ABSTRACT

Virtual View Mirror, a face detection and filter based system which detects various components of face within the window. On detecting a component it performs some functions on it and finally yield a filtered frame.

Virtual View Mirror enables the user to view how wearables such as spectacles look on them even without trying them physically. It reduces the physical wear and tear of wearables and thus act as a virtual platform for the viewer/user to make the best choice in a high range of products.

It requires a basic camera located in front of the system, windows (with the most recent release) as operating system and a powerful processor for making recognition work and transform. Virtual View Mirror is coded in python with libraries OpenCV, Numpy, Math and other Python libraries. OpenCV is a free library of programming features mainly aimed at real-time computer vision developed by Intel (in 1995).

In Future, it can be used to viewed other utilities and wearables besides spectacles.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

Abbreviations	Descriptions
HCI	Human Computer Interaction
RTI	Real Time Initiator
OpenCV	Open Source Computer Vision
IDE	Integrated Development Environment
PC	Personal Computer
GB	Giga Byte
HD	Hard Disk
ROI	Region of Interest

Chapter 1

INTRODUCTION

Virtual View Mirror is a Face Recognizing Application written in Python with OpenCV for Windows Operating System. It recognizes face and eyes and place a product on the face Created in Python.

1.1 Problem Statement

Human Computer Interaction (HCI) is getting more and more important with the improvement of the technology. In the history, different perspectives inspired people to develop some innovative systems. Static keys and buttons, track path devices, touch and multi-touch screens have developed respectively. The next innovation should let people use computers, game consoles or mobile devices without touching in order to control those devices easily and effortlessly. Thus face recognition are the next improvement of interaction with computers and mobile devices.
Viewing products through this application will become ease to buy those products, this recognition system will help to operate the application on an operating system.

- To design a software/interface/utility which detects face patterns instead of physical touch.
- The camera is positioned such that it recognizes the movement of face and performs some operations.

1.2 Purpose

This Project enables a user to overcome the physical barriers between the user and the system and thus reduces the mechanical wear and tear of the products. Also it increases durability of that system, making process of maintenance easier for a user.

1.3 Scope

- The goal is to manage computers and view products virtually rather than pointing and clicking a mouse or touching a display directly.
- User friendly application which minimizes the requirement of physical connection.

- Powerful media application that minimizes the timing consumption.
- This System is a Real Time System, which works on the real-time data and processed simultaneously.

1.4 Overview

"**Virtual View Mirror**" is a Face Recognizer which detects position of face when placed near the active webcam or a simple camera.

It places the product on the face after recognition and detection of face and eyes. This system is written in Python with OpenCV Library, using Pycharm as an IDE.

Chapter 2

SYSTEM OVERVIEW

The main goal of the project is to track and recognize basic hand gestures and process them to view products on face. The functionality of Virtual View Mirror is to provide an interaction between the product and its user without need to touch the device. Thus Laptop/PC users can control the devices when they are doing their daily activities. The user has to go to market and view the product and wear it to check how that products look on them. The Virtual View Mirror offers a solution to people to control their Laptop/PC even when they are doing that kind of activities. The system will also work properly in some specific conditions that make people use their Laptop/PC difficult.

2.1 Goals and Objectives

- **Reliability**

  Our system is highly reliable as its requirements are simple that is webcam which is present in every laptop and computer. It will contain a dataset of various products with different types and colors in order to increase the variety.

- **Low-Cost**

  Implementation of the stereovision technology by using low cost cameras provides an inexpensive solution with respect to the current technologies which are already available for all version of devices for free.

- **Maintainability**

  Any error occurred while the system is in use will be tolerated and system recovers itself and continues properly.

- **Portability**

  This software is portable as it is just to be put on the devices which have python and OpenCV installed on them.

These are free and platform independent which makes them portable.

- **Performance**

  Since the system will work on real time, performance is one of the most important topics of the system. Performance of the system will be high enough that user can use the system without noticeable delays or performance problems.

- **Usability**

  Usability for the system is quite easy as the person just needs to perform various hand gestures and move hand in various directions which makes the particular software very user-friendly.

- **Time**

  Software will work on real time. Response of the system after the user makes an action will be fast enough that does not cause a problem. Although delay is about 0.4 seconds to remove any confusion.

- **Security**

  As this software is based on python thus the internal security manager provides it a level of security making it highly efficient also when it will work on UNIX it will become more secure from external attacks.

- **User – Friendly**

  This system is user friendly, thus easy to learn and easy to operate.

- **Availability**

  This project is available to all as this project is based on OpenCV and python which are open-source that is anybody can use and operate it easily without facing any issues.

Chapter 3

DESIGN CONSIDERATIONS

Virtual View Mirror comes with many assumptions, dependencies and especially constraints.

3.1 Designs Assumptions and Dependencies

This Project comes with many assumptions and dependencies,

- The system requires a camera or a webcam located in front, that might not be present in few systems.
- This Python based system is limited to windows operating system.
- It is needed that the performance and time limits of the software will be applied considering that the software would be working on a laptop/PC.
- It is assumed that this project will operate well on new generation systems without any problems if the systems have powerful processors and a camera located in front.

3.2 Design Constraints

Virtual View Mirror has many hardware and software constraints.

3.2.1 Hardware Constraints

The system needs a powerful processor in order to make a recognition work and transform. A powerful processor like Intel Atom will be sufficient.

The hardware system should have atleast 2 GB RAM and minimum of 5 GB HD space. System memory consumptions never exceed 20% of the total memory, approximately 200 MB.

A webcam or a camera with atleast 30 frames per second and minimum 640x480 pixels located in front of laptop/PC is the most essential part of this system.

3.2.2 Software Constraints

When the hardware part of the system is completed, software should work on Windows x32 or higher (with most recent release).

Webcam Drivers which are device specific should be installed in the system.
This System is coded in Python along with libraries OpenCV, Math, Numpy and Pyglet. Therefore, all libraries should be imported and installed in the system.

3.2.3 Performance Constraints

When the system is activated, system will be in a state that interaction between end-users will be the most frequent event; thus, the system should process 6-7 frames/sec for a proper and error-free interaction.

