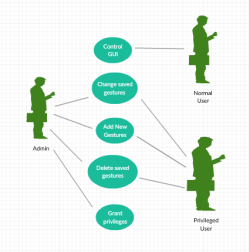
Project title

CS18L1 Project

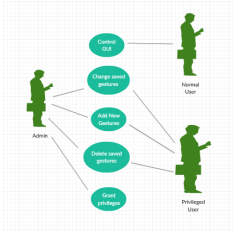
regno rollno name
B. Tech Computer Science & Engineering



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 7, 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

List of Tables

Chapter 1

Introduction

For decades the mouse and keyboard have been the primary choice of input device to use computers. We propose a complete methodology of controlling the operating system using fluid natural of hand gestures using the normal web camera instead of the conventional mouse. This can have wide range of applications.

Evolution of ubiquitous computing, current user interaction approaches with keyboard, mouse and pen are not sufficient for the still widening spectrum of Human computerinteraction. Gloves and sensor based trackers are unwieldy,constraining and uncomfortable to use. Due to the limitation of these devices the useable command set based diligences isalso limited. Direct use of hands as an input device is an innovative method for providing natural Human Computer Interaction which has its inheritance from textbased interfaces through 2D graphical-based interfaces, multimedia-supported interfaces, to full-fledged multiparticipant Virtual Environ- ment (VE) systems. The proposed system will allow users to manipulate his operating systems GUI with his/her hands to do virtually anything. From moving around elements on the GUI such as windows and manipulating them to zooming in on pages, rotating them and using your hands as your own custom cursor. The research effort centralizes on the efforts of implementing an application that employs computer vision algorithms and gesture recognition techniques which in turn results in developing a low cost interface device for interacting with objects in virtual environment using hand gestures.

The prototype architecture of the application comprises of a central computational module that applies the camshift technique for tracking of hands and its gestures. Haar like technique has been utilized as a classifier that is creditworthy for locating hand position and classifying gesture. The patterning of gestures has been done for recognition by mapping the number of defects that is formed in the hand with the assigned gestures. The virtual objects are produced using Open GL library. This hand gesture recognition technique aims to substitute the use of mouse for interaction with the virtual objects. This will be useful to promote controlling applications like virtual games, browsing images etc in virtual environment using hand gestures.

1.1 Proposed Project

1.1.1 Problem Statement

The existing Hand Gesture Recognition systems recognize only single frame hand gestures and are not responsive. The better systems which track such movement require special hardwares which

are very costly. The existing Mouse/Keyboard combination of input is lacking in fluidity of motion.

1.1.2 Proposed Solution

The proposed solution is developing a system called "MultiFrame Hand Gesture GUI Control System using Convex Hull Algorithm" which uses the standard webcam built in laptops to record multiple frame hand gestures and execute corresponding GUI commands in the OS.

Chapter 2

System Study Report

2.1 Literature Survey

1. K. Oka, Y. Sato, Real-Time Fingertip Tracking and Gesture Recognition - Their method uses a filtering technique, in addition to detecting fingertips in each image frame, to predict fingertip locations in successive image frames and to examine the correspondences between the predicted locations and detected fingertips. This lets us obtain multiple fingertips trajectories in real time and improves fingertip tracking. This method can track multiple fingertips reliably even on a complex background under changing lighting conditions without invasive devices or color markers.

2. Lai, H. Y., Lai, H. J., Real-Time Dynamic Hand Gesture Recognition - The eleven kinds of hand gestures have been dynamic recognized, which represent the number from one to nine. The dynamic images are caught by a dynamic video. We use the YCbCr color space transformation to detect the skin color and to find the hand contour from the complex background. Convex defect character points of the hand contour are defined, and the finger angles and fingertip positions are calculated to recognize the hand gestures. The OpenCV is used to perform our research. Ten tested users produce 330 cases to recognize eleven hand gestures, and each hand gestures have three different poses of hand gestures. The accurate recognition rate is 95.1%.

3. Z. Ren, J. Yuan, J. Meng , Z. Zhang, Robust Part-Based Hand Gesture Recognition Using Kinect Sensor -The recently developed depth sensors, e.g., the Kinect sensor, have provided new opportunities for human-computer interaction (HCI). Although great progress has been made by leveraging the Kinect sensor, e.g., in human body tracking, face recognition and human action recognition, robust hand gesture recognition remains an open problem. Compared to the entire human body, the hand is a smaller object with more complex articulations and more easily affected by segmentation errors. It is thus a very challenging problem to recognize hand gestures. This paper focuses on building a robust part-based hand gesture recognition system using Kinect sensor. To handle the noisy hand shapes obtained from the Kinect sensor, we propose a novel distance metric, Finger-Earth Mover's Distance (FEMD), to measure the dissimilarity between hand shapes. As it only matches the finger parts while not the whole hand, it can better distinguish the hand gestures of slight differences. The extensive experiments demonstrate that our hand gesture recognition system is accurate (a 93.2% mean

accuracy on a challenging 10-gesture dataset), efficient (average 0.0750 s per frame), robust to hand articulations, distortions and orientation or scale changes, and can work in uncontrolled environments (cluttered backgrounds and lighting conditions)

4. Ji-Hwan Kim,Nguyen Duc Thang,Tae-Seong Kim, 3-D Hand Motion Tracking and Gesture Recognition Using a Data Glove- Developed a 3-D hand motion tracking and gesture recognition system via a data glove (namely the KHU-l data glove consisting of three tri-axis accelerometer sensors, one controller, and one Bluetooth). The KHU-l data glove is capable of transmitting hand motion signals to a PC through wireless communication via Bluetooth. Also we have implemented a 3-D digital hand model for hand motion tracking and recognition. The implemented 3-D digital hand model is based on the kinematic chain theory utilizing ellipsoids and joints. Finally, we have utilized a rule-based algorithm to recognize simple hand gestures namely scissor, rock, and paper using the 3-D digital hand model and the KHU-l data glove.

