP setup

3.1.1 Infrared Light Source

Infrared (IR in short) is a portion of the light spectrum that lies just beyond what can be seen by the human eye. It is a range of wavelengths longer than visible light, but shorter than microwaves. 'Near Infrared' (NIR) is the lower end of the infrared light spectrum and typically considered wavelengths between 700nm and 1000nm (nanometers). Most digital camera sensors are also sensitive to at least NIR and are often fitted with a filter to remove that part of the spectrum so they only capture the visible light spectrum. By removing the infrared filter and replacing it with one that removes the visible light instead, a camera that only sees infrared light is created. For the LLP setup lasers will be required which emit light of wavelength residing in 'Near Infrared' region.

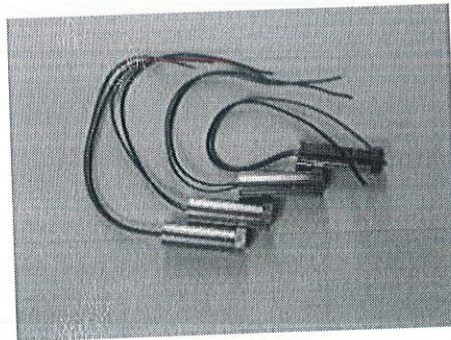


Figure 3.1

Different laser modules available are:

- 780nm 25mW Infrared Laser
- 780nm 80mW Infrared Laser
- 850nm 5mW Infrared Laser
- 850nm 10mW Infrared Laser
- 850nm 28mW Infrared Laser
- 880nm 10mW Infrared Laser

The laser used in our project has the following specifications:

- Wavelength: 780nm
- Power/Intensity: 25mW
- Voltage: 3.2VDC
- Max Current: 30mA
- Dimensions: 12x30mm Laser Module

3.1.2 Line Generating Lens

A line generating lens is required to spread the laser light so that it covers the complete surface. The line generator is thread onto the lasers to create a plane of laser light needed to make a LLP infrared optical-based multi touch setup. This line generator is composed of two parts, a grooved piece of acrylic which converts the laser dot to a line and a outer piece which keeps it attached onto the laser. This lens is fixed in front of the laser to widen the spectrum of the laser.

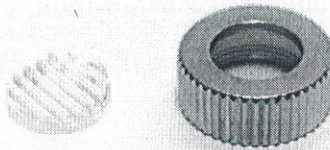


Figure 3.2

Different lenses available are:

- Laser Line Generator – 89 Degree
- Laser Line Generator – 120 Degree

The lens used in this project is 120 degree line generator.

3.1.3 Laser Power Adapter

This adapter can power the lasers. Each laser draws at most 30mA of current, so one can safely power up to about 16 individual lasers ($16 \times 30\text{mA} = 480\text{mA}$).

It has a 2 prong (non-ground) USA plug, and alligator clips to attach to the lasers. One can cut the clips off and solder to your laser circuit if needed.



Figure 3.3

Specifications:

Voltage Input: 110-240VAC

Voltage Output: 3.2VDC

Max Current: 500mA

Frequency: 50/60Hz

3.1.4 Infrared Cameras

Simple webcams work very well for multi-touch setups, but they need to be modified first. Regular webcams and cameras block out infrared light, letting only visible light in. The opposite is required. Typically, by opening the camera up, we can simply pop the filter off and you the camera as per our need. Most cameras will show some infrared light without modification, but

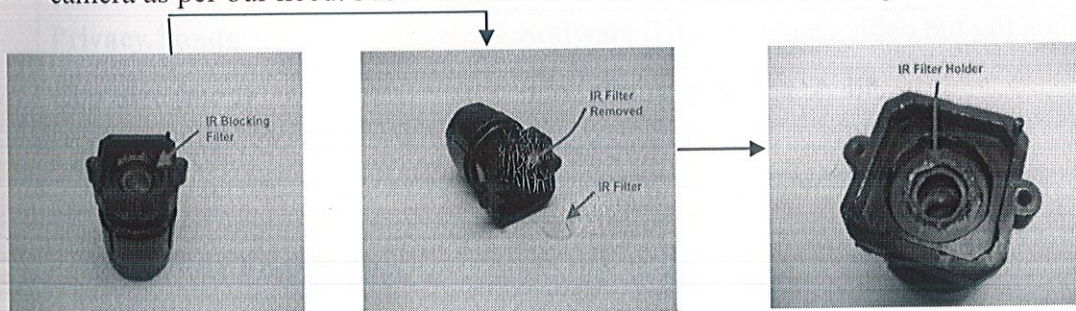


Figure 3.4



Figure 3.5

much better performance can be achieved if the filter is replaced. The Logitech webcam was modified as per the needs and used for the setup.

Specifications:

Connection Type	Corded USB
USB Type	High Speed USB 2.0
USB VID_PID	VID_046D&PID_0817

Microphone	N/A
Lens and Sensor Type	Plastic, CMOS
Focus Type	Fixed
Field of View (FOV)	60°
Focal Length	N/A
Optical Resolution (True)	640x480 VGA
Image Capture (4:3 SD)	320x240, 640x480 (JPG - True) 1.3 MP (1280x1024 (JPG - Software Enhanced)
Image Capture (16:9 W)	N/A
Video Capture (4:3 SD)	320x240, 640x480 (WMV - True) 1280x1024 (WMV - Software Enhanced)
Video Capture (16:9 W)	N/A
Frame Rate (max)	30fps @ 640x480 (Hardware Limit)
Video Effects (VFX)	Fun Filters
Right Light	Right Light 1
Buttons	Snapshot
Indicator Lights (LED)	Activity/Power
Privacy Shade	Software (NOTE: Mutes video but not audio)
Clip Size (max)	0 to infinity, Not Detachable
Cable Length	6 Feet or 2 Meters

Table 3.1

3.1.5 Clear Flat Surface

Glass, plexiglass can be used to prepare the surface for the multi-touch wall.

3.1.6 Projector

Using projector is one of the methods to display visual feedback on the screen (touch wall).

Any video projection device (LCDs, OLEDs, etc.) should work but projectors tend to be the most versatile and popular in terms of image size.

3.2 Software Requirements

3.2.1 OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions for real time computer vision. OpenCV is released under a BSD license; it is free for both academic and commercial use. It has C++, C and Python and soon Java interfaces running on Windows, Linux, Android and Mac. The library has >2500 optimized algorithms. Applications range from interactive art, to mine inspection, stitching maps on the web and also advanced robotics.

3.2.2 MATLAB

MATLAB is a programming environment for algorithm development, data analysis, visualization, and numerical computation. Using MATLAB, we can solve technical computing problems faster than with traditional programming languages, such as C, C++, and FORTRAN. MATLAB can be used in a wide range of applications, including signal and image processing, communications, control design, test and measurement, financial modeling and analysis, and computational biology.

3.2.3 Linux Operating Systems

Linux is a Unix-like computer operating system assembled under the model of free and open source software development and distribution. The defining component of Linux is the Linux kernel, an operating system kernel.

3.2.4 Code::Blocks (Open frameworks Library)

Code::Blocks is a free C++ IDE built to meet the most demanding needs of its users. It is designed to be very extensible and fully configurable. Code::Blocks is an IDE with varied features, having a consistent look, feel and operation across platforms. Built around a plugin framework, Code::Blocks can be extended with plugins. Any kind of functionality can be added by installing/coding a plugin. For instance, compiling and debugging functionality is already provided by plugins.

3.2.5 Pymt

Pymt is an opensource language for multi-touch application development. Python library is an opensource library; it has very updated APIs with users' active contribution. It provides

support for TUIO transmission; touch events, gesture recognition and many more touch related features.

3.2.6 TUIO Protocol

TUIO (Tangible User Interface Objects) is a simple yet versatile protocol designed specifically for the Touch tangible user interfaces. This protocol is used for communication between the program that is receiving the blob information and an application based on Action Scripts/Pynt which uses these blob's data to project some information. TUIO delivers binary OSC (Open Sound Control) data by sending UDP format packets via an open port and the sending part is done from the host C program.

4. MULTI-TOUCH DETECTION AND PROCESSING

Video Image Processing

To perform camera based multi-touch detection, the following steps are performed in the order depicted.

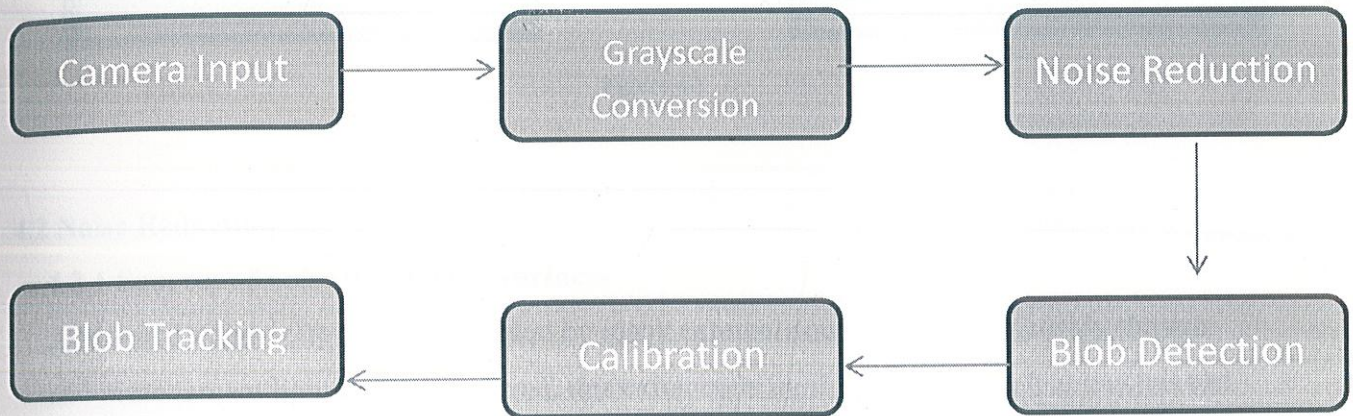


Figure 4.1

Each of the modules performs various functions ranging from capturing the video to removing the noise from the input image depending on the type of noise.

4.1 Grayscale Conversion

Combining of color channels reduces the camera noise to a large extent. The input color image from the camera is converted to grayscale for further reduction of noise. This helps us in removing excess noise, although this removes a lot of information from the input image. Also this step does not affect our scenario as our objects of interest in the image are white finger blobs. RGB to grayscale conversion is done by creating a new matrix from the individual R,G and B matrices by using the formula

$$\text{RGB}[A] \text{ to Gray: } Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

Figure 4.2 shows conversion of a input color image to grayscale (No noticeable change is seen visibly because the input image mostly contains black and white.)

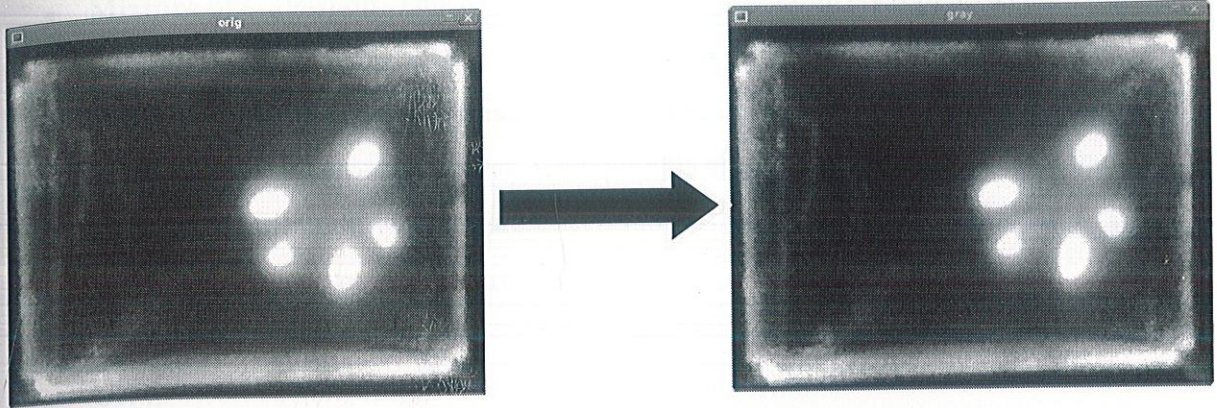


Figure 4.2

4.2 Noise Reduction

4.2.1 Sources of noise in tabletop surfaces

Ambient IR light- IR light is produced by many external sources, most noticeably the sun and incandescent lights. If not removed, this extraneous light can obscure actual touches and generate false touches. Example - sunlight, bulbs and tube light.

Camera noise- Commodity cameras modified for IR sensing introduce significant random variation in pixel values. Many cameras, particularly those communicating over USB, perform lossy compression of the video, adding artifacts that changes actual touches.

Shadows- As users move their hands and objects over the screen; they can produce shadows in the ambient IR light. These rapid changes in the background are more difficult to exclude than stable environmental light.

4.2.2 Steps involved in reducing noise

1. **Background Subtraction-** This method is used to identify blobs (touch) which are continuously moving. It removes the static noise and thus makes it easy to detect the blobs. Also it leads to removing of maximum static noise.

Algorithms available for Background Subtraction are:-

- **Simple Background subtraction-** A frame is chosen as the balance frame and absolute difference between the balance frame and every subsequent frame is calculated. It evidently works only in particular conditions of object's speed and frame rate.

- **Mixture of Gaussians**-This technique discusses modeling each pixel as a mixture of Gaussians and using an on-line approximation to update the model. The Gaussian distributions of the adaptive mixture model are then evaluated to determine the most likely to be i background process. Each pixel is classified based on whether the Gaussian distribution which represents it most effectively is considered part of the background model or the foreground model

Comparison between the two above mentioned algorithms.

Mixture of Gaussian	Simple Background subtraction
<input type="checkbox"/> Complex method.	<input type="checkbox"/> Relatively less Complex.
<input type="checkbox"/> Computationally heavy.	<input type="checkbox"/> Comparatively less heavy.
<input type="checkbox"/> Inacceptable output for tabletop surface in visual terms.	<input type="checkbox"/> Better visual output.

Table 4.1

Inference-By observing the differences between the two algorithms Simple Background Subtraction method is being used in this project.

2. **Image Filtering**- This is the second step involved in the noise reduction process. After background subtraction, most of the remaining noise consists of random camera noise and changed ambient lighting. The camera noise is either single pixels from its sensor, or distorted blocks introduced by its lossy compression. Both affect small areas and have sharp edges, so can be reduced by blurring the image. Changes to ambient lighting, such as the sun coming out from behind a cloud, are only visible through the table's semi-opaque top surface, so affect large areas and have much softer edges. The camera noise is considered high frequency noise and ambient light is considered low frequency noise. User touches lie in intermediate frequencies. Thus the standard approach used to detect the intermediate frequencies is to use a band pass filter. A band pass filter can be constructed by subtracting results of two Gaussian low pass filters. Figure4.3(c) shows a common 1-D mid-pass filter based on the

difference of two Gaussian filters. The narrower (first) filter in Figure 4.3(a) discards high frequencies, while the wider (second) in Figure 4.3(b) removes remaining low frequencies.

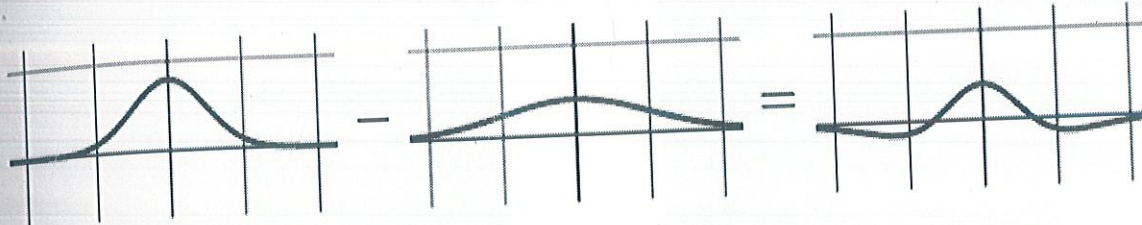


Figure 4.3(a)

Figure 4.3(b)

Figure 4.3(c)

4.3 Comparative Study of filtering techniques

To result out the best filter that should be used for the multi-touch application, PSNR was calculated of every resultant image with respect to the original image as shown below. All the computations have been performed by initially converting the image into frequency domain.

- **Lowpass Filters:** These filters are used to create a blurred or smoothed image; they attenuate the high frequencies and leave the low frequencies of the Fourier transform relatively unchanged.
- **Highpass Filters:** These filters are basically to block high frequency noise. Applying such a filter outputs the edges in the input image. Therefore these filters are used for edge detection as well.

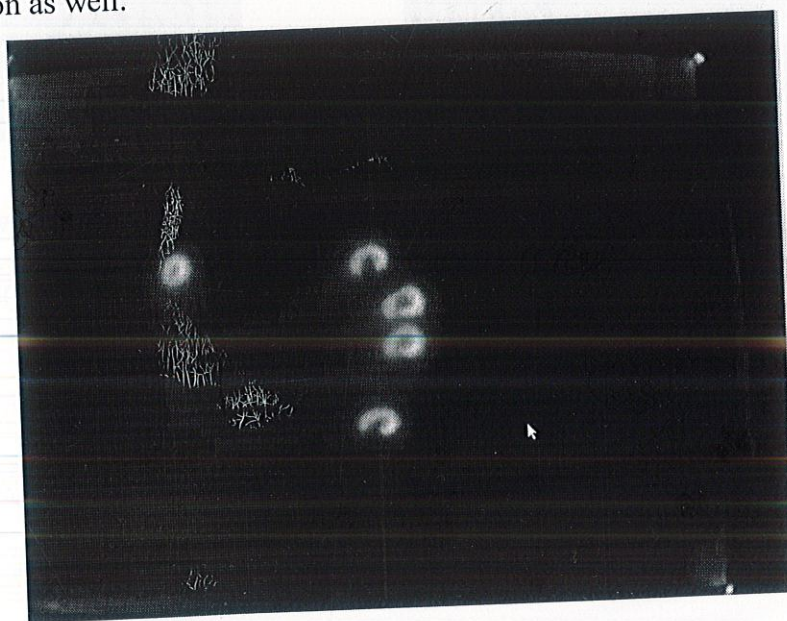


Figure 4.4 Original Image

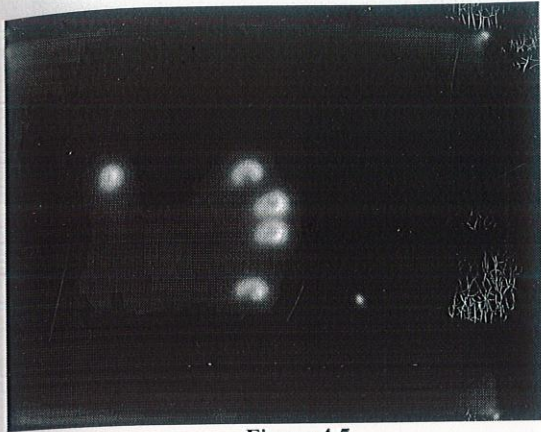


Figure 4.5

Lowpass_butterworth (PSNR=4.6423)

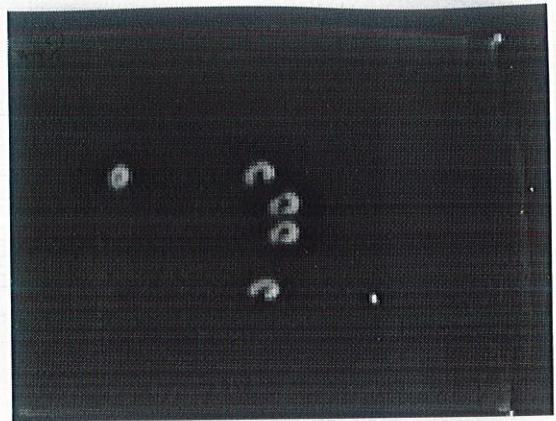


Figure 4.6

Highpass_butterworth (PSNR=1.4716)

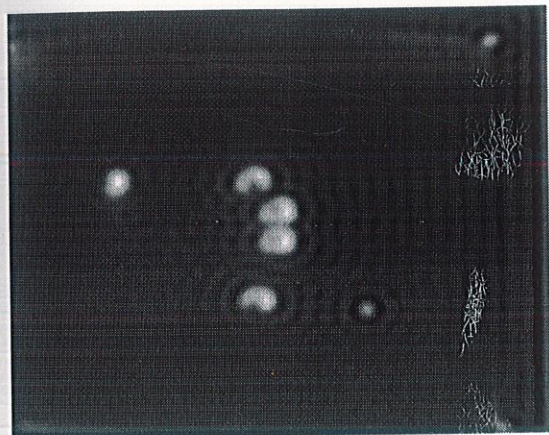


Figure 4.7

Lowpass_ideal(PSNR=4.6262)

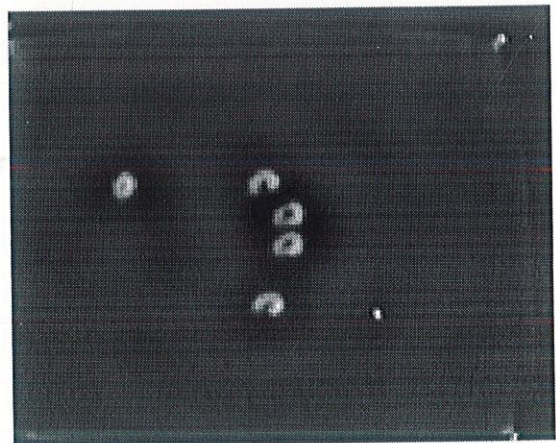


Figure 4.8

Highpass_ideal (PSNR=1.4490)

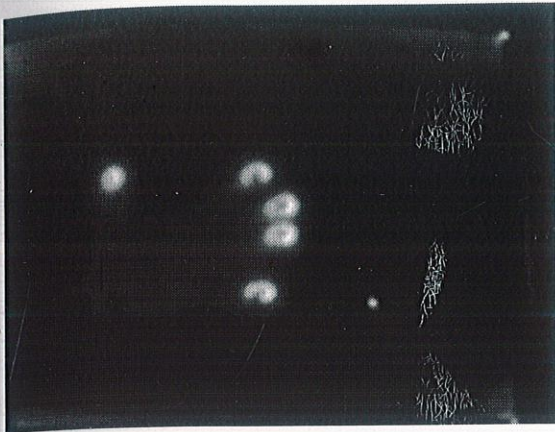


Figure 4.9

Lowpass_gaussian (PSNR=4.6431)

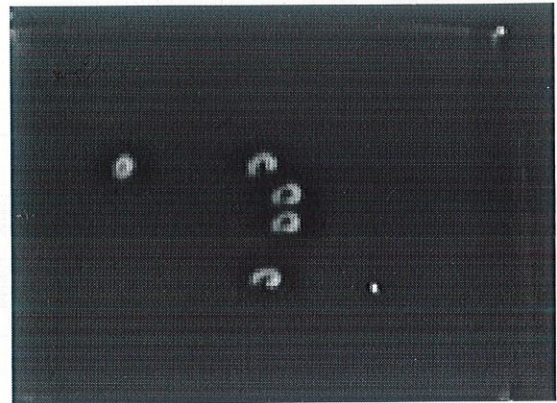


Figure 4.10

Highpass_gaussian (PSNR=1.4655)

4.3.1 Butterworth Filter

The classical method of analog filters design is Butterworth approximation. The Butterworth filters are also known as maximally flat filters. It is designed to have as flat a frequency response as possible in the pass band. Squared magnitude response of a Butterworth low-pass filter is defined as follows

$$|H(j, \omega)|^2 = \frac{1}{1 + \left(\frac{\omega}{\omega_{scale}} \right)^{2N}}$$

Where $\omega = 2 \cdot \pi \cdot f$ - radian frequency, $\omega_{scale} = 2 \cdot \pi \cdot f_{scale}$ - constant scaling frequency, N - order of the filter.

Some properties of the Butterworth filters are:

- $|H(j, \omega)|$ has a maximum at $\omega = 0$
- The first $2N-1$ derivatives of the equation are equal to zero at $\omega = 0$. This is why Butterworth filters are known as maximally flat filters.

LOWPASS

MESH

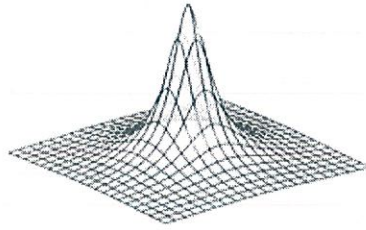


Figure 4.11

IMAGE

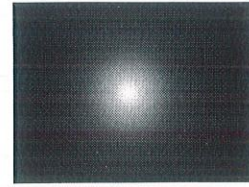


Figure 4.12

HIGHPASS



Figure 4.13

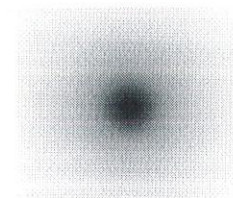


Figure 4.14

4.3.2 IDEAL FILTER

An ideal low-pass filter completely eliminates all frequencies above the cutoff frequency while passing those below unchanged: its frequency response is a rectangular function, and is a brick-wall filter. The transition region present in practical filters does not exist in an ideal filter. An ideal low-pass filter can be realized mathematically (theoretically) by multiplying a signal by the rectangular function in the frequency domain or, equivalently, convolution with its impulse response, a sinc function, in the time domain. However, the ideal filter is impossible to realize without also having signals of infinite extent in time. Hence it generally needs to be approximated for real ongoing signals, because the sinc function's support region extends to all past and future times. The filter would therefore need to have infinite delay, or knowledge of the infinite future and past, in order to perform the convolution. It is effectively realizable for pre-recorded digital signals by assuming

extensions of zero into the past and future, or more typically by making the signal repetitive and using Fourier analysis. The formula for an ideal lowpass filter is:-

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$$

Where D_0 is a specified nonnegative number and $D(u, v)$ is the distance from point (u, v) to the center of the filter.