3.3 Design Goals and Guidelines

The highest priority of the software is to design the recognition module. After the module is designed, Face Recognition module will be inherited and Product module will be manipulated. Thus, design of face recognition module is the first main step of the software. Module descriptions are explained in detailed at the fifth chapter. One of the most important goals of the software is to keep the memory use in reasonable degree.
Since the aim of the software is the software can be implemented into laptops and PCs, memory use is one of the most critical points that the software will faced. The upper limit of the memory use will be 20% of the memory.
The other important goal is the speed of the software. The process of the software, which is recognition of the gesture, processing it and providing the action, will be done in at most 0.2 second in order that the software will work in reasonable time.

Chapter 4

DATA DESIGN

4.1 Data Description

Virtual View Mirror system consists of two different components that are working together. Real-Time initiator and Product Interface are the mentioned components.

At first the Real-Time initiator part tracks face and different parts of face and sends it to the interface. Interface translates the gesture to the appropriate command and if it is valid, the corresponding transition or the corresponding action will be done. The ER diagram of the perfect cooperation of these two parts, Real-Time initiator and Product Interface can be seen on Figure 1.

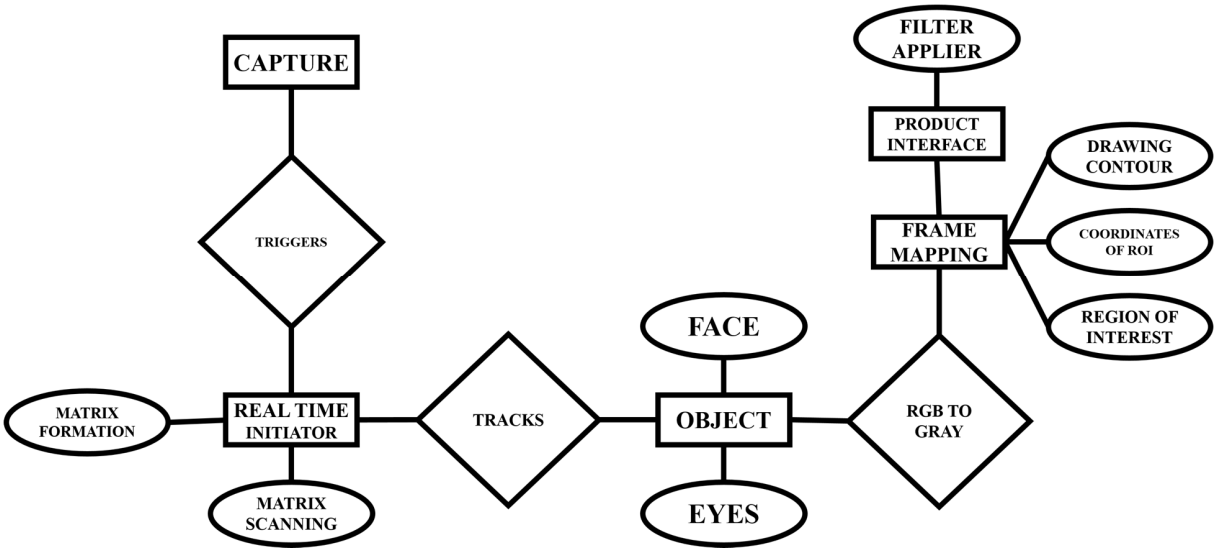

Figure 4.1 Entity Relationship Diagram of System Entities

Virtual View Mirror System consists of a main data structure named face recognition, this data structure is created, modified or used by the two modules of the system, Real-Time Initiator and Product Interface.

Real-Time Initiator part of Virtual View Mirror System tracks face and different parts of face. RTI sends that detected and recognized object to the Product Interface and it is reflected on the screen by this module and will be changed according to the module.

Whenever the face is detected the position of the different part of face is calculated and according to which the matching function is called. For every function there is different position and product is defined.

In the above ER Diagram (Figure 4.1), entities RTI and product interface performs different roles in completion of the whole process. The output generated by RTI is the face detection and recognition created by the user and manipulated by the system which is then invokes functions of the product interface. Functions such as calculation of position, resizing of product according to detected face and its part and final output.

For RTI, video is captured and simultaneously processed into noise free threshold image which is then used to detect face and eyes which then invokes Haar Cascade functions. For clear and understandable video capturing the user should place its face in the frame so that they are easily detected.

Chapter 5

SYSTEM ARCHITECTURE

Description of system architecture is given below.

5.1 Architectural Design

Our system is consists of two main modules, namely, RTI and Product Interface. RTI is responsible for recognizing of face and recognizing eyes and other parts. Once, suitable contour is drawn, RTI interprets related action and triggers product objects to do the action of the suggested frame. In basic, Product Interface is the subsystem which enables the user of the device to give the freedom of control without touching the device and RTI is the subsystem that is responsible for capturing, analyzing and detecting the desired action for the incoming face movement and position.