5. Takuro Niidome, Rokuya Ishii, A GUI Support System for a Sight Handicapped Person By Using Hand Shape Recognition - A GUI Support System for a Sight Handicapped Person By Using Hand Shape Recognition - An algorithm of hand-shape recognition and applies to a Graphical User Interface (GUI) support system for a sight handicapped person. This algorithm is to recognize the angle of a hand and ten hand-shapes. As one application of this algorithm, we implement a system in which the angle of a hand controls a reading point in a text on a screen in real time, and a text is read out with a sound syntheses software. Further, in this system, more complicated directions to control a text reader can be done by recognized hand-shapes

6. X. Wu, C. Yang, Y. Wang, H. Li, and S. Xu, An Intelligent Interactive System Based on Hand Gesture Recognition Algorithm and Kinect, - A cascade control algorithm integrating electromyography bio-feedback into hand gesture recognition is proposed. The outer loop is the trajectory motion tracking with Kinect-based gesture decoding classifier, and the inner loop is torque control with electromyography bio-feedback in the real time. This proposed method improves the tracking accuracy. The tracking error is effectively reduced from 70.56 to 28.07 in the simulation experiment.

7. Hayato Takahashi, Yuhki Kitazono, Integrated multi-touch gesture with hand gesture and make new input method. Multi-touch gesture and hand gesture have been studied separately, however, these could be integrated, and it would be new interaction way for computer. We made a large scale multi-touch screen by FTIR method that uses infrared ray to detect the touched point, and a hand gesture recognition device with sensors, pressure sensor and bend sensor. In the end, we integrated the input information to interact with computer. User can input coordinate by touching, and simultaneously input action by performing hand gesture.

8. Hamid A. Jalab, Herman .K. Omer, Human Computer Interface Using Hand Gesture Recognition Based On Neural Network, A hand gesture interface for controlling media player using neural network. The proposed algorithm recognizes a set of four specific hand gestures, namely: Play, Stop, Forward, and Reverse. The algorithm is based on four phases, Image acquisition, Hand segmentation, Features extraction, and Classification. A frame from the webcam camera is captured, and then skin detection is used to segment skin regions from

background pixels. A new image is created containing hand boundary. Hand shape features extraction, are used to describe the hand gesture. An artificial neural network has been utilized as a gesture classifier, as well. The proposed algorithm develops an alternative input device to control the media player, and also offers different gesture commands and can be useful in real-time applications. Comparisons with other hand gesture recognition systems have revealed that our system shows better performance in terms accuracy.

9. Mykyta Kovalenko, Svetlana Antoshchuk, Juergen Sieck, Real-Time Hand Tracking and Gesture Recognition Using Semantic-Probabilistic Network, A system for real-time hand gesture recognition for use as a human-computer interface. Firstly, hand detection is performed using a Viola-Jones algorithm. We use the Continuously Adaptive Mean Shift Algorithm (CAMShift) to track the position of each detected hand in each video frame. The hand contour is then extracted using a Border Following algorithm, which is preceded by skin-colour thresholding, performed in HSV colour space. Finally, a semantic-probabilistic network is introduced, which uses an ontological gesture model and Bayesian network for gesture recognition. We then demonstrate the effectiveness of our technique on a training data set, as well as on a real-life video sequence.

10. Yee Yong Pang, Nor Azman Ismail, Phuah Leong Siang Gilbert, A Real Time Vision-Based Hand Gesture Interaction, Moving beyond mouse and keyboard, the evolution of human-computer interaction (HCI) has been an interest research in recent years which witnessed the development from text-based like using a keyboard to graphic user interface (GUI) based on a mouse, from cumbersome data gloves and tracking devices to visual-based computer application. One of the interest fields is by using hand gestures to interact with computer. However, the complexity of a hand set a lot of challenges to be tracked. In real-time, the application requires high accurate detection and recognition. In additional the real and clutter environments have a big impact on recognition process because it included with irrelevant information from the application point of view. In this paper, a real time vision based hand gesture interaction prototype was proposed. Currently a prototype has built for controlling the desktop cursor and concerned the tasks involving in navigation the desktop cursor by using hand gesture input modality.

2.2 Proposed System

The system will implement the basic GUI features using fluid Hand gestures. The system will be implemented as a daemon continuously running in the background. We require a good system as well as a decent web cam. Good lighting conditions are also favourable. The system will be advantageous over the existing system as it tends to be a more natural form of interaction. The main disadvantages we face over here is the lack of good web cams and system performances. The operating system is currently limited only to Linux. Future work includes incorporating the rest of the operating systems.

Chapter 3

Software Requirement Specification

3.1 Introduction

3.1.1 Purpose

Hand Gesture GUI Control System version 1 revison 1 - A Real time dynamic Hand Gesture Recognition GUI system to improve the interaction between user and computer. The user can use his webcam to record hand gestures which our system can use to perform Linux GUI functions in realtime.

3.2 Document Conventions

3.2.1 Intended Audience and Reading Suggestions

This document is intended for the developers of the system. Developers may use document to get a better understanding of the requirements of the system and the prerequisites for building the system.

3.2.2 Project Scope

The software specified has the purpose of implementing natural and easy to understand hand gestures to control the operating system. The operating system we use here is Linux. We are implementing fluid hand gestures rather than static hand gestures which is tracked by the standard webcam rather than using specialized equipment.

3.2.3 Overview of Developer's Responsibilities

The developers must design and implement the Hand Gesture Recognition System and the Control System for the GUI. Developers must determine valid gestures and mappings for each gesture to a appropriate function in the GUI of the computer.

3.3 Overall Description

3.3.1 Product Perspective

This system arose from the need for a more fluid and convenient way to interact with a computer. It allows you to interact with the computer with no physical contact with any peripheral. Similar systems have been developed but most required additional peripherals or did not offer a wide range of Linux GUI functions that could be accessed via gestures.

