SHIVAJI UNIVERSITY

A Vision Based Color Code Gesture Interface For Controlling VLC Media Player

A thesis submitted in partial fulfillment for the

in the

Department of Computer Science and Engineering
Sharad Institute of Technology College of Engineering,Yadrav-Ichalkaranji

April 2015

SHIVAJI UNIVERSITY

Abstract

Department of Computer Science and Engineering
Sharad Institute of Technology College of Engineering,Yadrav-Ichalkaranji

Human Computer Interaction can acquire several advantages with the introduction of different natural forms of device free communication. Gestures are a natural form of actions which we often use in our daily life for interaction, therefore to use it as a communication medium with computers generates a new paradigm of interaction with computers. This paper implements computer vision and gesture recognition techniques and develops a vision based low cost input device for controlling the VLC player through gestures. VLC application consists of a central computational module which uses the Principal Component Analysis for gesture images and finds the feature vectors of the gesture and save it into a XML file. The Recognition of the gesture is done by K Nearest Neighbour algorithm. The theoretical analysis of the approach shows how to do recognition in static background. The Training Images are made by cropping the hand gesture from static background by detecting the hand motion using Lucas Kanade Pyramidical Optical Flow algorithm. This hand gesture recognition technique will not only replace the use of mouse to control the VLC player but also provide different gesture vocabulary which will be useful in controlling the application.

Acknowledgements

We take this opportunity to express our profound gratitude and deep regards to our guide M.D Jilan and Mr. M. I. Mullani for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessing, help and guidance given by him time to time shall carry us a long way in the journey of life on which we are about to embark. We also take this opportunity to express a deep sense of gratitude to our Principal, Dr. S. A. Khot and our H.O.D., Mr. O. D. Joshi for their cordial support, valuable information and guidance, which helped us in completing this task through various stages. We are obliged to all faculty and staff members of all departments of our Sharad Institute of Technology, College of Engineering, Yadrav for the valuable information provided by them in their respective fields. We are grateful for their cooperation during the period of our assignment. Lastly, we thank almighty, our parents, brothers, sisters, friends and our colleagues for their constant encouragement without which this assignment would not be possible.

Mr. Tejas R. Amate

Mr. Rohit P. Tambwekar

Mr. Prasad K. Desai

Mr. Aniket A. kadlokar

Contents

List of Figures

Chapter 1

INTRODUCTION

1.1 Introduction of Project

As the ubiquitous computing paradigm that is predicted for the future is brought closer by technological advances, designers of new smart homes mediated spaces and sentient computer systems will have to consider new techniques to interact with users. The main Objective of this application is to implement software application for playing media files with the help of Media Player which can be controlled by using Gesture with the help of Webcam. This project is basically about using Web cam technology using Webcam API for real time interaction .This project broadly includes three main concepts in java namely Threading Color Detection ,basic operations of Media Player.[1] With the increasing use of computing devices in day to day life, the need of user friendly interfaces has lead towards the evolution of different types of interfaces for human computer interaction. Real time vision based hand gesture recognition affords users the ability to interact with computers in more natural and intuitive ways[2].

Human Computer Interaction can acquire several advantages with the introduction of different natural forms of device free communication. Gestures are a natural form of actions which we often use in our daily life for interaction, therefore to use it as a communication medium with computers generates a new paradigm of interaction with computers. This paper implements computer vision and gesture recognition techniques and develops a vision based low cost input device for controlling the VLC player through gestures[3].

Chapter 2

LITERATURE REVIEW

2.1 Literature Review

2.1.1 Human Computer Interaction

Human Computer Interaction can acquire several advantages with the introduction of different natural forms of device free communication. Gestures are a natural form of actions which we often use in our daily life for interaction, therefore to use it as a communication medium with computers generates a new paradigm of interaction with computers. This paper implements computer vision and gesture recognition techniques and develops a vision based low cost input device for controlling the VLC player through gestures. This hand gesture recognition technique will not only replace the use of mouse to control the VLC player but also provide different gesture vocabulary which will be useful in controlling the application.

2.1.2 User Friendly Interfaces

With the increasing use of computing devices in day to day life, the need of user friendly interfaces has lead towards the evolution of different types of interfaces for human computer interaction. Real time vision based hand gesture recognition affords users the ability to interact with computers in more natural and intuitive ways. [1]

2.1.3 Real Time Interaction

Object recognition technologies using PCA (principal component analysis) recognize objects by deciding representative features of objects in the model image, extracting feature vectors from objects in a image and measuring the distance between them and object representation. Given frequent recognition problems associated with the use of point-to-point distance approach, this study adopted the k-nearest neighbor technique (class-to-class) in which a group of object models of the same class is used as recognition unit for the images inputted on a continual input image. However, the robustness of recognition strategies using PCA depends on several factors, including illumination. When scene constancy is not secured due to varying illumination conditions, the learning performance the feature detector can be compromised, undermining the recognition quality. This paper proposes a new PCA recognition in which database of objects can be detected under different illuminations between input images and the model images.

2.1.4 Hand-Shape Recognition

First we gather a data set of all the hand-shapes we wish to recognize. A naive approach to recognizing a new image D would be to simply compare it with all the images stored in the data set and find the target image T with the closest match. But because there are so many images in the data set this will take far too long. We can reduce the time by using a multi-scale approach. We divide up the data set into groups of images, which are similar to one another by blurring the images at different levels so that small differences between similar images will be eroded. Thus a whole group of original images may become reduced to just one image, which represents the entire group. [2]

Chapter 3

OBJECTIVE AND SCOPE

3.1 Objective and Scope

3.1.1 Objectives

The Objectives of our proposed system are:

- The file is a Java-based PC application to develop Media Player for playing any format media files.
- The second part is to control the media player with gesture (by color detection techniques).
- The application uses a hybrid approach for hand gesture recognition which recognizes static hand gestures.
- The images are captured from camera and then passed to different algorithms for learning and recognition.
- The computer vision techniques used for the application. Gesture recognition can be conducted with techniques from computer vision and image processing.
- This computing not only going to reduce the hardware impact of the system but also it increases the range of usage of physical world object instead of digital object like keyboards, mouse.
- Using this we can implement and can create a new thesis of creating of new hardware no requirement of monitors too.