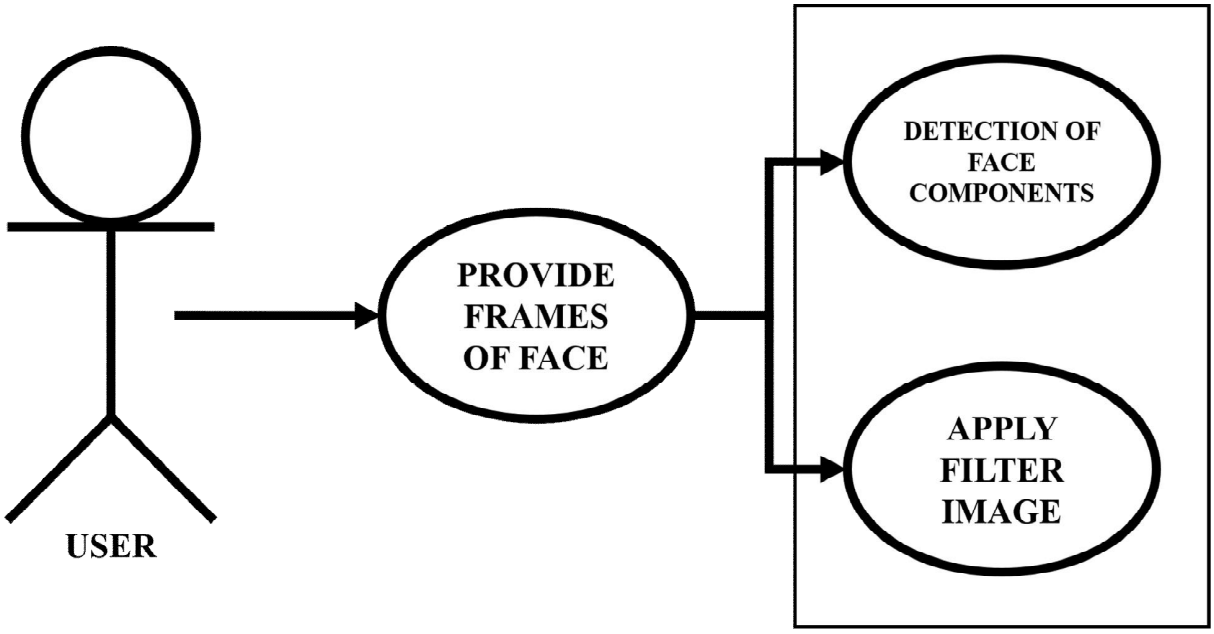

Figure 5.1 Use case Diagram of the system

In Figure 5.1, Use Case Diagram of Virtual View Mirror, the actor – User provides frames of face to detect facial components and then apply filter image on it.

5.2 Description of Components

In the upper section it's briefly said that Virtual View Mirror has two subsystems, namely Real-Time Initiator module and Product Interface module. The responsibility of Real-Time Initiator module is to capture consecutive frames, try to recognize the face of the user and gets the movements and position of face from camera. If a face is found, Product Interface is notified. Product Interface looks for the face and takes in action the desired action.

5.2.1 Processing Narrative

Two responsibilities of the Frame Detection component is to detect frame and create contour objects and send it to filter module for further processing with position of face with respect to centroid of the frame.

5.2.2 Frame Detection Interface Description

Frames that comes from the camera and trigger from frame detector are the input to this component and output is the frame object that is created after the recognition of the face movement.

5.2.3 Processing Details

The frames Recognized by the Frame Detection component transform the input and create noise free frames which then transferred to frame processor as output and proper functions are invoked.

5.2.4 Real Time Initiator

RTI transforms the frame captured into threshold and calculates the position of the face components (eyes, nose etc.) with respect to the face.

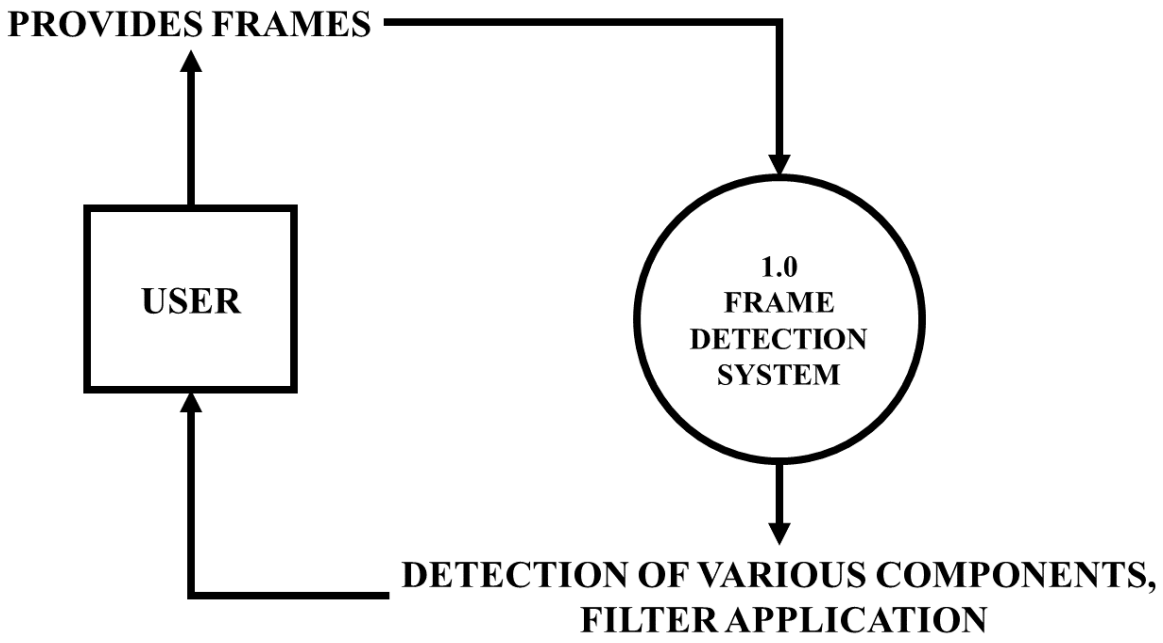

Figure 5.2 Data Flow diagram for Gesture Recognition (Level 0)

In figure 5.2, when the user provides face frames, it is sent to the system for function calling. The gestures provided by the user are matched with the defined gestures in the system it then is sent to filter module.

After the recognition of the frame it transform the image in several steps, as explained in detailed design.

