A

Project Report
(Semester I)

on

Handling Basic Windows OS Operations using Gesture Input

by
Saket Joshi (BE-A 53)
Shounak Gujarathi (BE-A 55)
Abishek Mirge (BE-A 56)

Under the guidance of
Prof. Chaitali Patil

Department of Computer Engineering

K K Wagh Institute of Engineering Education and Research , Nashik

University of Pune
YEAR 2013-2014

CERTIFICATE

This is to certify that the project entitled "Handling Basic Windows OS Operations using Gesture Input" is a bona fide record by Saket Joshi(BE-A 53), Shounak Gujarathi(BE-A 55) and Abhishek Mirge(BE-A 56) in partial fulfilment of the requirements for the Degree of Computer Engineering of University of Pune from K K Wagh Institute of Engineering Education and Research, Nashik for the semester I of academic year 2013-2014

Prof. Chaitali Patil	**Prof.Dr.S.S.Sane**
Project Guide	**Head,**
Dept. of Comp. Engg.	**Dept. of Comp. Engg.**

Date: 14 th October 2013

Abstract

We propose handling basic Windows OS operations using gesture input in which object is decided by the user and based on its trajectory, we detect gestures. These gestures are used as a real time input to the windows operating system based on which basic windows tasks like window minimization, maximization, close, restore volume control, opening start menu, accessing command prompt etc. can be performed. The choice of assigning a specific gesture to a specific operation is customizable to the user.

The first phase is the training phase which includes selecting an object and using it for input there onwards so that gesture input can be recorded.

In the second phase, the specified object is detected and tracked using CAMshift and SURF algorithm, based on its trajectory, matching is done with the pre-recorded gestures, of which, one is chosen.

In the third phase, based on the accepted gesture input, the assigned operation is performed.

In this project, the main features of our software include:

- Recognition based on motion cues of an object specified by the user
- Invariance to object rotation, scaling, occlusion
- Invariance to light conditions

The CAMshift algorithm is a fast and effective algorithm for real-time object tracking based on its color. In case of losing the object, we are using the SURF algorithm since it is gray scale based and transformation invariant.

Keywords: object detection, gesture, tracking

Contents

List of Tables

List of Figures

Chapter 1

Introduction

- **Detailed Problem Definition**

  Depending on the available environmental circumstances it is not always feasible to use a primitive input device like Mouse or Keyboard. In addition to this if we use mouse or keyboard as an input then it is static to that particular position location. Hence to all above problems we provide a solution that is an Object based Gesture Input which will be useful to the user to perform his basic windows OS operations from a distance without using primitive input devices.

  So the problem statement is Handling of basic windows operations such as switching between the tabs of windows, minimize, maximize and closing a window, opening the start menu, task manager, command prompt, etc. using Object based Gesture Input.

  The front end of this system is implemented using Visual Studio. The APIs required for video and image processing are provided by a library named Open CV (Open Computer Vision). The system will be able to perform all the functionalities of as specified in real time and hence provide a suitable alternative to primitive input devices.

- **Current Market Survey**

  - ▹ **Leap Motion**
    - ◦ Cost is very high.
    - ◦ Requirement of an external hardware device.
    - ◦ Works only with Airspace apps.
  - ▹ **Sony Vaio Gesture Input**
    - ◦ Works only on Sony Vaio Devices.
    - ◦ Inaccurate in object recognition.

▹ **Hand based Gesture Input**

◦ System limitations restrict the applications such as ges tures are made with the hand only, the palm is facing the camera, and background is plain and uniform.

- **Need of the System**
  - ◦ Control the computer from a distance.
  - ◦ Secondary input as compared to primitive keyboard and mouse to control the Windows OS operations.

- **Advances to the previous systems**

  As stated earlier in the Current Market Survey, existing Gesture Based Techniques require either an additional hardware or they are required to satisfy the constraints.

  Object based Gesture Input can be viewed as advancement to the existing ones as it works without additional hardware and could work with any object having definite properties like dimensions and color.

- **Organization of the report**

  This report is divided into three main section. Section I focuses on Requirement Analysis. Methodology used is discussed in section II and section III consists of Modeling and Design.

Chapter 2

RequirementAnalysis

2.1 Requirement analysis

▹ Purpose: Purpose of the project is to handle basic windows OS opera tions using Object based Gesture Input.

▹ Scope:
- ◦ Basic windows OS operations.
- ◦ User uses one object at a time.

▹ Assumptions:
- ◦ User inputs one gesture at a time
- ◦ One Object is used by the User at a time.