3.1.2 Scope

- We will develop a java based media player application for any media format.
- We will control this media player with hand gesture and color detection.
- Media Player which can be controlled by using Gesture with the help of Webcam.
- This project is basically about using Webcam technology using Webcam API for real time interaction.
- This project broadly includes three main concepts in java namely Threading, Color Detection, basic operations of Media Player.

3.1.3 Out of Scope

- The present application is less robust in recognition phase.
- Some more robust algorithms can be used to reduce noise and blur motion.
- We are not implementing pure hand gesture.

Chapter 4

REQUIREMENT ANALYSIS

4.1 Hardware Requirements

Computer with following minimum configuration:

1. Processor:

   Recommended: core i3 and above

2. RAM:

   Recommended: minimum 2GB and above.

4.2 Software Requirements

4.2.1 Operating System

Windows 7 (For a 32-bit and 64-bit operating system)

Recommended: Windows XP Professional x64, windows 7 and 8.

4.2.2 Platform

- Front End: J2SE: (VLCJ API, JDK) Application architecture. JAVA: Application architecture. Eclipse IDE Framework Webcam: to capture real time images.

- **Eclipse IDE Framework** In computer programming, Eclipse is an integrated development environment (IDE). It is developed by Eclipse Foundation. The stable release is Stable release 4.4.1 (Luna) / 26 September 2014. It contains a base

workspace and an extensible plug-in system for customizing the environment. Written mostly in Java, Eclipse can be used to develop applications. By means of various plug-ins, eclipse may also be used to develop applications in other programming languages: Ada, ABAP, C, C++, COBOL, Fortran, Haskell, JavaScript, Lasso, Natural, Perl, PHP, Prolog, Python, R, Ruby (including Ruby on Rails framework), Scala, Clojure, Groovy, Scheme, and Erlang. It can also be used to develop packages for the software Mathematica. Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++ and Eclipse PDT for PHP, among others.

Chapter 5

SYSTEM DESIGN

5.1 An Overview of UML

The UML is a language for

- Visualizing
- Specifying
- Constructing
- Documenting

THE UML LANGUAGE

A language provides a vocabulary and the rules for combing words in that vocabulary for the purpose of the communication. A modeling language is a language whose vocabulary and rules focus on conceptual and physical representation of a system. A modeling language such as the UML is thus a standard language for software blueprints. In this context, specifying means building models that are precise, unambiguous, and complete. In particular, the UML addresses the specification of all the important analysis, design and implementation decision that must be made in developing and deploying a software intensive system. The UML is not a visual programming language, but its model can be directly connected to a verity of programming languages. This means that its possible to map from a model in the UML to a programming language such as java, cpp, or visual basic or even to tables in a relational database. Things that are best expressed graphically are done so graphically in the UML,whereas things that best expressed textually are done so in the programming language. A healthy software organization produces all sorts of artifacts in addition to raw executable code. These artifacts include requirements,

architecture, design, source code, project plans, tests, prototypes, releases. The UML addresses the documentation of a systems architectures and all of its details. The UML also provides for expressing requirements and for tests. Finally, The UML provides a language for modeling the activities of project planning and release management.

5.2 Goals of UML

The primary goals in the design of the UML were:

- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models. Provide extensibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development processes.Provide a formal basis for understanding the modeling language
- Encourage the growth of the OO tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices

5.3 A Conceptual Model of UML

To understand the UML, you need to form a conceptual model of the language, and this requires learning three major elements: the UMLs basic building blocks, the rules that dictate how those building blocks may be put together, and some mechanisms that apply throughout the UML. Once you have grasped these ideas, you will be able to read UML models and create some basic ones. As you gain more experience in applying the UML, you can build on this conceptual model, using more advanced features of the language.

5.3.1 Building Blocks of UML

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships

- Diagrams

These are the abstractions that are first-class citizens in a model relationships tie these things together; diagrams groups interesting collections of things.

5.4 Diagrams in UML

A diagram is the graphical presentation of a set of elements, most often rendered as a connected graph of vertices (things) and arcs (relationships). You draw diagrams to visualizing a system from different perspectives, so a diagram is a projection into a system. For all but the most trivial systems, a diagram represents an elided view of the elements that make up a system. The same element may appear in all diagrams. In theory, a diagram may contain any combination of things and relationships. The views that comprise the architecture of software intensive system. For this reason, the UML includes following diagrams:

- Use case diagram
- Class diagram
- Sequence diagram
- Deployment diagram

5.5 Use Case Diagram

A use case diagram is a diagram that shows a set of use cases and actors and their relationships. A use case diagram is a just special kind of diagram and shares the same common properties as do all other diagram-a name and graphical contents.