LOWPASS

MESH

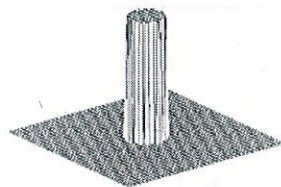


Figure 4.15

IMAGE

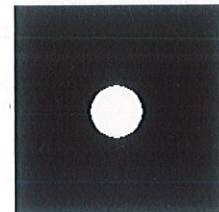


Figure 4.16

HIGHPASS



Figure 4.17



Figure 4.18

4.3.3 Gaussian Filter

The low-pass filters we focus on replace each pixel with a weighted average of its surroundings. The size of this surrounding region is the diameter of the filter. The Gaussian function provides a common weighting, because it removes high frequencies with minimal artifacts. Naively computing the weighted average for each pixel would require $O(p \times d \times d)$ time, where p is the number of pixels in the image, and d is the diameter of the filter. Fortunately, the Gaussian function is separable, meaning it can be applied as a 1-D function to rows and columns in turn. This reduces the complexity to $O(p \times d)$.

Working of Gaussian filter

The Gaussian smoothing operator is a 2-D convolution operator that is used to 'blur' images and remove minute detail and noise. The kernel is of the shape of a Gaussian ('bell-shaped') hump. The Gaussian distribution in 1-D defined as:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

where σ is the standard deviation of the distribution. The distribution has a mean of zero (*i.e.* it is centered on the line $x=0$). The distribution is illustrated as in figure 4.19

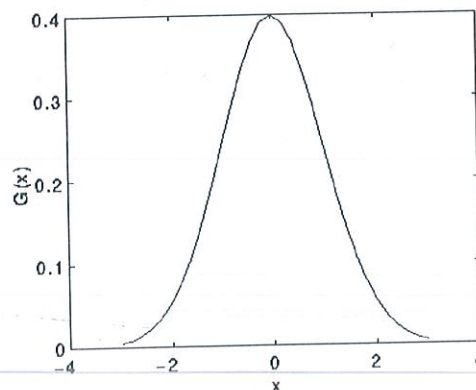


Figure 4.19

In 2-D, an isotropic (*i.e.* circularly symmetric) Gaussian has the form:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

This distribution is 2-D Gaussian distribution with mean (0,0) and $\sigma=1$. The distribution is illustrated in figure 4.20

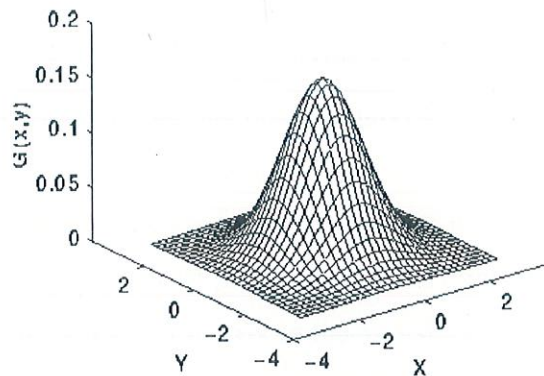


Figure 4.20

The idea of Gaussian smoothing is to use this 2-D distribution as a 'point-spread' function, and this is achieved by convolution. Since the image is stored as a collection of discrete pixels we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel, but in practice it is effectively zero more than about three standard deviations from the mean, and so we can truncate the kernel at this point.

Integer valued convolution kernel with a σ of 1.0.

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

Figure 4.21

Once a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution methods. The convolution can in fact be performed fairly quickly since the equation for the 2-D isotropic Gaussian shown above is separable into x and y components. Thus the 2-D convolution can be performed by first convolving with a 1-D Gaussian in the x direction, and then convolving with another 1-D Gaussian in the y direction. (The Gaussian is in fact the *only* completely circularly symmetric operator which can be decomposed in such a way). Figure 4.22 below shows the 1-D x component kernel that would be used to produce the full kernel shown in kernel above (after scaling by 273, rounding and truncating one row of pixels around the boundary because they mostly have the value 0. This reduces the 7×7 matrix to the 5×5 shown above). The y component is exactly the same but is vertical.

.006	.061	.242	.383	.242	.061	.006
------	------	------	------	------	------	------

Figure 4.22

The effect of Gaussian smoothing is to blur an image. The degree of smoothing is determined by the standard deviation of the Gaussian. (Larger standard deviation Gaussians, of course, require larger convolution kernels in order to be accurately represented.)

The Gaussian outputs a 'weighted average' of each pixel's neighborhood, with the average weighted more towards the value of the central pixels. One of the principle justifications for using the Gaussian as a smoothing filter is due to its *frequency response*. Most convolution-based smoothing filters act as lowpass frequency filters. This means that their effect is to remove high spatial frequency components from an image.

LOWPASS

MESH

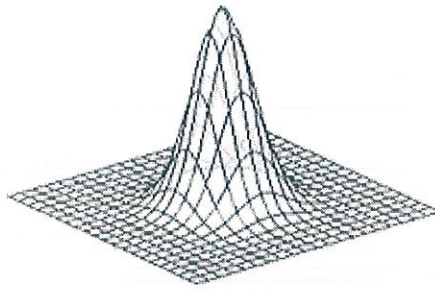


Figure 4.23

IMAGE

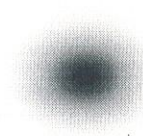


Figure 4.24

HIGHPASS

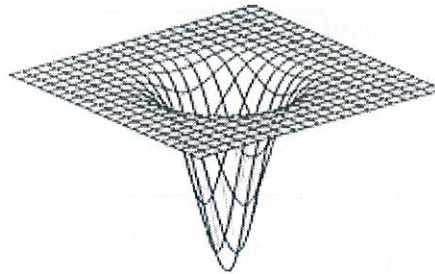


Figure 4.25

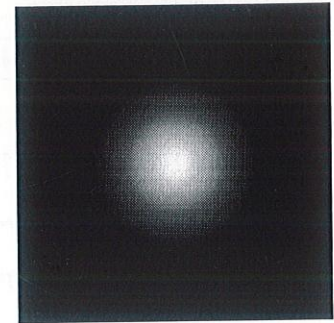


Figure 4.26

4.4 Peak Signal to Noise Ratio

Peak signal to noise ratio often abbreviated as PSNR, is a term for the ratio between the maximum possible power of signal and the power of corrupting noise that affects the fidelity of its representation. Due to the wide dynamic range of signals, it is expressed in logarithmic decibel scale. The PSNR is most commonly used as a measure of quality of reconstruction of lossy compression. The signal in this case is the original data, and the noise is the error introduced by compression.

It is most easily defined via the mean squared error (**MSE**) which for two $m \times n$ monochrome images I and K where one of the images is considered a noisy approximation of the other is defined as:

$$MSE = \frac{1}{m \cdot n} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} [I(i, j) - K(i, j)]^2$$

The PSNR is defined as:

$$\begin{aligned} PSNR &= 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \\ &= 20 \cdot \log_{10} \left(\frac{MAX_I}{\sqrt{MSE}} \right) \\ &= 20 \cdot \log_{10} (MAX_I) - 10 \cdot \log_{10} (MSE) \end{aligned}$$

Here, MAX_I is the maximum possible pixel value of the image. When the pixels are represented using 8 bits per sample, this is 255. More generally, when samples are represented using linear PCM with B bits per sample, MAX_I is $2^B - 1$. For color images with three RGB values per pixel, the definition of PSNR is the same except the MSE is the sum over all squared value differences divided by image size and by three. Alternately, for color images the image is converted to a different color space and PSNR is reported against each channel of that color space, e.g., YCbCr or HSL.

4.5 Calibration

Calibration is of utmost importance for a setup using wide-angle lens suffering from radial distortion. It is done using Barycentric coordinate system. The barycentric coordinate system is a coordinate system in which the location of a point is specified as the center of mass, or barycenter, of masses placed at the vertices of a simplex (a triangle, tetrahedron, etc.). Barycentric coordinates are a form of homogeneous coordinates. The system was introduced (1827) by August Ferdinand Möbius.

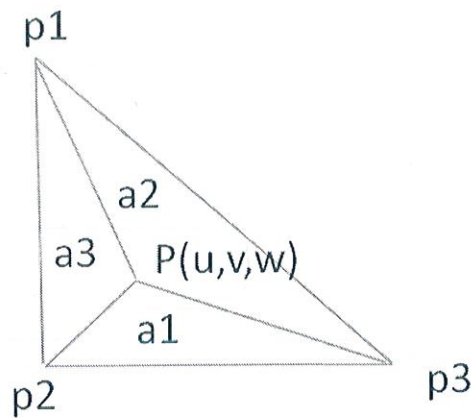


Figure 4.27

Figure 4.27 shows the barycentric coordinates of a point P which lies inside a triangle in the Cartesian coordinate system. The barycentric coordinates u, v, w are calculated using the formula given below

$$U = a_1/a$$

$$V = a_2/a$$

$$W = a_3/a$$

where a_1, a_2, a_3 and a are triangle area's. For calculating the Cartesian coordinates from these barycentric coordinates we use the following formula.

$$\text{New } x = (\text{newA}.x * u) + (\text{newB}.x * v) + (\text{newC}.x * w)$$

$$\text{New } y = (\text{newA}.y * u) + (\text{newB}.y * v) + (\text{newC}.y * w)$$

4.6 Blob Detection

After a frame has been captured and processed, the resulting image is used for blob detection. Blob detection is done by finding the contours of a blob using the `findContours` function available in opencv in the current frame. The function retrieves contours from the binary image.

The contours are a useful tool for shape analysis and object detection and recognition. All found contours (and their properties) are stored in a contour list.

```
void findContours(InputOutputArray image, OutputArrayOfArrays contours,  
OutputArray hierarchy, int mode, int method, Pointoffset=Point())
```

Parameters:

- **image** – Source, an 8-bit single-channel image. Non-zero pixels are treated as 1's. Zero pixels remain 0's, so the image is treated as binary. These functions –compare(), inRange(), threshold(),adaptiveThreshold(), Canny(), and others can be used to create a binary image out of a grayscale or color one. The function modifies the image while extracting the contours.
- **contours** – Each detected contour is stored as a vector of points.
- **hierarchy** – Optional output vector containing information about the image topology. It has as many elements as the number of contours. For each contour contours[i], the elements hierarchy[i][0], hierarchy[i][1], hierarchy[i][2], and hierarchy[i][3] are set to 0-based indices in contours of the next and previous contours at the same hierarchical level: the first child contour and the parent contour, respectively. If for a contour i there are no next, previous, parent, or nested contours, the corresponding elements of hierarchy[i] will be negative.
- **mode** – Various contour retrieval mode's available. It filters contours based on the mode and delivers returns them.
 - **CV_RETR_EXTERNAL** retrieves only the extreme outer contours. It sets hierarchy[i][2]=hierarchy[i][3]=-1 for all the contours.
 - **CV_RETR_LIST** retrieves all of the contours without establishing any hierarchical relationships.
 - **CV_RETR_CCOMP** retrieves all of the contours and organizes them into a two-level hierarchy. At the top level, there are external boundaries of the components.

At the second level, there are boundaries of the holes. If there is another contour inside a hole of a connected component, it is still put at the top level.

- **CV_RETR_TREE** retrieves all of the contours and reconstructs a full hierarchy of nested contours. This full hierarchy is built and shown in the OpenCV contours.c demo.
- **method** – various contour approximation method. Tells if approximation is to be made when finding contours.
 - **CV_CHAIN_APPROX_NONE** stores absolutely all the contour points. That is, any 2 subsequent points (x_1, y_1) and (x_2, y_2) of the contour will be either horizontal, vertical or diagonal neighbors, that is, $\max(\text{abs}(x_1 - x_2), \text{abs}(y_2 - y_1)) = 1$.
 - **CV_CHAIN_APPROX_SIMPLE** compresses horizontal, vertical, and diagonal segments and leaves only their end points. For example, an up-right rectangular contour is encoded with 4 points.
 - **CV_CHAIN_APPROX_TC89_L1**, **CV_CHAIN_APPROX_TC89_KCOS** applies one of the flavors of the Teh-Chin chain approximation algorithm.
- **offset** – Optional offset by which every contour point is shifted. This is useful if the contours are extracted from the image ROI and then they should be analyzed in the whole image context.

An ellipse is put over the contour; the properties of the ellipse are used to determine the position, orientation, and the size of the blob. If the size of the blob fits the minimum and maximum requirements on height and width, the blob will be added to the blob list.

4.7 Blob Tracking

- To report any action it is desired that the blob is detected in the correct frame corresponding to the touch.
- We make the reasonable assumption that a touch by a finger can move only short distances between frames. So the algorithm associates the closest pairs within a threshold distance.
- The number of blobs in typical images is small enough so that the efficiency of this algorithm is not crucial.
- The algorithm assigns the same blob id to the blobs in two consecutive frames if the distance between them is below a certain threshold. If the distance is greater than a certain threshold the blob id is incremented by 1 and assigned to the new blob.
- Tracking is particularly useful when detecting gestures. This is because one complete gesture would have the same blob id assigned to the blob while tracking it.

5. MULTIPLEXER MODULE

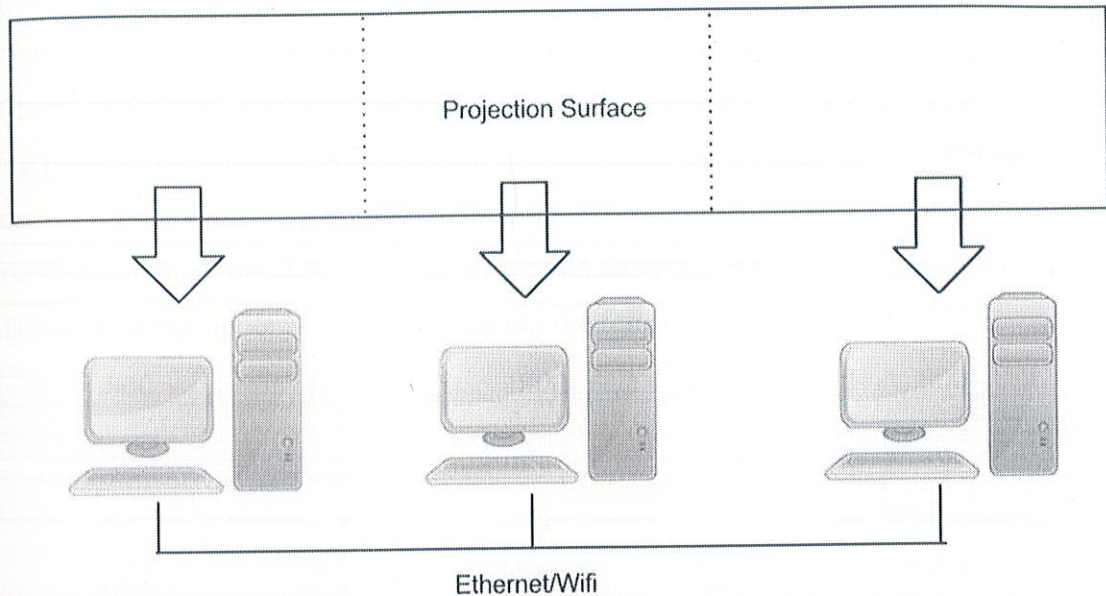


Figure 5.1

This module enhances the whole system by making it scalable. For implementing scalability, on such a table-top surface, there are two options. The first one is multiple camera approach; combining multiple camera streams and then sending it to the tracking pipeline. The disadvantage of this approach lies in the fact that only a limited number of cameras can be multiplexed together before running out of USB bandwidth (around 8mbps). Also, such a setup is computationally intensive as we have to stitch all these camera feeds. Second is multiplexing of data. It involves running application on different computers and then combining them.

Taking into consideration these disadvantages attached to the multi-cam approach of scalability, the second method is used i.e. multiplexing data. It involves acquiring finger coordinates from our tracking applications running on different computers and coordinating with each other over the existing Ethernet protocol.

Advantages of this approach over the multi-cam approach are:-

- 1) No bandwidth limitations therefore no limit on scalability
- 2) Not resource hungry, as it is only concerned with combining coordinate data from various sources.

The proposed method successfully overcomes the limitations of the earlier method. The following section explains how the system works.

The projection surface is divided into n parts ($n=3$ in this example). Each part is allotted to a computer node which runs a tracking application in itself. All the nodes perform their usual task of tracking pipeline for their part of the screen and then send the tracked data coordinates to the multiplexer for further tracking with respect to main screen. The effect of this is explained later. For differentiating input from different sources we add an offset based on the position of the node. This can be better understood by considering the following diagram.

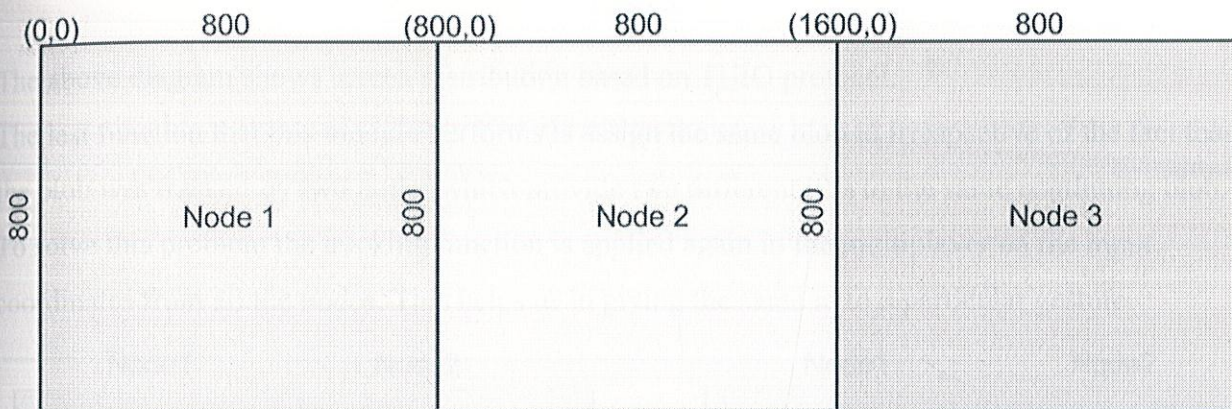


Figure 5.2

In accordance to the diagram, when a node is sending data it will add the his x offset to differentiate between his coordinate and other nodes coordinate. In the above scenario

node1.x = x

node2.x = x + 800

node3.x = x + 1600

the y coordinates would be sent as it is.

We are able to divide the screen based on the TUIO protocol which converts the coordinate data to touch inputs. According to the TUIO specification the value of length and breadth of any multi touch client application varies from 0.0 to 1.0.

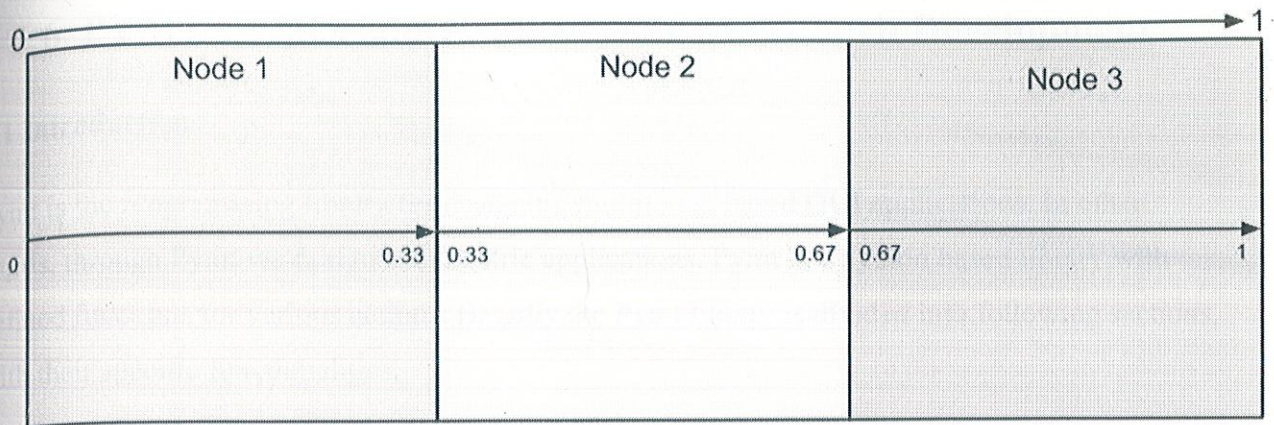


Figure 5.3

The above diagram shows screen distribution based on TUIO protocol.

The last function that this module performs is assign the same blob id irrespective of the fact that the blob was tracked by two nodes which provide two different id's to the same continuing blob. To solve this problem the tracking function is applied again in the multiplexer on the input coordinates from all the nodes. This helps us in giving the same id to a particular gesture.

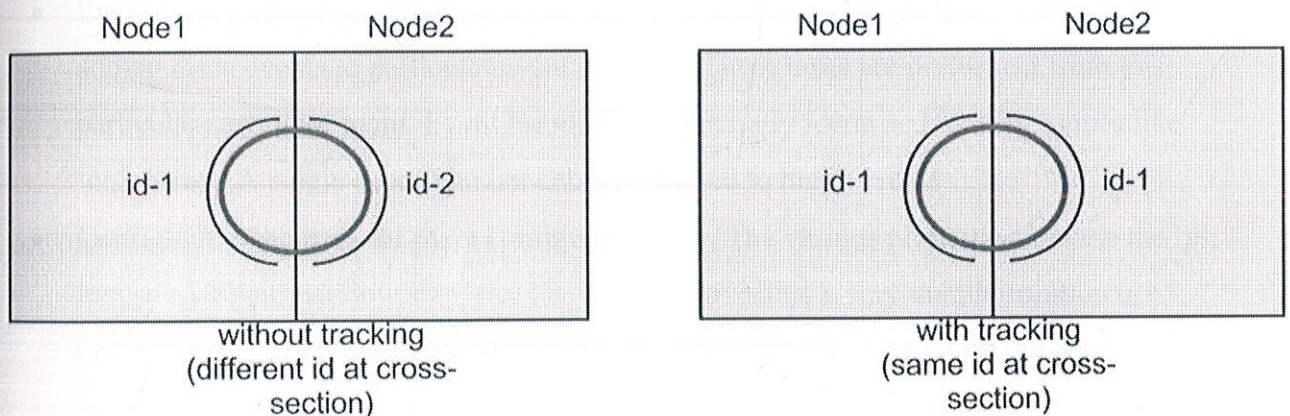


Figure 5.4

Thus the multiplexer performs all the necessary functions for increasing the scalability of our multi touch setup.

6. PYMT – Open source library for Multitouch Development

6.1 Introduction

Pymt is the programming library for designing multitouch based GUI applications. In other words, through Pymt we design user centric applications. Pymt is a python based library with defined functions for various actions. Broadly the Pymt library is divided into following sections, with their specific functionalities:

- **Base**-This is the main section for managing the windows, main event looping in which all the objects and functions are defined.
- **Gesture**-This is an important section with main aim of gesture recognition. It can be used to create new gesture and match any user input gesture with the stored gestures. The gestures can be multi-stroke i.e. they needn't be in a single flow and could be broken down into multiple user strokes. This enables user to make a cross sign in two strokes.
- **Event**-Every object can produce events like a 'touchdown', etc. so there is a need for adding these events to particular event handlers. These tasks are performed from this particular section of pymt. Event Handlers are generally identified by their names like 'on_resize'. A single event-handler can be assigned to many events.
- **Config** – It is the base for pymt configuration file. The various properties include the fps, mouse enabling, window size, etc. The config base offers a very unique solution to upgrade the configurations by editing the properties and thus upgrading the config version. If the version is not set i.e. it's value is zero then default values are fed to these config parameters.
- **Geometry**- It is the base for computing some of the trivial calculations like calculating the minimum size of circle that will enclose all the points through its defined functions of `circumcircle` and `minimum bounding circle`.
- **UserInterface (ui)** – This base provides effects and various user-interfacing options in pymt. The various buttons, sliders, menus are provided through this base. Also the font-styles, border-styles can be controlled through this base. This base creates the various user-interfacing buttons, sliders through `Opengl`. The animation effects are also provided

through this base as the classes which need to be animated and variables of initial state and final state of the widget is added to the function.