5.2.4.1 Detailed Design

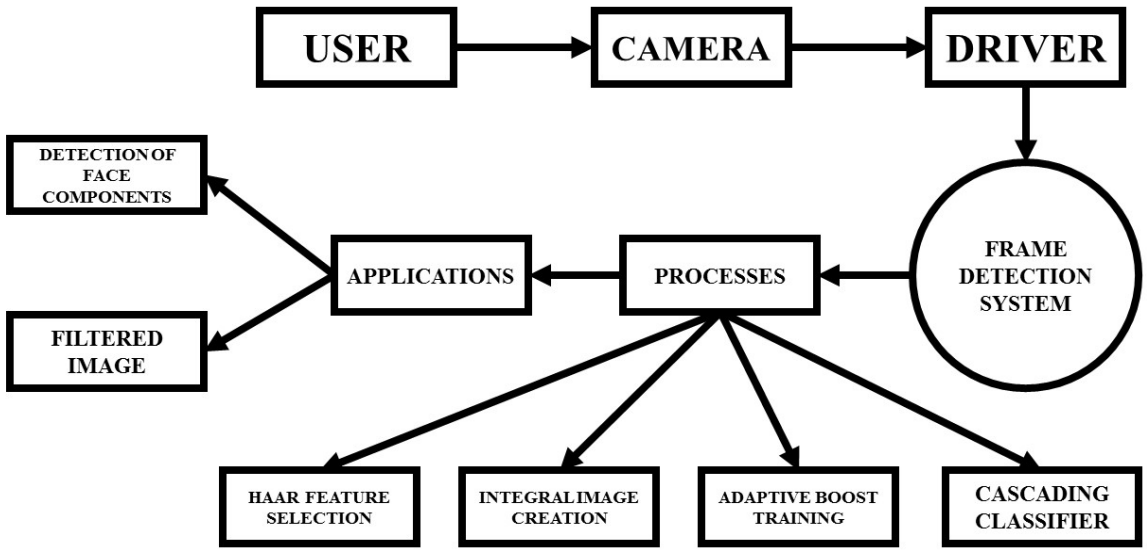

Figure 5.3 Detailed Data Flow Diagram for RTI (Level 1)

In figure 5.3, The frame captured are convert into grayscale then it is processed after going through selection of haar feature to cascading classifier and finally components of face are determined and then filters are applied.

5.2.5 Filter Applier

Filter applier takes processed frame as input and then place various on different components of face accordingly after making a rough overall contour.

Chapter 6

REAL TIME INITIATOR (MODULE 1)

Face detection is most of the times a necessary preprocessing step for face alignment, as most of the regression models assume a bounding b ox as an input, although there are now methods that perform detection and alignment jointly.

6.1 Viola Jones Face Detector [8]

The Viola-Jones detection framework was the first object detection framework to provide competitive detection rates in real-time. Although it can be applied for general object detection, the primary motivation behind the method was face detection. The detection algorithm uses a sliding window approach - the algorithm essentially asks "Does this region contain a face?" for many possible regions within the image. As there are many candidate regions, this classification task has to be handled extremely fast.

Figure 6.1 The Viola-Jones face detector is a sliding-window algorithm that classifies rectangles of different sizes as either "face" or "non-face"

In Figure 6.1, matrix of the captured frame is formed and scanned to determine the Region of Interest.

6.2 Feature Extraction

Due to the need for high-speed classification of each region, the features has to be simple and fast to extract. Viola-Jones uses **Haar-like features** that involve sums of pixel intensities within rectangular regions. They exploit the fact that certain regions of a face have higher intensity than others, e.g. the eye region is generally darker than upper part of the cheeks.

Different Haar features are depicted as rectangles of black and white areas. To calculate the value of a particular feature, sum of all pixels within white regions is subtracted from sum of pixels within black regions. These features can be computed extremely fast using a pre-processing technique called **integral image**. By blurring, the image is created smooth transition from one color to another and reduce the edge content. Thresholding is used for image segmentation and to create binary images from grayscale images.

Figure 6.2 Haar-like features on Eyes

In figure 6.2, Haar like features is used to detect eyes on a face and plotting the coordinates of the detected region of interest.

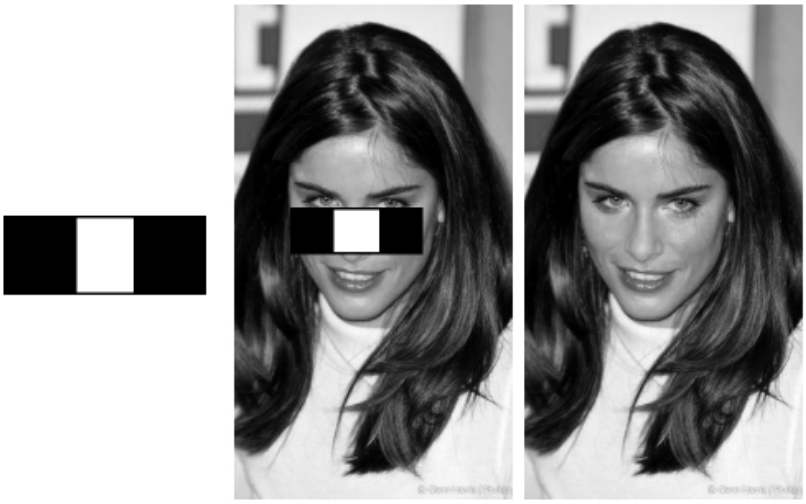

Figure 6.3 Haar-like features on Nose

In figure 6.3, Haar like features is used to detect nose on a face and plotting the coordinates of the detected region of interest.

6.3 Drawing Contour for full face

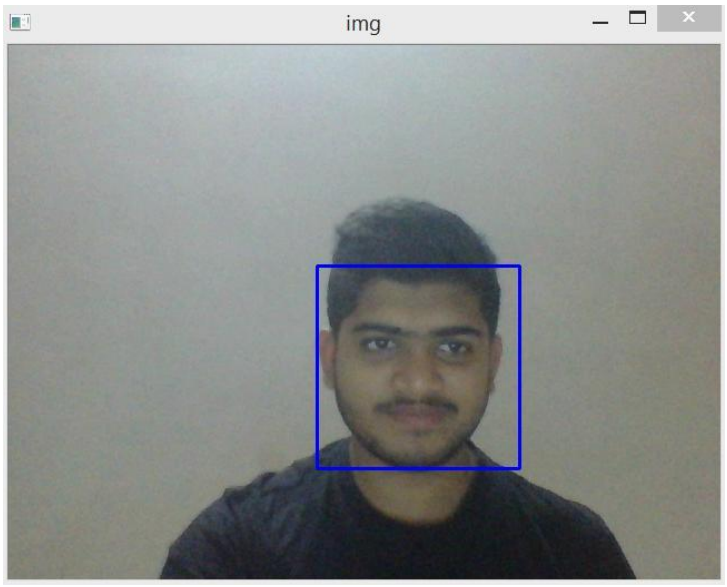

Figure 6.4 Drawing Contour for Face

In figure 6.4, Contour of face in the captured region of interest is drawn.

