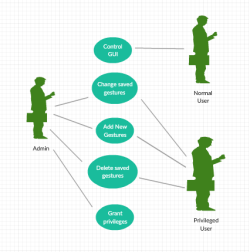
Project title

CS18L1 Project

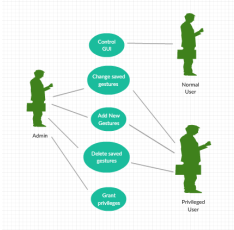
regno rollno name
B. Tech Computer Science & Engineering



Department of Computer Engineering
Model Engineering College
Thrikkakara, Kochi 682021
Phone: +91.484.2575370
http://www.mec.ac.in
hodcs@mec.ac.in

April 2017

Model Engineering College Thrikkakara
Department of Computer Engineering



C E R T I F I C A T E

This is to certify that, this report titled ***Project title*** is a bonafide record of the work done by

regno rollno name

Eighth Semester B. Tech. Computer Science & Engineering

students, for the course work in **CS18L1 Project**, which is the second part of the two semester project work, under our guidance and supervision, in partial fulfillment of the requirements for the award of the degree, B. Tech. Computer Science & Engineering of **Cochin University of Science & Technology**.

Guide

Name of guide
Assistant Professor
Computer Engineering

Coordinator

Dr. Priya S
Associate Professor
Computer Engineering

Head of the Department

Manilal D L
Associate Professor
Computer Engineering

May 6, 2019

Acknowledgements

This project would not have been possible without the kind support and help of many individuals. We would like to extend my sincere thanks to all of them.

First of all, We would like to thank our esteemed Principal, Prof. (Dr.) V.P Devassia, for his guidance and support in maintaining a calm and refreshing environment to work in and also for providing the facilities that this work demanded.

We am highly indebted to our Project Coordinator, Dr. Priya S, Associate Professor and Head of the Department, Dr. Manilal D L, Associate Professor for their guidance, support and constant supervision throughout the duration of the work as well as for providing all the necessary information and facilities that this work demanded.

We would like to thank our Project Guide, ... for his/her support and valuable insights and also for helping me out in correcting any mistakes that were made during the course of the work.

We offer our sincere gratitude to all our friends and peers for their support and encouragement that helped me get through the tough phases during the course of this work.

Last but not the least, we thank the Almighty God for guiding me through and enabling us to complete the work within the specified time.

`name`

Abstract

Abstract of the project

Contents

List of Figures **iv**

List of Tables **v**

1 Introduction **1**
1.1 Proposed Project . . . 1
1.1.1 Problem Statement . . . 1
1.1.2 Proposed Solution . . . 1

2 System Study Report **2**
2.1 Literature Survey . . . 2
2.2 Proposed System . . . 2

3 Software Requirement Specification **3**
3.1 Introduction . . . 3
3.1.1 Purpose . . . 3
3.1.2 Document Conventions . . . 3
3.1.3 Intended Audience and Reading Suggestions . . . 3
3.1.4 Project Scope . . . 3
3.1.5 Overview of Developers Responsibilities . . . 3
3.2 Overall Description . . . 4
3.2.1 Product Perspective . . . 4
3.2.2 Product Functions . . . 4
3.2.3 User Classes and Characteristics . . . 4
3.2.4 Operating Environment . . . 4
3.2.5 Design and Implementation Constraints . . . 4
3.2.6 User Documentation . . . 4
3.2.7 General Constraints . . . 4
3.2.8 Assumptions and Dependencies . . . 4
3.3 External Interface Requirements . . . 5
3.3.1 User Interfaces . . . 5
3.3.2 Hardware Interfaces . . . 5
3.3.3 Software Interfaces . . . 5
3.3.4 Communication Interfaces . . . 5
3.4 Hardware and Software Requirements . . . 6
3.4.1 Hardware Requirements . . . 6

3.4.2 Software Requirements . . . 6
3.5 Functional Requirements . . . 7
3.6 Non-functional Requirements . . . 8
3.7 Other Requirements . . . 8

4 System Design **9**
4.1 System Architecture . . . 9
4.2 Input Design . . . 9
4.3 Database Design . . . 9
4.4 Libraries and Packages Used . . . 9
4.5 Module Description . . . 9