6.2 Pynt Applications

Few applications that are designed in this project for users using Pynt library are described below:-

6.2.1 Paint

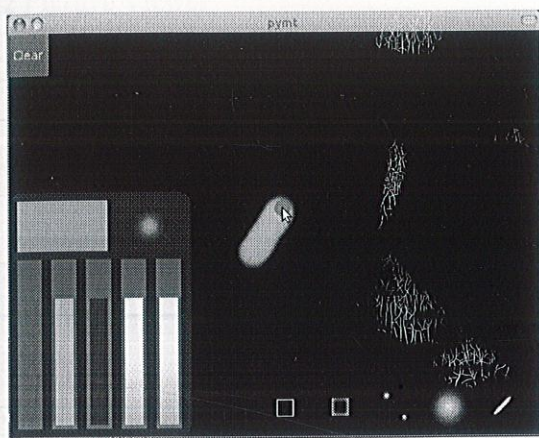


Figure 6.1

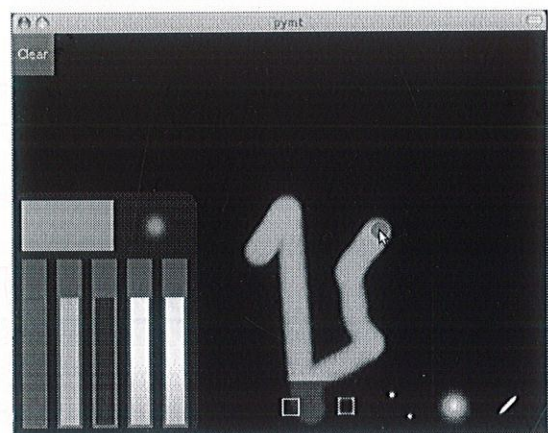


Figure 6.2

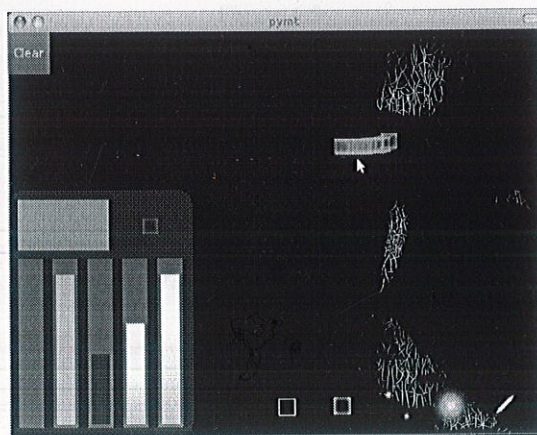


Figure 6.3

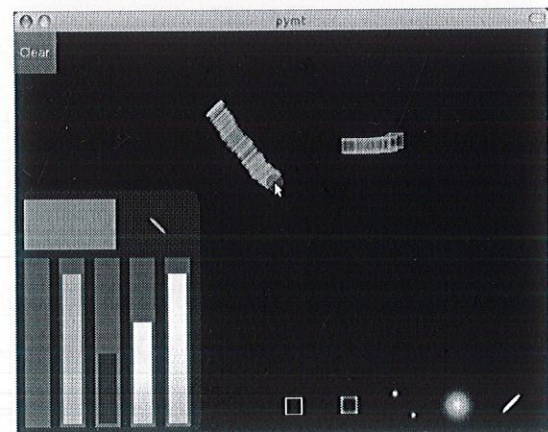


Figure 6.4

This is a simple paint application which has colour selector in the bottom left hand side according to the RGB color scheme. Different types of crosshairs could be selected from the bottom bar.

The colour selector is a slider interface so that the user can touch in the particular row and scroll up and down to reduce the particular value of that color. This slider was created using the MTSlider function in the 'ui' section of the Pymt library.

The Clear button gives the user an option to clear all drawings at any given instance.

6.2.2 Picture Viewer

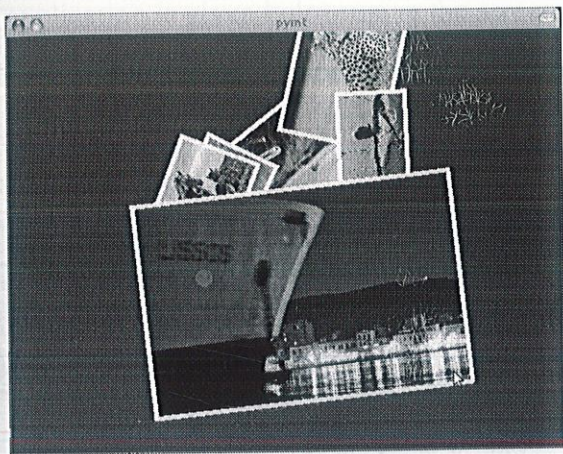


Figure 6.5

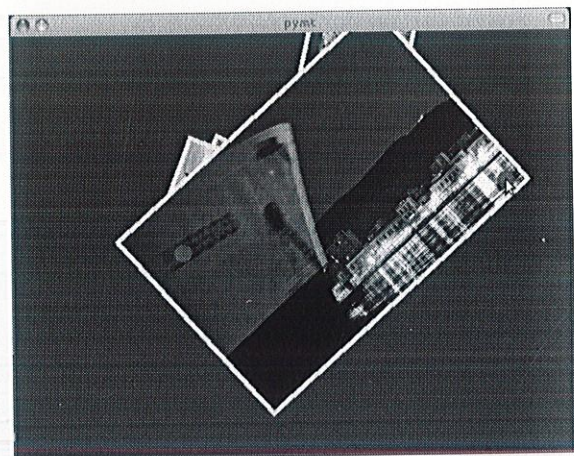


Figure 6.6

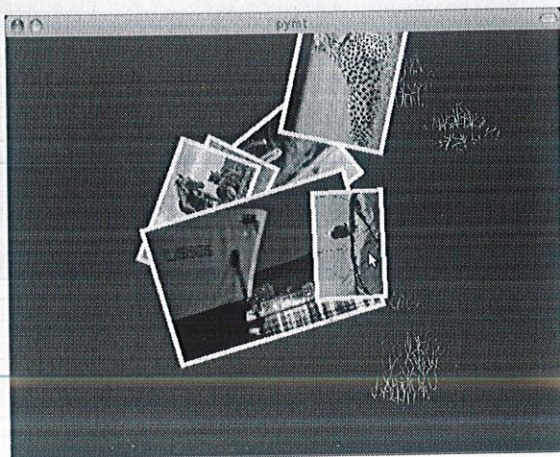


Figure 6.7

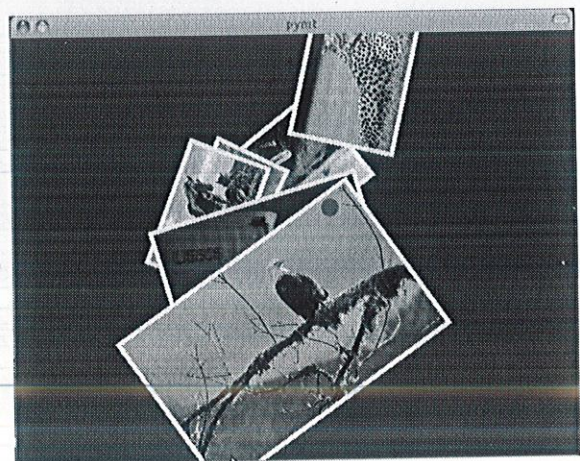


Figure 6.8

This application provides a unique interface to the users to browse through the picture galleries and at the same time zoom-in and zoom-out of any particular picture. The user can also rotate any particular picture. By default the picture stored in the program directory gets loaded so to add or remove any picture we need to modify this directory form the OS.

6.2.3 Calculator

The following application provides a basic calculator with gesture recognition. A user writes a equation as he would by hand on a paper and the application provides the output. A demo of the application is given below using the screenshots.

First number is input.

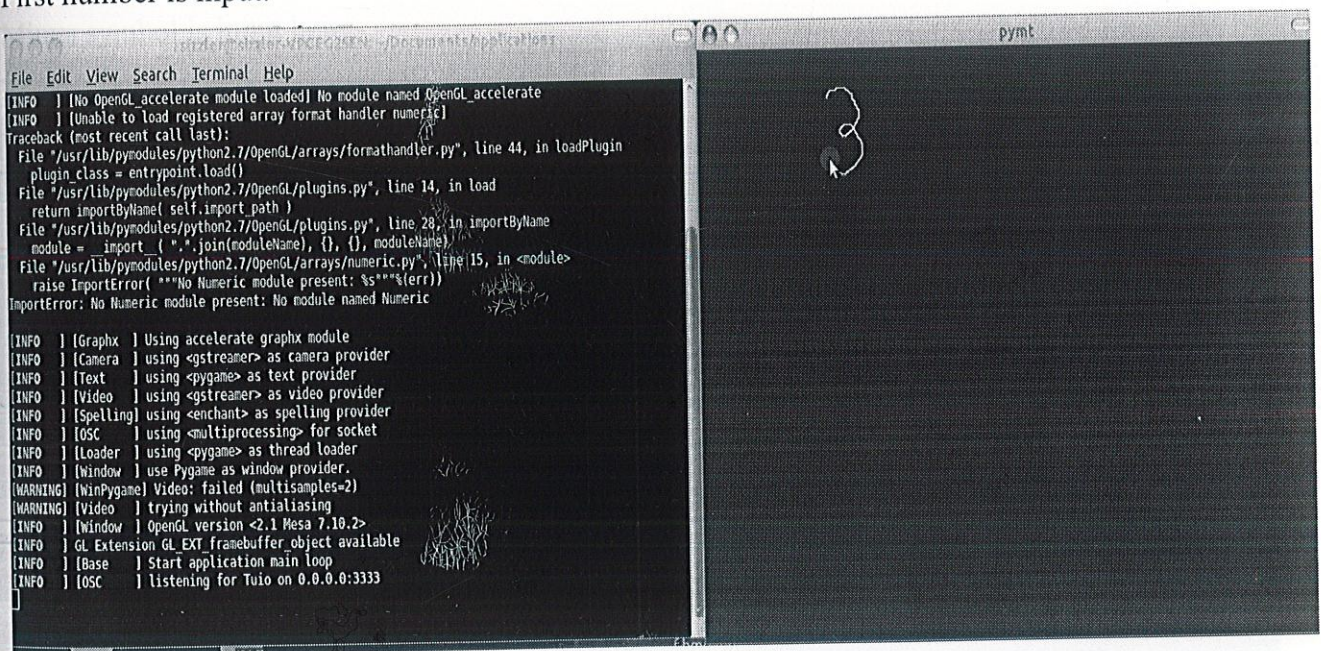


Figure 6.9

Number is recognized by our application

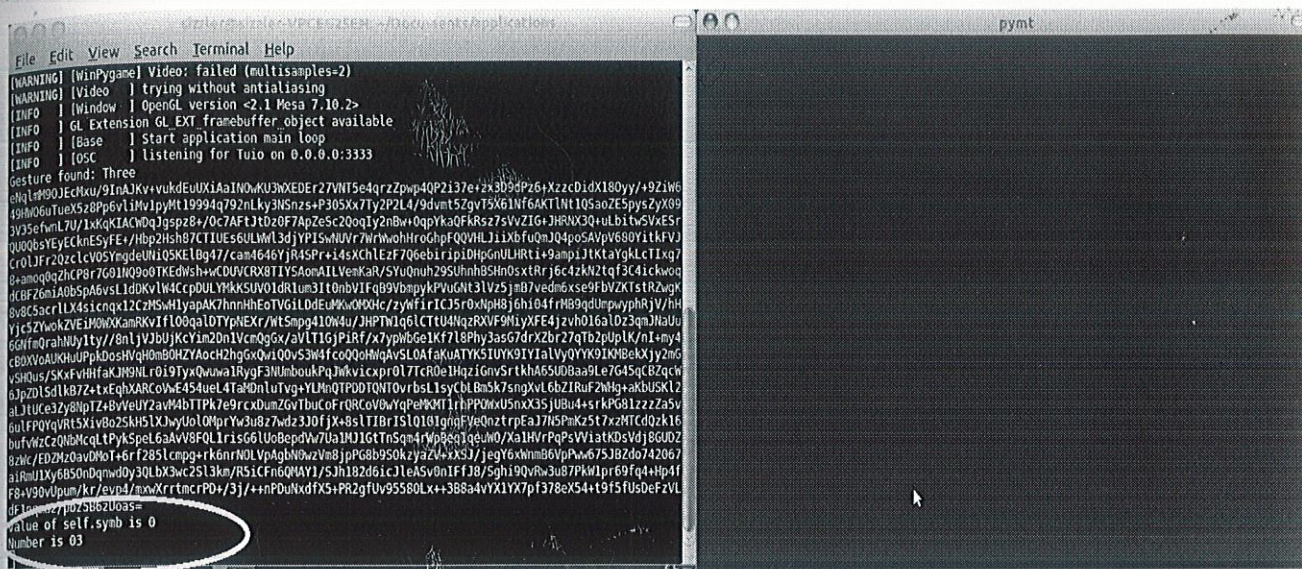


Figure 6.10

Minus symbol is input here:-

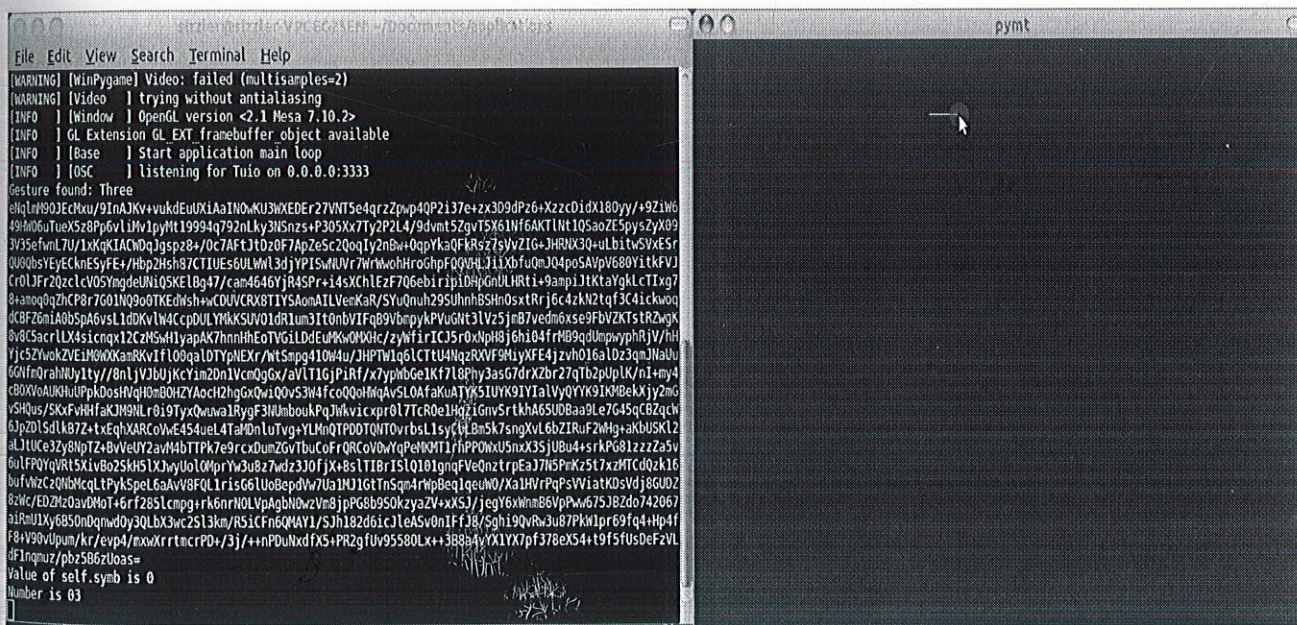


Figure 6.11

As can be seen the symbol is recognized by the application as minus.



Figure 6.12

The second number is entered below:-

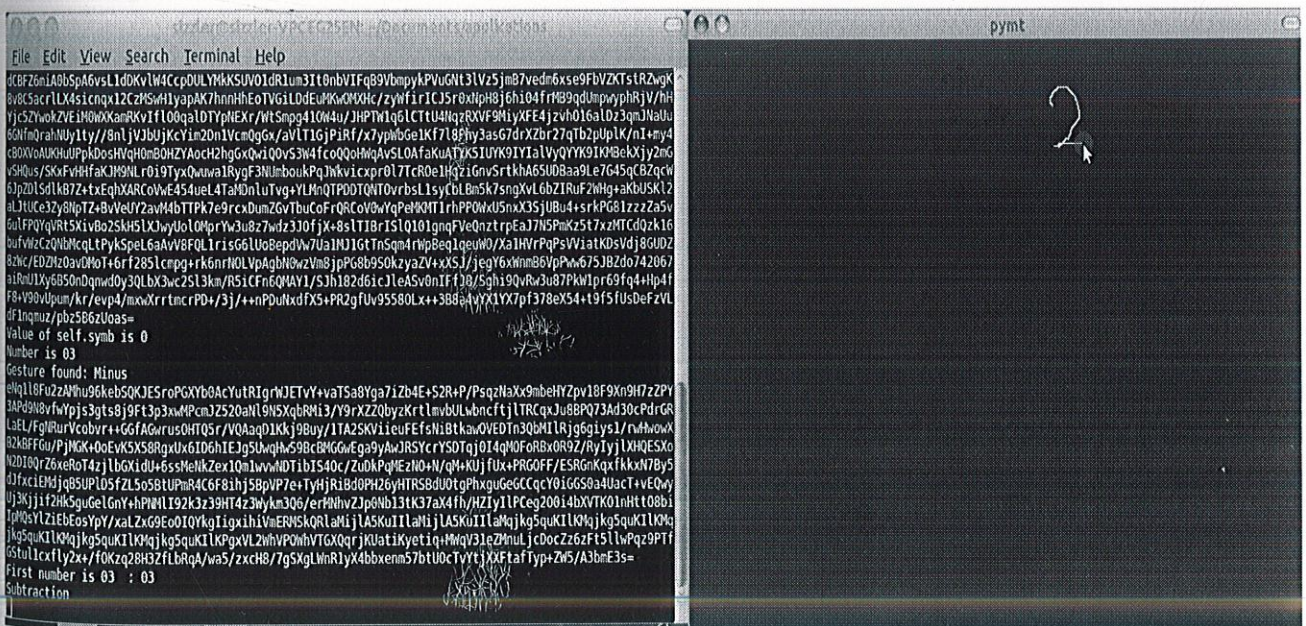


Figure 6.13

The number is recognized by the application (as can be seen below)

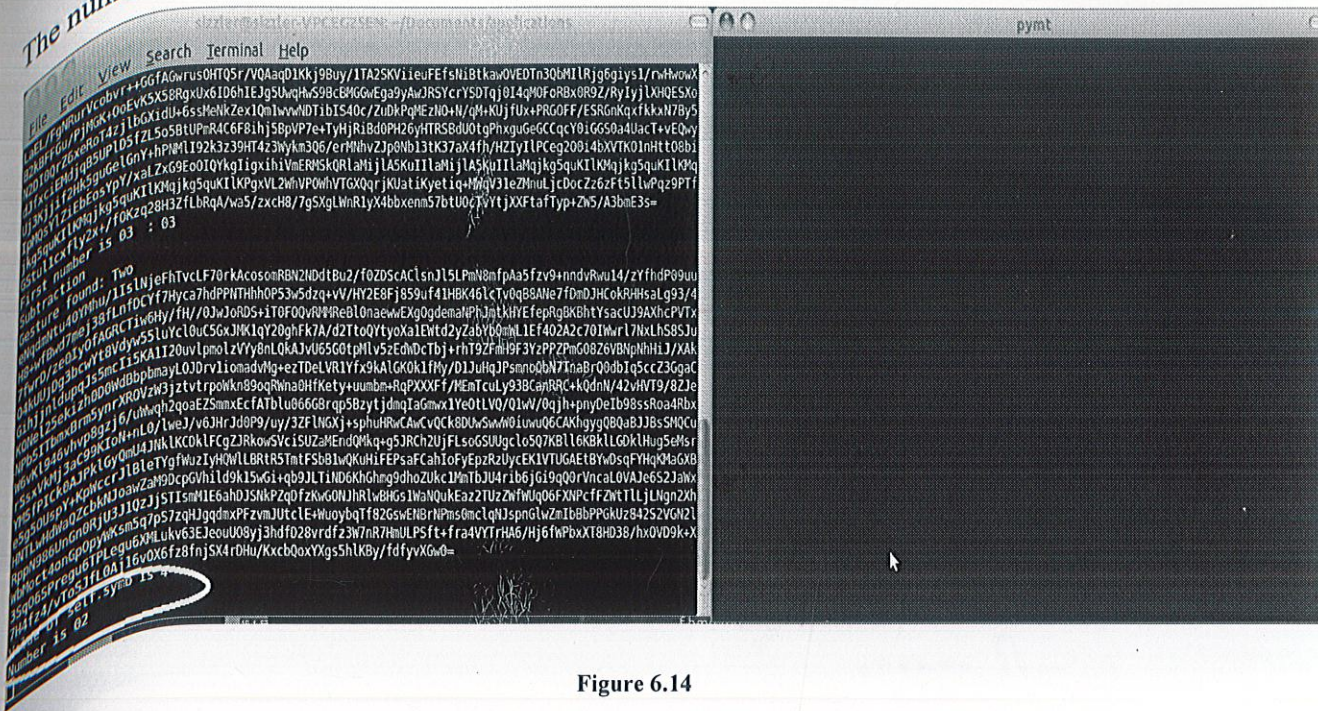


Figure 6.14

Next '=' sign is entered as two instances of '-' sign.

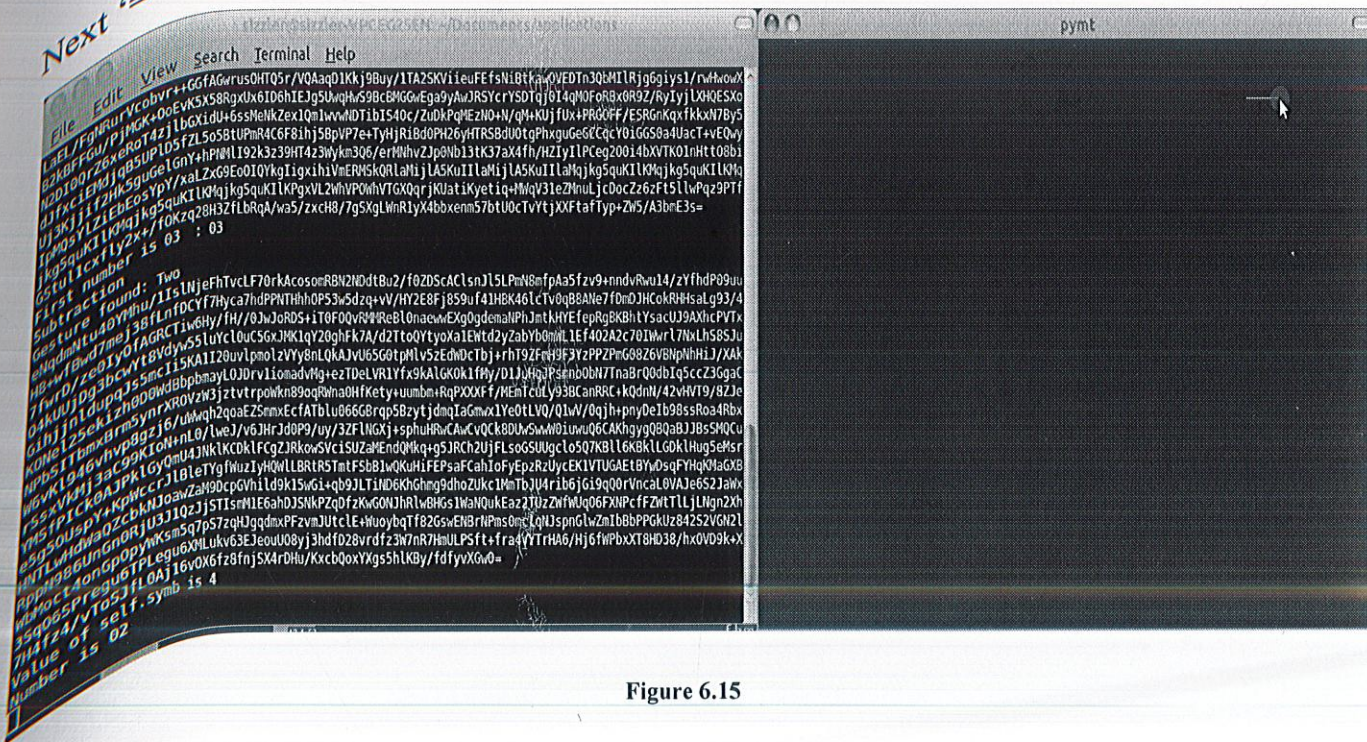


Figure 6.15

It is recognized as a single minus (as can be seen below)

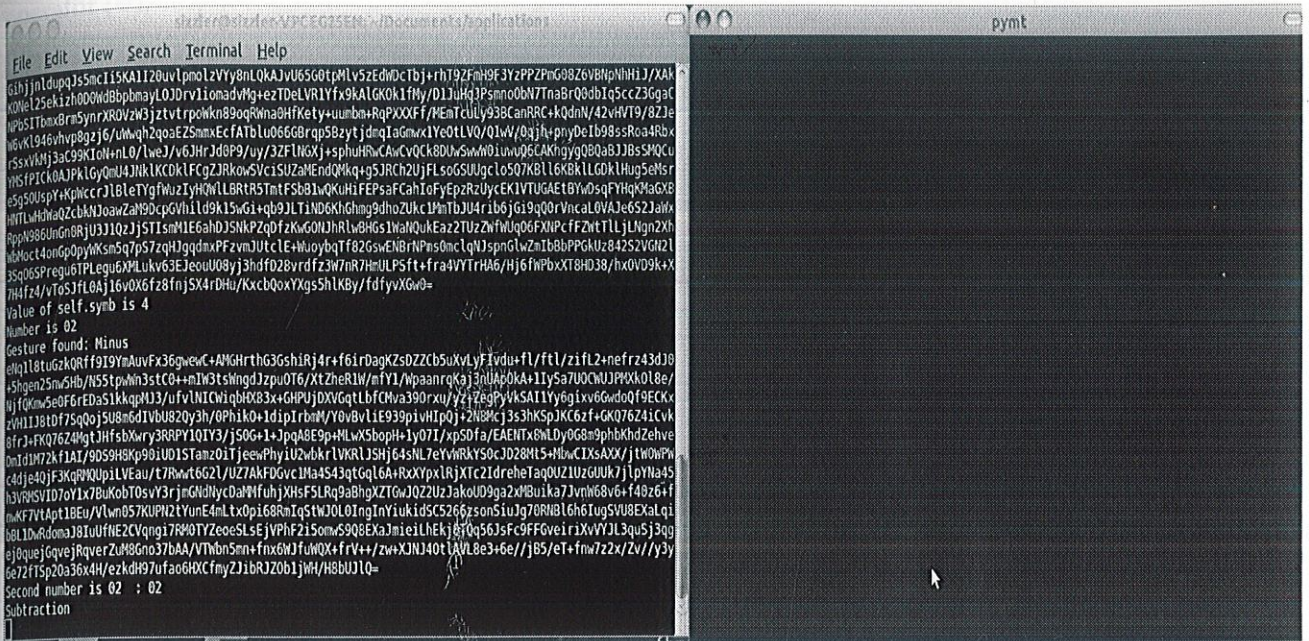


Figure 6.16

Next, second minus sign is entered.