6.4 Drawing Contour for Eyes

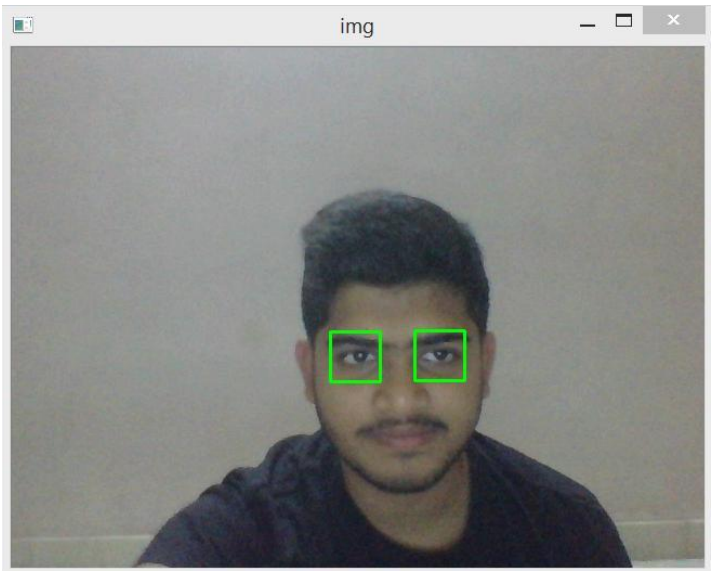

Figure 6.5 Drawing Contour for Eyes

In figure 6.5, Contour of eyes in the captured region of interest is drawn.

6.5 Drawing Contour for Eyes and Mouth

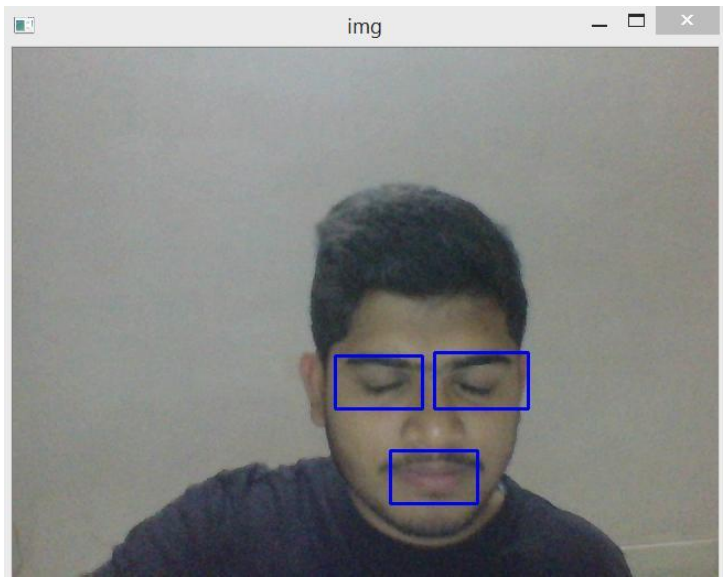

Figure 6.6 Drawing Contour for Eyes and Mouth

In figure 6.6, Contour of eyes and mouth in the captured region of interest is drawn.

6.6 Contour for Face and Eyes

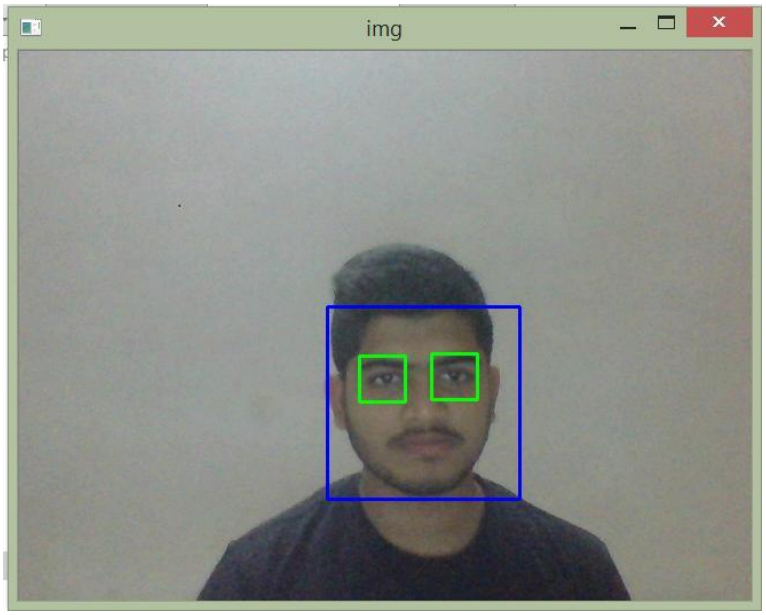

Figure 6.7 Drawing Contour for Face and Eyes

In figure 6.7, Contour of face and eyes in the captured region of interest is drawn.

Chapter 7

PRODUCT INTERFACE

This module works on the processed image from previous module. This module places the product image on the face of the user after recognition, detection and drawing contours on the specified position on Face.

7.1 Product 1 (Eye Glasses)

7.1.1 Eye Glass Frame Filter 1 on Eyes

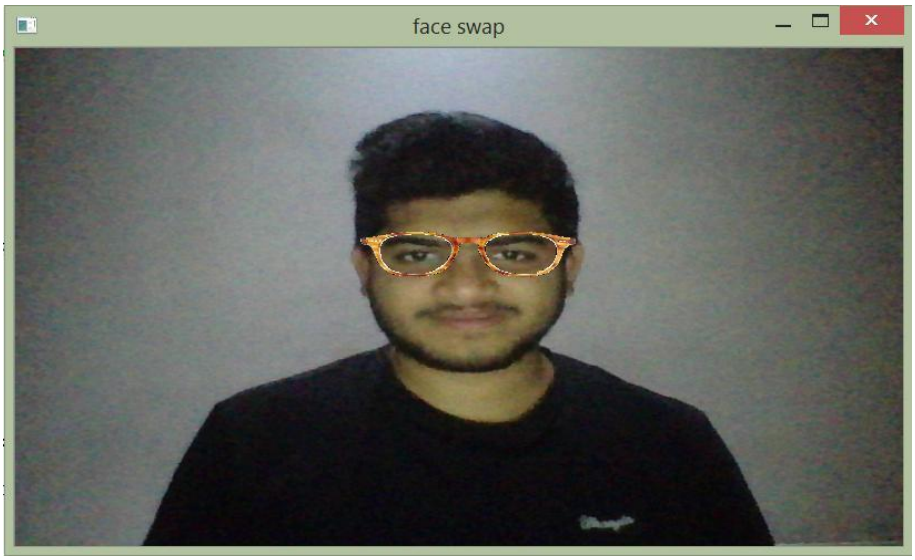

Figure 7.1 Eye Glass Frame Filter 1 on Eyes

In figure 7.1, Filter image 1 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

