Vasu Gupta :- Ph No. 8140046063
Shubham Bharadwaj :- Ph No. 8840794209

A Project Report

on

Hand Gesture Recognition System

Submitted in partial fulfillment of the

requirement for the award of the degree of

Bachelor of Technology in Computer Science and Engineering

(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision of
DR. KAVITA SAINI
Professor
Department of Computer Science and Engineering

Submitted By

18SCSE1010123 - VASU GUPTA

18SCSE1010592 - SHUBHAM BHARADWAJ

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA

DECEMBER - 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

We hereby certify that the work which is being presented in the project, entitled **"Hand Gesture Recognition System " in partial fulfillment of the requirements for the award of the BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of **JULY-2021 to DECEMBER-2021**, under the supervision of **DR. KAVITA SAINI, Professor, Department of Computer Science and Engineering** of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the project has not been submitted by us for the award of any other degree of this or any other places.

18SCSE1010123- VASU GUPTA
18SCSE1010592 - SHUBHAM BHARADWAJ

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Supervisor
(DR. KAVITA SAINI, Professor)

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of **18SCSE1010123- VASU GUPTA 18SCSE1010592 - SHUBHAM BHARADWAJ has been held on _____________ and his/her** work is recommended for the award of **BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING**.

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date:

Place:

TABLE OF CONTENTS

ABSTRACT

Gesture recognition, along with facial recognition, eye tracking and lip movement recognition are components of what developers refer to as a perceptual user interface (PUI). The goal of PUI is to enhance the efficiency and ease of use for the underlying logical design of a stored program, a design discipline known as usability.

In personal computing, gestures are most often used for input commands. Recognizing gestures as input allows computers to be more accessible for the physically-impaired and makes interaction more natural in a gaming or 3-D virtual reality environment. Hand and body gestures can be amplified by a controller that contains gyroscopes to sense tilting, rotation and acceleration of movement or the computing device can be outfitted with a camera so that software in the device can recognize and interpret specific gestures. A wave of the hand, for instance, might terminate the program.

In addition to the technical challenges of implementing gesture recognition, there are also social challenges. Gestures must be simple, intuitive and universally acceptable. The study of gestures and other nonverbal types of communication is known as kinesics.

Gesture Recognition cases the level of handling a broad range of devices such as personal navigation devices, computers, laptops and mobile handsets. A growing trend towards touchless gesture recognition in niche applications including gaming is likely to amplify the growth of the touchless sensing market over the projected period.

Gesture recognition was implemented extensively in gaming consoles, with the segment accounting for over 50% share in 2016. Increasing sales of gaming consoles such as Xbox and PlayStation is expected to bolster the growth of the market over the projected period.

Smartphones are expected to witness the highest growth over the projected period on account of increasing implementation of the technology for safety & security

purposes. Moreover, the high adoption of digitalization coupled with ease of use of mobile phones is expected to fuel the growth of the market over the forecast period.

Smart televisions are expected to witness above average growth on account of high disposable income of consumers and an increasing trend towards purchasing televisions with the latest technology. The segment is expected to grow at a CAGR of over 28% from 2017 to 2024.

Applications of gesture recognition in the hospitality sector, which accounted for a comparatively modest 8.1% revenue share of the market in 2015, are expected to witness expansion at the highest, 17.0% CAGR, across other application areas over the said period. The flourishing travel industry and the rising numbers of high-end hotels across the globe will favor the adoption of systems such as touchless biometric recognition, hand dryers, and automated faucets and flushes.

ACKNOWLEDGEMENT

In the accomplishment of this project successfully, many people have best owned upon us their heart pledged support and well wishes. We as a group would like to utilize this time to thank all the people who have been concerned with this project. Primarily we would like to thank our University to present us with the opportunity to work on such a wonderful project.

Next, we would like to thank our very supporting Dean of our concerned department who was extremely supportive in helping us with choosing our projects on time and letting us know about due deadlines well in advance.

Further we would like to thank all our teachers throughout our academic years who taught us extremely well and made us able to understand and execute the needs of the concerned project.