▹ Dependencies:
- ◦ Computational Power
- ◦ Quality of Camera

▹ Product Functions:
- ◦ Perform Object Registration.
- ◦ Object Detection and Tracking.
- ◦ Defining Gestures and associated Operations.

▹ User Classes and Characteristics:

◦ Users of this system are only considered to be naive users, they are assumed to have basic knowledge of computers. Users will be differentiated into four as Registered and Unregistered Users and in registered ones again we bifurcate into two as those users who are Trained and Untrained users. Almost all the specifications concerning the flawless execution of the system will be hard coded and only the part of accurately interacting with the system through the provided interface will be left to the user hence freeing him from unnecessary concerns and making the interaction simple for the user.

▹ Operating Environment:

- Software Requirements
    - ⋆ Microsoft .NET Framework.
    - ⋆ EMGU OpenCV libraries.
    - ⋆ Windows API libraries.
    - ⋆ Microsoft Visual Studio.

- Hardware specifications
    - ⋆ Intel Core2Duo 2.66 GHz.
    - ⋆ 2 GB of system RAM.
    - ⋆ 4 GB HDD free space.
    - ⋆ Webcam/External Camera.
    - ⋆ Custom Object.

- Platform
    - ⋆ Windows 7 OS and above.

2.1.1 System Features

- Detection of a customizable object.
- Tracking of object..
- Recording custom gestures.
- Assigning gestures to tasks.
- Accepting gesture input.
- Matching with pre-recorded gestures.
- Executing the task associated with that gesture.

2.1.2 Nonfunctional Requirements

Table 2.1: Non-functional requirements

Performance	Response Time	
Capability	Broadness/Throughput	1 gesture at a time
Extensibility	Can be changed	New Gestures can be added
Flexibility	Can be modified	Objects and gestures can be customized
Security	Privacy	Only user selected object works
Usability	Easy to use	Can be used with initial help

2.1.3 Design and Implementation Constraints

The proposed system should be written in a language with strong GUI links and a simple, accessible windows API. Front end can be designed by using Visual Studio. The Open CV (Open computer vision) library is used to provide an interface between the application and the hardware component and also to implement various computations via inbuilt functions. The system must provide a capacity for executing real time operation. The design does not depend on scalability as the application is intended for a stand-alone system. The end system should also allow for seamless recovery, without data loss, from input, output or operational errors. It is worth noting that this system is likely to conform to what is available. With this assumed, the most adaptable and portable technologies should be used for the implementation. The system has criticality as it has to provide required functionality in real time. The system must be reliable enough to execute without crashing or producing any glitches in the computation, or if any implementing a recovery mechanism so that the system occurs flawless to then end user.

2.1.4 User Documentation

We are going to provide the user with-

- Tutorial Video
- Tutorial File

2.2 Project plan

2.2.1 Project Plan

Table 2.2: Project Plan

Month	Week	Task
July	1st week	Project Search
	3rd week	Project Selection and Search
	5th week	Check Feasibility
August	1st week	Abstract Preparation
	2nd week	Synopsis Preparation
	3rd and 4th week	Software Requirement Specification
September	1st week	Mathematical Model
	3rd week	Preparation of GUI
October	1st week	UML Behavioral Diagrams
November	1st week	UML Structural Diagrams
	2nd week	Design and Model
December	3rd week	Implement Modules
	4th week	Implement Modules
January	1st week	Implement Modules
	2nd week	Implement Modules
	3rd week	Test Cases
	4th week	Testing Various methods
February	1st week	Testing Various methods
	2nd week	Testing Various methods
	3rd week	Testing Various methods
	4th week	Testing Various methods
March	1st week	Software Deployment

2.3 Team structure

Table 2.3: Team Structure

1	Saket Joshi	UML, Code Design, Testing
2	Shounak Gujarathi	Code Design, Testing and Deployment, Survey code design, Documentation
3	Abhishek Mirrge	Survey, code design, Literature, Mathematical Model, Domain Study, Testing

Chapter 3

Methodology

3.1 Software Requirements

1. Microsoft .NET Framework
2. EMGU OpenCV libraries
3. Windows API libraries
4. Microsoft Visual Studio

3.2 Hardware specification

1. Intel Core2Duo 2.66 GHz
2. 2 Gb of system RAM
3. 4 Gb HDD freespace
4. WebCam/External Camera
5. Custom Object

3.3 Programming language

1. Visual Basic
2. C#
3. Visual C++

3.4 Platform

Windows 7 and above

3.5 Components

1. Registration and Training Module
2. Object Detect and Tracking Module
3. Operational Module

3.6 Tools

1. Microsoft Visual Studio
2. OpenCV

3.7 Coding style followed

1. Proper indentation done wherever necessary
2. Program divided into several modules
3. Good elaborative documentation
4. Object oriented programming approach followed