7.1.2 Eye Glass Frame Filter 2 on Eyes

Figure 7.2 Eye Glass Frame Filter 2 on Eyes

In figure 7.2, Filter image 2 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

7.1.3 Eye Glass Frame Filter 3 on Eyes

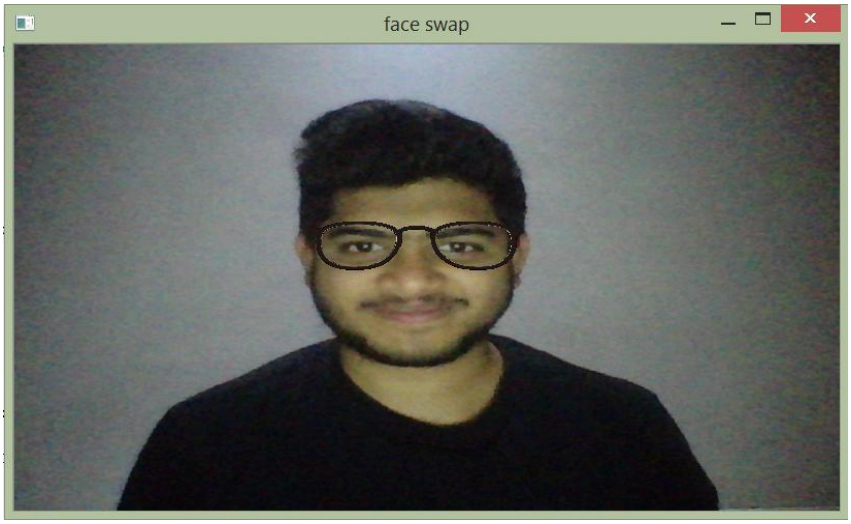

Figure 7.3 Eye Glass Frame Filter 3 on Eyes

In figure 7.3, Filter image 3 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

7.1.4 Eye Glass Frame Filter 4 on Eyes

Figure 7.4 Eye Glass Frame Filter 4 on Eyes

In figure 7.4, Filter image 4 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

7.1.5 Eye Glass Frame Filter 5 on Eyes

Figure 7.5 Eye Glass Frame Filter 5 on Eyes

In figure 7.5, Filter image 5 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

7.2 Product 2 (Mustaches)

7.2.1 Mustache Filter 1 below Nose

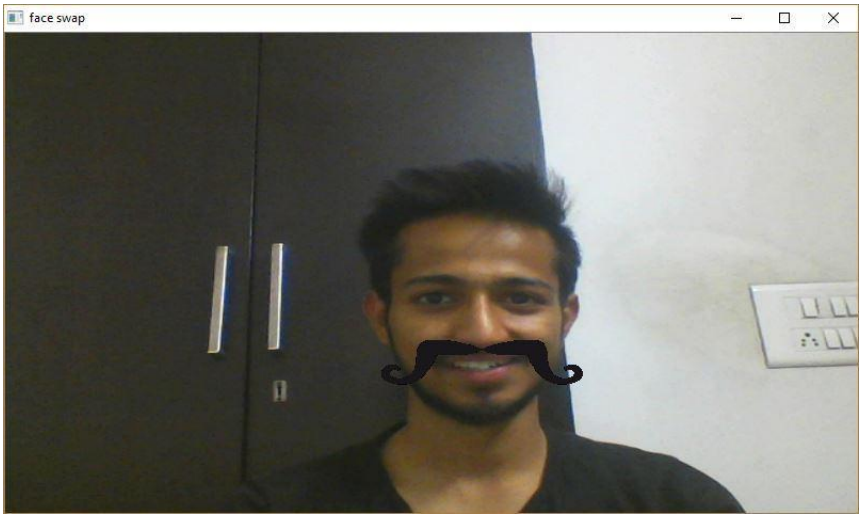

Figure 7.6 Mustache Filter 1 below Nose

In figure 7.6, Filter image 6 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

7.2.1 Mustache Filter 2 below Nose

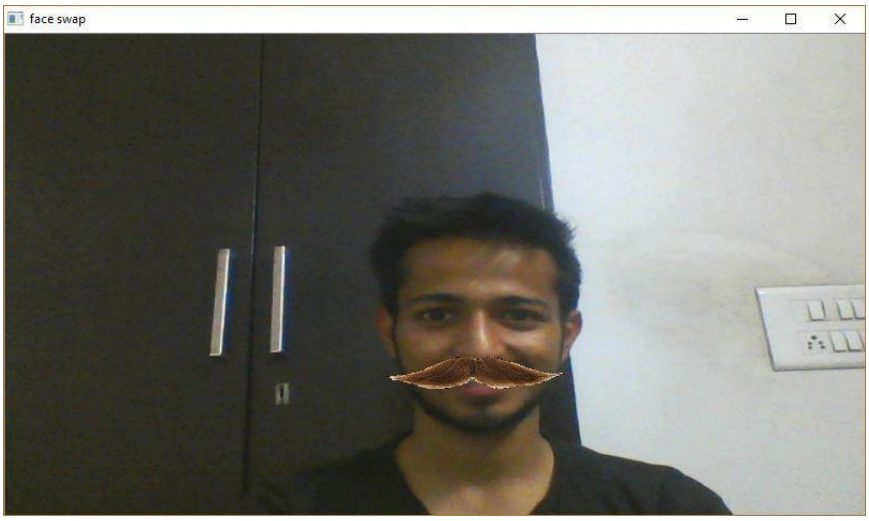

Figure 7.7 Mustache Filter 2 below Nose

In figure 7.7, Filter image 7 on region of interest is superimposed. Thus, creating a virtual image on the real time video/image frames.

Chapter 8

LIBRARIES AND TOOLS

Virtual View Mirror uses different tools, and library with python as a programming language.