We further thank our mentor Dr. Kavita Ma'am for helping us choose the project and guiding us constantly on how to execute it. We thank our evaluators who constantly monitored and evaluated our projects and helped us identify the shortcomings and made us work on it.

Lastly, we would like to thank each other (team members) for staying and working together on the project and helping each other out in every way possible.

CHAPTER 1

INTRODUCTION

Computer is used by many people either at their work or in their spare-time. Special input and output devices have been designed over the years with the purpose of easing the communication between computers and humans, the two most known are the keyboard and mouse. Every new device can be seen as an attempt to make the computer more intelligent and making humans able to perform more complicated communication with the computer. This has been possible due to the result oriented efforts made by computer professionals for creating successful human computer interfaces. As the complexities of human needs have turned into many folds and continues to grow so, the need for Complex programming ability and intuitiveness are critical attributes of computer programmers to survive in a competitive environment. The computer programmers have been incredibly successful in easing the communication between computers and human. With the emergence of every new product in the market; it attempts to ease the complexity of jobs performed. For instance, it has helped in facilitating tele operating, robotic use, better human control over complex work systems like cars, planes and monitoring systems. Earlier, Computer programmers were avoiding such kind of complex programs as the focus was more on speed than other modifiable features. However, a shift towards a user friendly environment has driven them to revisit the focus area. The idea is to make computers understand human language and develop a user friendly human computer interfaces (HCI). Making a computer understand speech, facial expressions and human gestures are some steps towards it. Gestures are the non-verbally exchanged information. A person can perform innumerable gestures at a time. Since human gestures are perceived through vision, it is a subject of great interest for computer vision researchers. The project aims to determine human gestures by creating an HCI. Coding of these gestures into machine language demands a complex programming algorithm. An overview of gesture recognition system is given to gain knowledge.

1.1. OVERALL DESCRIPTION

A hand gesture recognition system provide a natural, innovative and modern way of non verbal communication. It has a wide area of application in human computer interaction and sign language. The intention of this project is to discuss a novel approach of hand gesture recognition based on detection of some shape based features. The setup consist of a single camera to capture the gesture formed by the user and take this hand image as an input to the proposed algorithm. The overall algorithm divided into four main steps, which includes segmentation, orientation detection, feature extraction and classification. It is independent of user characteristics. It does not require any kind of training of sample data It takes a less computation time as compare to other approaches.

Gesture recognition has been a very interesting problem in Computer Vision community for a long time. This is particularly due to the fact that segmentation of foreground object from a cluttered background is a challenging problem in real-time. The most obvious reason is because of the semantic gap involved when a human looks at an image and a computer looking at the same image. Humans can easily figure out what's in an image but for a computer, images are just 3-dimensional matrices. It is because of this, computer vision problems remains a challenge.

As per the context of the project, gesture is defined as an expressive movement of body parts which has a particular message, to be communicated precisely between a sender and a receiver. A gesture is scientifically categorized into two distinctive categories: dynamic and static. A dynamic gesture is intended to change over a period of time whereas a static gesture is observed at the spurt of time. A waving hand means goodbye is an example of dynamic gesture and the stop sign is an example of static gesture. To understand a full message, it is necessary to interpret all the static and dynamic gestures over a period of time. This complex process is called gesture recognition. Gesture recognition is the process of recognizing and interpreting a stream continuous sequential gesture from the given set of input data.

1.2. PURPOSE

We are going to recognize hand gestures from a video sequence. To recognize these gestures from a live video sequence, we first need to take out the hand region alone removing all the unwanted portions in the video sequence. After segmenting the hand region, we then count the fingers shown in the video sequence to instruct a robot based on the finger count. Thus, the entire problem could be solved using 2 simple steps -

1. Find and segment the hand region from the video sequence.

2. Count the number of fingers from the segmented hand region in the video sequence.

1.3. MOTIVATION AND SCOPE

This project is related to two significant fields, computer vision and machine learning. Both of these field are of immense importance in contemporary times due to their widespread use in various disciplines. Computer vision can be defined as a field that incorporates methods for acquiring, processing, understanding and using images and in general any high dimensional real-world data in order to produce useful information. Computer vision has been extensively used in various fields like Human computer Interaction, Medicine, Physics, Image reconstruction etc. over the past years and lately it has gained much more traction as it has been used in mainstream devices like Xbox, PS4, smart phones, Tablets, medical devices etc. Machine Learning on the other hand is a subfield of Computer Science that evolved from studying pattern recognition and computational learning in Artificial Intelligence. Machine learning is closely related to computational statistics, prediction making and mathematical optimization has been widely used for applications like spam filtering, Computer Vision, OCR, search predictions and other prediction-based applications.