Our product serves as an add-on to the current user interface, the mouse and the keyboard

3.3.2 Product Functions

The major Linux GUI features to be implemented are:

- Switch Window to Next one.
- Drag operations.
- Minimize window
- Close window
- Open Apps Launcher
- Open Terminal
- Scroll down / Up
- Zoom in/ out
- Switch Tabs.

3.3.3 User Classes and Characteristics

The user base intended for this product are all the Linux users. Linux Developers are encouraged to add-on their own gestures and tinker with the product and contribute.

The normal Linux users can use the product freely without restrictions.

3.3.4 Operating Environment

The software will run on a Linux machine as a daemon and is not architecture dependant. The only hardware dependant is the camera which must be compatible with the linux operation system with appropriate drivers.

3.3.5 Design and Implementation Constraints

Implementation constraints include the speed of the computer being used so as to provide a smooth implementation. A good system is required to optimally recognize the gestures from the video input.

A webcam that is compatible with the Linux OS is required to capture the gestures performed by the user and appropriate drivers for the same.

The OpenCV library will be used to recognize the hand gestures from the input.

3.3.6 User Documentation

An online documentation will be provided consisting of the Hand Gestures that are included.

3.3.7 General Constraints

The software can be limited by several factors.

- Lack of light
- Complex backgrounds
- Poor camera quality
- Poor System Performance
- System Memory Constraints
- Camera is already is use by another application

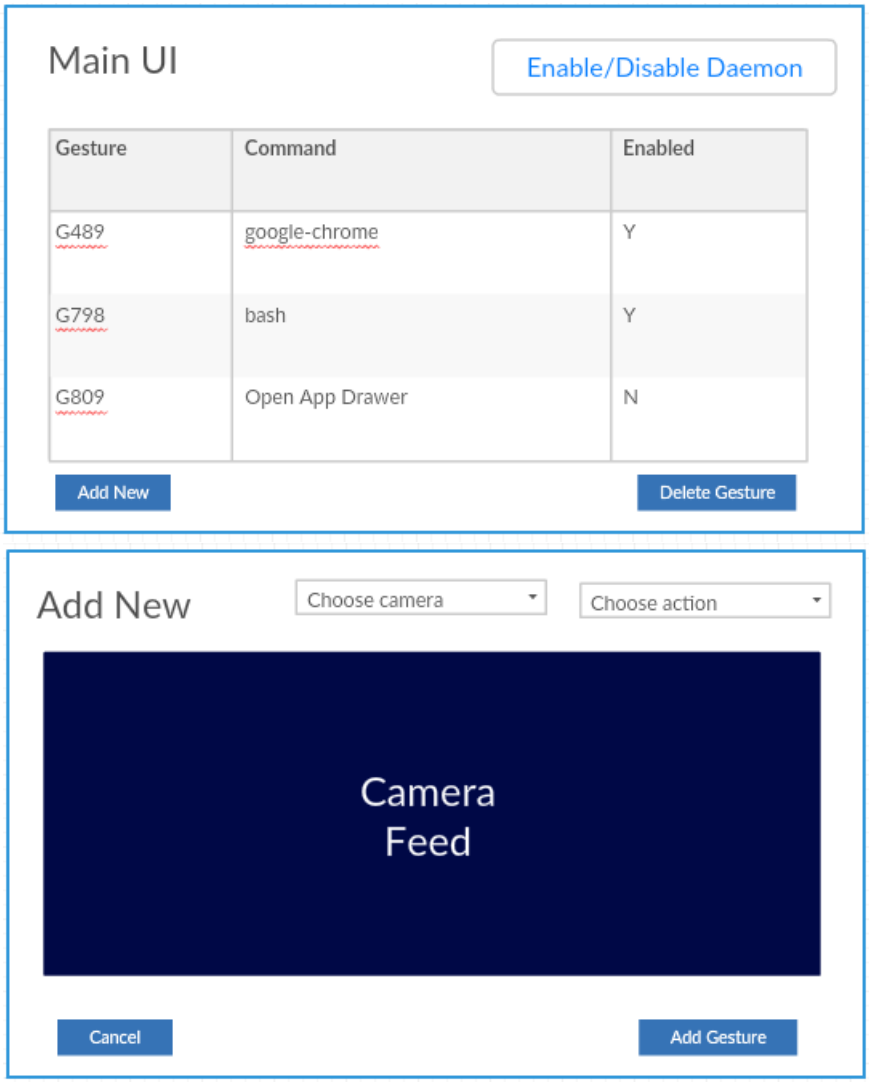
3.3.8 Assumptions and Dependencies

Our assumption is that the product will always be used on a system with a good graphics card as well as a good processor. The operating system will be Linux and will have OpenCV libraries installed in it. A good webcam will be present and will be always searching for hand gestures.

3.4 External Interface Requirements

3.4.1 User Interfaces

There will be single user interface where the user can enable and disable the daemon. Add and remove gestures. Change mappings between gestures and commands and map gestures to custom macros and keyboard strokes.



3.4.2 Hardware Interfaces

The hardware required for the application is the webcam. The images are then handled by our application and the webcam too is managed by our deamon running in the background.

3.4.3 Software Interfaces

The daemon must interface with the camera driver and the operating system to be able to control the GUI based on input gestures.

3.4.4 Communications Interfaces

The application will need internet connection to send bug reports in case of a crash or malfunction. Bug reports will be sent in the form of HTTP requests to a remote webserver.

Chapter 4

Hardware and Software Requirements

4.1 Hardware Requirements

- A computer with at least 2GHz processor, 1GB RAM and USB ports.
- A USB camera.

4.2 Software Requirements

- Linux operating system.

  The linux operating system was chosen due to the fact that it is easy to build software due to wide range of libraries and easy access to them and the compilers. Whereas in a OS like windows a special IDE like Visual Studio and proprietary libraries would be required to build software.

- C++ compliler eg:- gcc

  C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation.

  It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, Web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes). C++ is a compiled language, with implementations of it available on many platforms.