8.1 Python [4]

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

8.1.1 History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Smalltalk, UNIX shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

8.1.2 Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of UNIX.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.
- Apart from the above-mentioned features, Python has a big list of good features, few are listed below:
- IT supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

8.2 OpenCV [1]

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel's research center in Nizhny Novgorod (Russia), it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license.

8.2.1 History

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as:

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performance-optimized code available for free with a license that did not require to be open or free themselves.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. In mid-2008, OpenCV obtained corporate support from Willow Garage, and is now again under active development. A version 1.1 "pre-release" was released in October 2008.

The second major release of the OpenCV was on October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations.

In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

8.2.2 Methods Used

Table 8.1 OpenCV methods

Functions	Description
cv2.videocapture()	This function is for video capturing from video files, image sequences or cameras where cv2 is OpenCV object.
cv2.flip()	Flips a 2D array around vertical, horizontal, or both axes
cv2.cvtColor()	For grey scale conversion, we use the function cv2.cvtColor(input image, flag) where flag determines the type of conversion
Cv2.findContours()	Contours can be explained simply as a curve joining all the continuous points having same color or intensity. There are three arguments in cv2.findContours() function, first one is source image, second is contour retrieval mode, third is contour approximation method. And it outputs the contours and hierarchy.
Cv2.ContourArea()	Contour area is given by the function cv2.contourArea()
Cv2.rectangle()	Draws a simple, thick, or filled up-right rectangle
Cv2.imshow()	Use the function cv2.imread() to read an image
Cv2.drawContours()	To draw the contours, cv2.drawContours function is used. It can also be used to draw any shape provided you have its boundary points. Its first argument is source image, second argument is the contours which should be passed as a Python list, third argument is index of contours (useful when drawing individual contour.

Cv2.moments()	Image moments help you to calculate some features like center of mass of the object, area of the object etc. The function cv2.moments() gives a dictionary of all moment values calculated
Cv2.circle()	Draws a circle it has parameters image where circle is drawn, center of circle, radius of circle, color of circle
Cv2.line()	Draws a line segment connecting two points. Its parameters are image, first point, second point, thickness, color
Cv2.putText()	Draws a text string. Parameters are image, text string, font and font-face.
Cv2.imread()	Use the function cv2.imread () to read an image. The image should be in the working directory or a full path of image should be given. Second argument is a flag which specifies the way image should be read. cv2.IMREAD_COLOR: Loads a color image. Any transparency of image will be neglected. It is the default flag. cv2.IMREAD_GRAYSCALE: Loads image in grayscale mode. cv2.IMREAD_UNCHANGED: Loads image as such including alpha channel.
Cv2.destroywindow()	Destroys a window. Parameter is name of window to be destroyed.
Cv2.waitKey()	Waits for a pressed key. Parameters are delay in milliseconds.
Cv2.destroyAllWindows()	Destroys a window.

Cv2.Add()	To add two Images on each Other
Cv2.Resize()	To Resize the captured Image
Cv2.Randn()	To Generate a random Number
Cv2.EqualizeHist()	To Calculate Histogram of set of arrays.
Cv2.COLOR_BGR2GRAY()	To convert RGB to Gray

8.3 Numpy [2]

NumPy is the fundamental package for scientific computing with Python. It contains among other things: a powerful N-dimensional array object sophisticated (broadcasting) functions tools for integrating C/C++ and Fortran code useful linear algebra, Fourier transform, and random number capabilities. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

8.3.1 Limitations

NumPy's arrays must be views on contiguous memory buffers. A replacement package called Blaze attempts to overcome this limitation.

Algorithms that are not expressible as a vectorized operation will typically run slowly because they must be implemented in "pure Python", while vectorization may increase memory complexity of some operations from constant to linear, because temporary arrays must be created that are as large as the inputs. Runtime compilation of numerical code has been implemented by several groups to avoid these problems; open source solutions that interoperate with NumPy include Scipy.weave, numexpr and Numba.

8.3.2 History

The Python programming language was not initially designed for numerical computing, but attracted the attention of the scientific/engineering community early on, so that a special interest group called matrix-sig was founded in 1995 with the aim of defining an array computing package.

Among its members was Python designer/maintainer Guido van Rossum, who implemented extensions to Python's syntax (in particular the indexing syntax) to make array computing easier. An implementation of a matrix package was completed by Jim Fulton, then generalized by Jim Hugunin to become Numeric, also variously called Numerical Python extensions or NumPy. Hugunin, a graduate student at MIT, joined CNRI to work on JPython in 1997 leading Paul Dubois of LLNL to take over as maintainer. Other early contributors include David Ascher, Konrad Hinsen and Travis Oliphant.

A new package called Numarray was written as a more flexible replacement for Numeric

8.3.3 Methods Used

Table 8.2 NumPy Methods

Methods	Description
Numpy.hstack(tup)	Stack arrays in sequence horizontally (column wise).Take a sequence of arrays and stack them horizontally to make a single array. Rebuild arrays divided by hsplit. Tup parameter is sequence of ndarrays that install arrays must have shape except along second axis.
Numpy.zeroes()	Return a new array of given shape and type, filled with zeros Parameters: shape : int or sequence of ints Shape of the new array, e.g., (2, 3) or 2. dtype : data-type, optional The desired data-type for the array, e.g., numpy.int8. Default is numpy.float64. order : {‘C’, ‘F’}, optional

	Whether to store multidimensional data in C- or Fortran-contiguous (row- or column-wise) order in memory. Return: ndarray Array of zeros with the given shape, dtype, and orde
Numpy.Array()	To create an array.

8.4 Math [3]

This module is always available. It provides access to the mathematical functions defined by the C standard.

These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don't is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place.

The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

8.4.1 Methods Used

Table 8.4 Math Library Methods

Methods	Description
Math.sqrt()	Returns the square root of a function

Math.acos()	Return the arc cosine of *x*, in radians
Math.asin()	Return the arc Sine of *x*, in radians

8.5 Pycharm [5]

8.5.1 Why use PyCharm?

Unlike a regular text editor, PyCharm knows Python and is built from the ground up to deal with code, not text. Other editors tend to deal with only simple text matching (e.g. for syntax highlighting) and leave the hard work to separate plugins. With PyCharm, though, the various features share the same language-aware core, so the navigation, completion, validation, code generation, etc. are all aware of the language rules and the full code structure. Another difference between PyCharm and other editors is that PyCharm is developed by a team of people who are still actively working on it full-time. Plugin ecosystems, on the other hand, are often full of no-longer-maintained side projects that don't necessarily mesh well together.