This project focuses on gesture recognition and it uses computer vision and machine learning techniques to achieve this goal. Gesture recognition is important in the field of HCI and HCI plays an important role in applications like Gaming, User Interaction with software systems and accessibility support. We can use various body parts like hand, fingers, head and other objects to perform gestures, but this project focuses on hand gestures.

CHAPTER 2

LITERATURE SURVEY

Research has been limited to small scale systems able of recognizing a minimal subset of a full sign language. Christopher Lee and Yangsheng Xu developed a glove-based gesture recognition system that was able to recognize 14 of the letters from the hand alphabet, learn new gestures and able to update the model of each gesture in the system in online mode, with a rate of 10Hz, Over the years advanced glove devices have been designed such as the Sayre Glove, Dexterous Hand Master and PowerGlove. The most successful commercially available glove is by far the VPL DataGlove as shown in figure 1.2 It was developed by Zimmerman during the 1970's. It is based upon patented optical fiber sensors along the back of the fingers. Star-ner and Pentland developed a glove-environment system capable of recognizing 40 signs from the American Sign Language (ASL) with a rate of 5Hz. Hyeon Kyu Lee and Jin H. Kim presented work on real-time hand-gesture recognition using HMM (Hidden Markov Model). Kjeldsen and Kendersi devised a technique for doing skin-tone segmentation in HSV space, based on the premise that skin tone in images occupies a connected volume in HSV space. They further developed a system which used a backpropagation neural network to recognize gestures from the segmented hand images. Etsuko Ueda and Yoshio Matsumoto presented a novel technique a hand-pose estimation that can be used in this method, the hand regions are extracted from multiple images obtained by a multiviewpoint camera system, and constructing the "voxel Model." Hand pose is estimated. Chan Wah Ng, Surendra Ranganath presented a hand gesture recognition system, they used image furrier descriptor as their prime feature and classified with the help of RBF network. Their system's overall performance was 90.9 %. Claudia Nõlker and Helge Ritter presented a hand gesture recognition modal based on recognition of finger tips, in their approach they find full identification of all finger joint angles and based on that a 3D modal of hand is prepared and using neural network.

Model based Segmentation and recognition of dynamic gestures in continuos video stream:

In this paper the authors mainly concentrated on the segmentation and recognition of continuos gestures which effect from the spatiotemporal variations. The authors used two types of gestures one is two arm movements for contour extraction, and another one includes a single hand movement hands contour used as the feature vector. They proposed a Multi scale Gesture Model. This model proposes 3 approaches which mainly differs in end point Localization. First proposed approach is used to find the end points with Multi scale search and Motion detection strategy. Second proposed approach is used to locate the end points of the fingers roughly with Dynamic Time wrapping. Third approach is based on Dynamic programming. Using these three approaches the recognition of the hand ranges from the 88% to 96%.

Improved Adaptive Gaussian Mixture Model for Background Subtraction:

Background subtraction is a common computer vision task. We analyze the usual pixel-level approach. We develop an efficient adaptive algorithm using Gaussian mixture probability density. Recursive equations are used to constantly update the parameters and but also to simultaneously select the appropriate number of components for each pixel.