5.5.1 Usecase Diagram

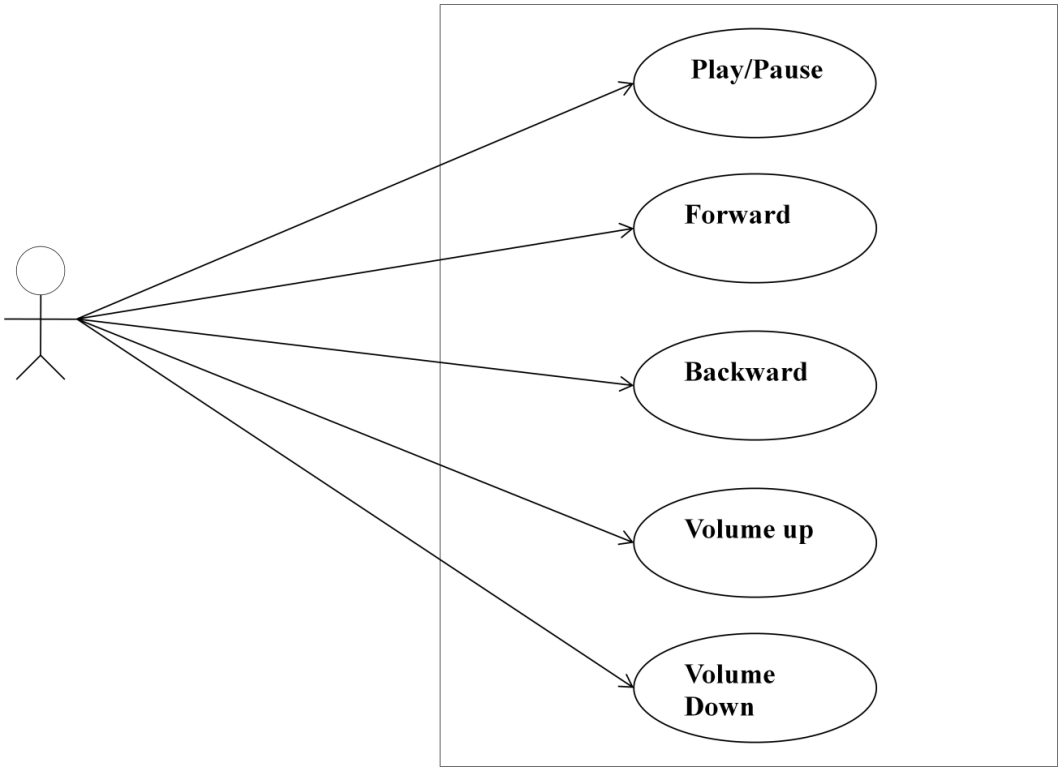

FIGURE 5.1: Usecase Diagram

5.5.2 Usecase Scenario:

5.5.3 Contents

Sequence diagram commonly contains

- Objects
- Links
- Messages

- **Use Case**

  Use case is a description of a set of sequence of actions that a system performs that yields an observable result of value to a particular actor. A use case is rendered as an ellipse with solid lines usually including its name.

TABLE 5.1: Use-Case Scenarios

USE CASE	**USE CASE SCENARIO**
Play/Pause	1) User holds the blue Color in-front of the Webcam. 2) Webcam captures the Color. 3) Color gesture motion detection. 4) Recognizing color gesture by Webcam. 5) Matching of color gesture. 6) Generating commands. 7) Giving the commands to VLC media player.
Forward/Rewind	1) User holds the red Color in-front of the Webcam. 2) Webcam captures the Color. 3) Color gesture motion detection. 4) Recognizing color gesture by Webcam. 5) Matching of color gesture. 6) Generating commands. 7) Giving the commands to VLC media player.
Volume up/Down	1) User holds the green Color in-front of the Webcam.
	2) Webcam captures the Color. 3) Color gesture motion detection. 4) Recognizing color gesture by Webcam. 5) Matching of color gesture. 6) Generating commands. 7) Giving the commands to VLC media player.

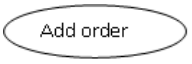

FIGURE 5.2: Use Cases

- **Dependency, generalization, and association relationships.**

  A dependency is a semantic relationship between two things in which a change to one thing may affect the semantics of the other thing.

FIGURE 5.3: Dependencies

  A *generalization* is a relationship in which objects of specialized elements (the child) are substitutable for objects of the generalized element.

  An *association* is a structural relationship that describes a set of links, a link being connection among objects

  Like all other diagrams, use case diagram may contain notes and constraints.

5.5.4 Common Uses

Use case diagram typically contain in one of two ways.

- To model the context of the system

  Hear system involves drawing line around the whole system and actors outside of the system and interact with it.

- To model the requirement of a system

  Hear specifies what the system should do, independent of how that system should do.

5.6 Sequence Diagram

A *sequence* diagram is an interaction diagram that emphasizes the time ordering of messages. A sequence diagram shows a set of objects and the messages sent and received by those objects. The objects are typically named or anonymous instances of classes, but may also represent instances of other things, such as collaborations, components, and nodes. You use sequence diagrams to illustrate the dynamic view of a system. An Actor models a type of role played by an entity that interacts with the subject (e.g., by exchanging signals and data), but which is external to the subject (i.e., in the sense that an instance of an actor is not a part of the instance of its corresponding subject). Actors may represent roles played by human users, external hardware, or other subjects. Note that an actor does not necessarily represent a specific physical entity but merely a particular facet (i.e., "role") of some entity that is relevant to the specification of its associated use cases.

Sequence diagram have two features that distinguish them from collaboration diagrams.

- First, there is the object lifeline. An object lifeline is the vertical dashed line that represents the existence of an object over a period of time. So these objects are at the top of the diagram. With their lifelines drawn from the top of the diagram to the bottom

- Second, there is the focus of control. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or through a subordinating procedure. The top of the rectangle is aligned with the start of the action; the bottom is aligned with its completion and also it can be marked by replay message.