Chapter 4

Modeling and Design

4.1 UML diagrams / DFDs

1. Sequence Diagram
2. Activity Diagram
3. Collaboration Diagram
4. Component Diagram
5. Deployment Diagram
6. Use Case Diagram
7. Data Flow Diagram
8. Class Diagram

Sequence Diagram

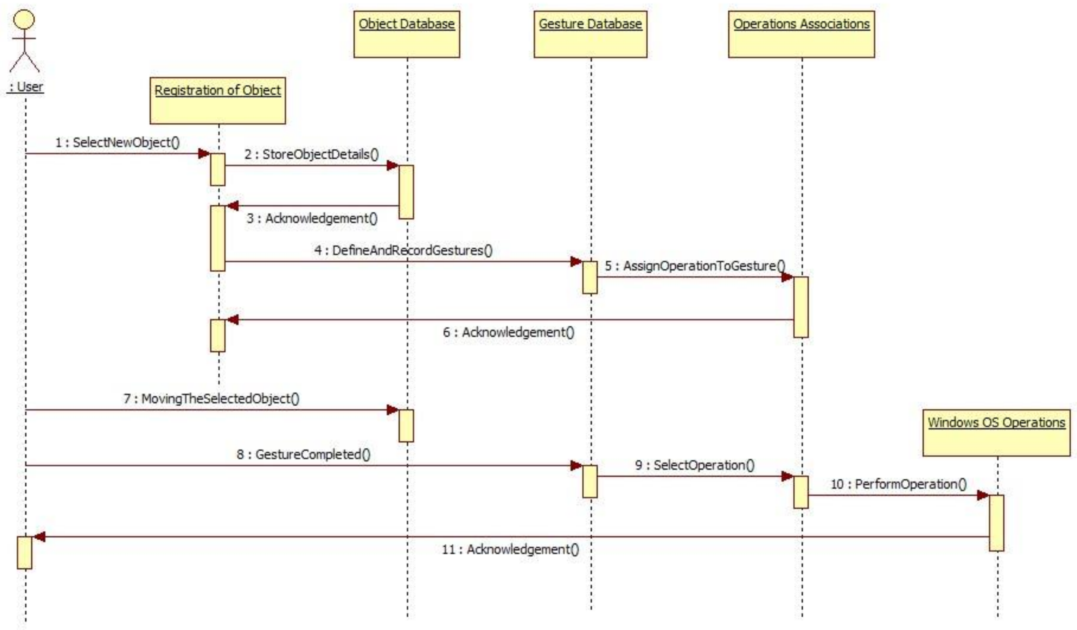

Figure 4.1: Sequence Diagram

Activity Diagram

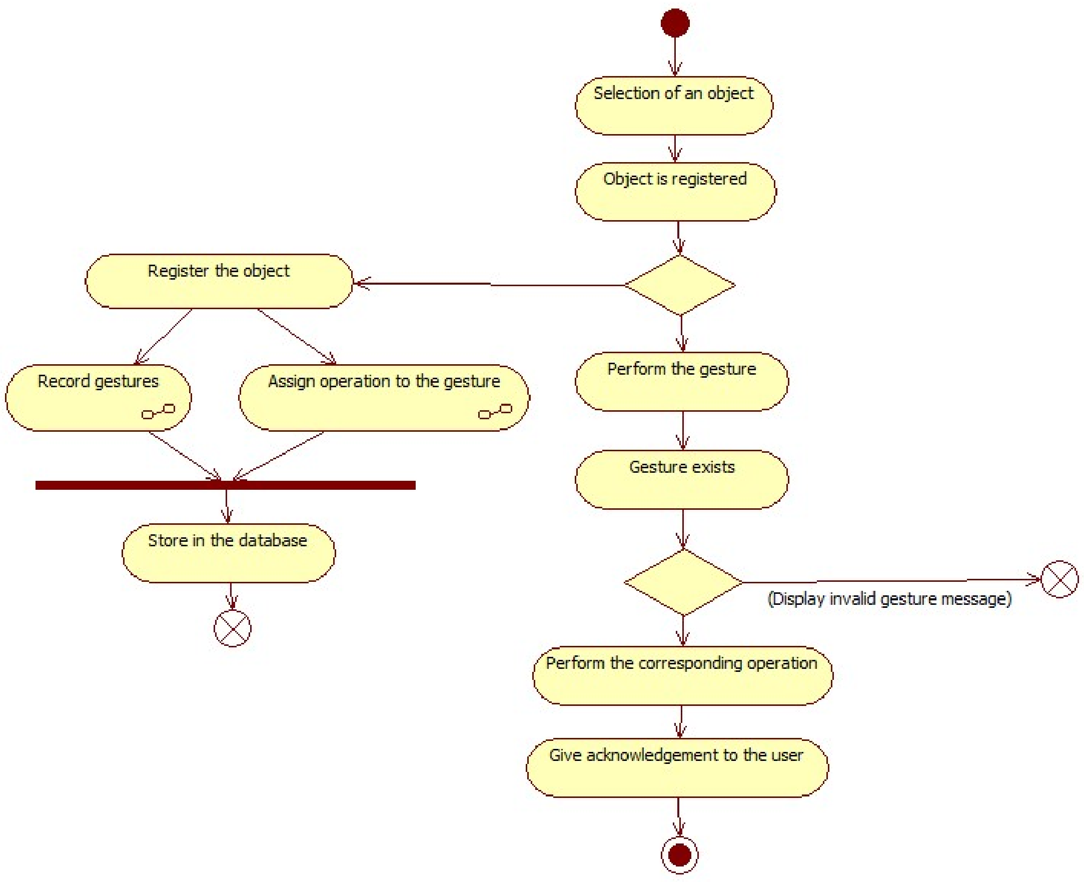
Selection of an object
Object is registered
Register the object
Record gestures
Assign operation to the gesture
Store in the database