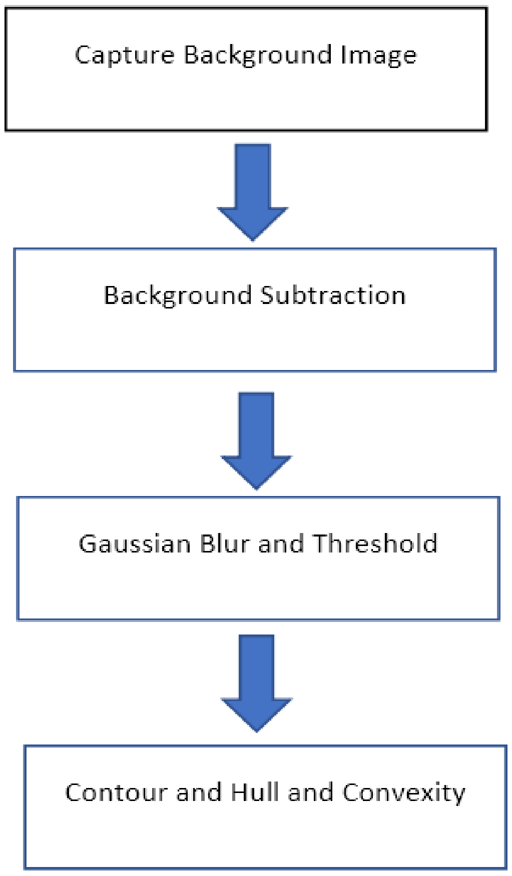
CHAPTER 3

PROPOSED MODEL

Hand Movements are recorded using a web cam, the web cam captures the images and sends them as input image for the image pre-processing.

First of all, we will create a binary mask of the hand in order to compute the hand contour. Furthermore, we will convert our frames, which are in BGR format by default as you read them from a file or capture in OpenCV, to HLS (Hue, Lightness, Saturation) color space. The Hue channel encodes the actual color information. By this we only have to figure out the proper Hue value range of the skin and then adjust the values for Saturation and Lightness. Next, we will tell OpenCV to find all contours in the mask. We will return the largest contour in case segmentation did not work out as well and still contains noise.Now that we have detected the contour, we start to discuss the actual algorithm for:detecting the fingertips and the number of fingers shown.

To achieve this, we will compute the convex hull as well as the convexity defect regions of the hand contour. Lastly, we count the hull points to find out the number of fingers being pointed at the camera to get our final result.



Background Subtraction

First, we need an efficient method to separate foreground from background. To do this, we use the concept of running averages. We make our system to look over a particular scene for 30 frames. During this period, we compute the running average over the current frame and the previous frames

After figuring out the background, we bring in our hand and make the system understand that our hand is a new entry into the background, which means it becomes the foreground object

After figuring out the background model using running averages, we use the current frame which holds the foreground object (hand in our case) in addition to the background. We calculate the absolute difference between the background model (updated over time) and the current frame (which has our hand) to obtain a difference image that holds the newly added foreground object (which is our hand). This is what Background Subtraction is all about.

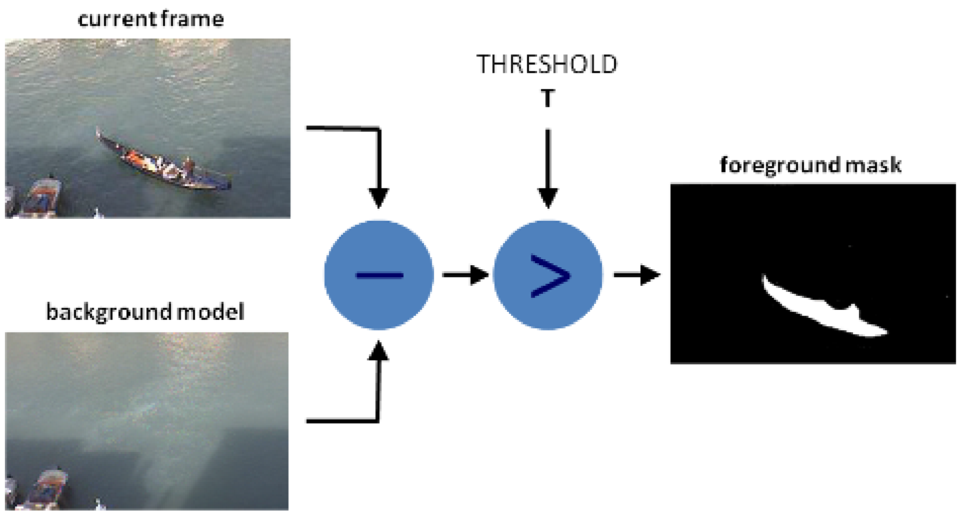


Fig 3.1. Background Subtraction

CHAPTER 4

IMPLEMENTATION

4.1. SOFTWARE USED

The project is build using Python, OpenCV and Numpy as the basic components. The architecture of the project consists of the following components:

4.1.1. PYTHON

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.