  Some of the interesting features of C++ are:

  - **Object-oriented**: C++ is an object-oriented programming language. This means that the focus is on objects and manipulations around these objects. Information about how these manipulations work is abstracted out from the consumer of the object.

  - **Rich library support**: Through C++ Standard Template Library (STL) many functions are available that help in quickly writing code. For instance, there are standard libraries for various containers like sets, maps, hash tables, etc.
  - **Speed**: C++ is the preferred choice when latency is a critical metric. The compilation, as well as the execution time of a C++ program, is much faster than most other general purpose programming languages.
  - **Compiled**: A C++ code has to be first compiled into low-level code and then executed, unlike interpreted programming languages where no compilation is needed.
  - **Pointer Support**: C++ also supports pointers which are widely used in programming and are often not available in several programming languages.

- OpenCV C++ library

  OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. OpenCV is written in C++ and its primary interface is in C++. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

  OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

  - **Core functionality (core)** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
  - **Image Processing (imgproc)** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
  - **Video Analysis (video)** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
  - **Camera Calibration and 3D Reconstruction (calib3d)** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
  - **2D Features Framework (features2d)** - salient feature detectors, descriptors, and descriptor matchers.
  - **Object Detection (objdetect)** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
  - **High-level GUI (highgui)** - an easy-to-use interface to simple UI capabilities.
  - **Video I/O (videoio)** - an easy-to-use interface to video capturing and video codecs.

  OpenCV has functions available for background subtraction. Several algorithms were introduced for this purpose. OpenCV has implemented three such algorithms which are very easy to use like BackgroundSubtractorMOG, BackgroundSubtractorMOG2 and BackgroundSubtractorGMG.

  OpenCV also has various feature extractors that can be used for detect the fingers of the user like CvFeatureEvaluator, CvFeatureParams, CvHaarEvaluator and CvHaarFeatureParams.

- GTK+ Library for C++

  GTK+, or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off tools to complete application suites.

  The GTK+ Library is needed to build an interactive GUI where users can configure the Gesture Control System. Functions to create windows and other GUI elements such as buttons, lists, pop-ups, drop-downs, frames to display video, etc are available in the library. The library also allows to bind functions to events in the GUI such as clicks, key-presses and so on.

  Features of GTK+:

  - **Stability**: GTK+ has been developed for over a decade to be able to deliver the enticing features and superb performance that it brings to your application development. GTK+ is supported by a large community of developers and has core maintainers from companies such as Red Hat, Novell, Lanedo, Codethink, Endless Mobile and Intel.
  - **Interfaces**: GTK+ has a comprehensive collection of core widgets and interfaces for use in your application.
  - **Cross Platform**: Originally GTK+ was developed for the X Window System but it has grown over the years to include backend support for other well known windowing systems. Today you can use GTK+ on Windows, Mac OS X and Linux.
  - **Accommodating**: GTK+ caters for a number features that today's developers are looking for in a toolkit including Native look and feel, Theme support, Thread safety and Object oriented approach.
  - **Foundation**: GTK+ is built on top of GLib. GLib provides the fundamental algorithmic language constructs commonly duplicated in applications.

4.3 Hardware and Software Requirements

4.3.1 Hardware Requirements

- A computer with at least 2GHz processor, 1GB RAM and USB ports.
- A USB camera.

4.3.2 Software Requirements

- Linux operating system.

  The linux operating system was chosen due to the fact that it is easy to build software due to wide range of libraries and easy access to them and the compilers. Whereas in a OS like windows a special IDE like Visual Studio and proprietary libraries would be required to build software.

- C++ compliler eg:- gcc

  C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation.

  It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, Web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes). C++ is a compiled language, with implementations of it available on many platforms.

  Some of the interesting features of C++ are:

  - **Object-oriented**: C++ is an object-oriented programming language. This means that the focus is on objects and manipulations around these objects. Information about how these manipulations work is abstracted out from the consumer of the object.
  - **Rich library support**: Through C++ Standard Template Library (STL) many functions are available that help in quickly writing code. For instance, there are standard libraries for various containers like sets, maps, hash tables, etc.
  - **Speed**: C++ is the preferred choice when latency is a critical metric. The compilation, as well as the execution time of a C++ program, is much faster than most other general purpose programming languages.
  - **Compiled**: A C++ code has to be first compiled into low-level code and then executed, unlike interpreted programming languages where no compilation is needed.
  - **Pointer Support**: C++ also supports pointers which are widely used in programming and are often not available in several programming languages.

- OpenCV C++ library

  OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision. OpenCV is written in C++ and its primary interface is in C++. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.

  OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

  - **Core functionality (core)** - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
  - **Image Processing (imgproc)** - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
  - **Video Analysis (video)** - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
  - **Camera Calibration and 3D Reconstruction (calib3d)** - basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
  - **2D Features Framework (features2d)** - salient feature detectors, descriptors, and descriptor matchers.
  - **Object Detection (objdetect)** - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
  - **High-level GUI (highgui)** - an easy-to-use interface to simple UI capabilities.
  - **Video I/O (videoio)** - an easy-to-use interface to video capturing and video codecs.

  OpenCV has functions available for background subtraction. Several algorithms were introduced for this purpose. OpenCV has implemented three such algorithms which are very easy to use like BackgroundSubtractorMOG, BackgroundSubtractorMOG2 and BackgroundSubtractorGMG.

  OpenCV also has various feature extractors that can be used for detect the fingers of the user like CvFeatureEvaluator, CvFeatureParams, CvHaarEvaluator and CvHaarFeatureParams.

- GTK+ Library for C++

  GTK+, or the GIMP Toolkit, is a multi-platform toolkit for creating graphical user interfaces. Offering a complete set of widgets, GTK+ is suitable for projects ranging from small one-off tools to complete application suites.