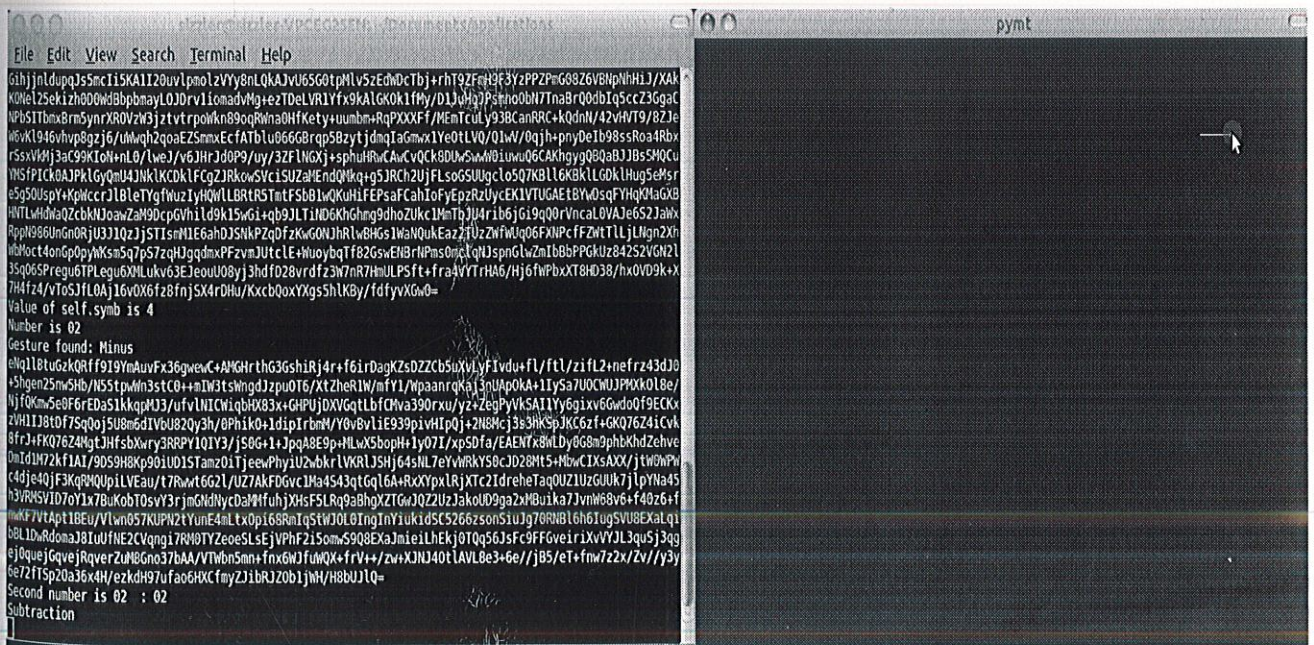
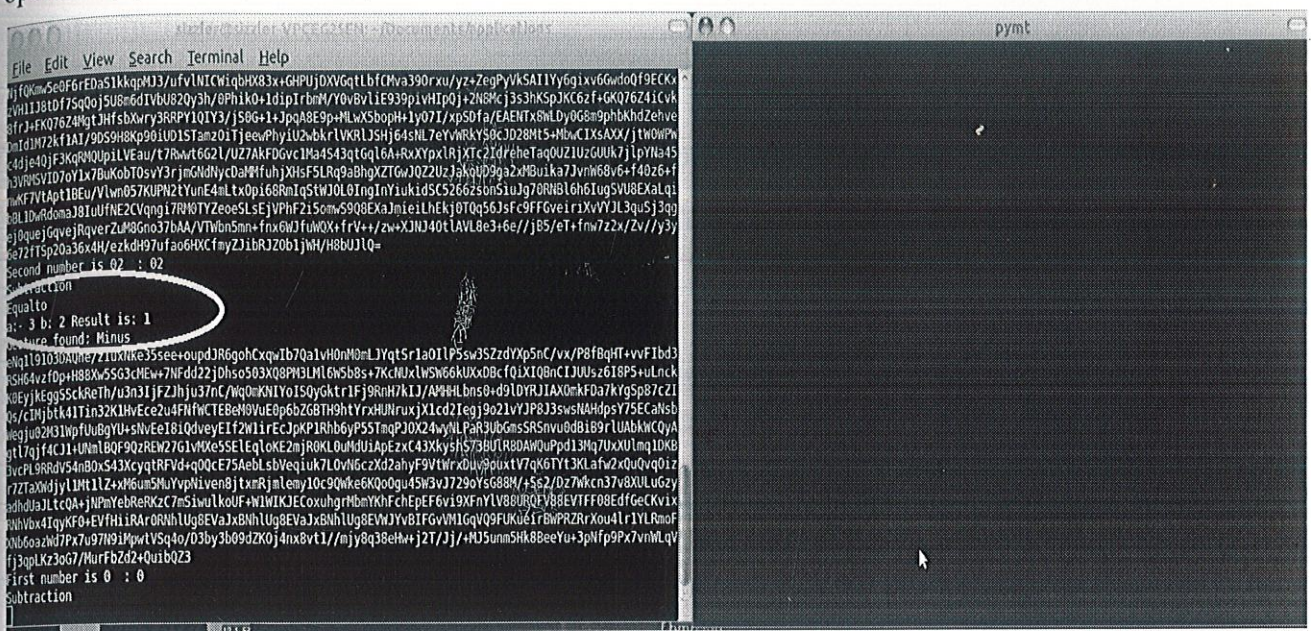


Figure 6.17

Our application outputs the result as it recognizes two simultaneous minus sign as an equal to operator (=).



```
File Edit View Search Terminal Help
pynt
Second number is 02 : 02
Equal to
a: 3 b: 2 Result is: 1
Feature found: Minus
```

Figure 6.18

7. TUIO – Tangible User Interface Object

7.1 Introduction

TUIO stands for touch user interface objects (TUIO). It is a protocol which helps in transferring touch related object's information over a network or between different applications running on the same computer. The figure below illustrates the working of TUIO protocol.

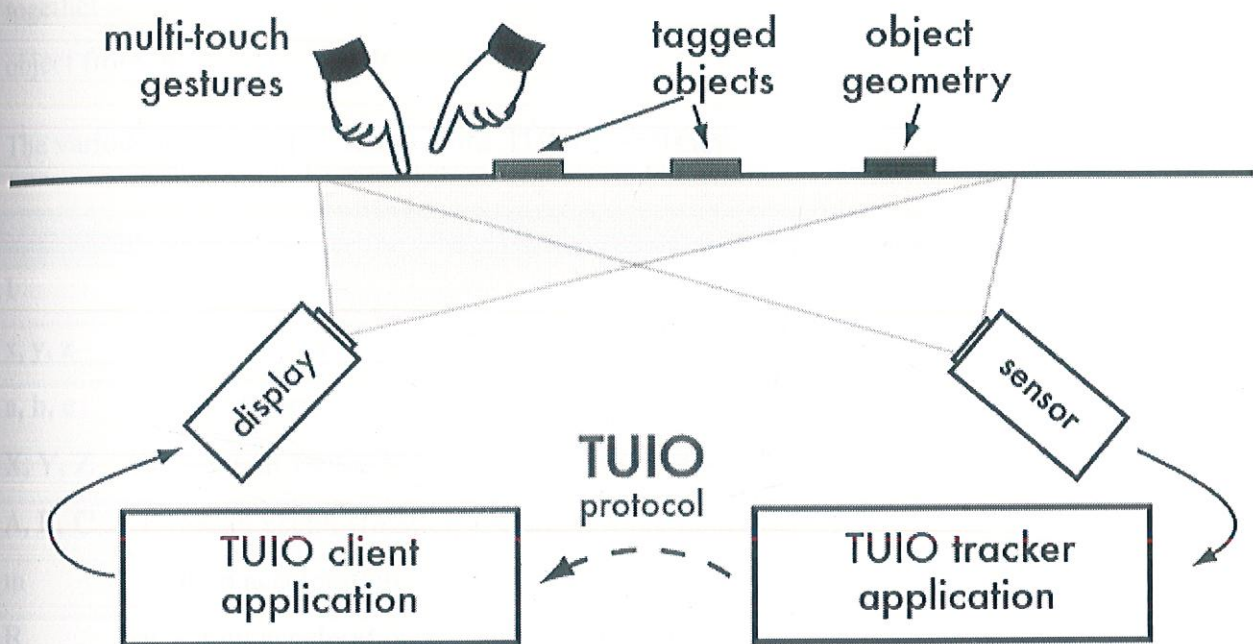


Figure 7.1

As explained in Figure 7.1 the camera is controlled by the main program which extracts the blob information from the captured frame. So, if any touch action has been performed by the user its information like x-coordinate, y-coordinate, etc. are extracted by this program. But now the challenge is to send these coordinates to the PYMT application which would perform corresponding actions to those events, this is where TUIO comes in. TUIO Protocol was designed specifically for the purpose of sending the touch information over a network. The protocol divides its messages into broadly two types:

- Set messages – This message is used to send the object details that are currently present on the touch-surface. Generally, the message is sent only when there is a change of state in the objects.
- Alive messages – This message contains information about all the current objects present on the touch-surface. Also, it needs to be sent only when there is an addition or removal of any object from the list of objects.

In addition to these two, there is also, a *fseq* message which numbers these two messages together so that lifetime of an object could be deduced from the addition and removal of an object from the *alive* messages.

The various parameters included in the TUIO packets are:

S	Session id(int32)
I	Class id(int32)
x, y, z	Position variables(float32)(range 0....1)
a, b, c	Angle (float32)(range 0....2PI)
X, Y, Z	Movement vector(Motion speed and direction)(float32)
A, B, C	Rotation vector (rotation speed and direction)(float32)
m	Motion acceleration
R	Rotation acceleration

Table 7.1

The TUIO protocol has a higher priority to provide low latency thus it is based on UDP transport. The UDP transport ensures low latency but at the risk of packet loss. To overcome this shortfall each *set* message are packed into a bundle and each bundle also includes a redundant *alive* message so as to provide the receiver with a consistent view of the objects present on the touch-surface. As a *fseq* message numbers these messages, loss of any single packet would mean resending that information again. So, for this purpose a cache of messages is always maintained by TUIO so that it would be able to send that information again if sending of those packets fails. There is also a concept of profiles used by this protocol which maintains the identification of the object as 2D or 3D, as TUIO can support sending information about both types of objects. The TUIO messages also contain an additional information about the profile of the object. As our

setup has to support touch, we would need the 2Dobj profile while transmitting through TUIO protocol.

So, the format of a TUIO set message would be something like this:

```
/tuo/[profileName] set sessionID [parameters]
```

And the format of a single bundle of message would be:

```
/tuo/[profileName] set sessionID [parameters]
```

```
/tuo/[profileName] alive [list of active sessionIDs]
```

A real-time TUIO message would look something like this:

```
/tuo/2Dobj set s I x y a X Y A m r
```

```
/tuo/2Dobj alive s1 s2 s3
```

The TUIO programming consists of mainly classifying the listeners and senders. The classification based on the APIs is listed below:

1. TUIOServer: This class consists of following important functions

- **initFrame:** This is the first command given while initially creating the frame. It initializes a frame according to the current TUIO time. So, every frame has a unique start time through which it can be uniquely identified.
- **addTuioObject:** This function is needed to add a new TUIO Object to a frame. This function is used after initializing a frame and this object is added to the frame. TUIO object is initialized with a serial number, x-coordinate, y-coordinate and z-coordinate.
- **updateTuioObject:** This function is used to update the existing TUIO object. The object is updated with the new x coordinate, y coordinate and angle of the TuioObject.
- **removeTuioObject:** This function is used to remove a TUIO object from the initialized frame.

- **commitFrame:** This function is used to commit the frame i.e. it is used to send the frames info over the network. The current set of objects present in the frame along with their information is sent.
 - **TUIOServer:** The default and initial constructor for this class. It assigns the port value on which the data would be sent by the TUIO.
2. **TUIOClient:** This class consists of following important functions
- **TUIOClient:** This function is the constructor function which initializes the port on which the client would be listening to for the TUIO objects.
 - **Connect:** After this function is executed the TUIO Client starts listening to the TUIO messages at the UDP port. All the received messages are decoded to be further processed.
 - **getTuioObjects:** This functions fetches all the TUIO Objects present in the message and creates a corresponding list of TUIO Objects for them. Thus the transmitted TUIO Objects are received at the receiver end.

8. APPLICATION WALKTHROUGH

The following section assists the user in understanding the functionalities that are offered by the application. This would help the user in setting up the table. He/She may use few of these functionalities to achieve optimum finger detection via the table

8.1 Running the application

The user has to navigate to the bin folder in the alladdonsexample folder, where the .exe is present. Double Clicking the executable opens the application. Also, there is folder named Data which holds the config.xml file. The user may use this to alter the settings instead of using the GUI to alter the settings. In both cases the settings would be saved in this config.xml file.

Figure 8.1 shows the greeting window when the application is run

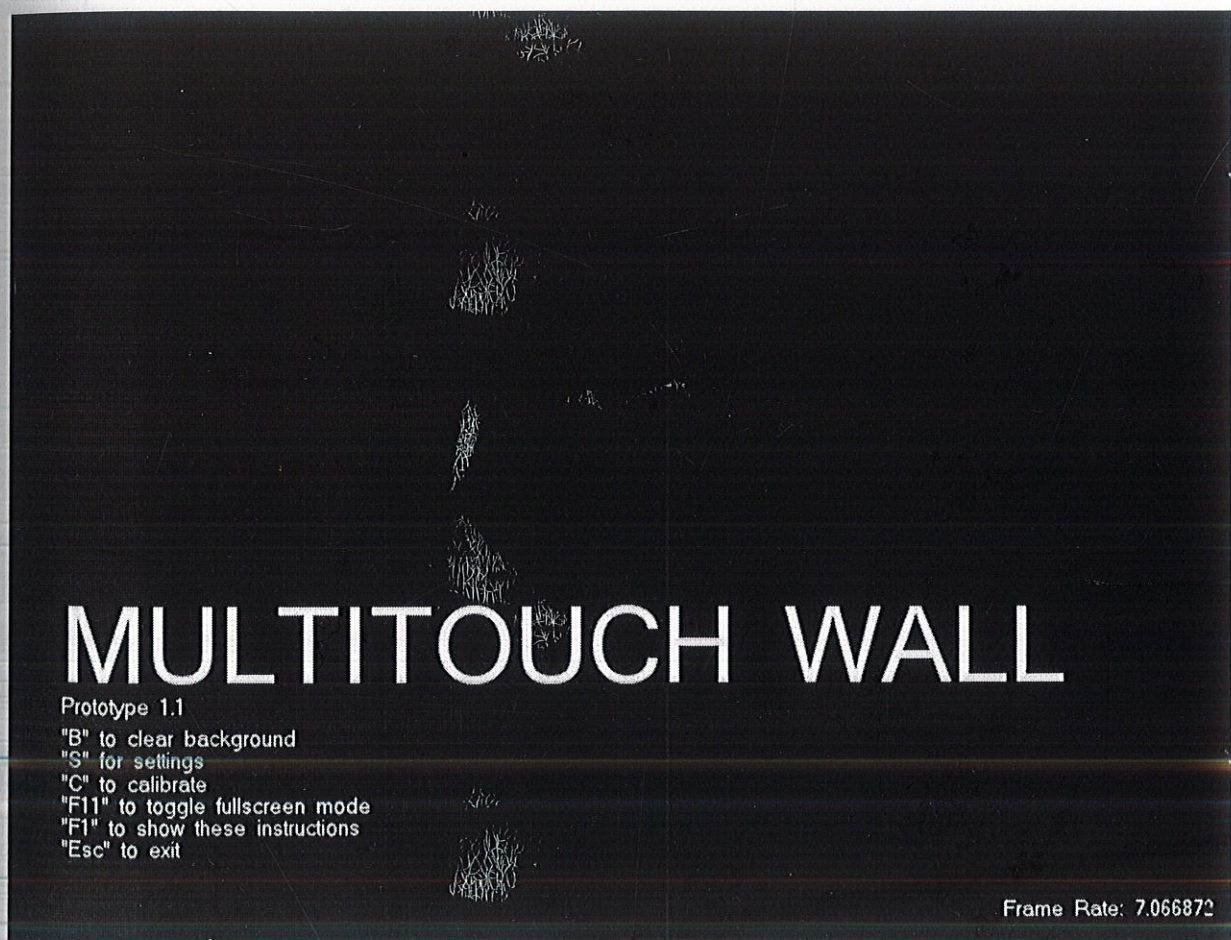


Figure 8.1

The First screen explains the version (1.1) of the application and provides keyboard options to navigate further. These include:-

“B” – Clear Background – This captures the current frame and then subtracts it from every subsequent frame, thus removing excess noise.

“S” – Opens the Setting Panel where further options are available

“C” – Opens the calibration mode

“F11” – Toggles the application between Fullscreen and normal screen mode

“F1” – Show’s these setting again, as they are available only for the first 5 seconds

“Esc” – Pressing Esc at any point of time exits the program

The captured frame rate is continuously shown in the lower right hand corner.

8.2 Calibration Mode

If the user chooses the calibration mode, Figure 8.2 appears. This is the landing page for the calibration module.

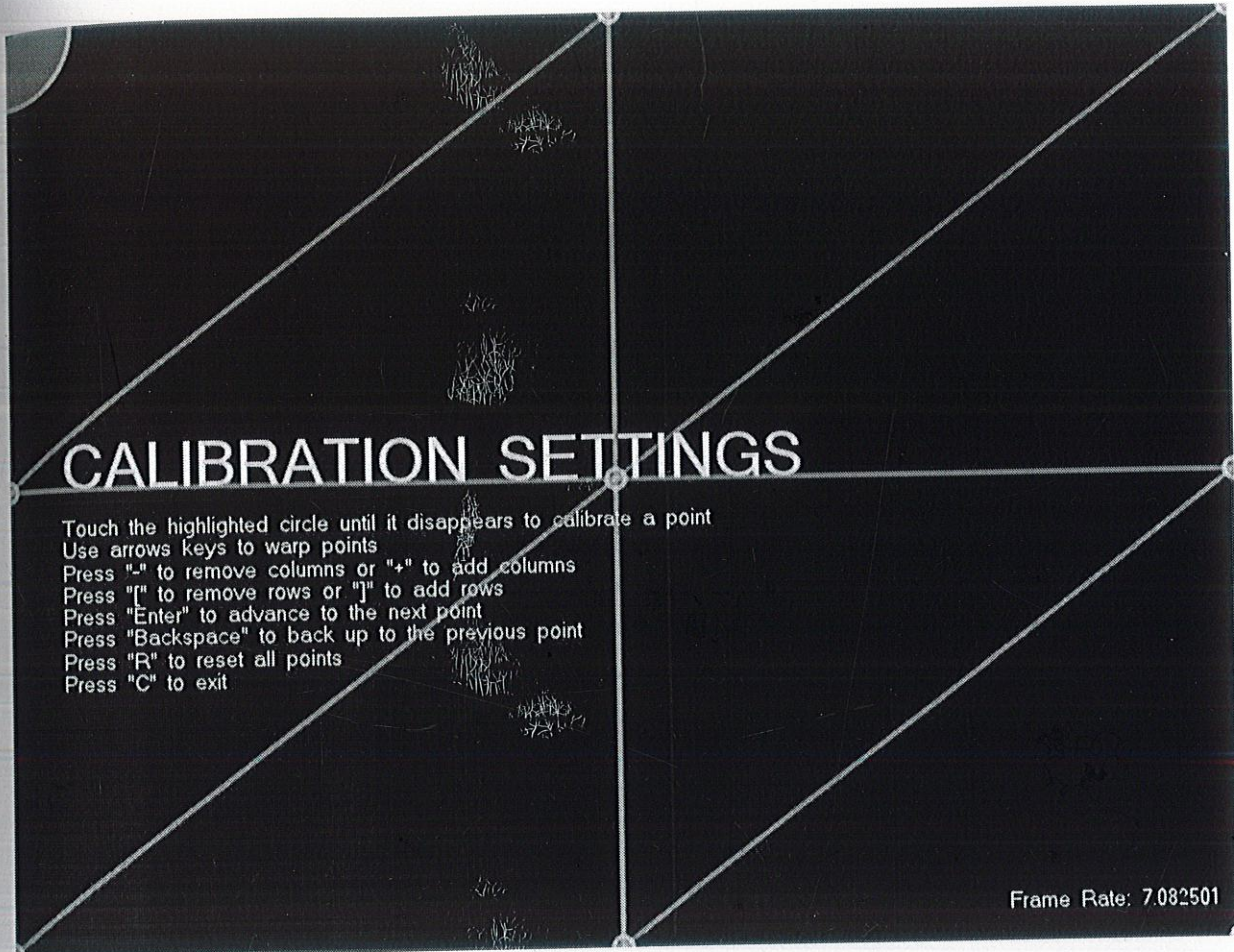


Figure 8.2

This module offers other functionalities which are triggered using the keyboard. All the keyboard bindings are shown the main calibration screen, as shown in Figure 8.2

8.2.1 Editing Number of Columns

If the user wants better calibration, in terms of accuracy, he can increase the number of columns used for calibration.

“-“ – Decreases the columns – The least value is 2

“+” – Increases the columns – The Maximum value is 12

Figure 8.3 shows the maximum possible columns that are possible for calibration.

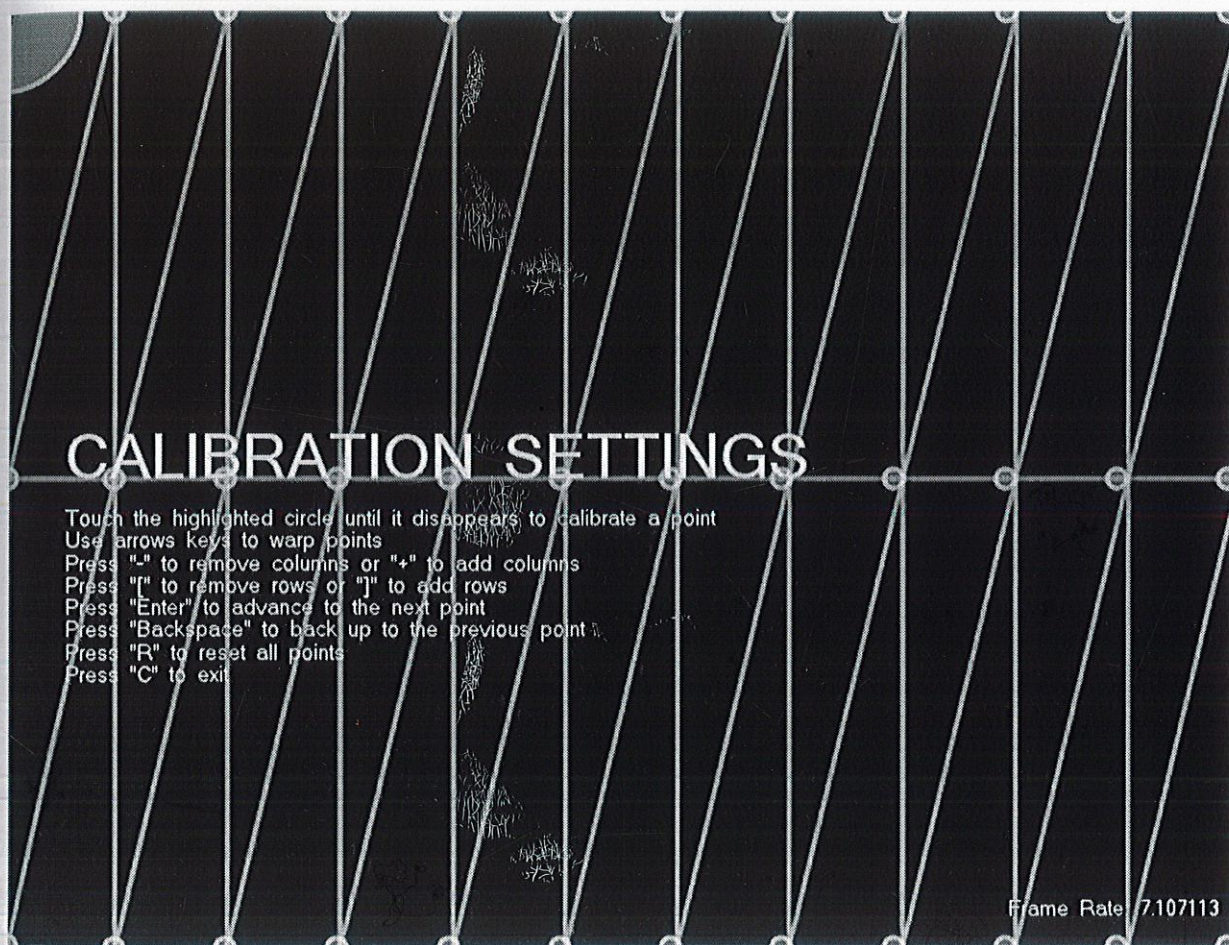


Figure 8.3

Columns alone are not responsible for increasing accuracy, thus increasing Rows is equivalently important.