4.1.1.1. Features

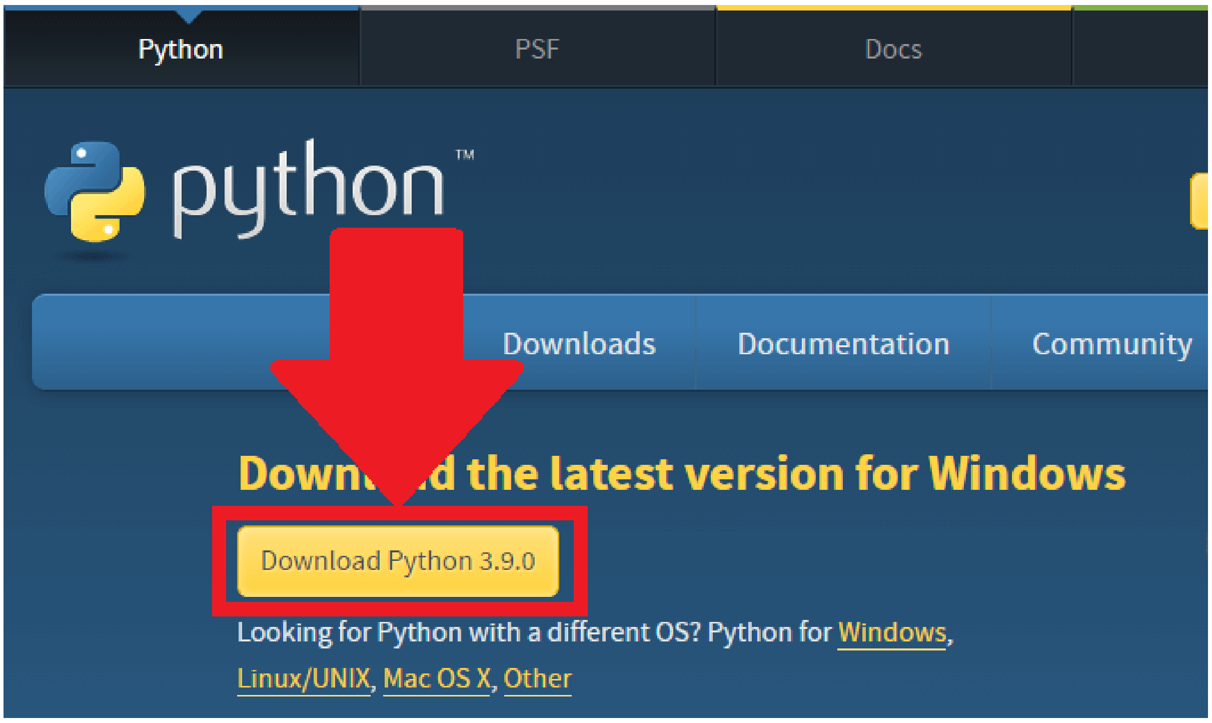
Python comes with the following features -

- 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.