  The GTK+ Library is needed to build an interactive GUI where users can configure the Gesture Control System. Functions to create windows and other GUI elements such as buttons, lists, pop-ups, drop-downs, frames to display video, etc are available in the library. The

library also allows to bind functions to events in the GUI such as clicks, key-presses and so on.

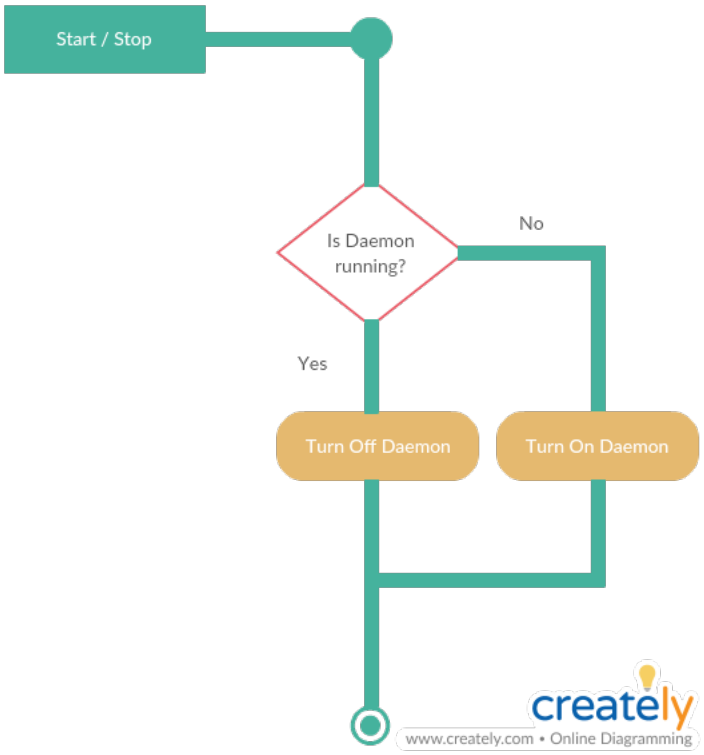
Features of GTK+:

- **Stability**: GTK+ has been developed for over a decade to be able to deliver the enticing features and superb performance that it brings to your application development. GTK+ is supported by a large community of developers and has core maintainers from companies such as Red Hat, Novell, Lanedo, Codethink, Endless Mobile and Intel.
- **Interfaces**: GTK+ has a comprehensive collection of core widgets and interfaces for use in your application.
- **Cross Platform**: Originally GTK+ was developed for the X Window System but it has grown over the years to include backend support for other well known windowing systems. Today you can use GTK+ on Windows, Mac OS X and Linux.
- **Accommodating**: GTK+ caters for a number features that today's developers are looking for in a toolkit including Native look and feel, Theme support, Thread safety and Object oriented approach.
- **Foundation**: GTK+ is built on top of GLib. GLib provides the fundamental algorithmic language constructs commonly duplicated in applications.

4.4 Functional Requirements

4.4.1 Start/Stop the Hand Gesture Daemon

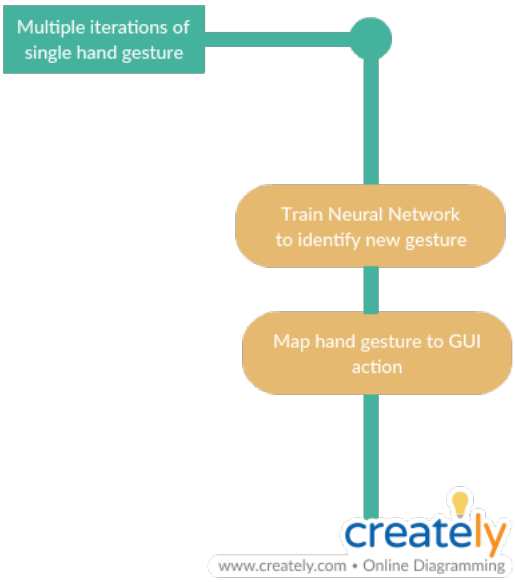
A toggle button that starts and stops the deamon which tracks hand gestures.



Input: Toggle button
Process: Will stop/start the daemon in the background
Output: The daemon shall be stopped/started.

4.4.2 Add new gestures

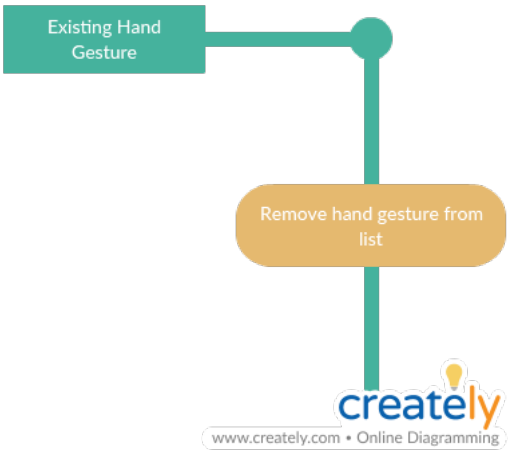
The user should be able to add new gestures by recording the hand movement and mapping the GUI feature to the current Hand gesture.



Input: Multiple Iterations of a gesture via Camera; Action to carry out
Process: The software will train the Neural Network to identify the new hand gesture and map the hand gesture to the Command given as input
Output: A new mapping from the hand gesture to the command will be added

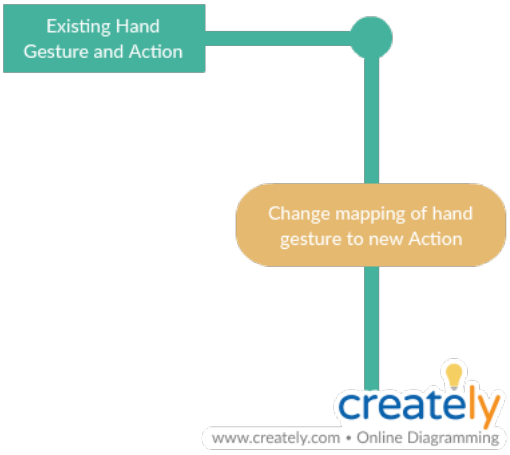
4.4.3 Remove gestures

The user can remove the existing gestures



Input: A hand gesture
Process: The software will remove the mapping of the input hand gesture from the list of hand gestures
Output: The hand gesture is removed