5.6.1 Sequence diagram

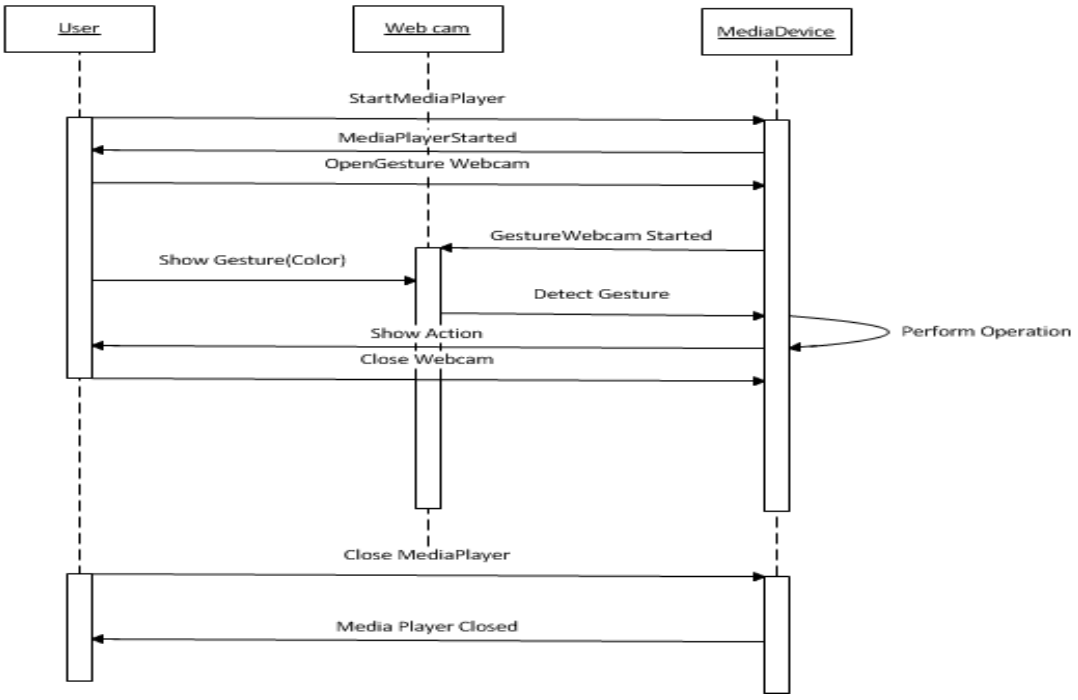

FIGURE 5.4: Sequence diagram

5.7 Class Diagram

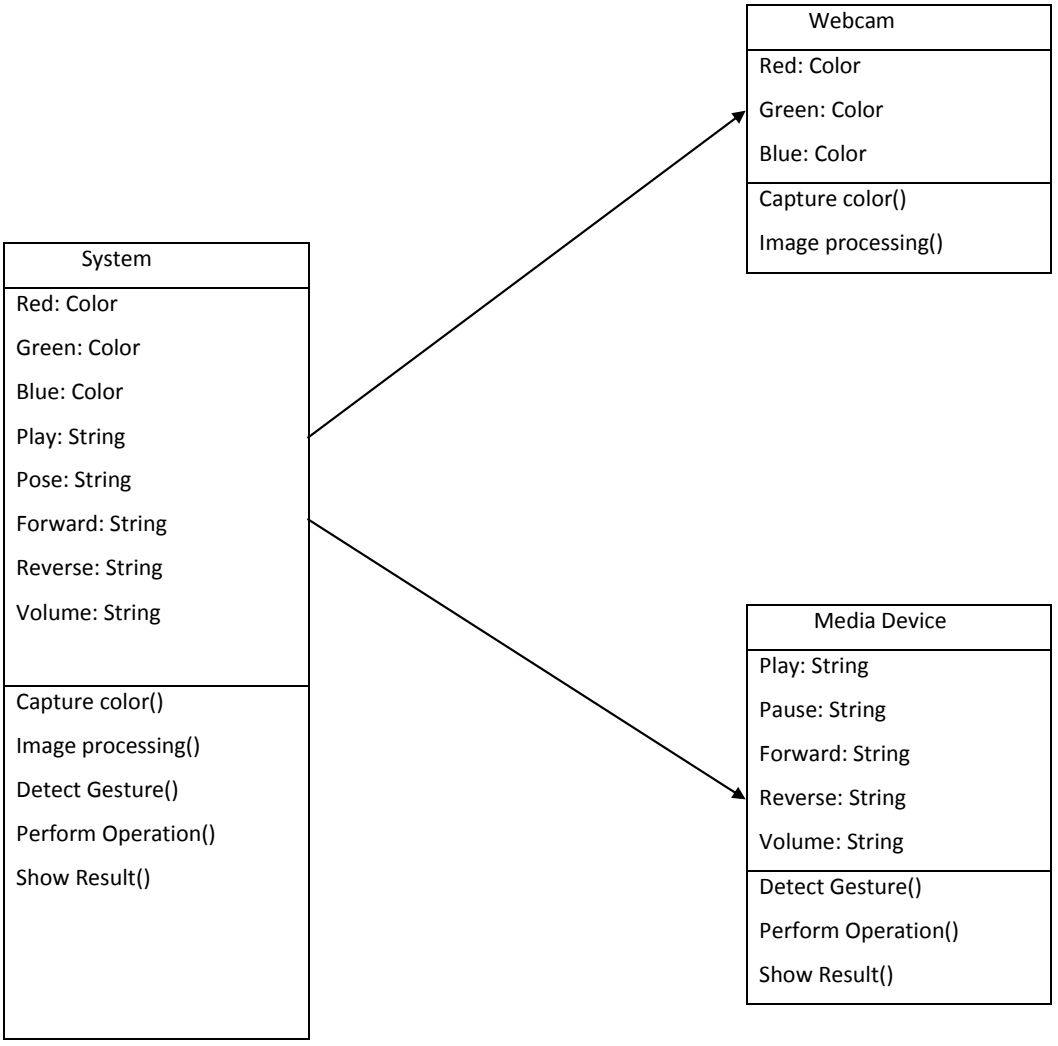

FIGURE 5.5: Class Diagram

5.7.1 Contents

Class diagram commonly contain the following things:

- Classes
- Interfaces
- Collaborations

- Dependency, generalization, and association relationships.

5.7.2 Definition and Common Uses

A class diagram is a diagram that shows a set of classes, interfaces and their relationships. Graphically, a class diagram is a collection of vertices and arcs. A class diagram will shares the same common properties as do all other diagrams. A class diagram is an illustration of the relationships and source code dependencies among classes in the Unified Modeling Language (UML). In this context, a class defines the methods and variables in an object, which is a specific entity in a program or the unit of code representing that entity. Class diagrams are useful in all forms of object-oriented programming (OOP). The concept is several years old but has been refined as OOP modeling paradigms have evolved. In a class diagram, the classes are arranged in groups that share common characteristics. A class diagram resembles a flowchart in which classes are portrayed as boxes, each box having three rectangles inside. The top rectangle contains the name of the class; the middle rectangle contains the attributes of the class; the lower rectangle contains the methods, also called operations, of the class. Lines, which may have arrows at one or both ends, connect the boxes. These lines define the relationships, also called associations, between the classes.

- Class: A definition of objects that share given structural or behavioral characteristics.
- Attribute: A typed value attached to each instance of a classifier.
- Operation: A method or function that can be performed by instances of a classifier

5.8 Deployment Diagram

subsectionDefinition A deployment diagram shows the configuration of run time processing nodes and the components that live on them. Deployment diagram address the static deployment view of architecture. They are related to component diagrams in that a node typically encloses one or more components

5.8.1 Nodes and Components

The UML provides a graphical representation of node. This canonical notation permits you to visualize a node apart from any specific hardware. Using stereotype this notation to represents specific kinds of processors and devices.