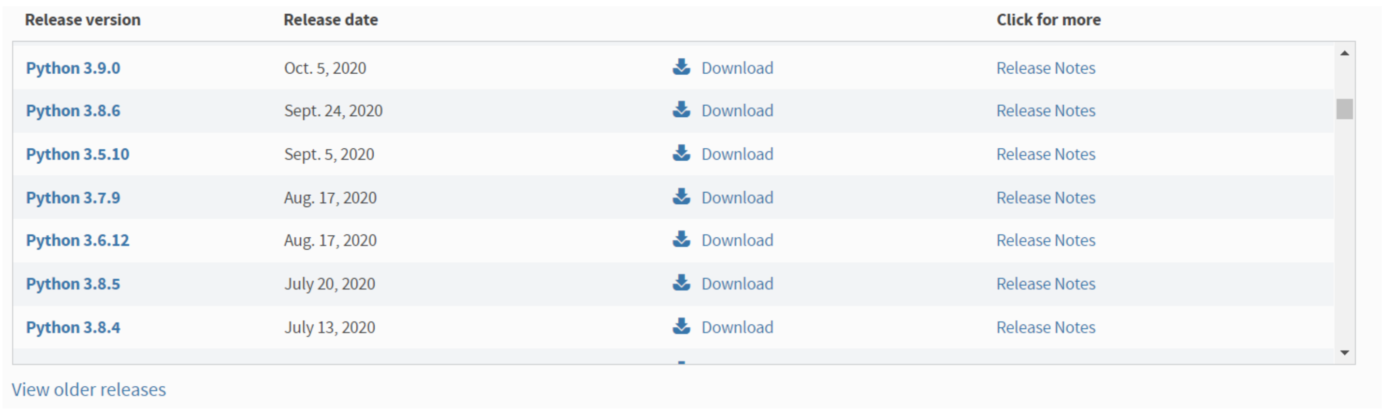
4.1.1.2. Python Installation

STEP 1: Visit the link https://www.python.org/downloads/ to download the latest release of Python. In this process, we will install Python 3.9.0 on our Windows operating system.


Python
PSF
Docs
python™
Downloads
Documentation
Community
Download the latest version for Windows
Download Python 3.9.0
Looking for Python with a different OS? Python for Windows, Linux/UNIX, Mac OS X, Other

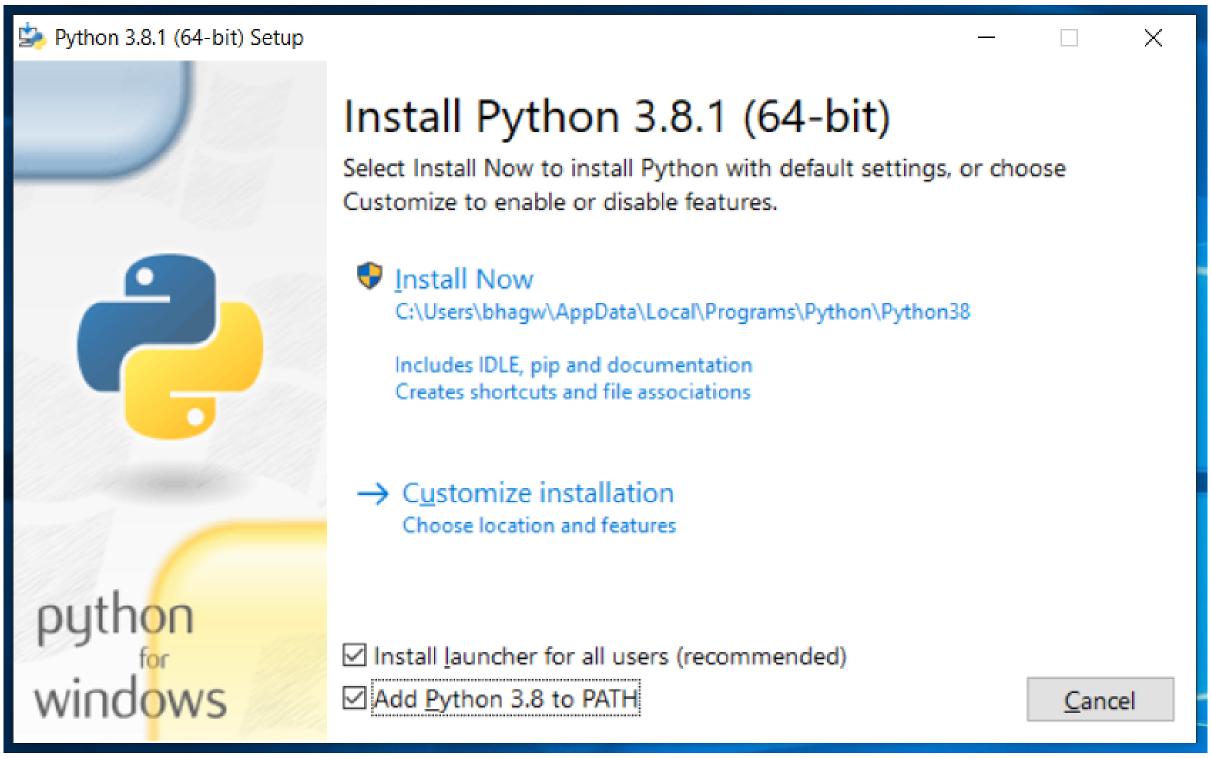
STEP 2: Then click on windows x84-64 web based installer.