Perform the gesture
Gesture exists
(Display invalid gesture message)
Perform the corresponding operation
Give acknowledgement to the user

Figure 4.2: Activity Diagram

Collaboration Diagram

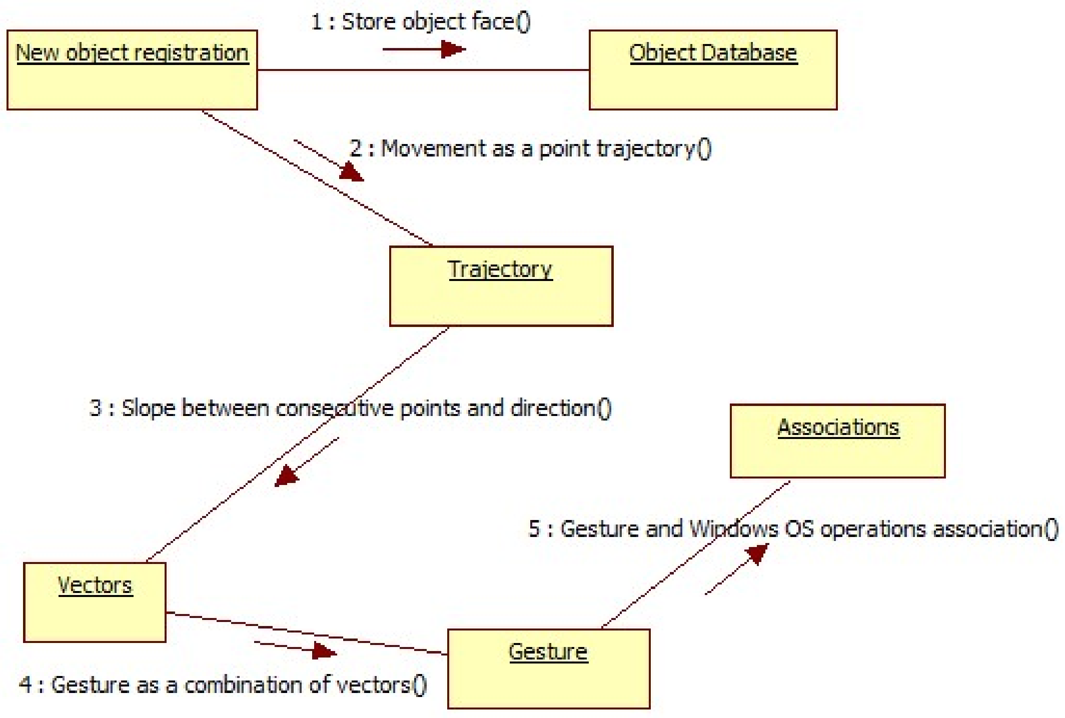
New object registration
1 : Store object face()
Object Database
2 : Movement as a point trajectory()
Trajectory
3 : Slope between consecutive points and direction()
Associations
5 : Gesture and Windows OS operations association()
Vectors
Gesture
4 : Gesture as a combination of vectors()