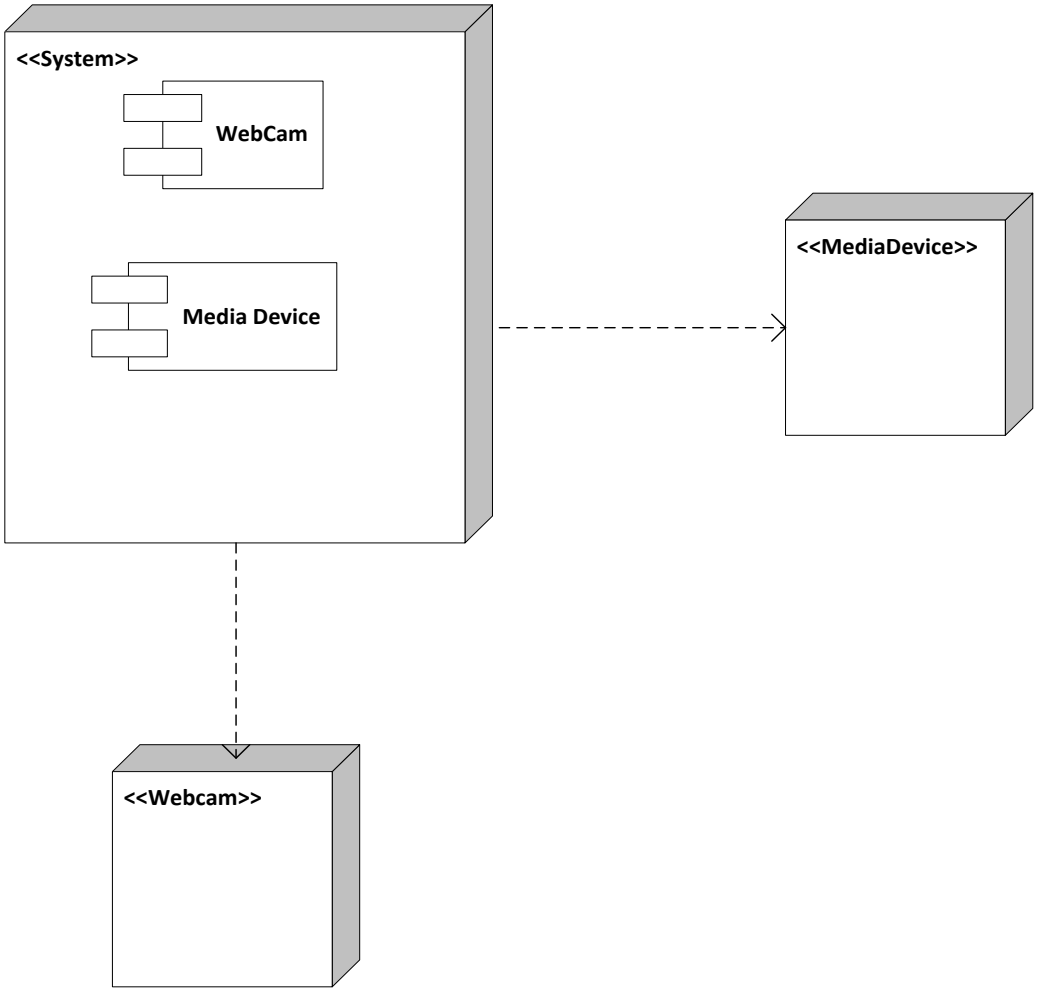

FIGURE 5.6: Deployment Diagram

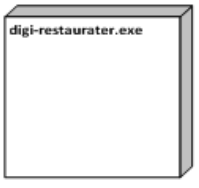

FIGURE 5.7: Node.

A *node* is a physical element that exists at run time and represents a computational resource, generally having at least some memory, and often processing capability. Graphically, a node is rendered as a cube. Every node must have a name that distinguishes it from other nodes. A name is a textual string. Components are things that participate in the execution of a system; nodes are things that execute components. Components represent the physical packaging of otherwise logical elements; nodes represent the physical development of components and components that things are executed by nodes. The

UML can often use stereotypes to specify new kinds of nodes that you can use to represent specific kinds of processors and devices. A *Processor* is a node that has processing capability, meaning that it can be executed by component. A*device* is a node that has no processing capability and general, represents something that interfaces to real world.

Chapter 6

SNIPPETS

6.1 Coding Standards

General coding standards pertain to how the developer writes code. The SISEPG has come up with a small set of items it feels should be followed regardless of the programming language being used.

6.1.1 Indentation

Proper and consistent indentation is important in producing easy to read and maintainable programs. Indentation should be used to:

- Emphasize the body of a control statement such as a loop or a select statement
- Emphasize the body of a conditional statement
- Emphasize a new scope block

A minimum of 3 spaces shall be used to indent. Generally, indenting by three or four spaces is considered to be adequate. Once the programmer chooses the number of spaces to indent by, then it is important that this indentation amount be consistently applied throughout the program. Tabs shall not be used for indentation purposes.

6.1.2 Structured Programming

Structured (or modular) programming techniques shall be used. GO TO statements shall not be used as they lead to spaghetti code, which is hard to read and maintain, except as outlined in the Standards and Guidelines.

6.1.3 Classes, Subroutines, Functions And Methods

Keep subroutines, functions, and methods reasonably sized. This depends upon the language being used. For guidance on how large to make software modules and methods, see section 4.0. A good rule of thumb for module length is to constrain each module to one function or action (i.e. each module should only do one thing). If a module grows too large, it is usually because the programmer is trying to accomplish too many actions at one time. The names of the classes, subroutines, functions, and methods shall have verbs in them. That is the names shall specify an action.

6.1.4 Source Files

The name of the source file or script shall represent its function. All of the routines in a file shall have a common purpose.

6.1.5 Variable Names

Variable shall have mnemonic or meaningful names that convey to a casual observer, the intent of its use. Variables shall be initialized prior to its first use.

6.1.6 Use of Braces

In some languages, braces are used to delimit the bodies of conditional statements, control constructs, and blocks of scope. Programmers shall use either of the following bracing styles: It is felt that the former brace style is more readable and leads to neater-looking code than the latter style, but either use is acceptable. Whichever style is used, be sure to be consistent throughout the code. When editing code written by another author, adopt the style of bracing used. Braces shall be used even when there is only one statement in the control block.