STEP 3: Double-click the executable file which is downloaded; the following window will open Select Customize installation and proceed.



Release version	Release date		Click for more
Python 3.9.0	Oct. 5, 2020	Download	Release Notes
Python 3.8.6	Sept. 24, 2020	Download	Release Notes
Python 3.5.10	Sept. 5, 2020	Download	Release Notes
Python 3.7.9	Aug. 17, 2020	Download	Release Notes
Python 3.6.12	Aug. 17, 2020	Download	Release Notes
Python 3.8.5	July 20, 2020	Download	Release Notes
Python 3.8.4	July 13, 2020	Download	Release Notes

View older releases


Python 3.8.1 (64-bit) Setup

Install Python 3.8.1 (64-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

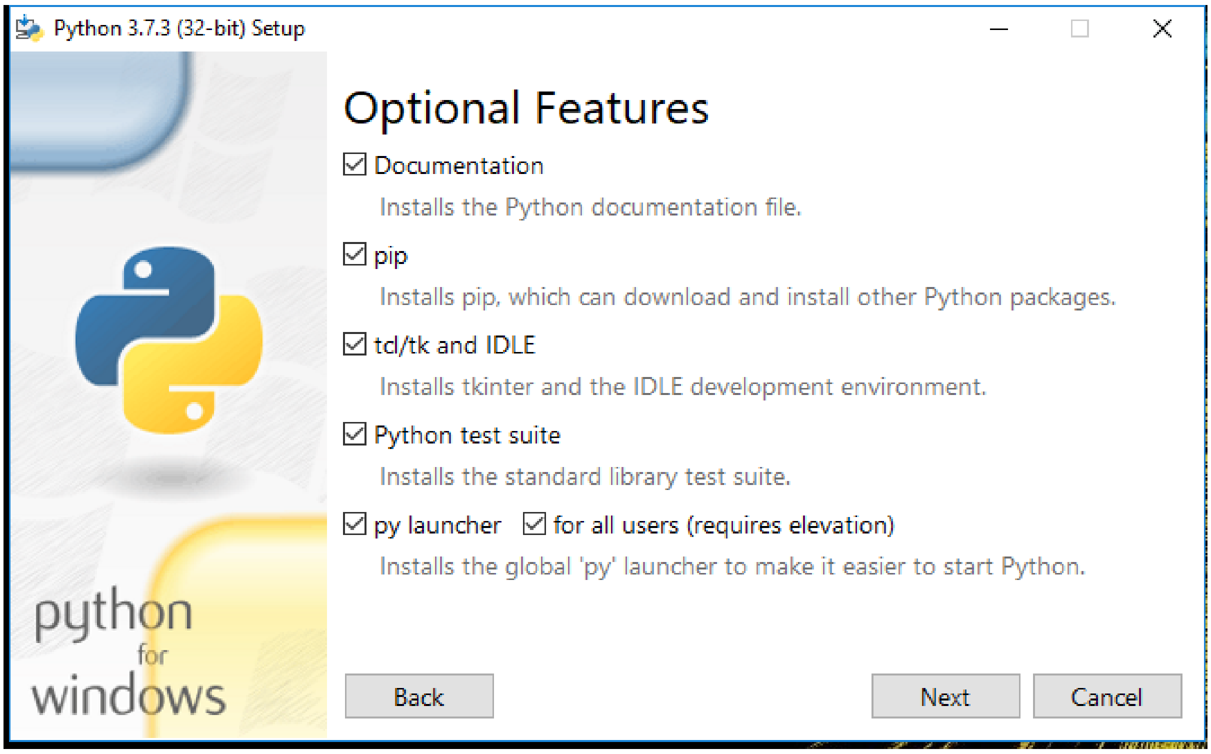
Install Now
C:\Users\bhagw\AppData\Local\Programs\Python\Python38