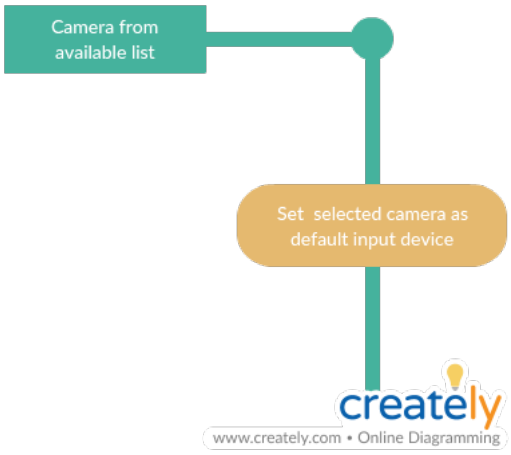
4.4.4 Modify Gestures



Input: Existing hand gesture; Action to execute
Process: The software will change the mapping of the input hand gesture to the input Action
Output: The hand gesture is modified

4.4.5 Choose webcam

For systems with multiple webcams connected the user must be able to choose which webcam the application must use.



Input: A camera detected by the daemon
Process: The software will set the selected camera as the default input for images
Output: The daemon will now watch for gestures through the selected camera when activated

4.5 Non-functional Requirements

4.5.1 Performance Requirements

The product must be fast enough to recognize and execute the hand gesture such that the action becomes fluid. The product must be responsive all the time and multiple hand gestures shown rapidly must be processed effectively.

4.5.2 Safety Requirements

No other application or user should access the camera feed.

4.5.3 Security Requirements

The camera feed must not be stored in the system. No one should be able to tap into the camera feed or be able to extract information from the camera.

4.5.4 Software Quality Attributes

Our product must be adaptable to all lighting conditions, poor webcams, and poor system capabilities. The product will be easy to maintain and review for other people. The product will run on newer updates of the operating system without any glitches.

4.6 Other Requirements

The product can be reused by other developers to create and maintain their own products. The GUI of the application must work with all languages so that people from all backgrounds can work seamlessly.

Chapter 5

System Design

5.1 System Architecture

5.2 Input Design

In the system, the hand gesture is captured by the web cam in multiple frames. After the gesture Is carried out, it is recognized and the corresponding GUI command is carried out.

5.3 Libraries and Packages Used

5.4 Module Description

5.4.1 Module 1 - Image Retrieval

The first step in this system is to retrieve the contiguous camera image frames. The user shows the hand-gesture and it is retrieved via the webcam. The images are captured at short intervals to accommodate multi-frame fluid hand-gestures.
The camera feed is retrieved from the user's webcam using the CaptureFromCAM function provided by OpenCV.
The function reads an image from the specified buffer in the memory. If the buffer is too short or contains invalid data, the empty matrix/image is returned.

5.4.2 Module 2 - Background Subtraction

The captured image from the camera needs to undergo background subtraction in order to recognize the gesture correctly. The different elements in the background has to be discarded and only the shape of the hand is retained.
The Mixture of Gaussians (MOG) method is used for foreground detection, and once the foreground is detected, we subtract the rest of the image, i. e., the background. we need to create a background object using the function, cv.createBackgroundSubtractorMOG(). It has some optional parameters like length of history, number of gaussian mixtures, threshold etc. Then we use backgroundsubtractor.apply() method to get the foreground mask.

5.4.3 Module 3 - Feature Extraction

The next step is to find the fingertip locations in the recorded image from the background subtracted image. To recognize the fingertips we use the Convex Hull algorithm.
The locations of the fingertips on the screen are detected by finding the Convex hull of the hand and marking the peaks. This process is done in realtime and the extracted location changes with every frame.

The function find the convex hull of a 2D point set using the Sklanskys algorithm that has O(N logN) complexity in the current implementation. The algorithm consisted of two parts. Part 1 was intended to take a simple polygon and return a maximal polygon (monotonic in both the horizontal and vertical directions). Part 2 was Sklansky's original algorithm, which was guaranteed to work on the output of part 1, since maximal polygons are weakly externally visible, and it is proven that Sklansky's original algorithm works for such polygons.

5.4.4 Module 4 - Finger-Tip Tracking

After extracting the finger tip information, we need to track the finger tips over multiple frames. cv.camShift() is the function that is used. Continuously Adaptive Mean Shift Algorithm (CAMShift) is used to detect the changes from frame to frame. The probability distribution of the first frame is taken as a reference for the second frame, the probability distribution of the second frame is taken for the third frame and so on and so forth.

5.4.5 Module 5 - Gesture Recognition

The convex hull and convexity defects along with the results of the CAMShift algorithm are used to create a mathematical state of the currently recorded gesture. This state is compared with the states of the saved gestures and the closest candidate is chosen.

We use Brute Force matching to find the closest saved gesture. First we create ORB descriptors (Oriented FAST (Features from accelerated segment test) and Rotated BRIEF (Binary Robust Independent Elementary Features)) and these descriptors are checked by using the Brute Force matcher.

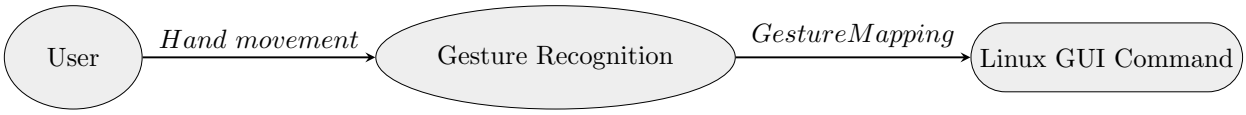
5.4.6 Module 6 - Gesture Mapping

Once the correct gesture has been identified, the corresponding class object is accessed to procure the GUI function that has been saved for that particular gesture. The function is first executed, and then the daemon starts listening for the next gesture.