6.2 Issues and Problem overcome:

6.2.1 Webcam problem:

At first time we faced the problem of slow capturing rate of Gestures by the webcam, Whenever at the second time color code is shown to the Webcam it is taking to much

time to capture and detect Gesture. Begining there arise the error of slow capturing rate of Gestures by the webcam

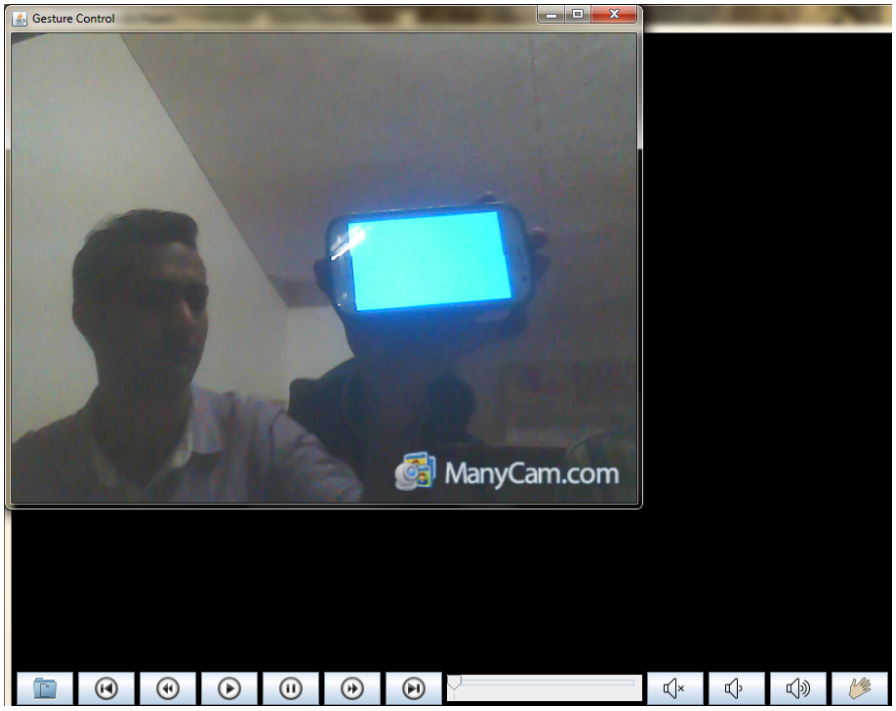

FIGURE 6.1: Webcam Error

6.2.1.1 Webcam problem overcome:

By minimizing the long distance of gesture to be shown to web camera and increasing the solid color gesture depth the slow capturing and processing is reduced.

6.2.2 Hand gesture color problem:

Hand gesture Problem color recognition based on shape features is highly influenced by some constraints like hand should be straight for orientation detection in image.

6.2.2.1 Problem overcome for hand gesture problem :

To detect the color movement position by using algorithm and perform specific media player operation such as play, pause, next song, previous song, forward song, backward song, control volume. By using the color gesture object the problem of slow capturing rate of gesture is minimized.

6.2.3 Compiler problem:

Compilers often issue two types of messages: warnings and errors. Compiler warnings normally do not stop the compilation process. However, compiler errors do stop the compilation process, forcing the developer to fix the problem and recompile Compiler and linker warnings shall be treated as errors and fixed.

6.2.3.1 Compiler problem overcome:

WebcamImageTransformed package is helful to compiler slow processing of gesture problem. Even though the program will continue to compile in the presence of warnings, they often indicate problems which may affect the behavior, reliability and portability of the code Some compilers have options to suppress or enhance compile-time warning messages. Developers shall study the documentation and/or man pages associated with a compiler and choose the options which fully enable the compilers code-checking features. It requires more time by web cam to capture gesture and more time to process the gesture one after the other by Media Player.

6.2.4 Time interval problem:

Whenever the color gesture is shown continuously to webcam it creates problem of detecting and processing the images as well as delay in operation.

6.2.4.1 Time interval problem overcome:

By reducing or minimizing the the set Interval time between gestures the problem of more time taken by webcam is minimized.

6.2.5 Video stream problem:

The input to the continuous video stream acquired by web camera.

6.2.5.1 Video stream problem overcome:

By creating the separate color gesture objects to be shown to web camera for image processing the the video streaming problem is minimized.

6.3 Review of Code

6.3.1 Snippest Code

6.3.2 Main:

In main method media GUI object is created and Thread is declared and socket is created to accept the requests from the clients. Data streams for accepts the input data. server socket is used to listen the input data such as gesture, acknowledgement to the client that it received.Thread object is created for handling multiple inputs from user. The connection is closed after the inputs are received.

```
public static void main(String[] args)
        {

final MediaGUI medGui = new MediaGUI();
medGui.setVisible(true);
        Thread t = new Thread()
        {
        public void run()
        {
        System.out.println("Server ready to listen.....");
        try
        {
        ServerSocket ss = new ServerSocket(1006);
        while (true)
        {
        Socket s = ss.accept();
        DataInputStream di = new
   DataInputStream(s.getInputStream());
        String rev = di.readUTF();
        medGui.Wifi(rev);
        System.out.println("Received from Client:" + rev);
        di.close();
        s.close();
        }
} catch (Exception e)
        {
```

```
        //e.printStackTrace();
        }
        }
};
t.start();

        }
```

6.3.3 Override:

This part of code performs action according to the Gesture shown by the user to the Webcam. The action event is performed, The x and y axis intersection and centre point is calculated to accept and capture the gesture. Media player timer is set to accept the gesture which is shown by the user to the webcam.

```
        int cnt=0;
        @Override
        public void stateChanged(ChangeEvent e)
        {
                cnt++;
        //      System.out.println("state changed");
                JSlider slider = (JSlider) e.getSource();
                if(cnt%2==0)
                mediaPlayer.getMediaPlayer().setTime((long)
   (slider.getValue()*Timer.factor));
        }

        public void mouseClicked(MouseEvent arg0)
        {
                // TODO Auto-generated method stub
                int x=arg0.getX();
                int y=arg0.getY();
//              System.out.println("In Click X="+x+"  Y="+y);

        }

        public void mouseEntered(MouseEvent arg0)
        {
                int x=arg0.getX();
```

```
                int y=arg0.getY();
        //      System.out.println("X="+x+"  Y="+y);
        //      System.out.println("Mouse Entered");
        //      panel2.setVisible(true);

        }

        public void mouseExited(MouseEvent arg0) {
                // TODO Auto-generated method stub

                //panel2.setVisible(false);
        }

        public void mousePressed(MouseEvent arg0) {
                // TODO Auto-generated method stub

        }

        public void mouseReleased(MouseEvent arg0) {
                // TODO Auto-generated method stub

        }
        public void mouseDragged(MouseEvent arg0) {
                // TODO Auto-generated method stub

        }
        public void mouseMoved(MouseEvent me)
        {
                int mouseX=me.getX();
                int mouseY=me.getY();
        //      System.out.println("Arrow Location
   X:"+mouseX+" Y:"+mouseY);

        }
}
```