5 Data Flow Diagram **10**
5.1 Level 0 . . . 10
5.2 Level 1 . . . 10
5.3 Level 2 . . . 11

6 Implementation **12**
6.1 Algorithms . . . 12
6.1.1 Overall algorithm . . . 12
6.1.2 Background subtraction . . . 12
6.1.3 Convex Hull . . . 13
6.1.4 CAMShift . . . 13
6.1.5 Brute Force Matching . . . 13
6.2 Development Tools . . . 14
6.2.1 Sublime Text . . . 14

7 Testing **15**
7.1 Testing Methodologies . . . 15
7.2 Unit Testing . . . 16
7.2.1 Background Subtraction Module . . . 16
7.2.2 Feature Extraction . . . 17
7.2.3 FingerTip Tracking and Gesture Recognition . . . 17
7.3 Integration Testing . . . 18
7.4 System Testing . . . 18

8 Graphical User Interface **19**
8.1 GUI Overview . . . 19
8.2 Main GUI Components . . . 19

9 Results **20**

10 Conclusion **21**

11 Future Scope **22**

12 Publication **23**

References **24**

List of Figures

Figure 7.1: Background Subtraction Module 16
Figure 7.2: Feature Extraction Module 17
Figure 7.3: Finger Tip Tracking and Gesture Recognition 18