Figure 4.3: Collaboration Diagram

Component Diagram

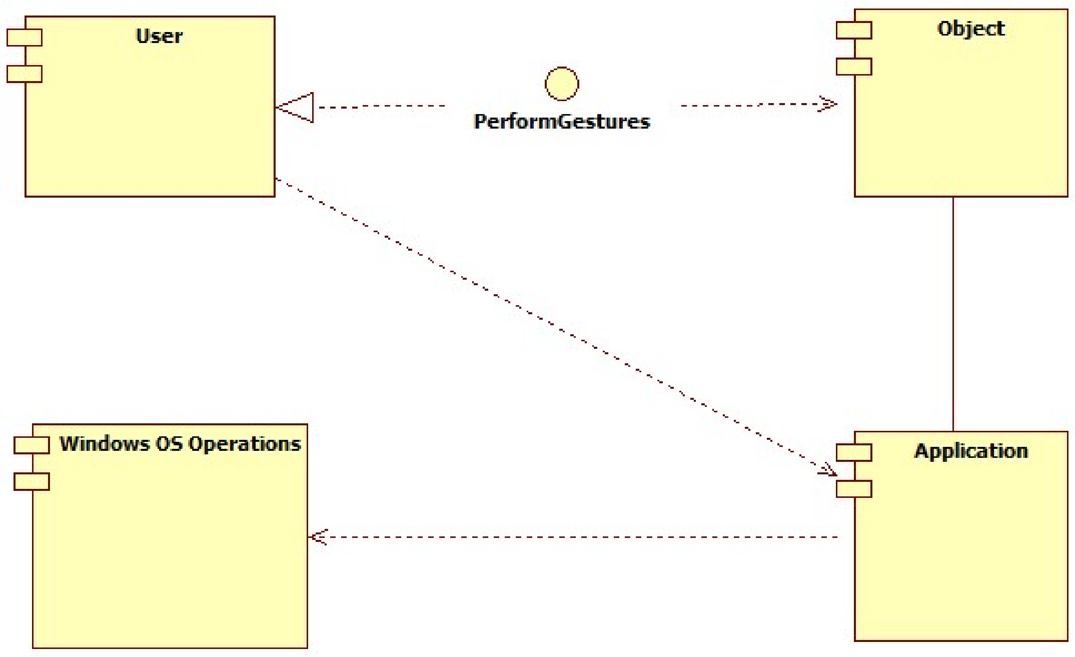

Figure 4.4: Component Diagram

Deployment Diagram

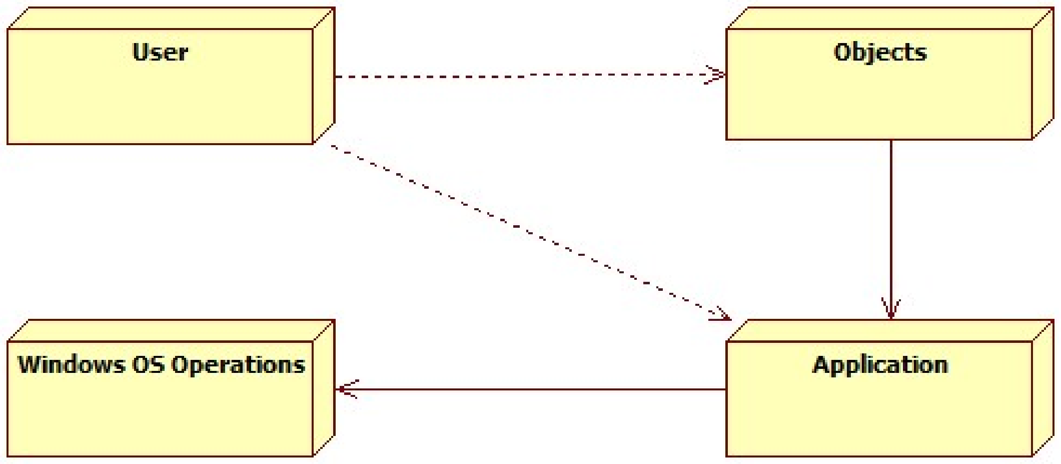

Figure 4.5: Deployment Diagram

Use Case Diagram

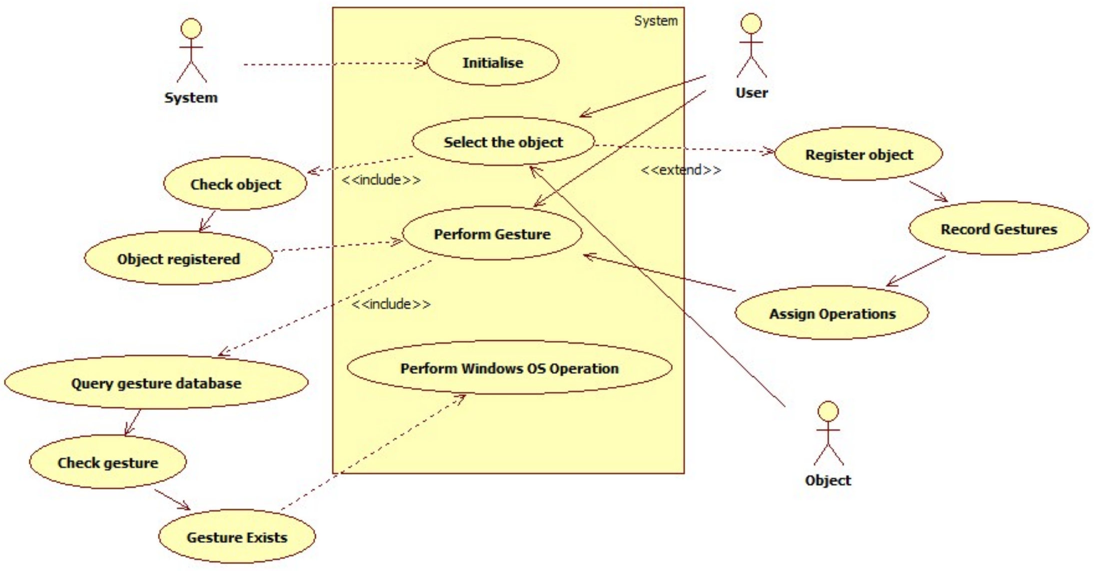

Figure 4.6: Use Case Diagram

Data Flow Diagram

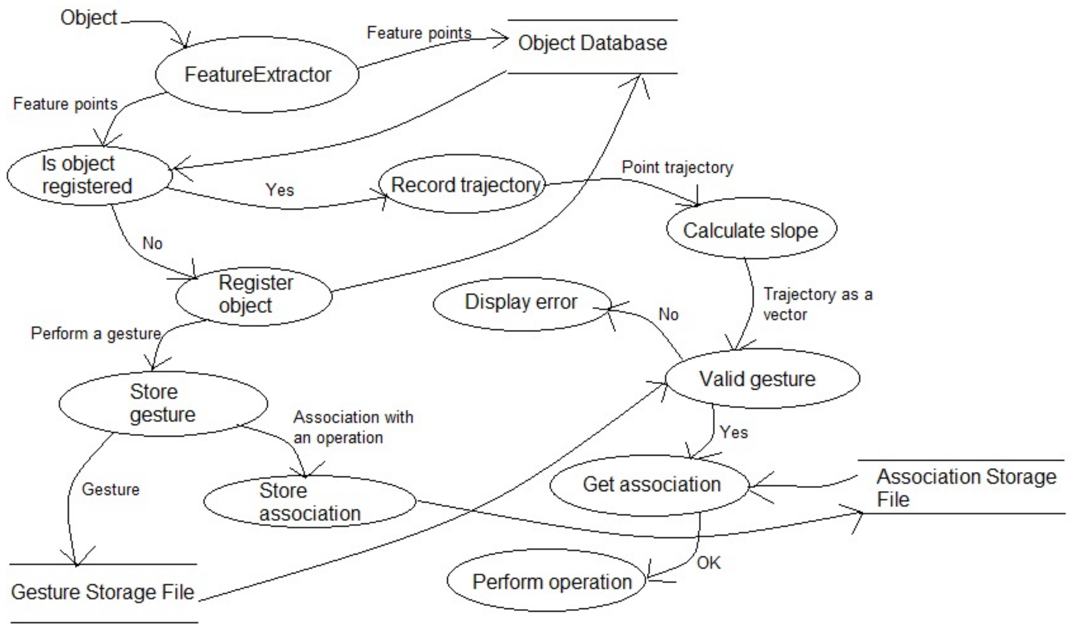

Figure 4.7: Data Flow Diagram

Class Diagram

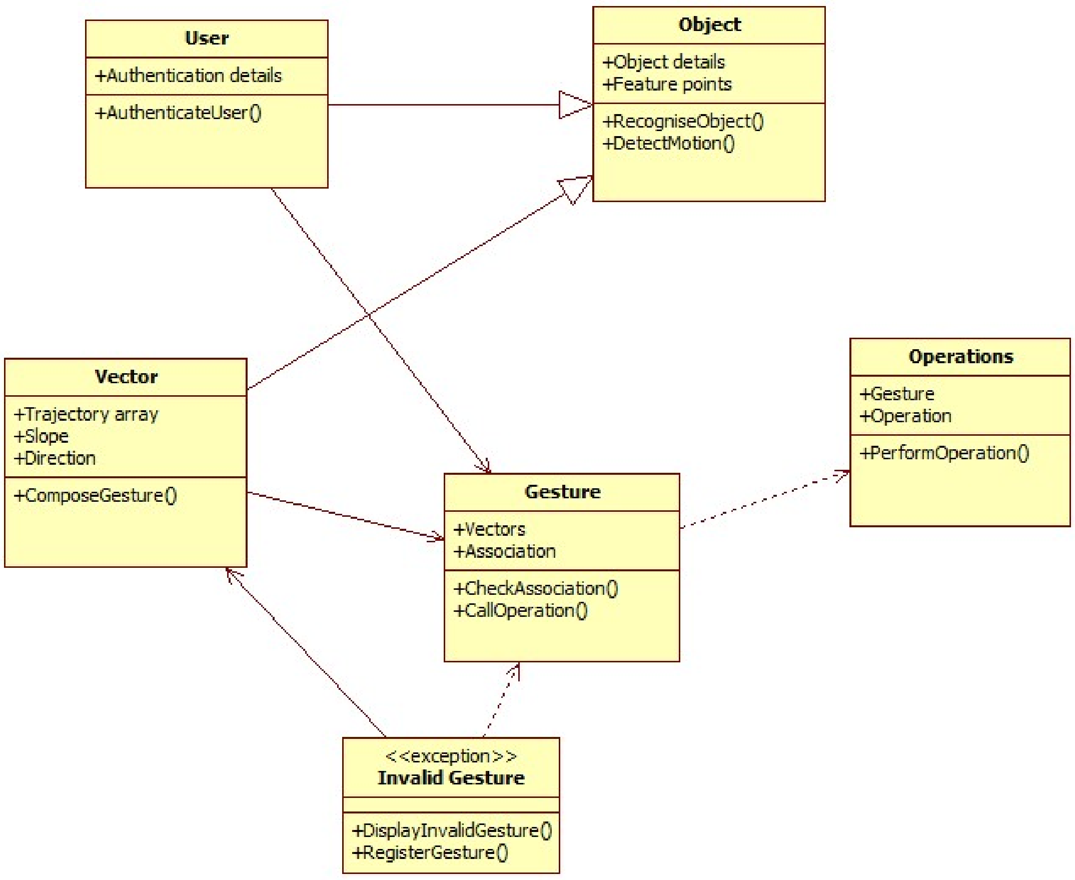

Figure 4.8: Class Diagram

Appendix A

Glossary

Table 1: Glossary

Object tracking	Detecting the Object in a real-time video.
Camshift	**Continuously Adaptive Mean Shift** is an algorithm for image segmentation color introduced by Gary Bradski in 1998, the Camshift algorithm cleverly exploits mean-shift by changing the size of the window when it came to convergence.