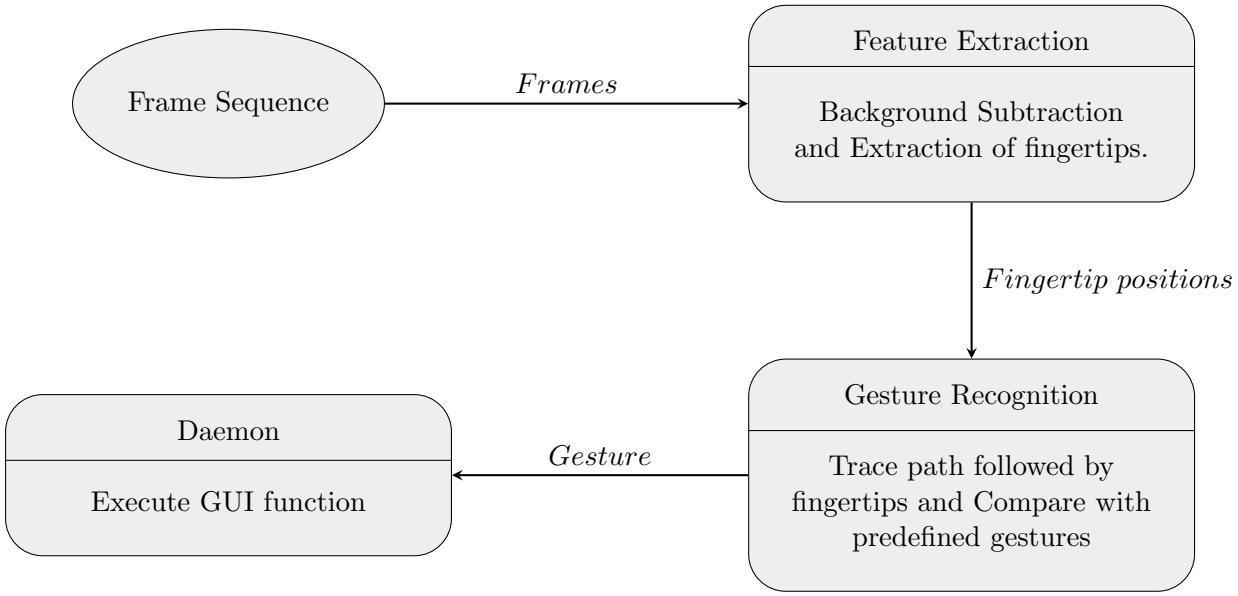
Chapter 6

Data Flow Diagram

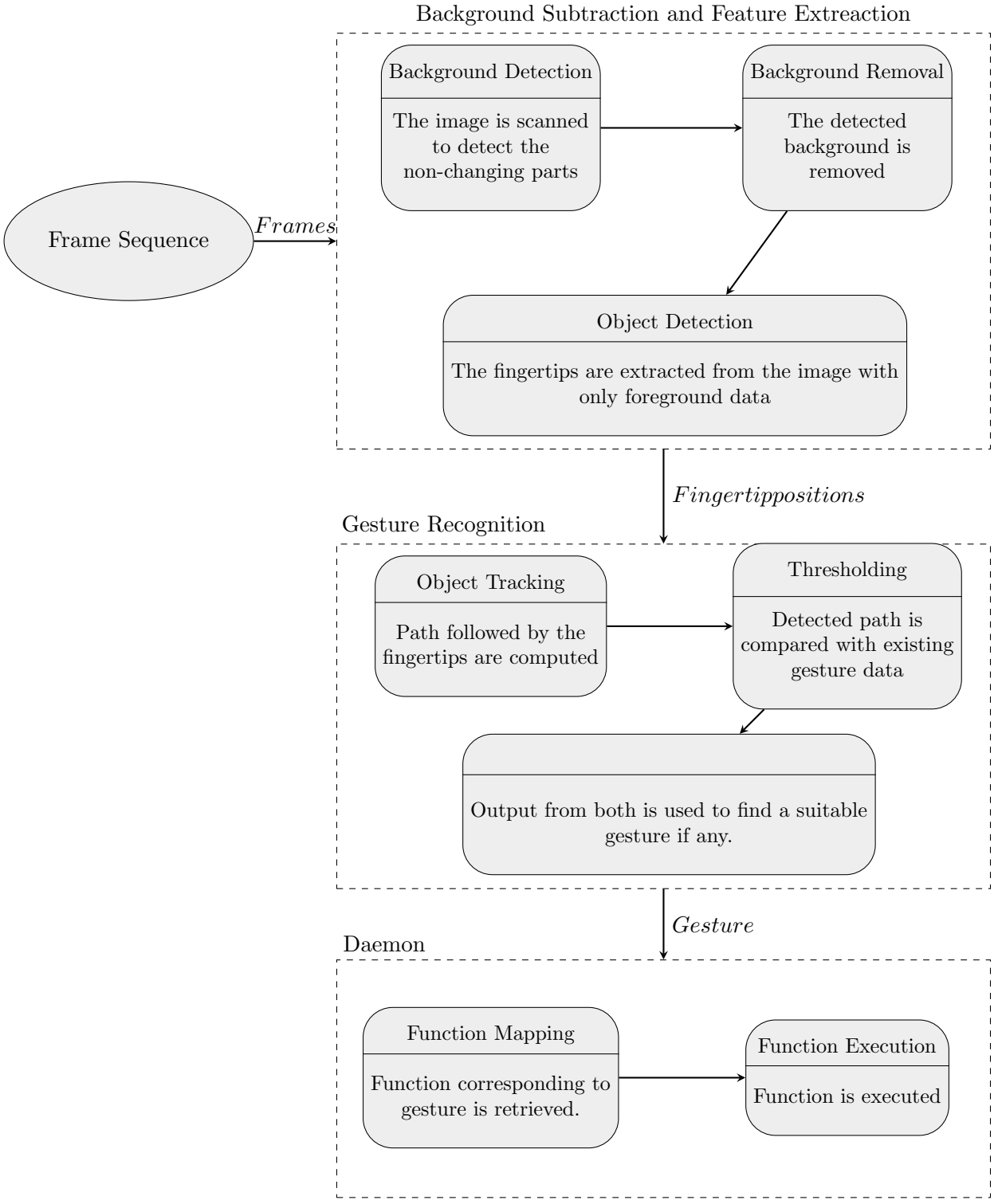
6.1 Level 0



6.2 Level 1



6.3 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 7

Implementation

7.1 Algorithms

7.1.1 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.