List of Tables

Chapter 1

Introduction

1.1 Proposed Project

1.1.1 Problem Statement

1.1.2 Proposed Solution

Chapter 2

System Study Report

2.1 Literature Survey

2.2 Proposed System

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

3.1.2 Document Conventions

3.1.3 Intended Audience and Reading Suggestions

3.1.4 Project Scope

3.1.5 Overview of Developers Responsibilities

3.2 Overall Description

3.2.1 Product Perspective

3.2.2 Product Functions

3.2.3 User Classes and Characteristics

3.2.4 Operating Environment

3.2.5 Design and Implementation Constraints

3.2.6 User Documentation

3.2.7 General Constraints

3.2.8 Assumptions and Dependencies

3.3 External Interface Requirements

3.3.1 User Interfaces

3.3.2 Hardware Interfaces

3.3.3 Software Interfaces

3.3.4 Communication Interfaces

3.4 Hardware and Software Requirements

3.4.1 Hardware Requirements

3.4.2 Software Requirements

3.5 Functional Requirements

3.6 Non-functional Requirements

3.7 Other Requirements

Chapter 4

System Design

4.1 System Architecture

4.2 Input Design

4.3 Database Design

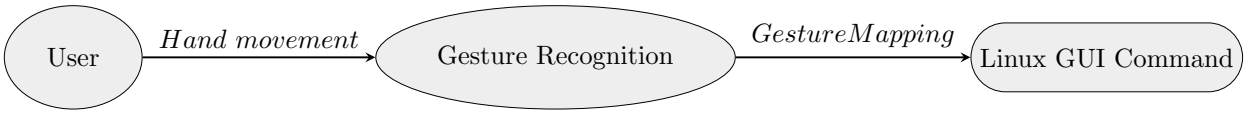
4.4 Libraries and Packages Used

4.5 Module Description

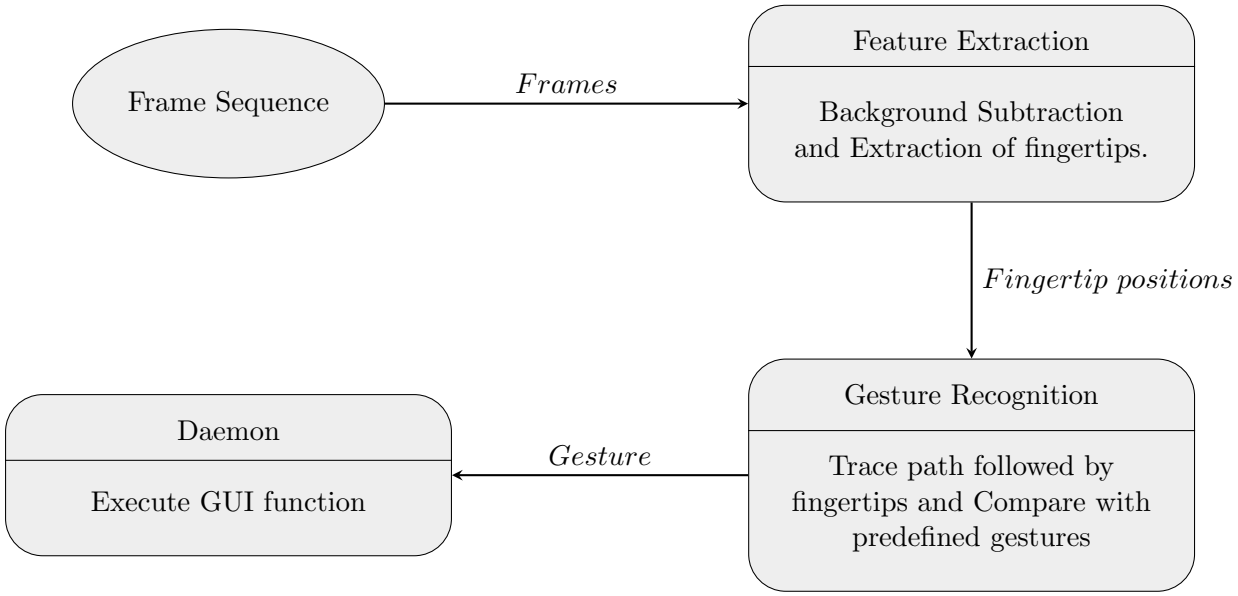
Chapter 5

Data Flow Diagram

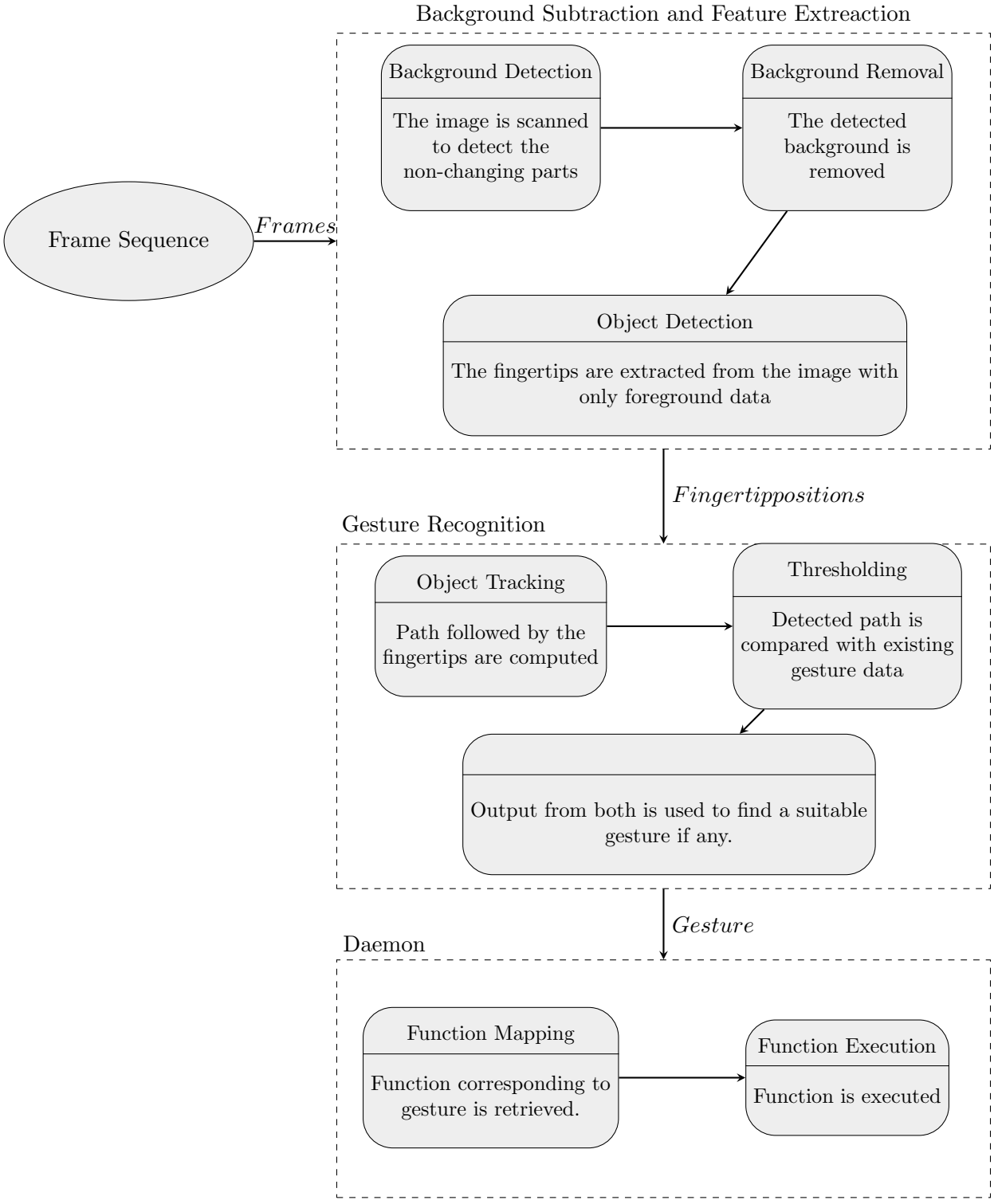
5.1 Level 0



5.2 Level 1



5.3 Level 2


Background Subtraction and Feature Extreaction
Background Detection
The image is scanned to detect the non-changing parts
Background Removal
The detected background is removed
Frame Sequence
Frames
Object Detection
The fingertips are extracted from the image with only foreground data
Fingertippositions
Gesture Recognition
Object Tracking
Path followed by the fingertips are computed
Thresholding
Detected path is compared with existing gesture data
Output from both is used to find a suitable gesture if any.
Gesture
Daemon
Function Mapping
Function corresponding to gesture is retrieved.
Function Execution
Function is executed

Chapter 6

Implementation

6.1 Algorithms

6.1.1 Overall algorithm

- Input: A set of contiguous image frames
- Output: The mapped GUI function
- Method:
    1. Calculate the foreground using MOG2 background subtraction algorithm.
    2. Detect the fingertips using the Convex Hull algorithm.
    3. Track the finger gesture using Continuously Adaptive Mean Shift (CAMshift) algorithm.