6.3.4 GestureWebCam:

Webcam class is created for capturing the real time images,which implements Webcam motion listener interface. Monitor is module which is used to detect the gesture as well as it functions the image processing mechanism. The Gesture control window is created as well as the Gesture webcam opening Icon is generated on the Media Player. Webcam panel window size is also set to display the Webcam window.

```
class GestureWebCam implements WebcamImageTransformer,
    WebcamMotionListener
{
        private static final JHGrayFilter RGB = new
    JHGrayFilter();
        Webcam webcam;

        public BufferedImage transform(BufferedImage image)
        {
                return image;
        }

        Monitor monitor;

        public GestureWebCam(MediaGUI mediaPlayer)
        {

                webcam = Webcam.getWebcams().get(0);

    webcam.setViewSize(WebcamResolution.VGA.getSize());

    webcam.setImageTransformer((WebcamImageTransformer) this);
                webcam.open();

                monitor = new Monitor(mediaPlayer, webcam);

                JFrame window = new JFrame("Gesture
    Control");
                WebcamPanel panel = new WebcamPanel(webcam);

//              panel.setFPSDisplayed(true);
                panel.setFillArea(true);
```

```
                window.add(panel);
                window.pack();
                window.setVisible(true);

    window.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);
```

6.3.5 Monitor:

This Monitors the real time images shown by the user to the webcam. And it starts detecting the Gesture. And the interval time is about 1000 microseconds. About after certain period of time the second input is shown to the webcam for further processing. The webcam motion detector object is used to detect the real time images which is captured by the camera.

```
WebcamMotionDetector detector = new
    WebcamMotionDetector(Webcam.getDefault());
                detector.setInterval(1000); // check every
    1000 microsec

    detector.addMotionListener((WebcamMotionListener) this);
                detector.start();

        }

        int i = 0;
        public void motionDetected(WebcamMotionEvent wme)
        {
                i++;
                System.out.println("Motion is Detected" + i);
        }
}
```

Chapter 7

TESTING

7.1 What is Software Testing

Software testing is the process of analyzing or operating software for the purpose of inclu ding bugs. Testing can be described as a process used for reveling defects in software,and for establishing that the software has attained a specified degree of quality with respect to selected attribute.The fundamental objective of testing is to and defects ,asearly as possible and get them fix.

Software Testing Process

- Test Planning high level plans which list test objectives, test approach, measurement criteria along with test schedule and resources.
- Test Design create test cases, identify test cases for automation(if applicable),prioritizetest cases and inalize test iterations.
- Test Implementation Create test scripts using automated testing tools
- Test Execution Execute the test cases on the test environment and test reports.
- The datausually will be obtained from your defect tracking system
- Postmortem reviews Discuss lessons learnt and identify strategies which will prevent such problems in future.

7.2 Test Methods

7.2.1 Black Box Testing

It is also called as functional testing, it is the process of giving the input to the system and checking the output of the system.Without bothering about the system that how the system generates the output. It is also called as Behavior testing.

Black box testing Process

- Approach to testing where the program is considered as a Black Box.
- Testing based solely on analysis of requirements user speci

  cation, user documentation etc.
- The test cases are based on the specifications.
- Black box testing techniques apply to all levels of testing.
- Test planning and design can begin early in the software process.
- Tests are done from a users point of view.

7.2.2 White Box Testing

White box testing or structural testing considers facets like programming style, control method, source language, database design. A test for remote monitoring routine can be an example of structural test. This type of testing helps to uncover defects at structural level. The tests go below the top or functional layer to uncover the defects.

- Testing that takes into account internal structure and low of a system or component.
- The testing is based on code structure or the algorithm.
- White box testing assumes that the procedural design and code is known to the tester.
- Obliviously test design can be done only after coding is complete.
- White box tests are inherently finite.
- Postmortem reviews Discuss lessons learnt and identify strategies which will prevent such problems in future.

7.3 Test Cases and Test Data

- Test data are inputs that have been devised to test the system.
- Test cases are inputs and output specification plus a statement of the function under test.
- Test data can be generated automatically or real.

Table 7.1 – Testcase Table

TC ID	OBJECTIVE	PREREQUISITES	STEPS TO BE FOLLOWED	RESULT	REMARK
1	Starting Media Player	User must Start Media Player	User should open the Media player..	Application successfully opened.	Pass
2	Starting Gesture Webcam	User must start Gesture webcam	User should open Gesture Webcam.	Webcam successfully opened.	Pass
3	Showing Gesture (color) to Webcam	User must show Gesture to webcam	User should show Gesture to Webcam.	Detection of Gesture by Webcam.	Pass
4	Showing Red color to Webcam	User must show Red color to Webcam.	User should show Red color to Webcam.	Forward/Backward operation performed	Pass
5	Showing Green color to Webcam	User must show Green color to Webcam	User should show Green color to Webcam.	Previous/Next song operation performed.	Pass
6	Showing Blue color to Webcam	User must show Blue color to Webcam.	User should show Blue color to Webcam	Play/Pause song operation performed.	Pass

TABLE 7.1: Test Data

Chapter 8

DEPLOYMENT

8.1 Control Media Player by using Gesture[Color Detection]

8.1.1 Color Detection

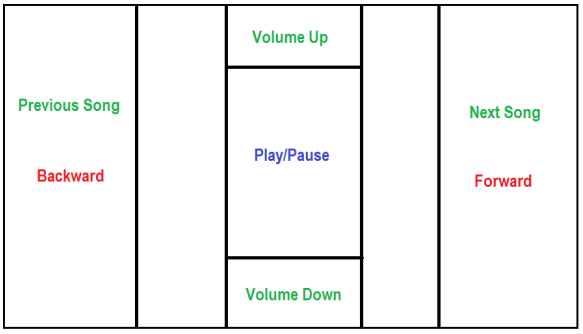

FIGURE 8.1: Shows the operation of media player by using gesture.