8.2.2 Editing number of Rows

If the user wants better calibration in terms of accuracy he/she can increase the number of rows used for calibration.

“[” – Decreases the columns – The least value is 2

“]” – Increases the columns – The Maximum value is 12

Figure 8.4 shows the maximum possible columns that are possible for calibration.

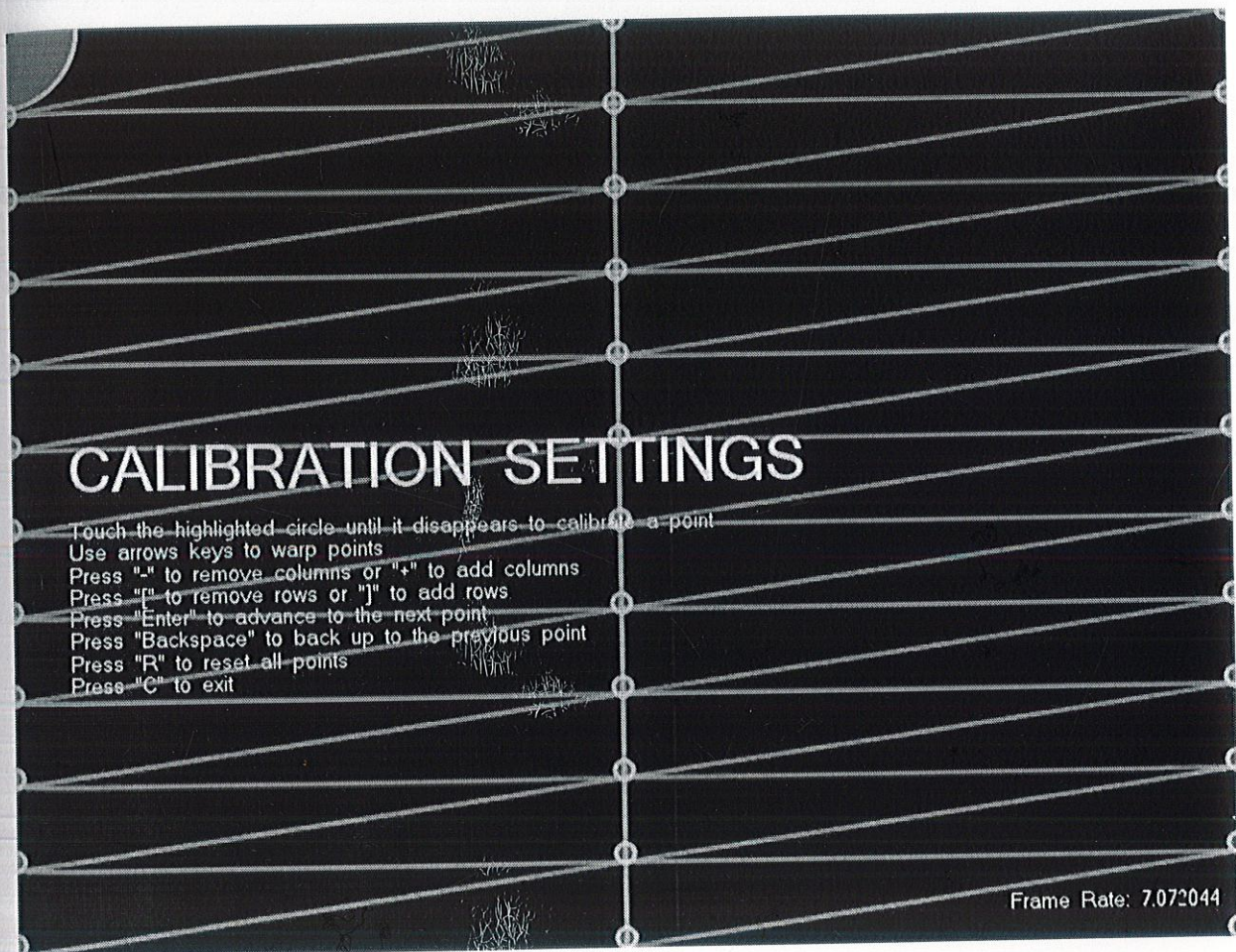


Figure 8.4

Figure 8.5 shows maximum possible rows and columns that are possible. In this case the accuracy of calibration will be maximum but the user will have to spend a considerable time in calibration of 144 points. But once done, these values are safely saved in calibration.xml.

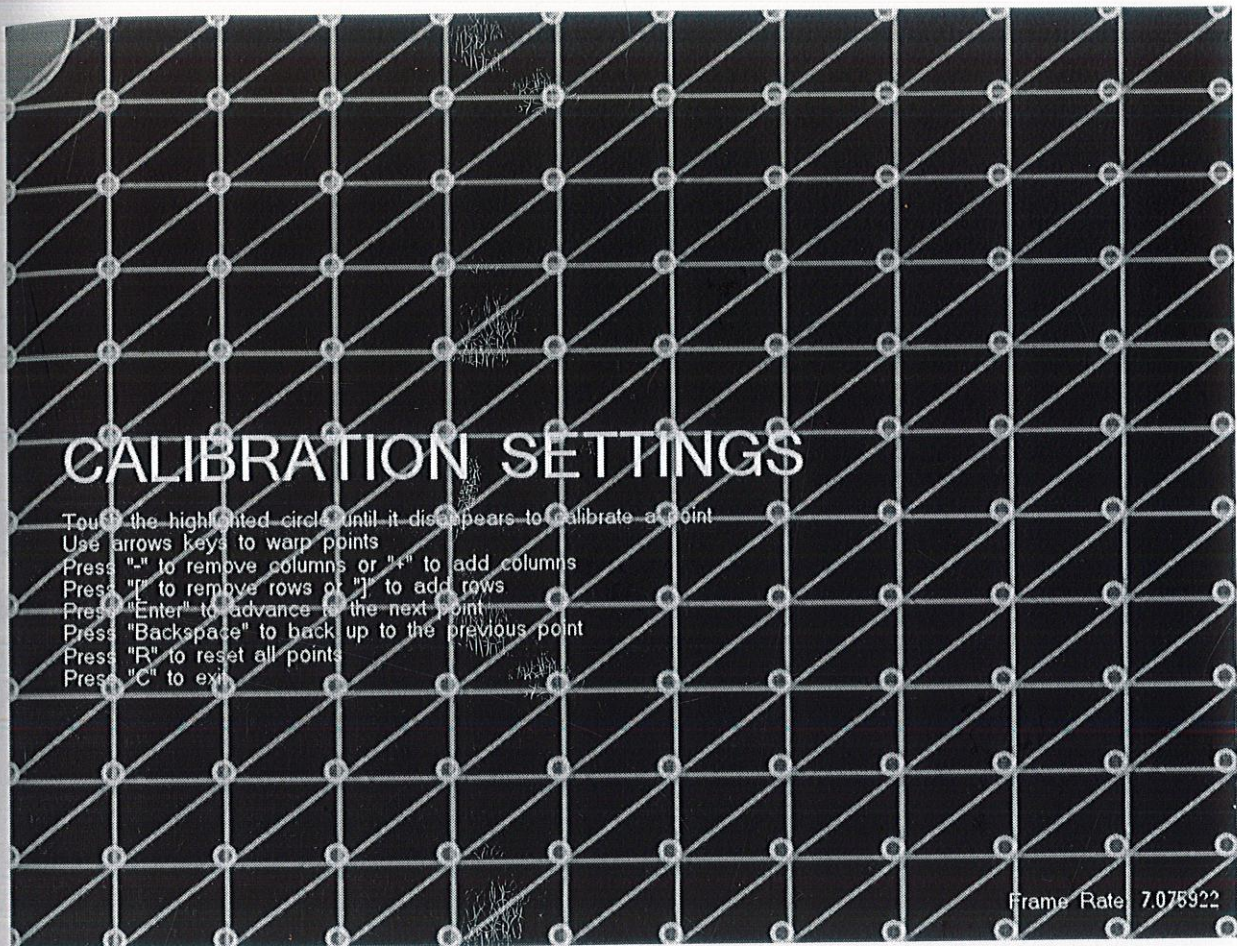


Figure 8.5

The calibration module works when the user touches the table and the finger is registered as a blob. This blob is used as the on-screen point for the on application point in consideration then. For example in the above figure 8.5 the on application point is the 1st one, when the user registers a blob, the on screen position of that finger is saved with respect to this point. Once successfully saved, the on application point in consideration becomes the 2nd point automatically. This process continues till the user has successfully calibrated the whole screen.

A situation may occur when the user may want to save the position of a particular point. Instead of repeating the whole procedure again, we offer him functionality to navigate to that particular point. This is done by the keyboard events:

“Enter” – Advance to the next point

“Backspace” – Go back to the previous point

Figure 8.6 shows the pressing of the “enter” button

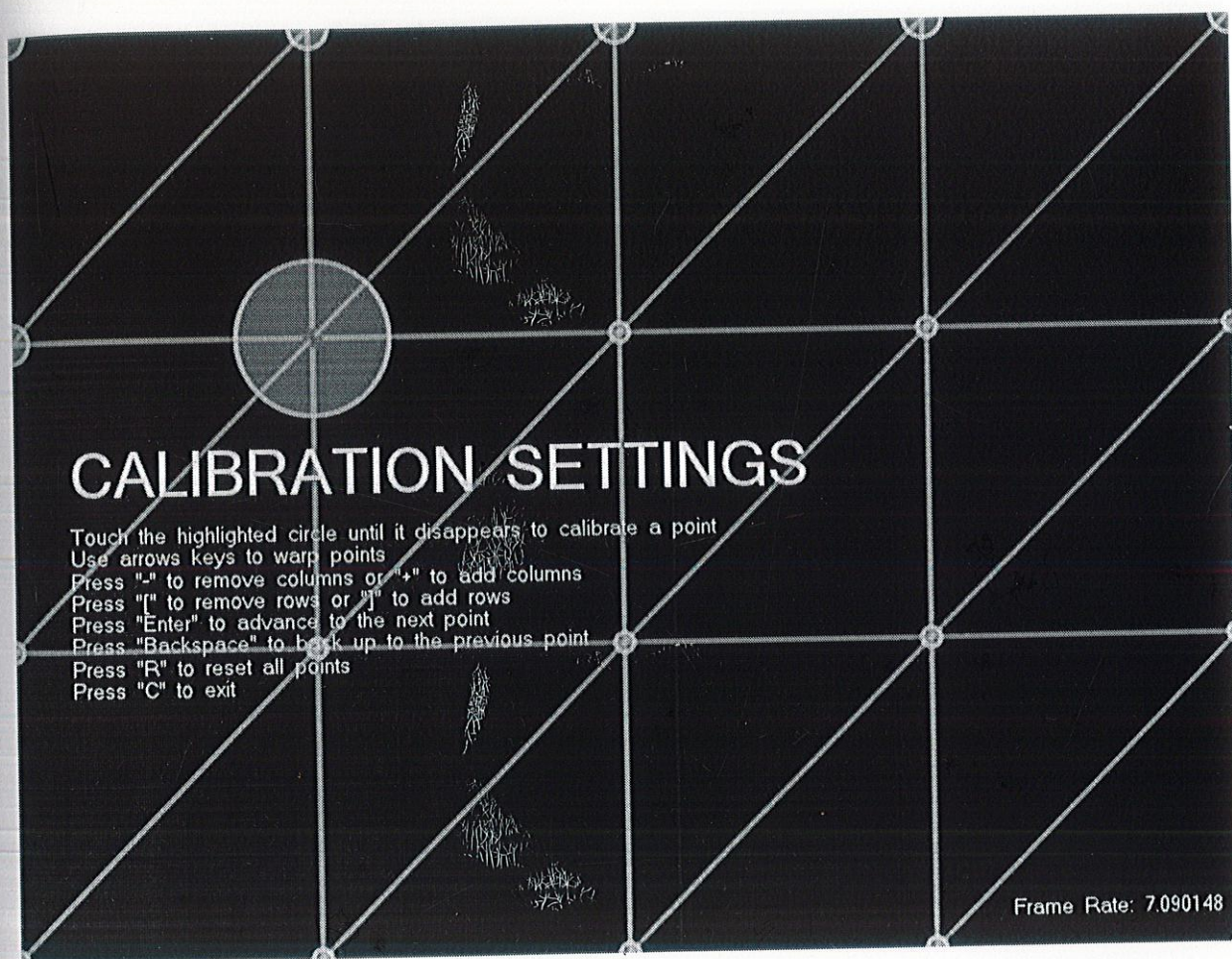


Figure 8.6

When a user wishes to recalibrate his screen at any point of time. He/She can use the reset button.

“R” – Resets all calibration data

Once “R” is pressed, all previous calibration data saved in the calibration.xml file is erased. The point of consideration becomes the 1st one and the system is ready to accept new calibration data.

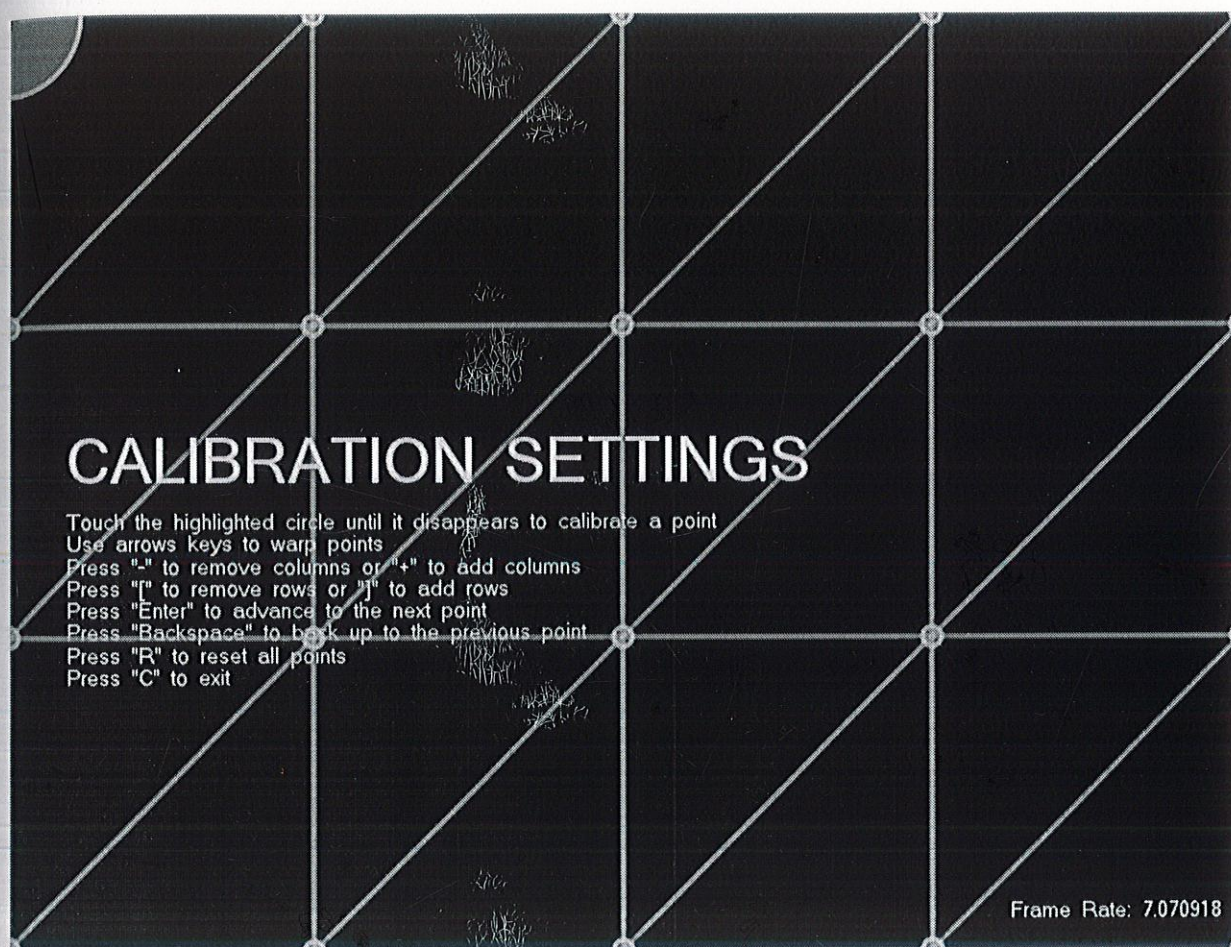


Figure 8.7

This concludes the calibration module. The user has to press “F1” to see the main menu options or “S” to go to the settings panel.

8.3 Settings Mode

The settings panel offers number of functionalities which the user can use to successfully detect, track and send the blobs data.

By pressing “Left” and “Right” arrow keys a user may shuffle between various functionalities.

By pressing “Up” and “Down” arrow keys a user may alter the parameters of that particular functionality.

8.3.1 Amplify

This multiplies the input image by itself to increase the intensity of the tracked image.

Figure 8.8 shows how amplification works

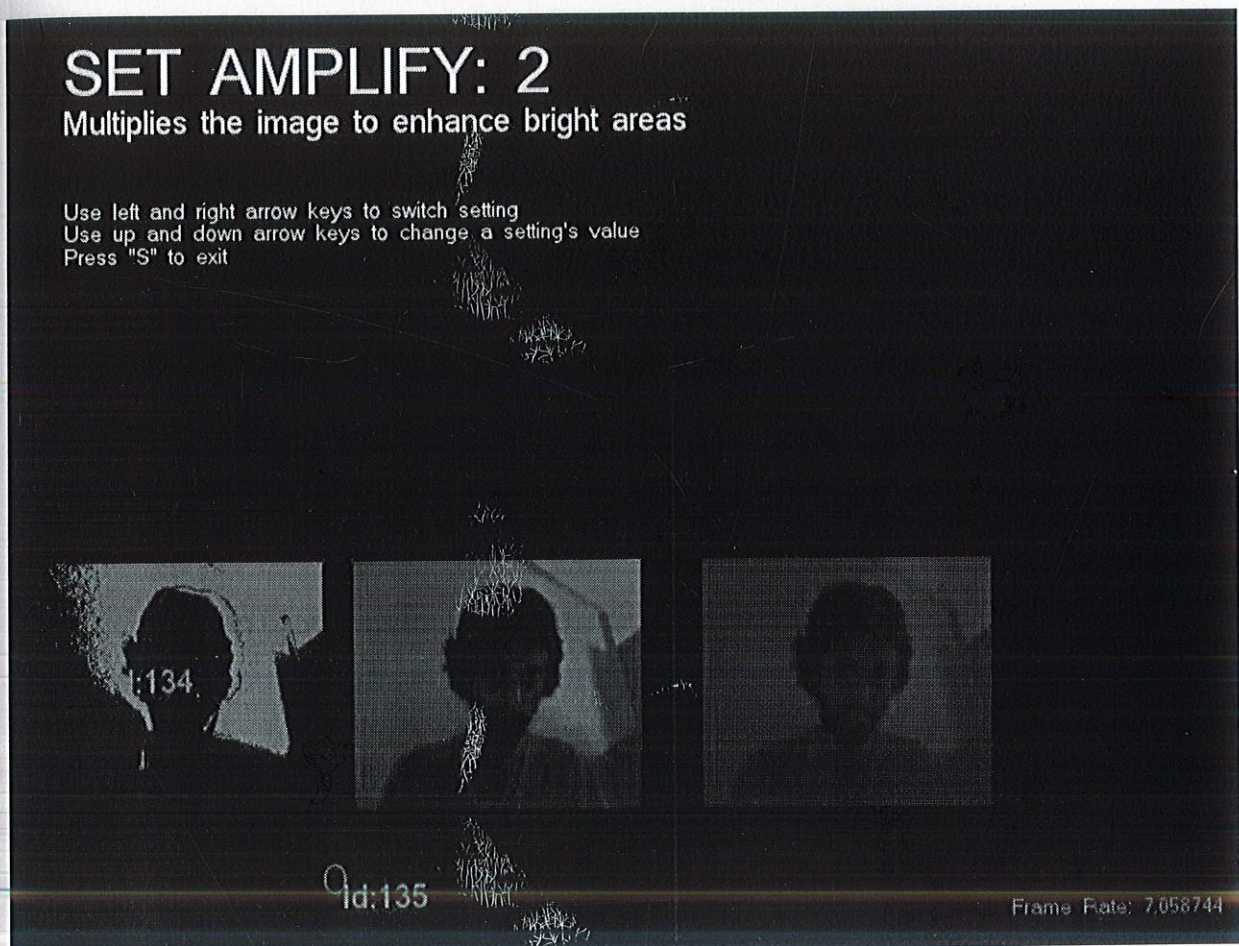


Figure 8.8

8.3.2 Simulated Blob Radius

Our application offers a testing functionality for the user called mouse blob. It can be used when a multi-touch screen is not available with the user. He may use the mouse as blob making device and test his multi-touch applications.