    4. Save the above tracked graph to a DetectedGesture object.
    5. Find the closest SavedGesture object and execute the GUI function.

6.1.2 Background subtraction

1. Purpose: To find the Region of Interest by subtracting the background from the image frame.
2. Input: Image frames with foreground and background.
3. Output: Image frames with only foreground.
4. Method:
    - Model each background pixel by a mixture of K Gaussian distributions (K = 3 to 5)
    - Calculate the weights of the mixture which represent the time proportions that those colours stay in the scene.
    - Remove the colours are which stay longer and more static.

6.1.3 Convex Hull

- Purpose: To find the positions of the fingertips in the image frame.
- Input: Image frames with background subtracted.
- Output: List of Convex Hull peaks
- Method:
  1. Threshold the image, and retrieve the points in the plane. Let's say there are P points.
  2. Initialize p as leftmost point.
  3. Do following while we dont come back to the first (or leftmost) point.
     (a) The next point q is the point such that the triplet (p, q, r) is counterclockwise for any other point r.
     (b) next[p] = q (Store q as next of p in the output convex hull).
     (c) p = q (Set p as q for next iteration).

6.1.4 CAMShift

- Purpose: To map the positions of the fingertips in the multiple frames to one fluid motion, in the form of lines/curves in a 2D plane.
- Input: Sequence of frames with Convex Hull peaks.
- Output: 2D plane with a set of lines/curves representing the fluid motion of the fingertips.
- Method:
  1. Set the calculation region of the probability distribution equal to the whole frame.
  2. Choose the initial location of the two-dimensional mean shift search window.
  3. Calculate the colour probability distribution in the 2D region centred on the search window location in an area slightly larger than the mean shift window size.
  4. Perform the search of the maximum density probability using the mean shift parameter for convergence or for setting the number of iterations. Store the zero moment (area or size) and middle position.
  5. For the next image frame, place the search window in the middle position fixed in step 4, and set the window size in conformity to the last moment. Go to step 3.