8.1.2 Open MediaPlayer

1. **Open Media Player,Selection video**
    - Click on the Open Logo Button to Selection of Video
    - Next Step is to open the Gesture Control Interface for this click on Hand Logo Button then one Camera Window will be pop up
    - Play/Pause/Forward/Backward/Next/Pervious song using Gesture/Color Object .
    - Next step to perform particular media player action (Play/Pause/Forward/Backward/Next/Pervious song) using Gesture/Color Object

FIGURE 8.2: Open MediaPlayer

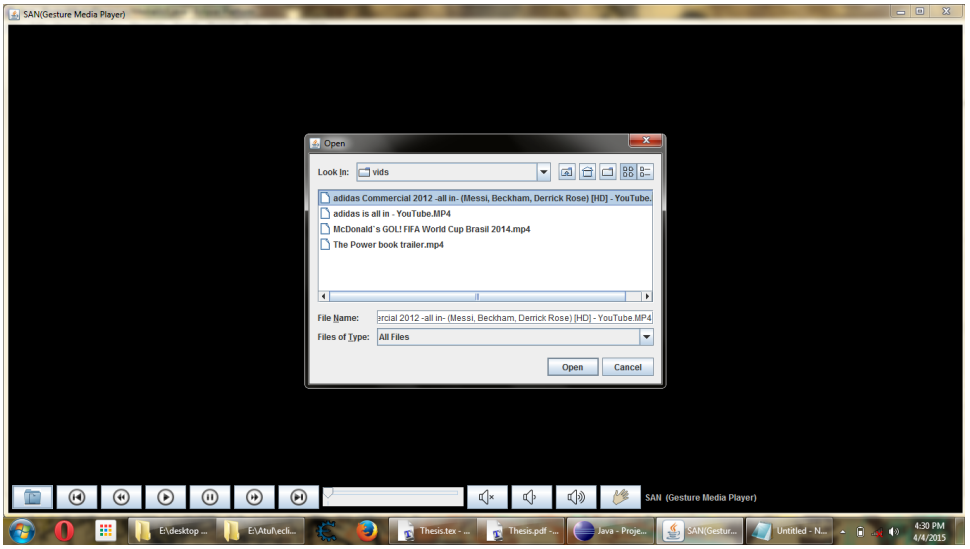

FIGURE 8.3: Selection of video

1. **Control Media Player by using Gesture(Color detection)**

   - The operations of media player controlled by using webcam and identifying the color position and that identified color position, media player operation will be performed.
   - User holds the blue Color in-front of the Webcam.
   - User holds the Red Color in-front of the Webcam.
   - User holds the Green Color in-front of the Webcam.

2. **Play/Pause:**

   - User holds the blue Color in-front of the Webcam.

FIGURE 8.4: Gesture Control Interface for Blue

- Webcam captures the Color.
- Color gesture motion detection.
- Recognizing color gesture by Webcam.
- Matching of color gesture.
- Generating commands.
- Giving the commands to VLC media player.

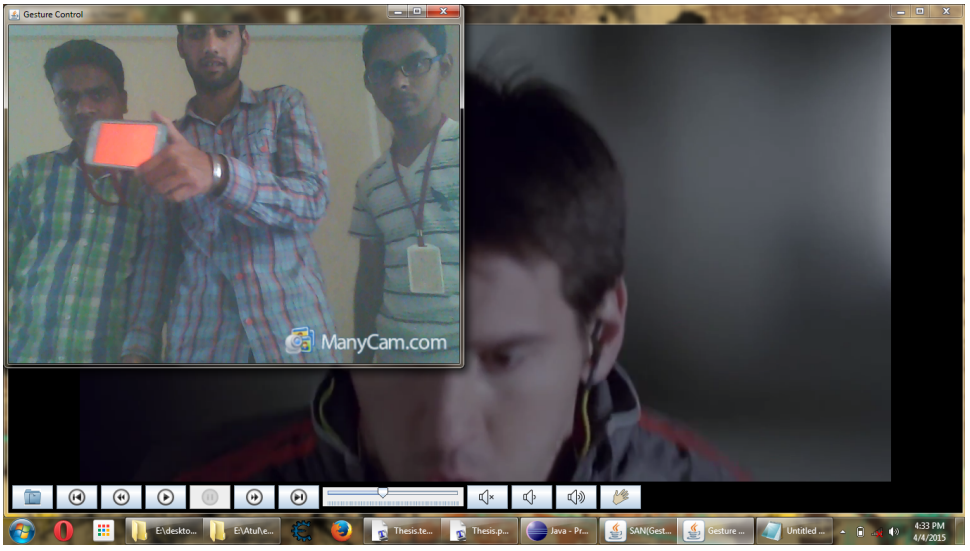

FIGURE 8.5: Gesture Control Interface for Red

3. **Forward/Rewind :**

- User holds the green Color in-front of the Webcam.

- Webcam captures the Color.
- Color gesture motion detection.
- Recognizing color gesture by Webcam.
- Matching of color gesture.
- Generating commands.
- Giving the commands to VLC media player.

FIGURE 8.6: Gesture Control Interface for Green

4. **Volume up/Down:**

- User holds the red Color in-front of the Webcam.
- Webcam captures the Color.
- Color gesture motion detection.
- Recognizing color gesture by Webcam.
- Matching of color gesture.
- Generating commands.
- Giving the commands to VLC media player.

Chapter 9

CONCLUSION

9.1 Conclusion

It is Helpful to control media player without using physical touch to any device. In this project, a novel method for vision based real time computer interaction system using bare hand and color detection of different color on different position is proposed. Media Player actions are identified based on the specific color detection and position of the detected color in the image. A vision-based hand gesture recognition method has been presented. Considering the relative infancy of research related to vision-based gesture recognition, remarkable progress has been made. To continue this momentum, it is clear that further research in the areas of feature extraction, classification methods and gesture representation are required, to realize the ultimate goal of humans interfacing with machines on their own natural terms The amount of literature on the problem of gesture recognition and the promising recognition rates reported, one would be led to believe that the problem is nearly solved. Sadly this is not so. A main problem hampering most approaches is that they rely on several underlying assumptions that may be suitable in a controlled lab setting but do not generalize to arbitrary settings. Several common assumptions include: assuming high contrast stationary grounds and ambient lighting conditions. Also, recognition results presented in the literature are based on each authorâĂŹs own collection of data, making comparisons of hand moment.

Bibliography

[1] Siddharth S Rautaray and Anupam Agrawal. Interaction with virtual game through hand gesture recognition. In *Multimedia, Signal Processing and Communication Technologies (IMPACT), 2011 International Conference on*, pages 244–247. IEEE, 2011.

[2] Siddharth S Rautaray and Anupam Agrawal. Real time hand gesture recognition system for dynamic applications. *Int J UbiComp*, 3(1):21–31, 2012.

[3] Siddharth S Rautaray and Anupam Agrawal. Design of gesture recognition system for dynamic user interface. In *Technology Enhanced Education (ICTEE), 2012 IEEE International Conference on*, pages 1–6. IEEE, 2012.