This particular functionality offers altering the size of this simulated mouse blob from 0-100 pixels

Figure 8.9 pictorially depicts the above explained functionality

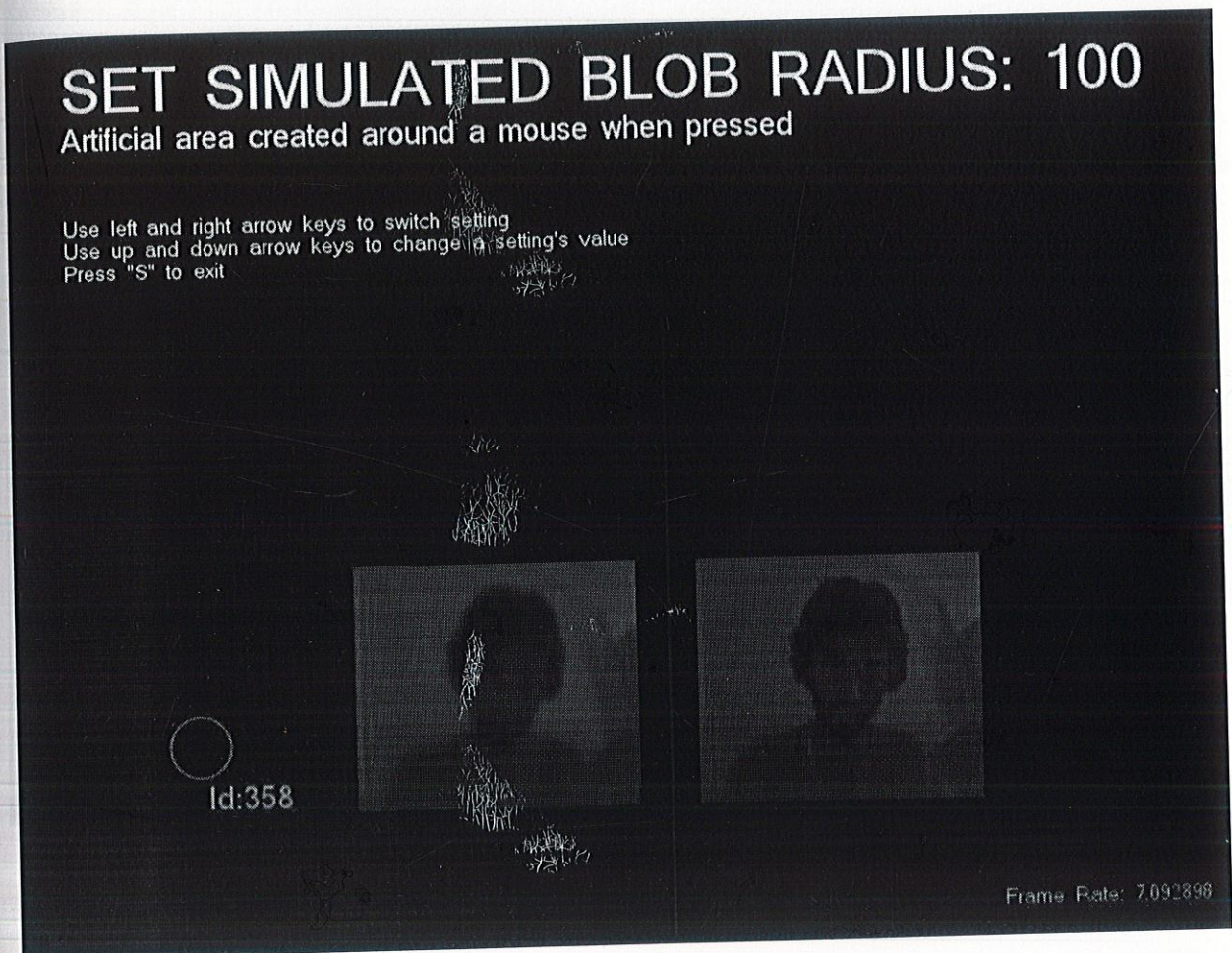


Figure 8.9

The above image shows a mouse blob with radius 100 and a blob id of 138

8.3.3 Setting Calibration Time

This functionality helps in altering the time required to capture one calibration point. This means a user has to keep his figure at the calibration point for this much amount of time

Figure 8.10 shows this functionality

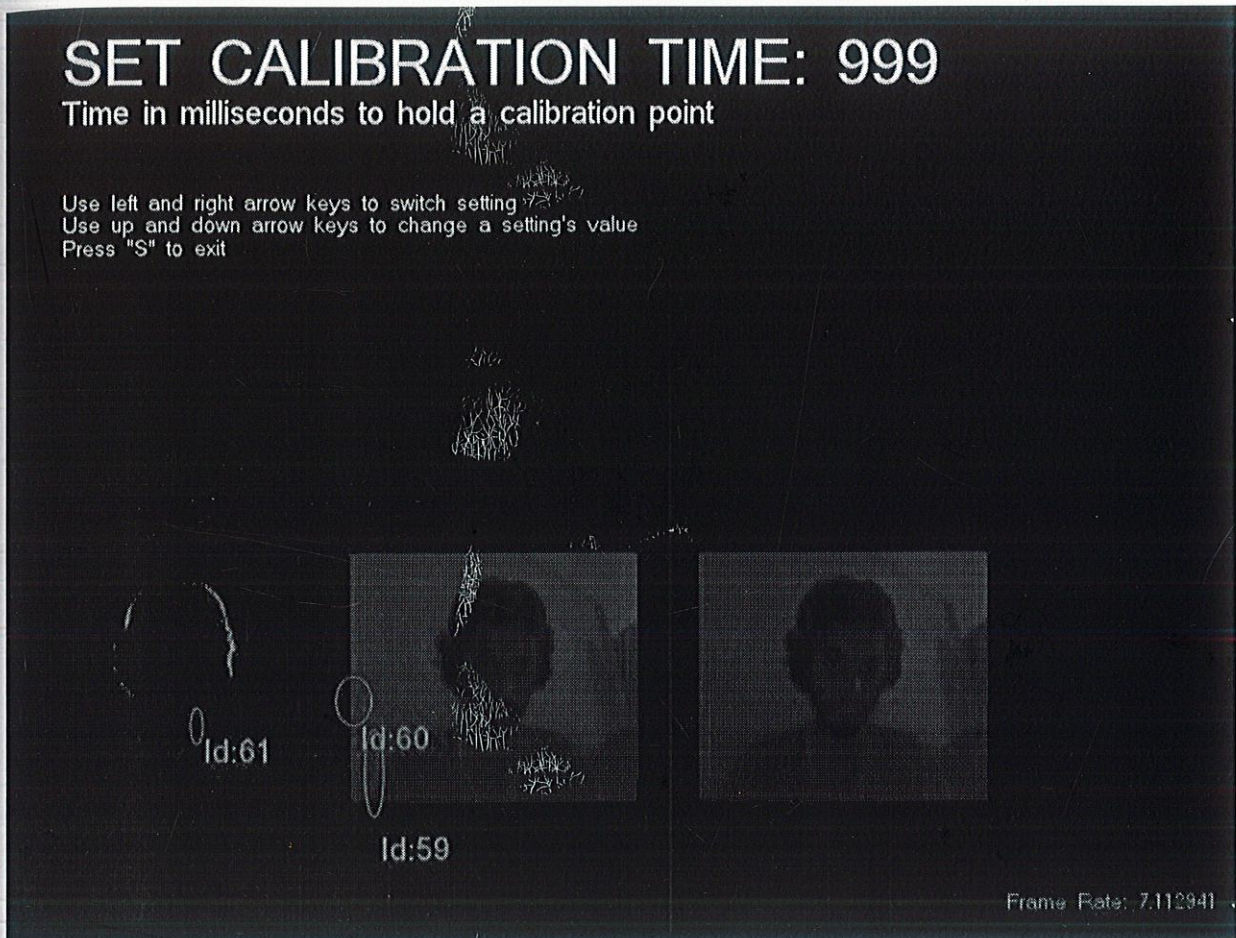


Figure8.10

The calibration time can range from 0-999 ms.

8.3.4 Contour Smoothing

The contours detected by the application connect the exact points. The user may need to smooth these connecting lines. This can be realized with the contour smoothing facility

Figure 8.11 depicts the same.

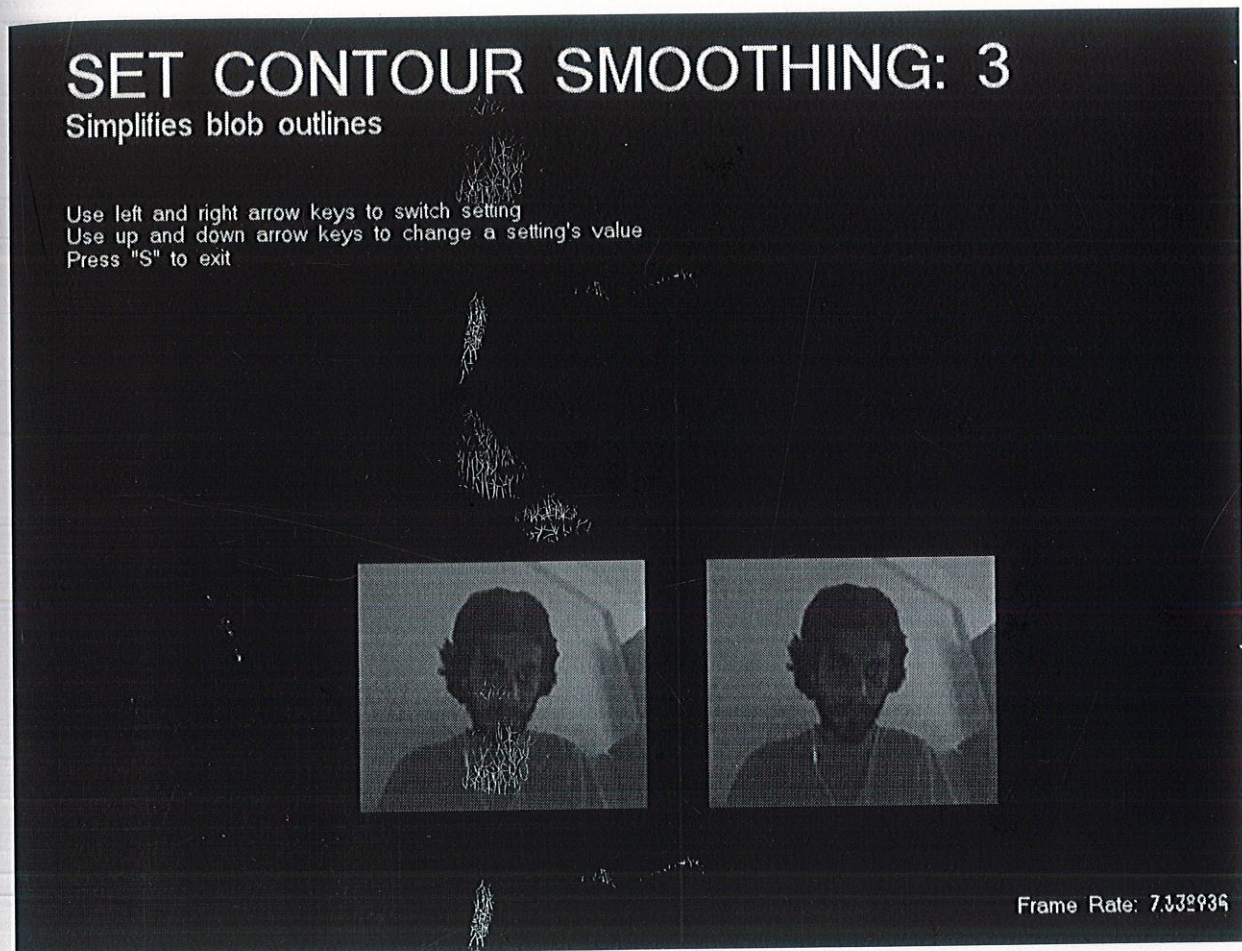


Figure 8.11

In the above image the contour smoothing value is set to 3. The user can increase/decrease this value according to what he desires.

8.3.5 Device Id

Our application allows capturing image from more than one camera. Thus, when more than one camera is connected to an application, the user may want different cameras to be used at different instances. He can do this by changing the device id and thus choosing the desired camera.

Figure 8.12 depicts how this can be done

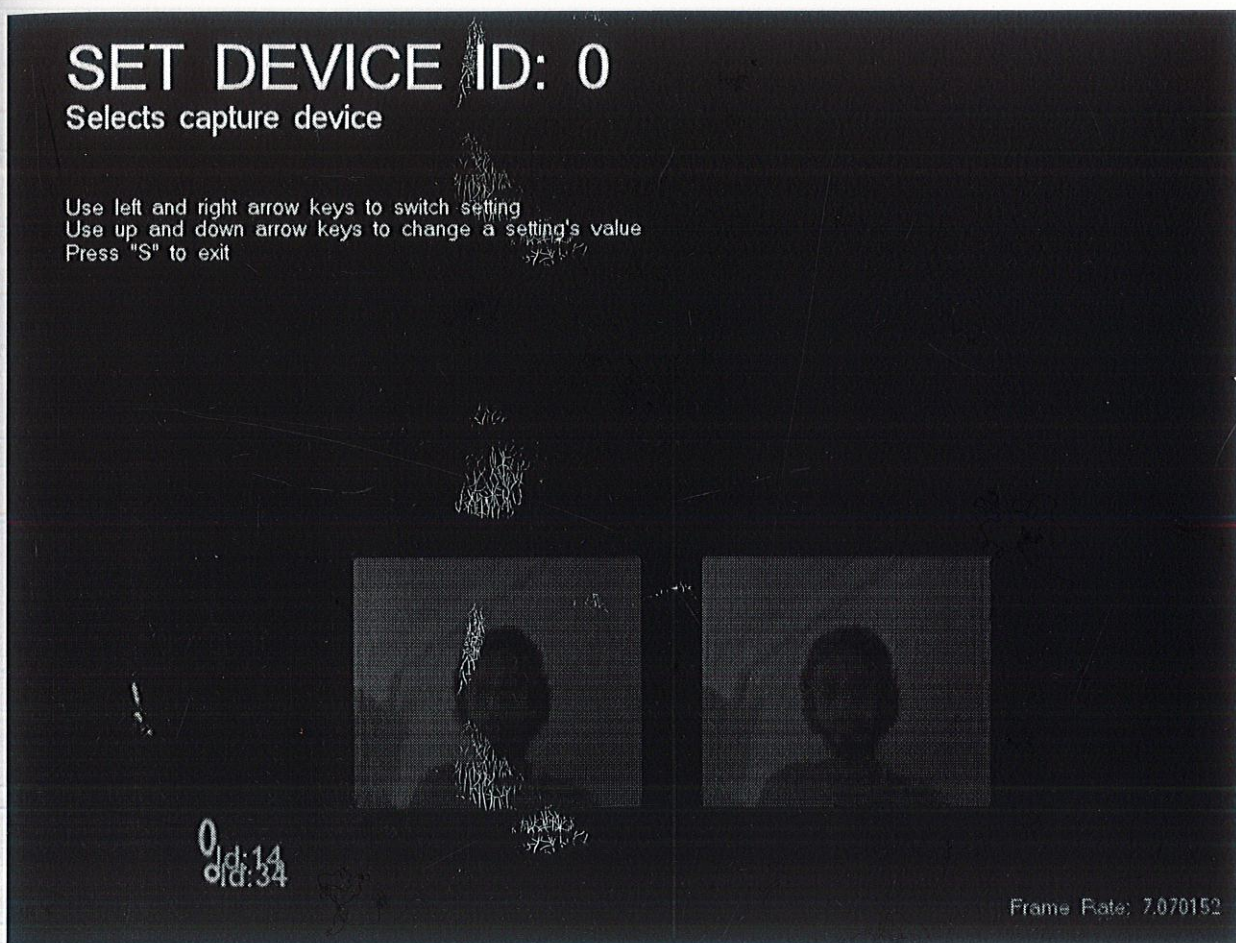


Figure 8.12

Default device id is 0, which is given to the first camera that is detected.

8.3.6 Flip Horizontal

When the camera is fixed to the touch screen, its direction may not be certain. Thus instead of rotating the camera, the application flips the live video feed and all the further processing is done using this flipped image.

Figure 8.13 shows how this is achieved.

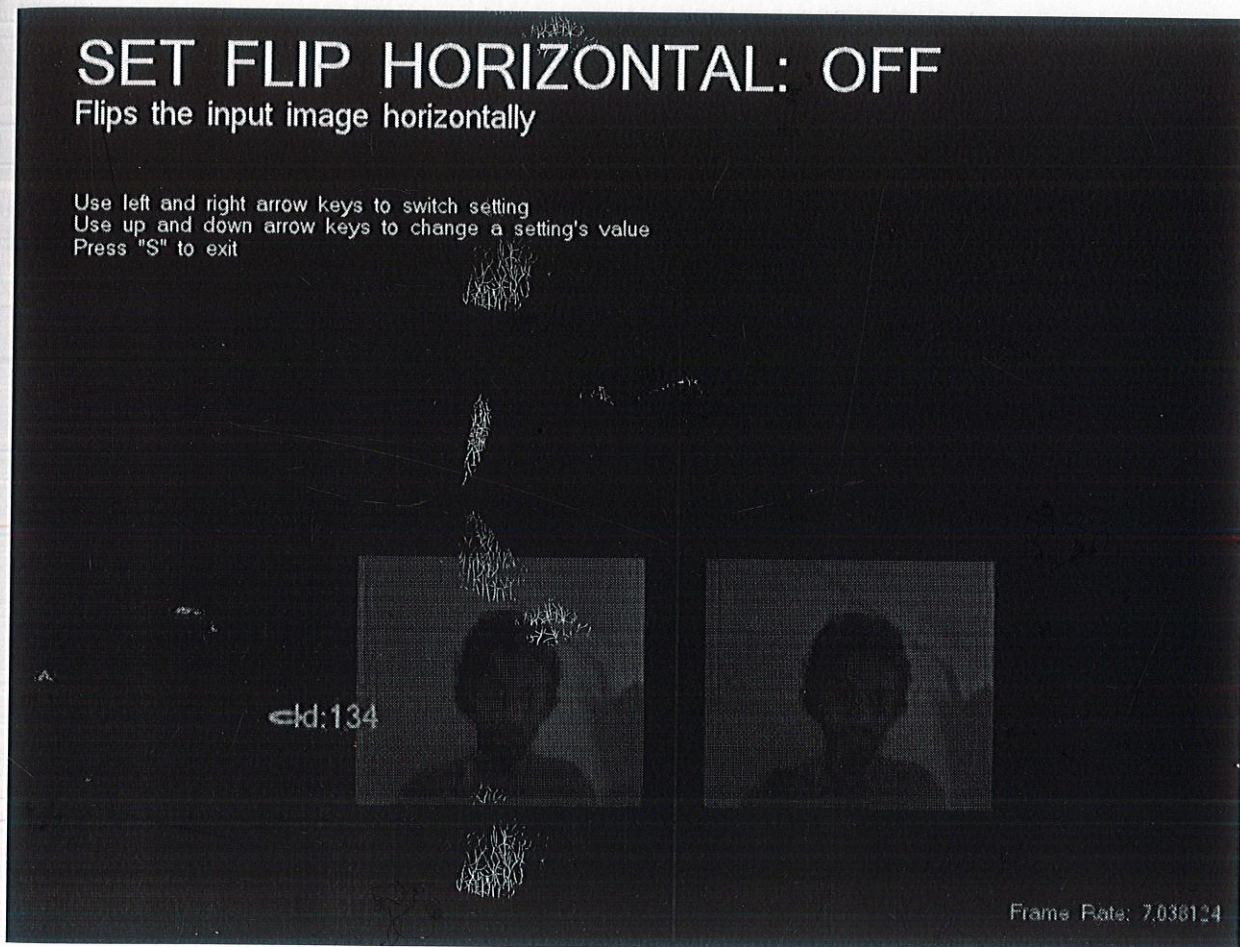


Figure 8.13

8.3.7 Flip Vertical

Is similar to 8.3.6 but, instead of flipping the video feed horizontally this function flips it vertically.

Figure 8.14 explains the same

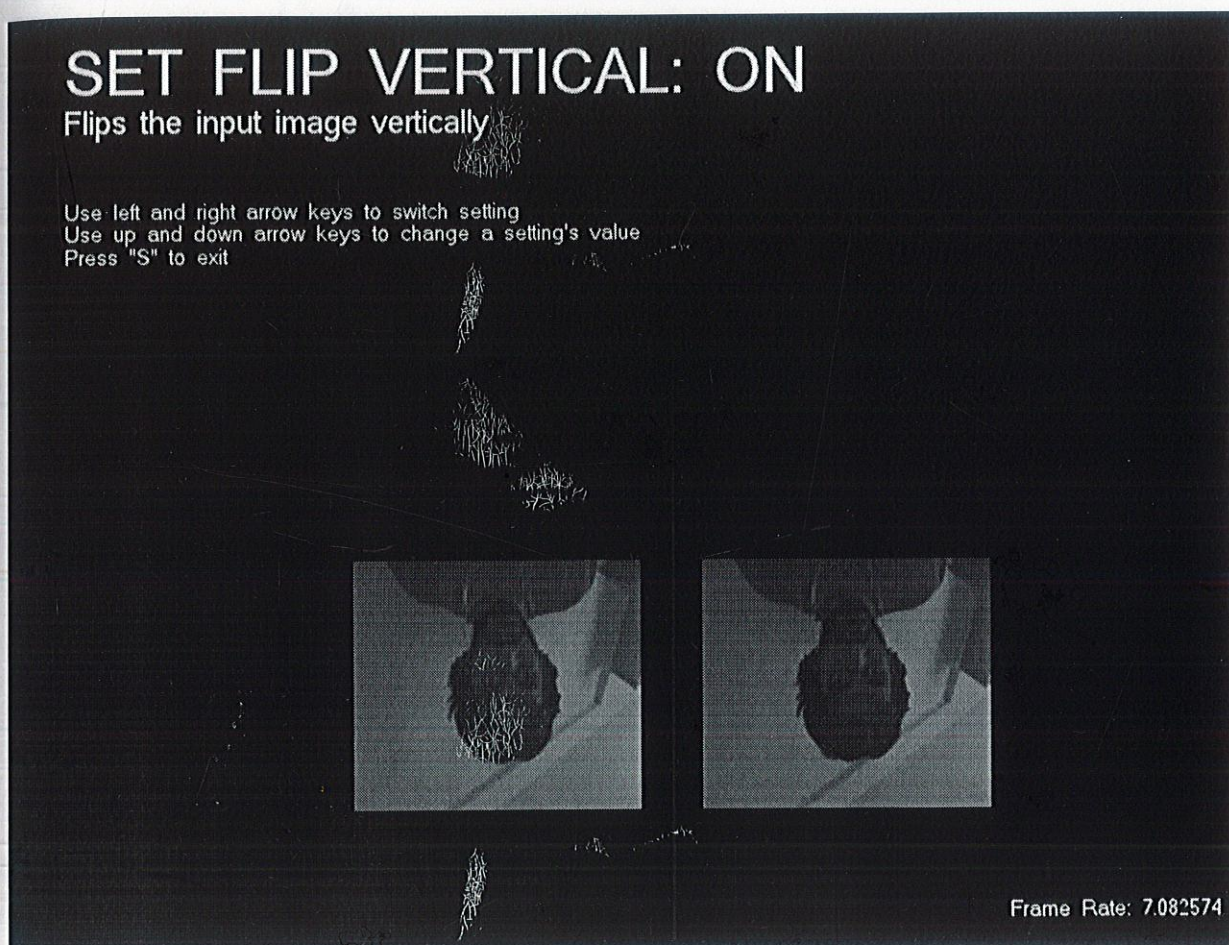


Figure 8.14

The flipped image is used for further image processing.

8.3.8 Dilate

It helps in dilation of the images. It thickens whatever is detected in the image.

Figure 8.15 shows this functionality

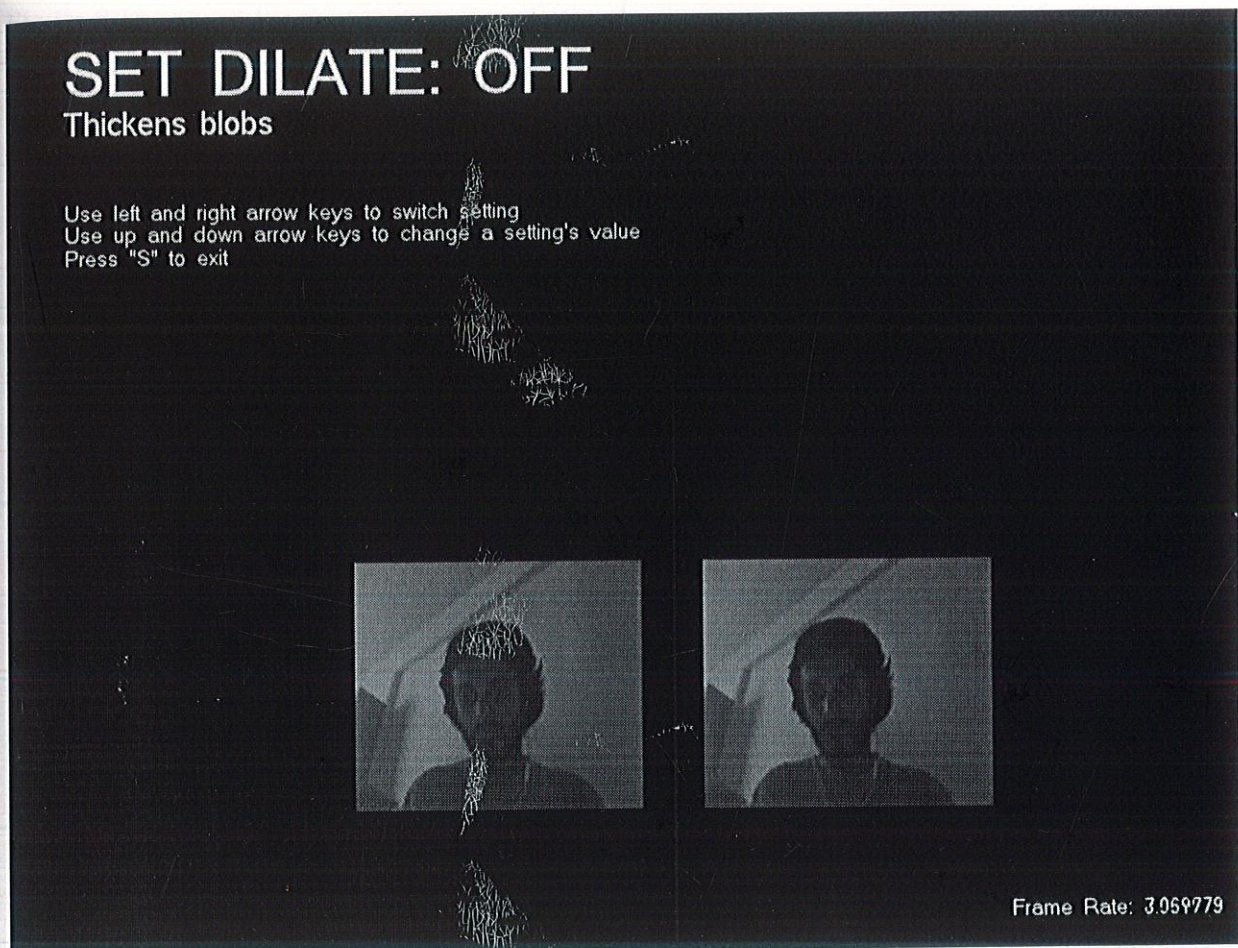


Figure 8.15

We use it when the detected blobs appear to be small in size.

8.3.9 Erode

Helps to implement the erode functionality of image processing. This is used to thin the detected blobs in the image.