The most common PyCharm features are code navigation (go to file/class/symbol by name, jump to implementation, find usages, etc.), autocomplete, problem detection/auto resolution, refactoring, integrated debugging, integrated unit tests, and clickable stack traces. There quite a bit of depth to these features as well as other secondary features that you discover when using PyCharm for longer.

8.5.2 Why might I not want to use PyCharm?

Every editor is its own beautiful snowflake, so there are plenty of little reasons why you might prefer one editor over another, but there are also some notable deficiencies in PyCharm that are worth mentioning up-front:

The JavaScript support is less powerful than the Python support. The IDE doesn't understand require statements, so it tends to resolve usages by just using naive name matching, and for common names it tends to overestimate the possible references. Also, JavaScript indexing is slow (it takes a few minutes to index webapp when starting PyCharm for the first time), and in some cases working with JS files is slow.

PyCharm isn't an "officially supported" dev setup, so typical steps may not work smoothly with PyCharm, and changes to the dev environment may cause even worse problems in the future. For example, Alan had to refactor the test infrastructure to get unit tests to run inside PyCharm, and they still don't completely work. If you want to take the simple, stable approach, stick with the command line for everything. Using PyCharm may require a fair amount of tinkering and will add complexity to your dev setup, but will hopefully make you more productive as well.

PyCharm costs money and is not completely open source (although a significant amount of it is open source).Even if you decide not to use PyCharm, it may provide some inspiration about what to expect from an editor, and you may be able to find (or develop) some equivalent plugins that work for your editor.

8.5.3 What are some unfair criticisms of PyCharm?

Well, these criticisms are at most half-fair.

"IDEs force you to use the mouse." Almost everything in PyCharm is accessible from just the keyboard. Most things you do are "actions", and you can perform any action by name through the keyboard or give a keyboard shortcut to any action (if it doesn't already have one). The main exception is that modifying preferences is usually best done with the mouse.

"IDEs are for beginners who are afraid of the command line." While it's true that PyCharm is probably more beginner-friendly because it gives immediate feedback about things like syntax mistakes, the features being actively developed make it clear that it's mostly targeted at professional software developers. Maybe some people use PyCharm to avoid the command line, but the better approach is to be aware of how to do anything on the command line and to use PyCharm when it makes you more productive. PyCharm even has a built-in terminal so you can use the command line inside its windowing system if you want.

"IDEs are slow and bloated." At least with Python, PyCharm is very rarely slow, and I've never run into memory problems in my 8GB MacBook Pro. Like I said above, though, it can be slow with JavaScript. Also, some operations can be slow the first time while PyCharm builds indexes, then are fast.

"PyCharm forces me to give up the great text editing provided by vim/emacs." PyCharm has a mode that gives emacs key bindings. There's also a plugin called IdeaVim that makes PyCharm

text editing feel like vim. However, it's unclear how good these modes are compared to actual vim/emacs.

"PyCharm isn't very extensible or configurable." PyCharm (and the IntelliJ platform in general) is really a combination of plugins built on a core IDE system, and there is an ecosystem of third-party plugins as well, and you can write your own plugins. It also has many, many settings, including custom keyboard shortcuts, live templates, color schemes, and code style options. Still, it is true that you occasionally run into cases where the implementation details of a feature don't quite do what you want and there's no way to fix it (e.g. auto-generated imports don't match our code style).

Chapter 9

CONCLUSIONS AND FUTURE SCOPE

The Virtual View Mirror Project is a small effort to reduce the physical communication between the User and the Product.

As the growing use of computers and other electronic devices would mean the growing demand on rapid and quick technical support, this system is carefully designed to fit with the rapid technical support.

Virtual View Mirror controls the Products (Glass Frame) through Face Recognitions. But in future it can be taken forward to make control over other Products and improve to reduce Physical Communication between the User and the Products.

Virtual View Mirror is designed to accommodate future upgrading and development without building a new system to fit with the growing needs and demands of the system.

REFERENCES

[1] OpenCV usage documentation, http://docs.opencv.org

[2] NumPy usage Documentation, https://www.numpy.org/

[3] Math-Python usage Documentation, https://docs.python.org/2/library/math.html

[4] Python usage Documentation, https://docs.python.org/2.7/

[5] Pycharm usage Documentation, https://www.jetbrains.com/pycharm/documentation/

[6] S.V. Viraktamath, Mukund Katti, Aditya Khatawkar & Pavan Kulkarni (2013), 'Face Detection and Tracking using OpenCV', CNCE, Vol-1, No-3, PP 45-50.

[7] Shervin EMAMI, Valentin Petruț SUCIU (2012), 'Facial Recognition using OpenCV', JMEDS, Vol-4, No-1, PP38-43.

[8] Andrej Maris, 'Implementation and Study of Cascaded-Regression Methods for Facial Feature Points Detection', ICL, PP 18-22.

[9] Nidhi, 'Image Processing and Object Detection', IJAOR, Vol-1, No-9, PP 369-399.

[10] Prathamesh Timse, Pranav Aggarwal, Prakhar Sinha and Neel Vora, 'Face Recognition Based Door Lock System Using Opencv and C# with Remote Access and Security Features', IJERA, Vol-4, No-4, PP 52-57.

[11] Rajashree Tripathy and R N Daschoudhury, 'Real-time Face Detection and Tracking Using Haar Classifier on SoC', IJECSE, Vol-3, No-2, PP175-184.

[12] Walaa Mohamed, Mohamed Heshmat, Moheb Girgis and Seham Elaw, 'A new Method for Face Recognition Using Variance Estimation and Feature Extraction', IJETTCS, Vol-2, No-2, PP 134-141.

[13] Manav Bansal, Sohan Garg, 'Facial Detection & Recognition Using Open CV library', IJCST, Vol-7, No-1, PP 24-26.

[14] Faizan Ahmad, Aaima Najam and Zeeshan Ahmed (2012), 'Image-based Face Detection and Recognition', IJCSI, Vol-9, No-6, PP169-172.