SURF	SURF **Speeded Up Robust Features** is a robust local feature detector, first presented by Herbert Bay et al. in 2006, that can be used in computer vision tasks like object recognition or 3D reconstruction.
OpenCV	OpenCV (for **Open Computer Vision**) is a graphics library free , originally developed by Intel, which specializes in image processing in real time. The company's robotic Willow Garage provides support for this library since 2008.
HSV	HSV stands for **hue, saturation, and value**, and is also often called HSB (B for brightness). Hue can be said as the *p*ure color, the saturation is the composition of the color whereas the value is the brightness

Appendix B

Design and Analysis
Algorithms used:

1. CAMshift Algorithm

2. SURF Algorithm

Description:

- **Algorithm 1:**

```
Input: Template Image
Output: Location of the target

– Step I: begin
– Step II: Read the Template image and the first frame of the Image
  sequence, calculate the location of the target using SURF method;
– Step III: Calculate the color histogram of the Template image,calculate
  the thresholds of S and V, filter the noise of the background;
– Step IV: while next frame do
    ∗ i: Update the size and the location of the search window using
      CAMshift;
    ∗ ii: Calculate Bhattacharrya distance
    ∗ iii: if Bhattacharrya distance < 0.8 then
              Lost the target, recalculate the location of the target using
              SURF method;
           else
              Return location of target
– Step V: end
```