Figure 8.16 shows this functionality

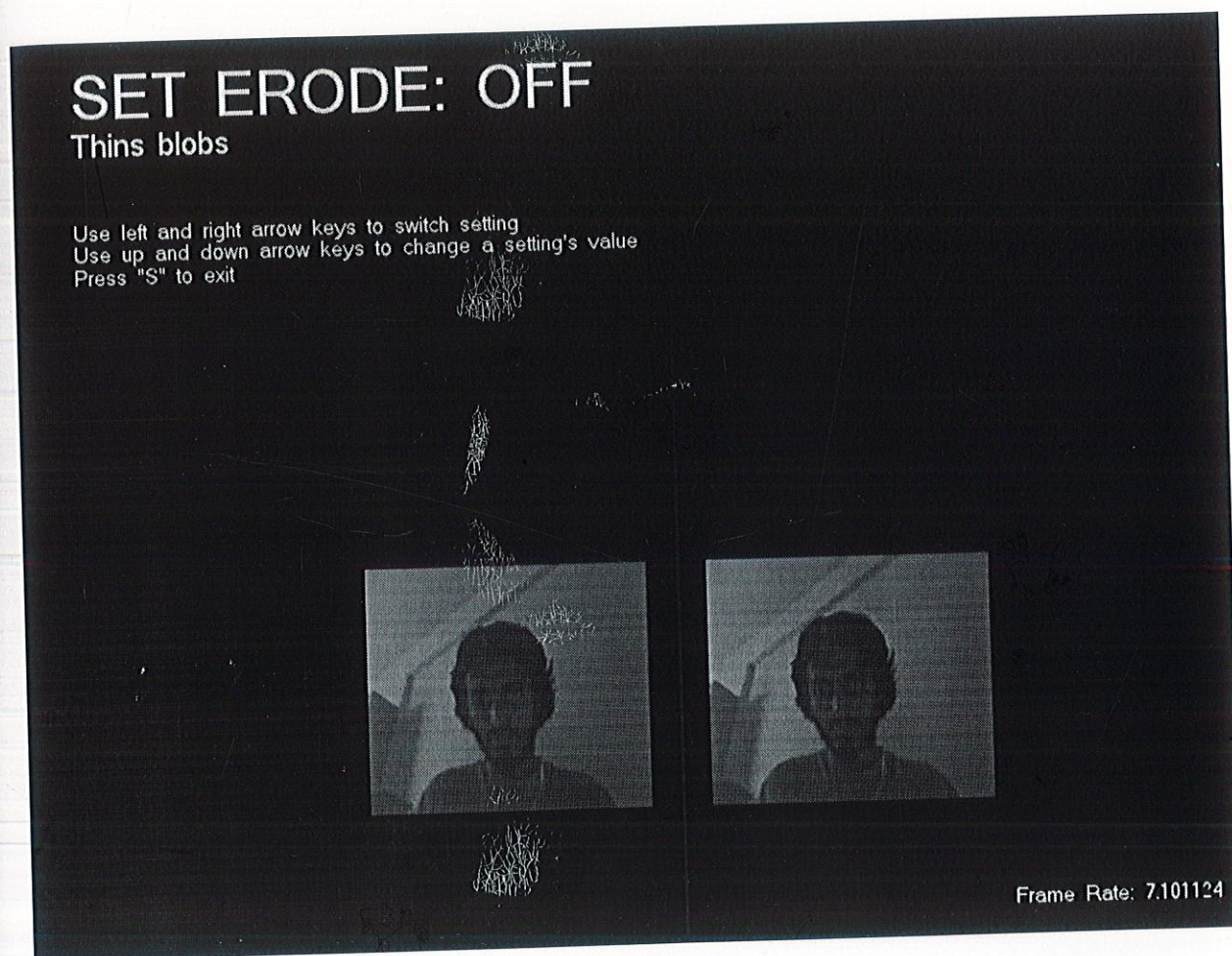


Figure 8.16

Used when the detected blobs are large.

8.3.10 High Pass Blur

Provides a high pass filter over the image. This is mostly used to detect the edges in the image. Only high frequencies are allowed to pass through.

Figure 8.17 shows this functionality

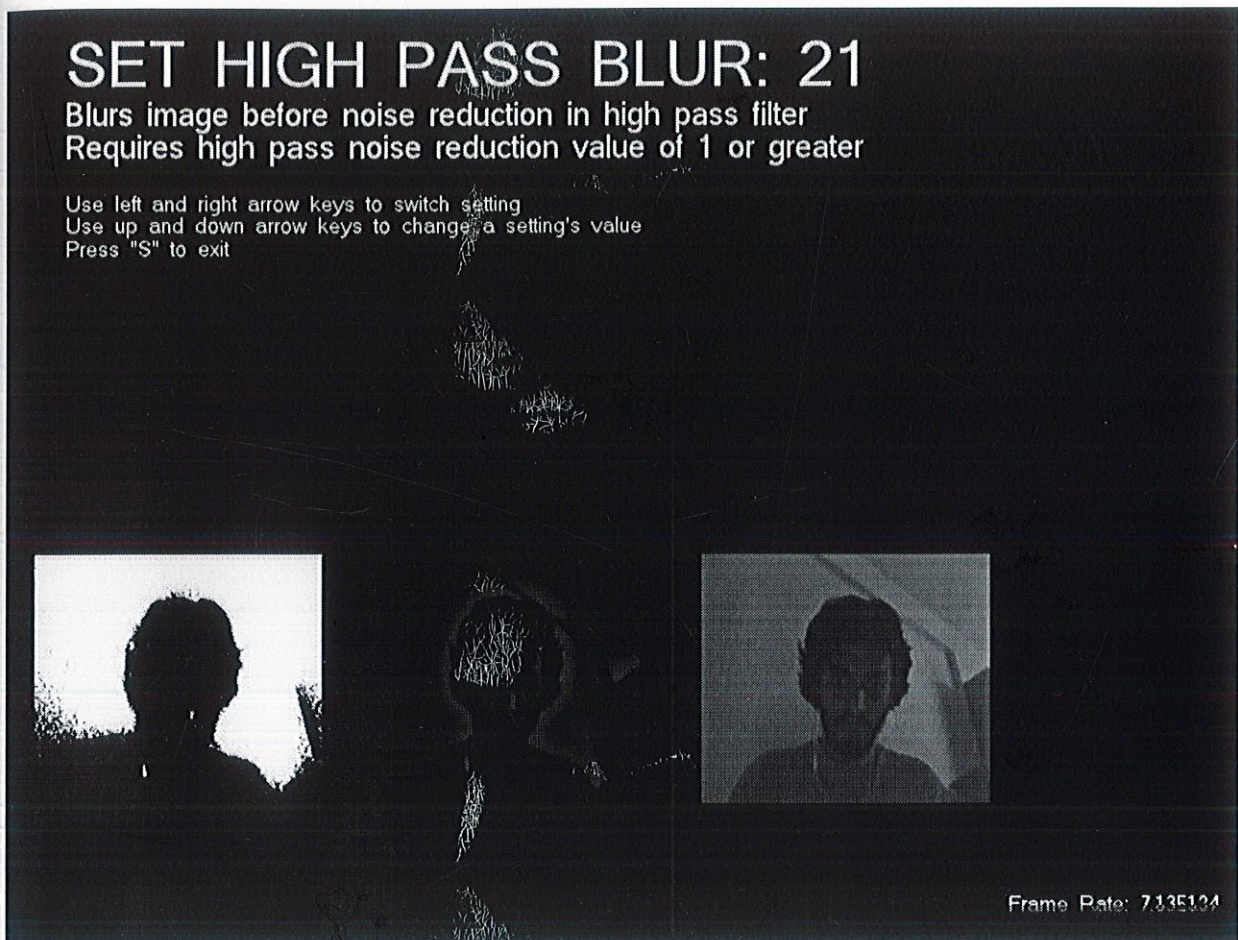


Figure 8.17

The above figure depicts that using these filters the edges of the object are obtained.

8.3.11 High Pass Noise Reduction

This provides blur functionality after the high pass filter has been applied. This is used to reduce the static noise after passing the video feed through the high pass filter. It is applicable only if the previous filter has been used.

Figure 8.18 shows this functionality

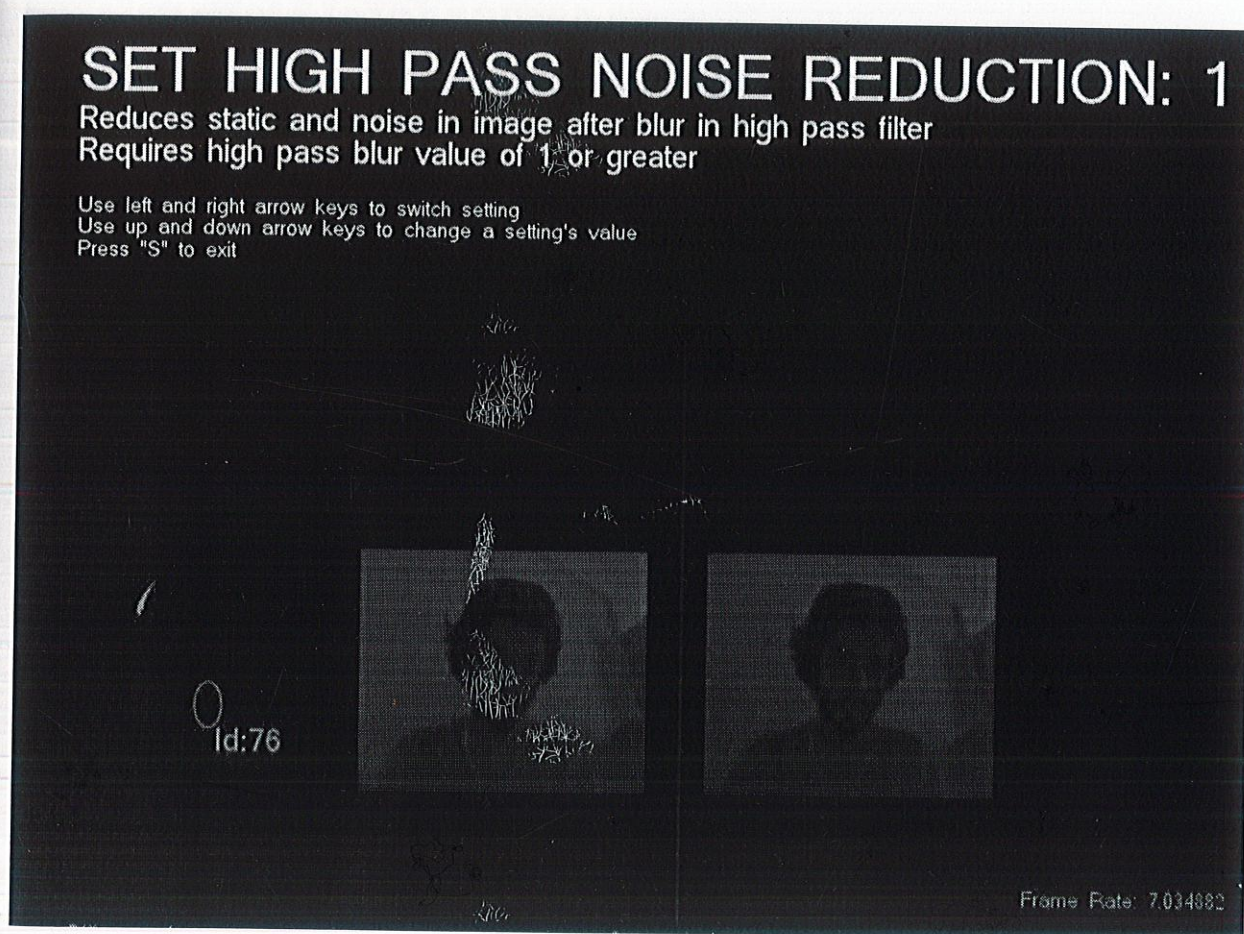


Figure 8.18

8.3.12 Smooth

This function helps in applying Gaussian blur to the video feed. Thus, removing most of the noise. This is the most important filter of the detection algorithm.

Figure 8.19 shows this functionality

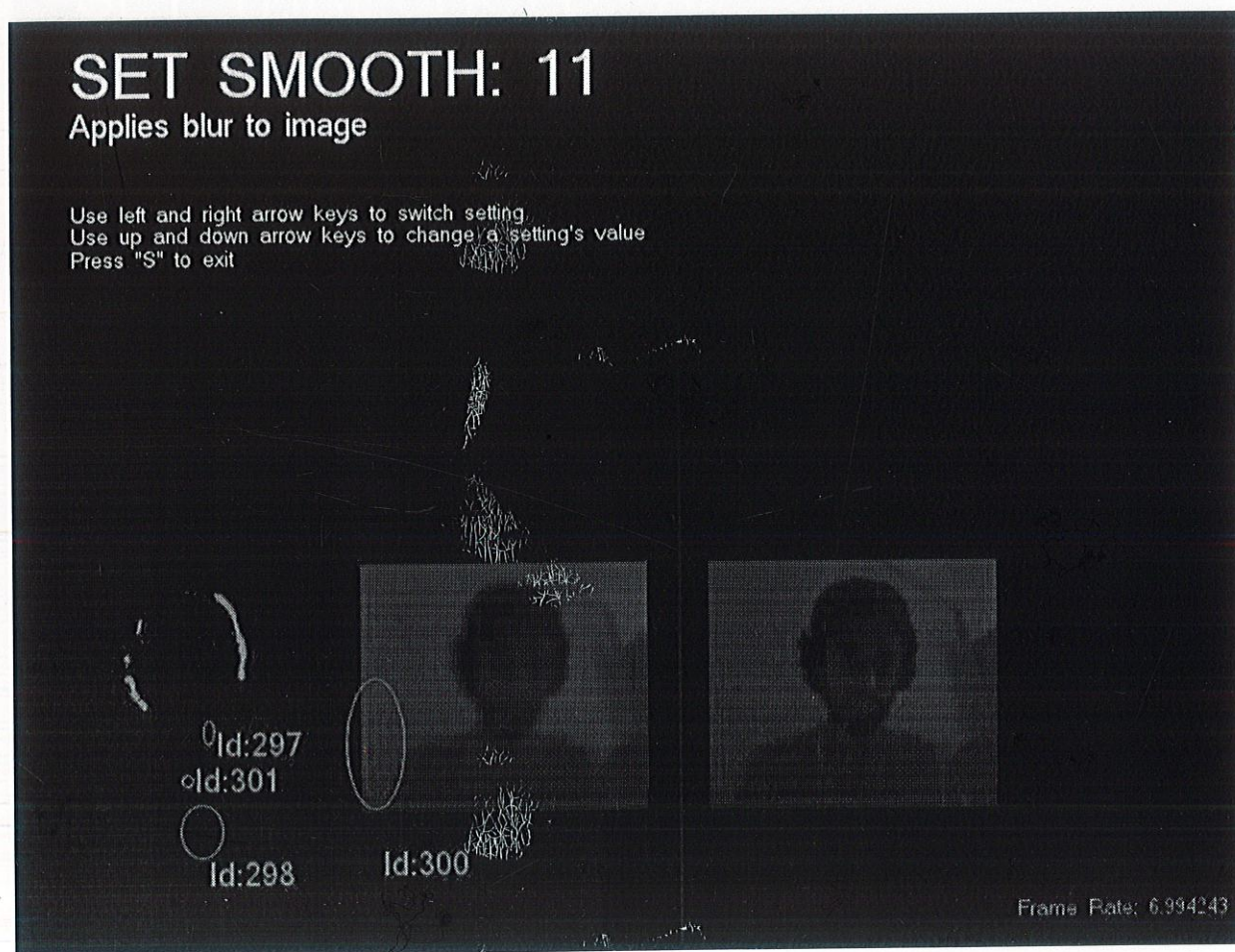


Figure 8.19

8.3.13 Threshold

This helps in applying a threshold value above which the grayscale values are to be treated as white. It plays an important role in detecting fingers from the video feed.

Figure 8.20 shows this functionality

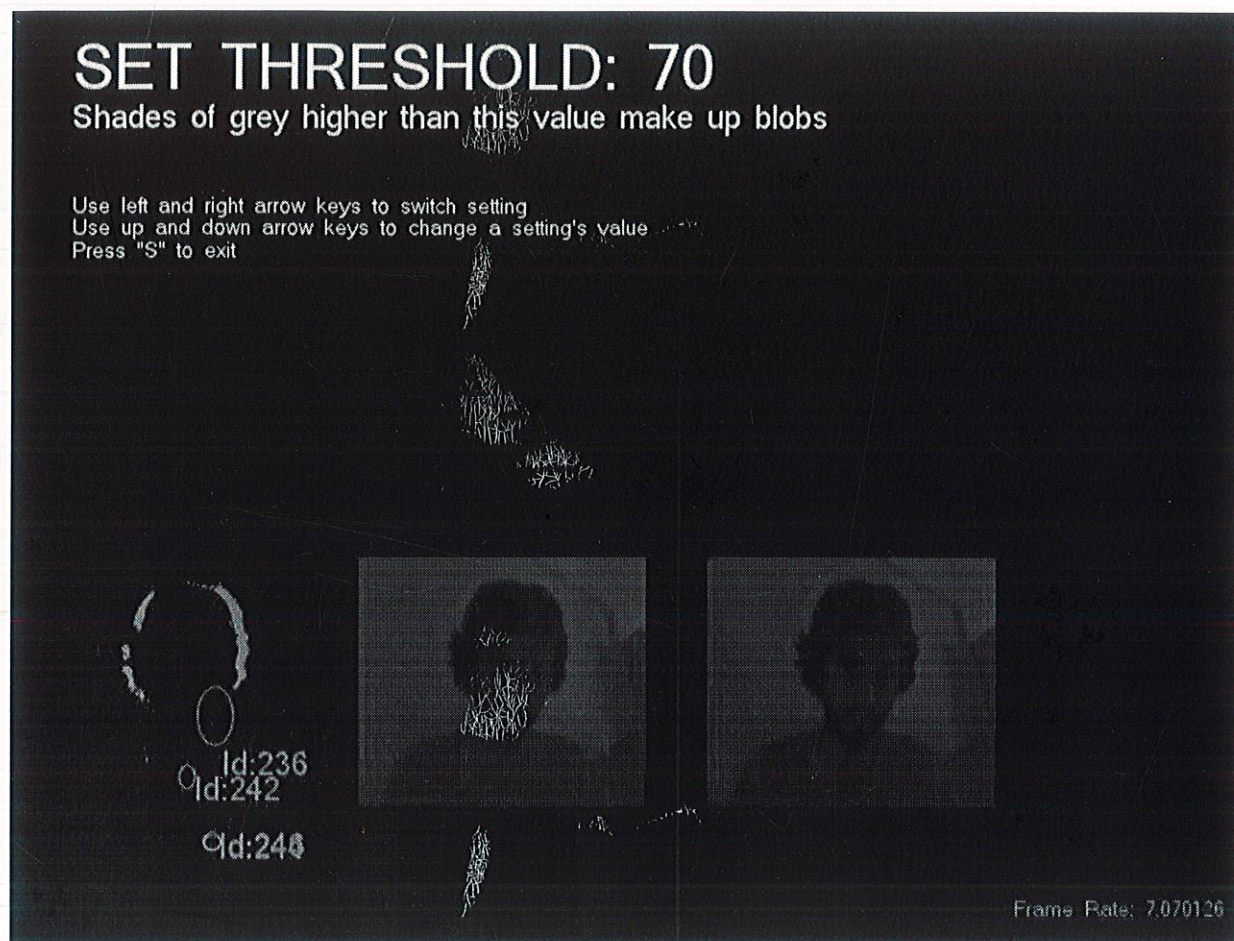


Figure8.20

Higher the value of threshold, less will be the visibility of weaker blobs and the stronger blobs will be detected and vice-versa.

8.3.14 Max Blobs

This defines the number of blobs that are to be detected. By choosing a value close to 5 which is for 5 fingers a user can remove false detecting effectively.

Figure 8.21 depicts the same

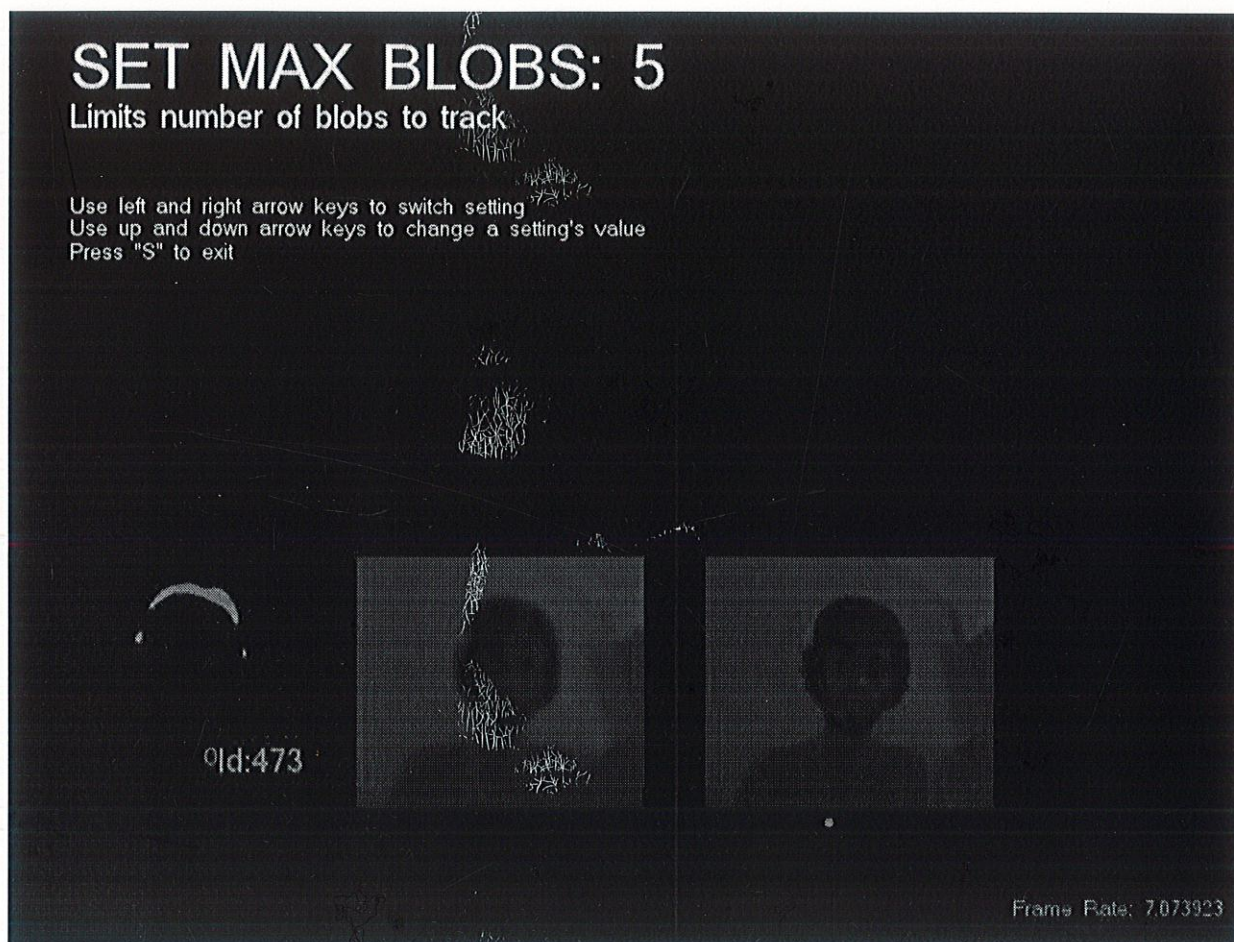


Figure 8.21

Here the max blobs that can be detected are 5

8.3.15 Max radius

This defines the maximum permissible radius (of the detected blobs). A blob with area greater than this will not be considered as a positive blob and will be safely discarded. The user may use the option to effectively track only finger sized blobs.