6.1.5 Brute Force Matching

- Purpose: To match the DetectedGesture to a SavedGesture, and execute the corresponding GUI function.
- Input: The feature objects encoded in the SavedFeature and DetectedFeature objects.
- Output: A quantity representing the closeness of the two gestures.
- Method:

1. For a given keypoint K1 from the first set, take every keypoint in the second set and calculate the distance.
2. Distance formula used is Euclidean distance. The sum of the distances on each dimension is finally used to calculate the returned quantity.
3. cv2.NORM2 can be used to find the BFMatch value between two sets (of points)
4. The Euclidean Distance sum is normalized and returned.

6.2 Development Tools

6.2.1 Sublime Text

Sublime Text is a proprietary cross-platform source code editor with a Python application programming interface (API). It natively supports many programming languages and markup languages, and functions can be added by users with plugins, typically community-built and maintained under free-software licenses. Some of the features of Sublime Text are

- Goto Anything,quick navigation to files, symbols, or lines
- Command palette uses adaptive matching for quick keyboard invocation of arbitrary commands
- Simultaneous editing: simultaneously make the same interactive changes to multiple selected areas
- Python-based plugin API
- Project-specific preferences
- Extensive customizability via JSON settings files, including project-specific and platform-specific settings
- Cross platform (Windows, macOS, and Linux)

Chapter 7

Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a program or application with the intent of finding software bugs, and to verify that the software product is fit for use. Software testing involves the execution of a software component or system component to evaluate one or more properties of interest. In general, these properties indicate the extent to which the component or system under test

- Meets the requirements that guided its design and development,
- Responds correctly to all kinds of inputs
- Performs its functions within an acceptable time
- Is sufficiently usable
- Can be installed and run in its intended environments, and
- Achieves the general result its stakeholders desire.

7.1 Testing Methodologies

Software testing methodology is for making sure that software products/systems developed have been successfully tested to meet their specified requirements and can successfully operate in all the anticipated environments with required usability and security. Software testing methods are traditionally divided into white and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases. White-box testing by seeing the source code tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses in- puts to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit. While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. Black-box testing treats the software as

a black-box , examining functionality without any knowl- edge of internal implementation, without seeing the source code. The testers are only aware of what the software is supposed to do, not how it does it. Here the black-box testing is used for the system. The testing methods applied were:

- Unit Testing
  Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.
- Integration Testing
  Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing.
- System Testing
  System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the systems compliance with its specified requirements. System testing falls within the scope of black-box testing, and as such, should require no knowledge of the inner design of the code or logic.

7.2 Unit Testing

In the unit testing phase, the Background Subtraction Module, Feature Extraction, Finger-tip Tracking, Gesture Recognition and Gesture mapping were separately tested.

7.2.1 Background Subtraction Module

The images from the camera feed is provided to the Background Subtraction module in RGB colour space. The output images were in HSV filtered with Colour Ranges as expected.

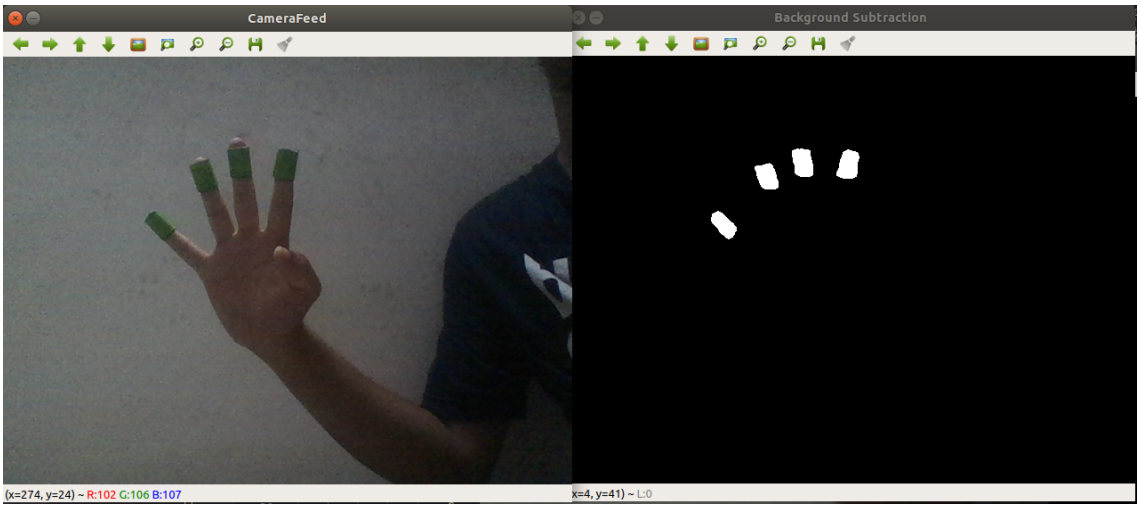


Figure 7.1: Background Subtraction Module

7.2.2 Feature Extraction

The output from the background subtraction module is fed into the feature Extraction module. This module computes the convex hull and finds the separate shapes and finds the centroids of each hull.