7.1.2 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.

7.1.3 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).

7.1.4 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.

7.1.5 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.

7.2 Development Tools

7.2.1 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 8

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.

8.1 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.

8.2 Unit Testing

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

8.2.1 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.

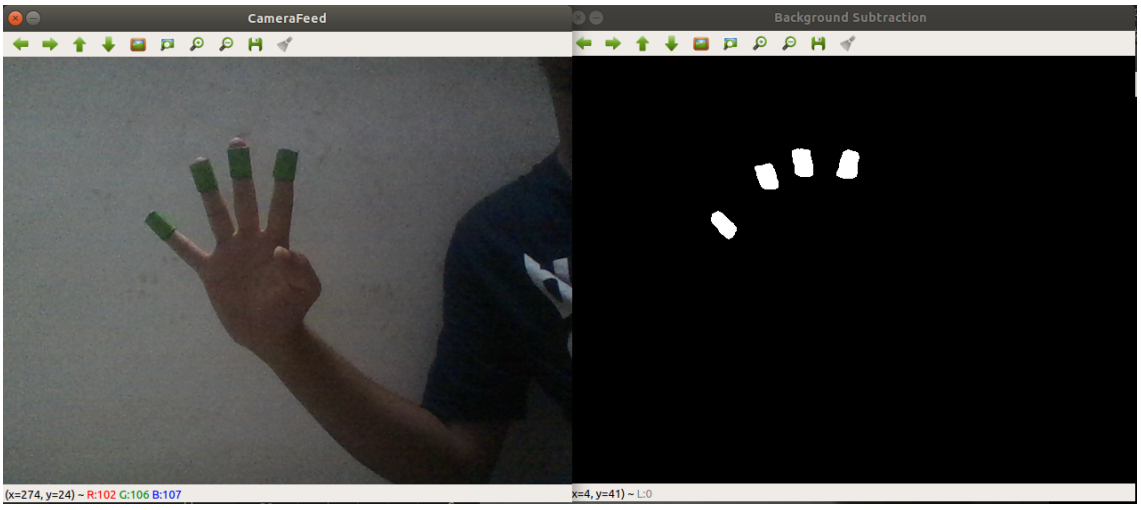


Figure 8.1: Background Subtraction Module

8.2.2 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 8.2: Feature Extraction Module

8.2.3 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.

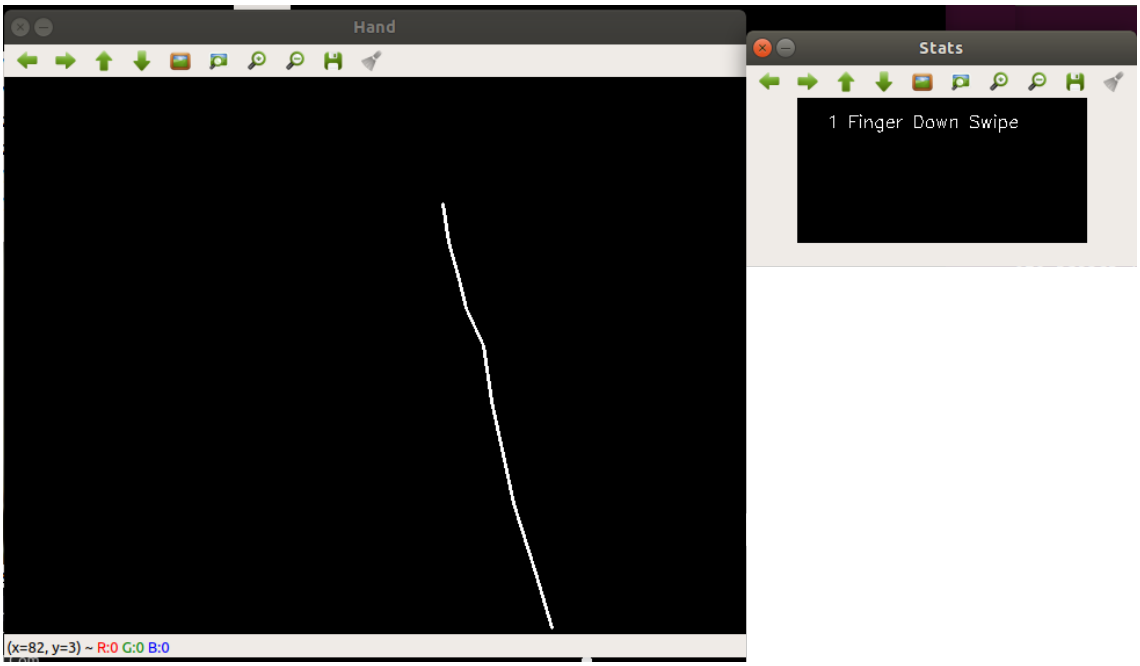


Figure 8.3: Finger Tip Tracking and Gesture Recognition

8.3 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.

8.4 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 9

Graphical User Interface

9.1 GUI Overview

9.2 Main GUI Components

Chapter 10

Results

Include screenshots of the project.

Chapter 11

Conclusion

Chapter 12

Future Scope

Chapter 13

Publication

References

[1]