Figure 8.22 shows this functionality

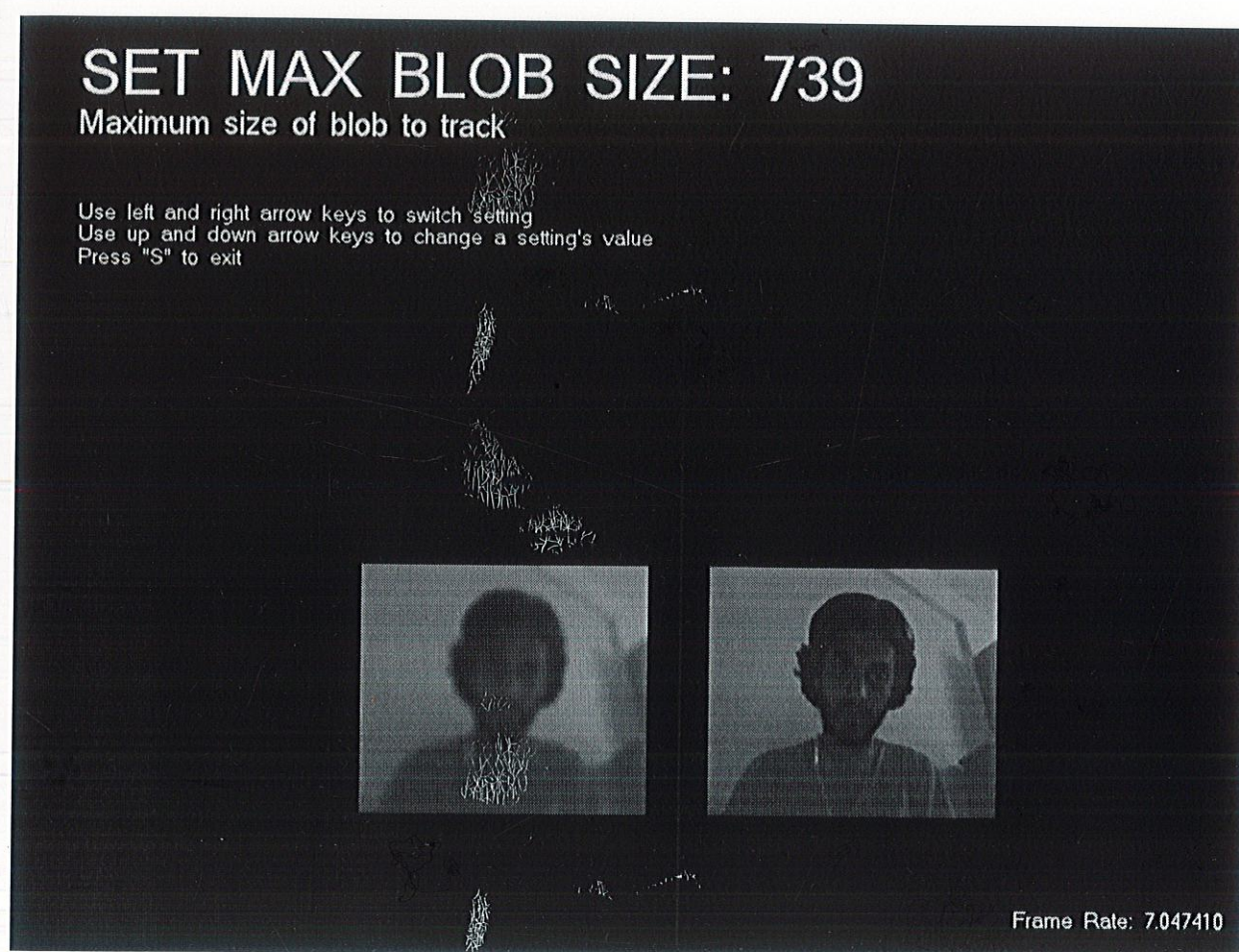


Figure 8.22

8.3.16 Minimum radius

This defines the minimum permissible radius (of the detected blobs). A blob with area lesser than this will not be considered as a positive blob and will be safely discarded. The user may use the option to effectively track only finger sized blobs.

Figure 8.23 shows this functionality

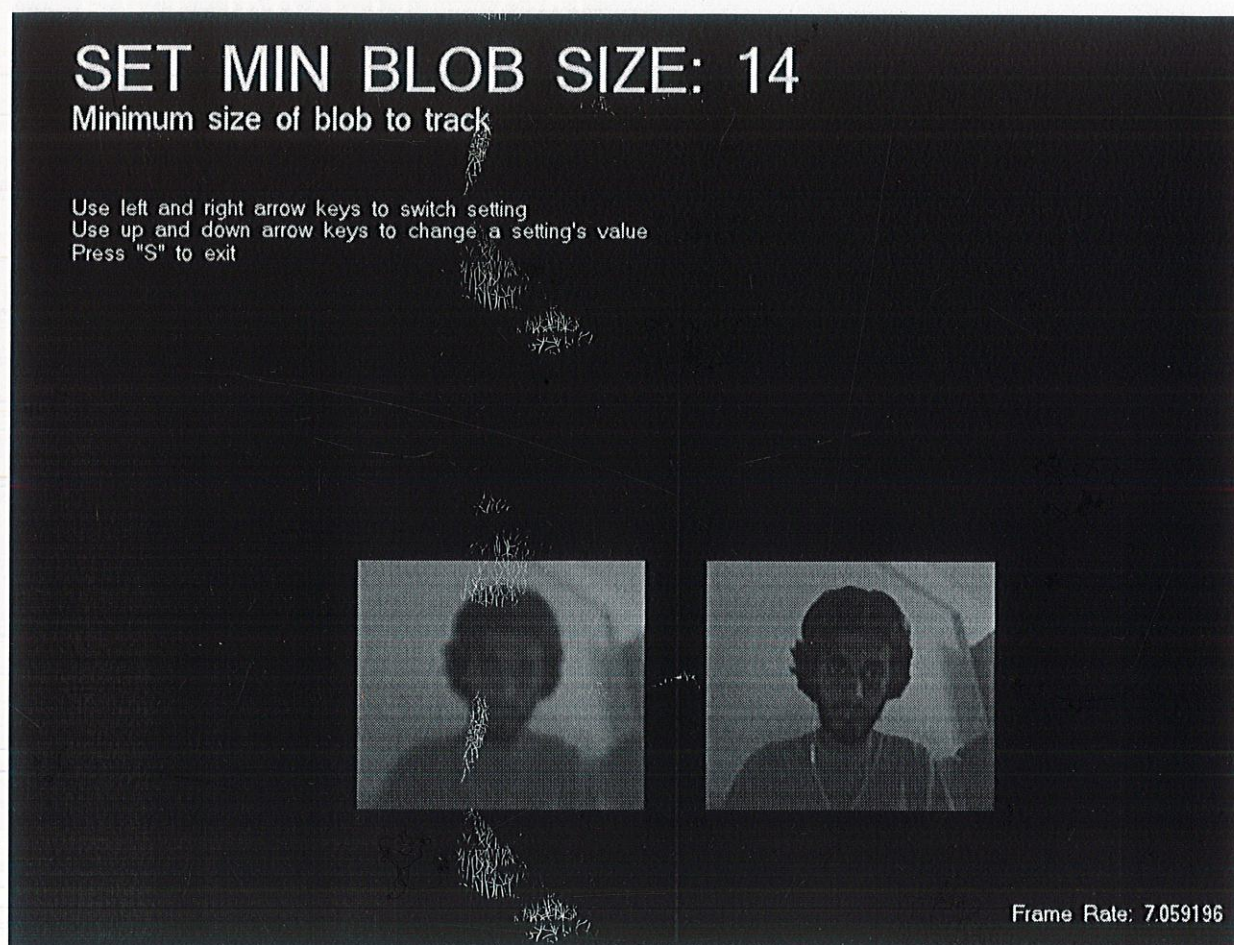


Figure 8.23

8.3.17 Invert

This functionality is for the rare event that the user wants to track dark blobs instead of the usual white ones.

Figure 8.24 depicts this functionality

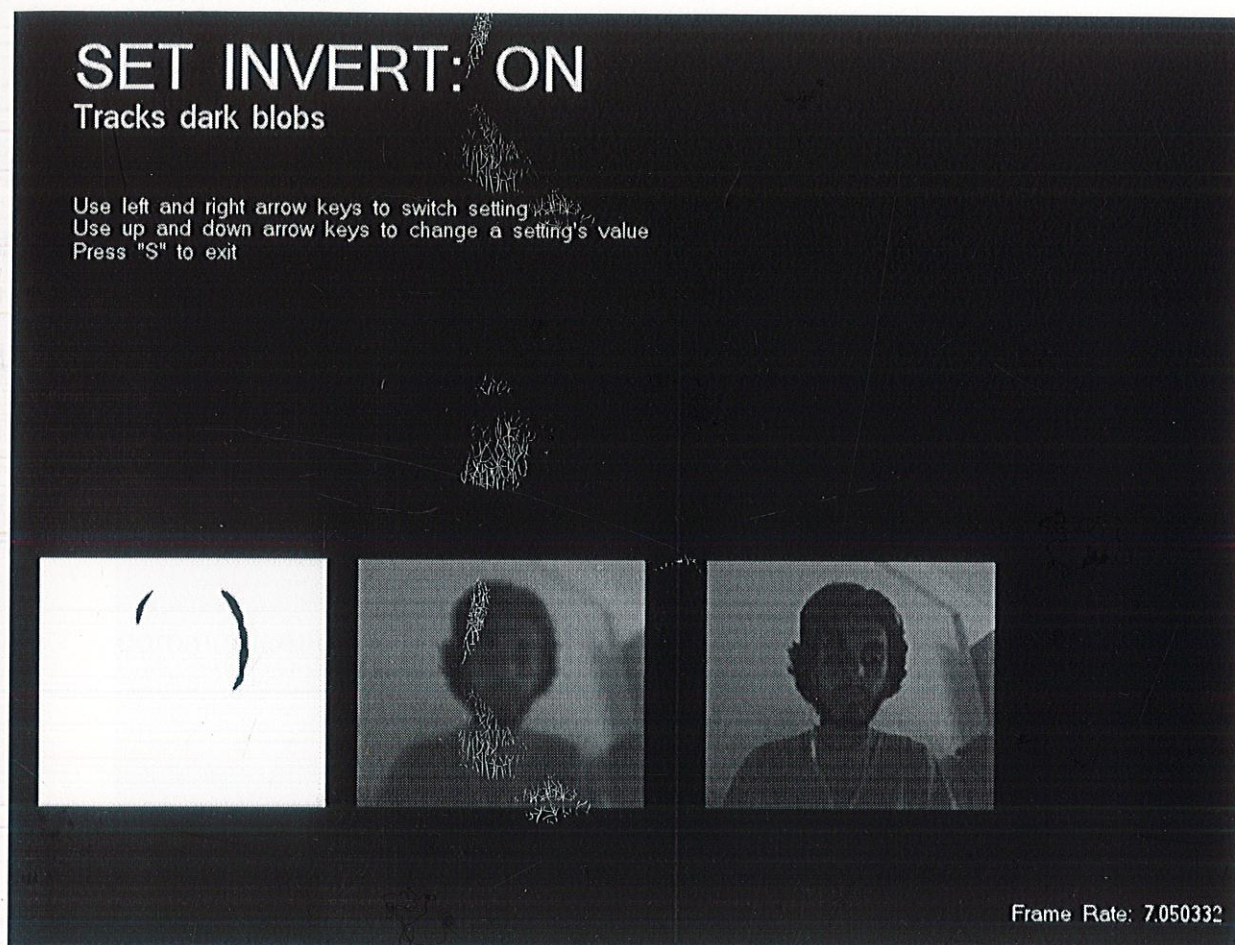


Figure 8.24

To send the blob data to the multi-touch application, the user has to press "T" after which the application starts sending the tracked data using the TUIO protocol.

9. SOFTWARE DEVELOPMENT LIFE CYCLE

9.1 Process Model

A software process model is an abstract representation of a process. It presents a description of a process from various perspective.

Generic software process models are:

- Prescriptive Models
There are separate and distinct phases of specification and development.
- Evolutionary Models
Specification, development and validation are interleaved.
- Specialized Process Models
The system is assembled from existing components.

Evolutionary Models: The Spiral

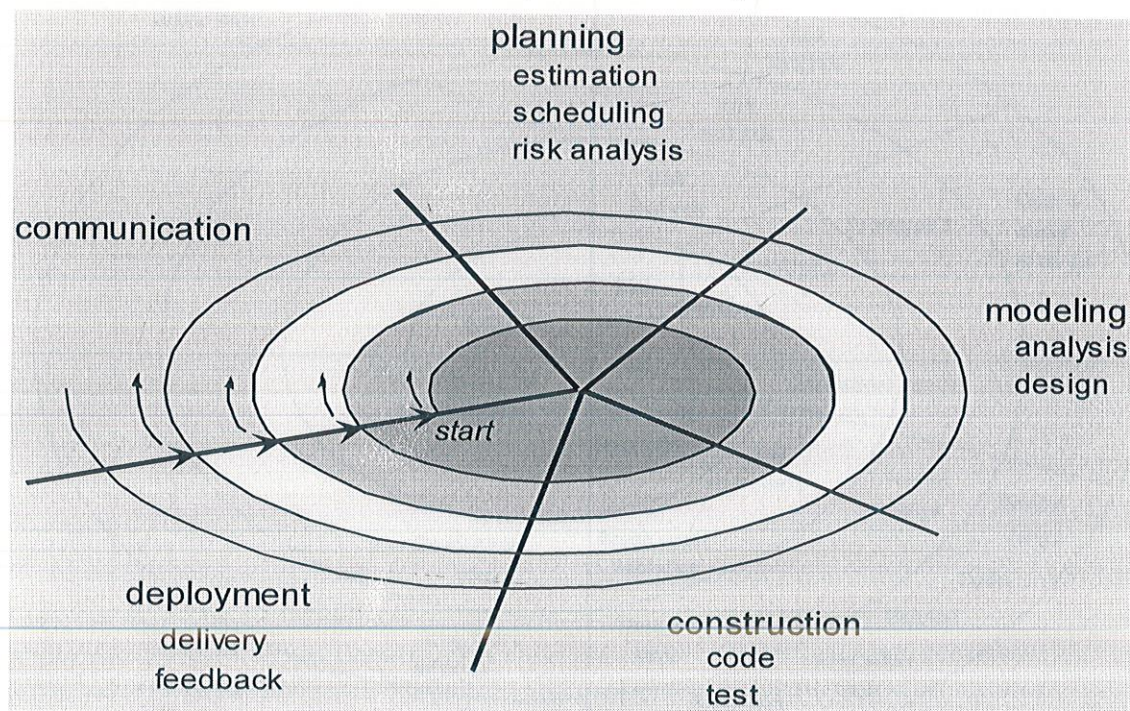


Figure 9.1

9.2 Spiral Development

- Process is represented as a spiral rather than as a sequence of activities with backtracking.
- Each loop in the spiral represents a phase in the process.
- No fixed phases such as specification or design, loops in the spiral are chosen depending on what is required.
- Risks are explicitly assessed and resolved throughout the process.

The process model followed during the course of this project is based on the Evolutionary model; it is the Spiral Model of software development.

Spiral Model couples the iterative nature of prototyping with controlled and systematic aspects of linear sequential model

Spiral Model of the software development process

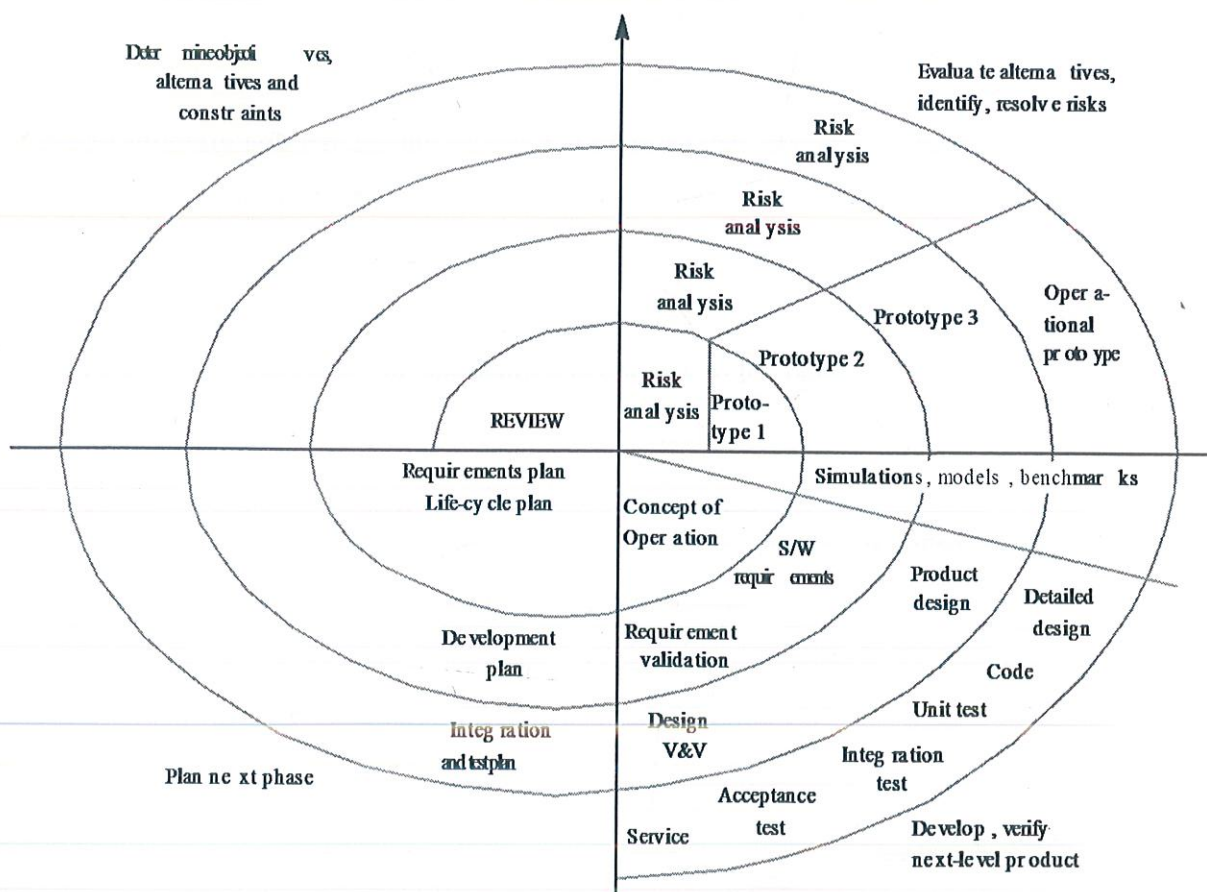


Figure 9.2

9.2.1 Spiral Model Sectors:

- Objective setting
Specific objectives for the phase are identified.
- Risk assessment and reduction
Risks are assessed and activities put in place to reduce the key risks.
- Development and validation
A development model for the system is chosen which can be any of the generic models.
- Planning
The project is reviewed and the next phase of the spiral is planned.

9.2.2 Spiral Model: Advantages

- The spiral model is a realistic approach to the development of large-scale systems and software. Before the assembling the hardware, mouse was used as an input device to provide blobs in place of input touches.
- Spiral model demands a direct consideration of technical risks at all stages of the project, and, if properly applied, should reduce risks before they become problematic. We implemented various algorithms and analysed them to obtain the best results. We discarded methods that increased the complexity.
- Incorporates iterative framework which reflects real world scenario. This was used in upgrading the modules of the application according to the need.

9.3 Use Case Model

The Diagram shows behavior of the system from the viewpoint of a user. It is concerned with what is provided rather than how it is provided.

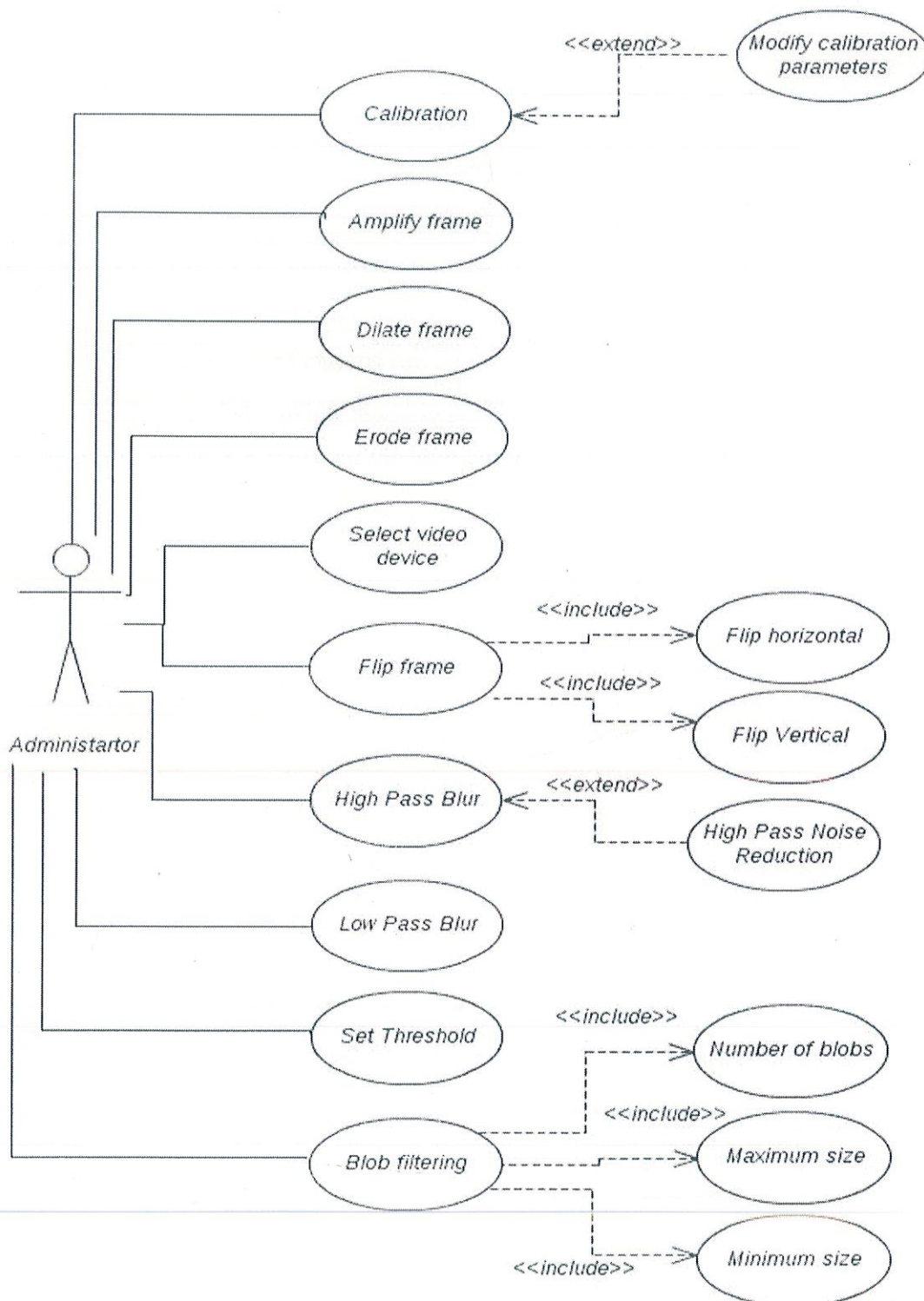


Figure 9.3

10. CONCLUSION

The project has been successfully concluded with the development of a prototype of the touch wall (screen). It is a multi touch table using glass as touch surface, LLP technology and a Logitech web camera, to detect the infrared blobs, and process them accordingly. The software has various features to control the image output. Also, Pymt applications are executed successfully on the multi touch table. We also implemented Multiplexing of data (finger coordinates) from the tracking applications running on different computers and coordinating with each other over the existing Ethernet protocol to provide scalability to the basic module. The only limitation of this setup is because of the ambient lighting around it due to external sources of infrared.

10. BIBLIOGRAPHY

- <http://www.nuiforum.net/>
- <http://www.highpassfilter.org/>
- <http://www.cs.uregina.ca/Links/class-info/425/Lab5/index.html>
- http://zone.ni.com/reference/en-XX/help/370051H-01/cvi/characteristics_of_an_ideal_filter/
- <http://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm>
- <http://en.wikipedia.org/wiki/Touchscreen>
- <http://www.mobileburn.com/definition.jsp?term=resistive+touchscreen>
- <http://stackoverflow.com/questions/4759194/finding-equivalent-gaussian-filter-mask-in-frequency-domain-for-given-mask-in-sp>
- <http://opencv.willowgarage.com/wiki/cvBlobsLib>
- <http://opencv.itseez.com>
- <http://www.lasermate.com/module.html>
- http://peauproductions.com/store/index.php?main_page=index&cPath=16&zenid=efe664a1b874e8dc81f147eba4d8606e
- <http://pymt.eu/>
- <http://pymt.eu/docs/api>
- <http://www.tuio.org/>
- <http://www.tuio.org/processing>
- Canny Edge Detection 09gr820 March 23, 2009
This document explains the working of a hignpass filter to detect edges of any image.
- Canny Edge Detection Tutorial Author: Bill Green (2002)
- <http://www.instructables.com/id/PIR-Motion-Sensor-Tutorial/>
This webpage explains the working of PIR motion sensors, these sensors have high cost and hence would not be fruitful in such kind of project.
- <http://www.gloolab.com/pirparts/infrared.html>

This webpage gives the explanation on how the infrared motion detector components work. This project does not use this methodology.

- <http://homepages.inf.ed.ac.uk/rbf/HIPR2/label.htm>

This webpage explains the concept of connected component labeling, 4-connected and 8-connected components.

- <http://www.keithlantz.net/2011/05/a-calibration-method-based-on-barycentric-coordinates-for-multi-touch-systems/>

This document explains the concept of using barycentric coordinate system for calibration

- Background subtraction techniques:a review Massimo Piccardi

Computer Vision Research Group (CVRG) University of Technology, Sydney (UTS)
e-mail: massimo@it.uts.edu

This set of slides gives an insight on how background subtraction is performed on images through various methods after an input is accepted.

- Rough fuzzy image analysis-- Foundations and Methodologies Shankar K. Pal and James F. Peters

This book explains the concept of image analysis using fuzzy logic techniques.

- ACM SIGGRAPH@UIUC ACM SIGGRAPH@UIUC Fast Image Convolutions Fast Image Convolutions by: Wojciech Jarosz

This set of slides explains the technique for proper and fast convolution of filters on images.