Figure 7.2: Feature Extraction Module

7.2.3 FingerTip Tracking and Gesture Recognition

The Finger tip tracking module tracks the centroids movement over multiple frames and the Gesture Recognition Module recognizes the gesture.

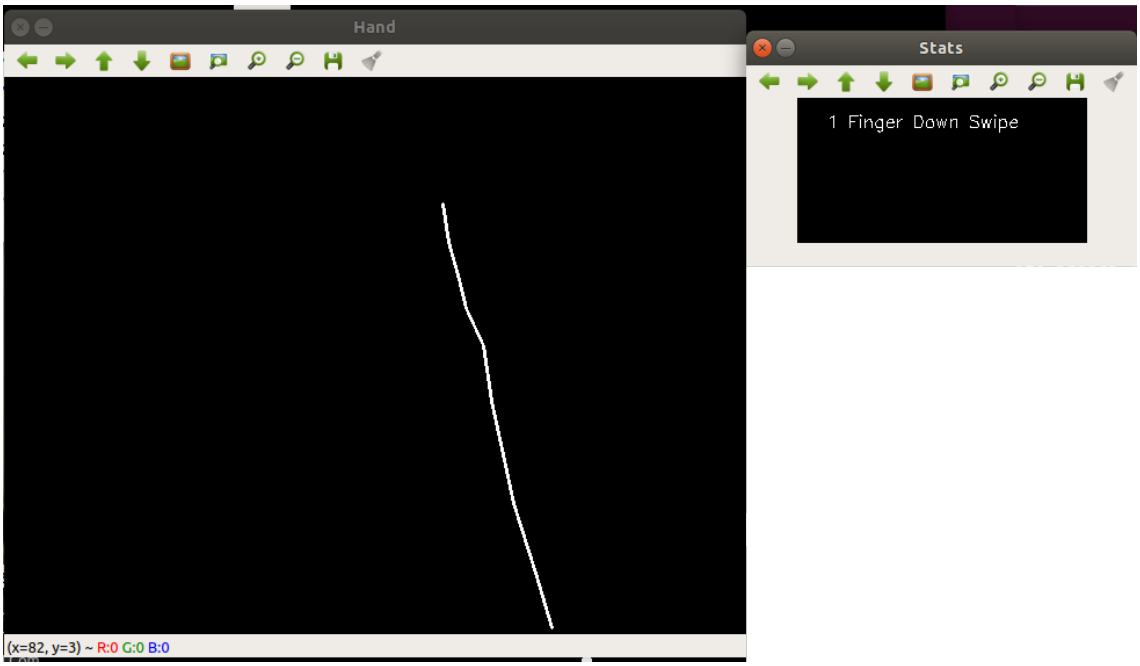


Figure 7.3: Finger Tip Tracking and Gesture Recognition

7.3 Integration Testing

Different modules were combined and was tested to see if the modules interact properly and produce correct output. The Background Subtraction Module provided its output to the feature extraction module which then finds the convex hulls. The output is passed to the finger-tip tracking module which tracks the direction and number of fingers. The output is successfully passed to the Gesture Recognition module which allots the gesture. Finally the output is passed to the Gesture Map module which executes the gesture in the Linux GUI.

7.4 System Testing

After the integration testing, we do the system testing. In system testing the whole modules are connected in order; the background subtraction module is integrated with the feature extraction module,the feature extraction module is integrated with the fingertip tracking and also the gesture recognition module and gesture mapping module. The whole system is inte- grated.

Chapter 8

Graphical User Interface

8.1 GUI Overview

8.2 Main GUI Components

Chapter 9

Results

Include screenshots of the project.

Chapter 10

Conclusion

Chapter 11

Future Scope

Chapter 12

Publication

References

[1]