- **Algorithm 2:**
  Input: Location of target
  Output: Trajectory Co-ordinates

  - **Step I: begin**
  - **Step II:** Set Timer to an appropriate value(say 500 ms)
  - **Step III: repeat** *algorithm I* for every cycle of the timer
  - **Step IV:** Store the co-ordinates in db1.xml
  - **Step V: repeat** *steps I to IV* until object out of view
  - **Step VI: end**

- **Algorithm 3:**
  Input: Trajectory Co-ordinates
  Output: Slope+Directon

  - **Step I: begin**
  - **Step II: while** the *whole sequence* has not been read,**do**
    - **i:** Read co-ordinates from db1.xml
    - **ii:** Calculate slope and direction for a pair of co-ordinates
  - **Step III:** Store the slopes and directions in db2.xml
  - **Step IV: end**

- **Algorithm 4:**
  Input: Slope+Direction
  Output: Vectors

```
- Step I: begin
- Step II: while the whole sequence has not been read do
    * i: Read slopes and directions from db2.xml
    * ii: Calculate vector a pair of slope and direction
- Step III: Calculate number of vectors and vector type and store in db3.xml
- Step IV: end
```

- **Algorithm 5:**
  Input: No. of Vectors, Vector Type
  Output: Gesture

```
- Step I: begin
- Step II: Read no. of vectors
- Step III: while all the no of vectors is not zero do
    * i: add vector type to array
    * ii: point to next array position, decrement vector no by 1
- Step IV: Match the array contents with the contents of gesture database(say gesturedefinations.xml)
- Step V: if match found then
          gesture found, search for the corresponding windows operation in the gesture-windows operation database(say ges-winop.xml);
      else
          gesture not found, initialize algorithm I
- Step VI: end
```

Appendix C

1. **Introduction:**
   A strategy for software testing integrates test case design methods into a well-planned series of steps that result into the successful construction of software. It provides a roadmap that describes the steps to be conducted as a part of testing. Following is the test plan used:

   - Testing begins at module level and works outward towards the integration of entire computer based system.
   - Different testing techniques are appropriate at different point in time.
   - Testing is conducted by all members of the project.

2. **Test Plan Objectives:**
   This test plan for Object based Gesture Input supports the following objectives:

   - Define the activities required for conduct System, Beta and User Acceptance testing.
   - Communicate to all responsible parties the System Test strategy.
   - Define deliverables.
   - Communicate to all parties various Dependencies and Risks.

3. **Test Strategy:**
   The test strategy consists of a series of different tests that will fully exercise the system. The primary purpose of these tests is to uncover the system limitations and measure its full capabilities. A list of the various planned tests and a brief explanation below:

   (a) **Unit Testing:**
   Unit testing focuses verification effort on the smallest unit of software design the software component or module. It also gives an advantage that the modules of software project get tested for bugs individually and can be modified according to functionality as specified in SRS.

(b) **Performance Testing:**

It will be conducted to ensure that our systems response time meets the user expectations and does not exceed the specified performance criteria. During these tests, response time will be tested under heavy load and volume.

(c) **Regression Testing:**

Each time a new module is added as part of integrating the software changes, regression testing is done to ensure that newly added module do not cause problems with functions that earlier worked flawlessly.

4. Test Plan Outline

Table 2: Test Plan Outline

Modules of Development Phase	Testing Strategy	Duration
Training Phase-Object and User Registration	Validation and Verification	4 days
Training Phase-Gesture Recognition and assigning operations	Unit Testing	2 weeks
Object Detection and Tracking Phase	Alpha Testing	2 weeks
Operation Phase	Integration Testing	2 weeks

Acknowledgements

We gratefully acknowledge the contributions to the project under the guidance of Prof. Chaitali Patil.

We also thank our Principal Prof. Dr. K.N.Nandurkar, Head of Department Prof. Dr.S.S.Sane, Project Coordinator Prof. S.M.Kamlapur, our parents and all well wishers for their support which played significant role in completion of this project.

Bibliography

- Object tracking using improved Camshift with SURF method - Jianhong Li, Zhenhuan Zhou, Wei Guo, Bo Wang, Qingjie Zhao

- Speeded Up Robust Features (SURF) - Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool

- en.wikipedia.com - Wikipedia, The online free encyclopedia

- www.ieeexplore.ieee.com

- Computer vision face tracking for use in a perceptual user interface - G. R. Bradski

- Introduction to programming with OpenCV - Gady Agam

- Notes on OpenSURF library - C. Evans