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

☑ Install launcher for all users (recommended)
☑ Add Python 3.8 to PATH

STEP 4: The following window shows all the optional features. All the feat installed and are checked by default; we need to click next to continue.


Python 3.7.3 (32-bit) Setup

Optional Features

☑ Documentation
Installs the Python documentation file.

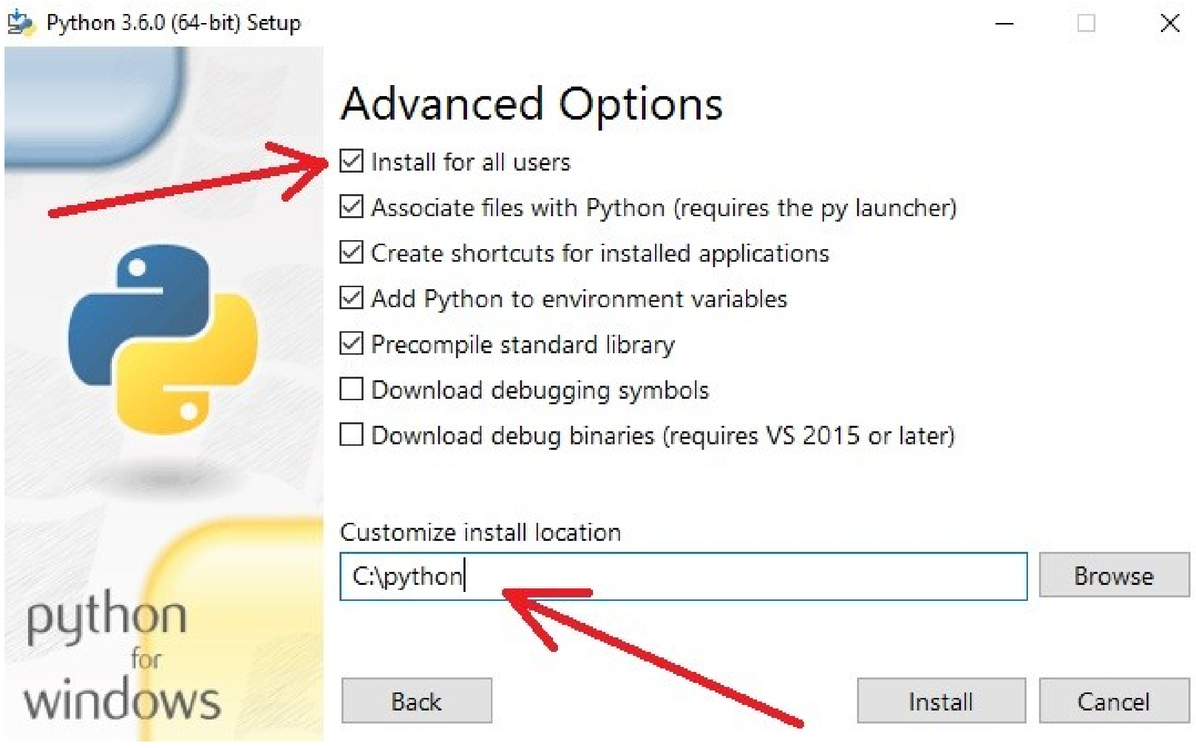
☑ pip
Installs pip, which can download and install other Python packages.

☑ tcl/tk and IDLE
Installs tkinter and the IDLE development environment.

☑ Python test suite
Installs the standard library test suite.

☑ py launcher ☑ for all users (requires elevation)
Installs the global 'py' launcher to make it easier to start Python.

STEP 5: The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, We must noticed that the first checkbox (install for all user) must be checked.



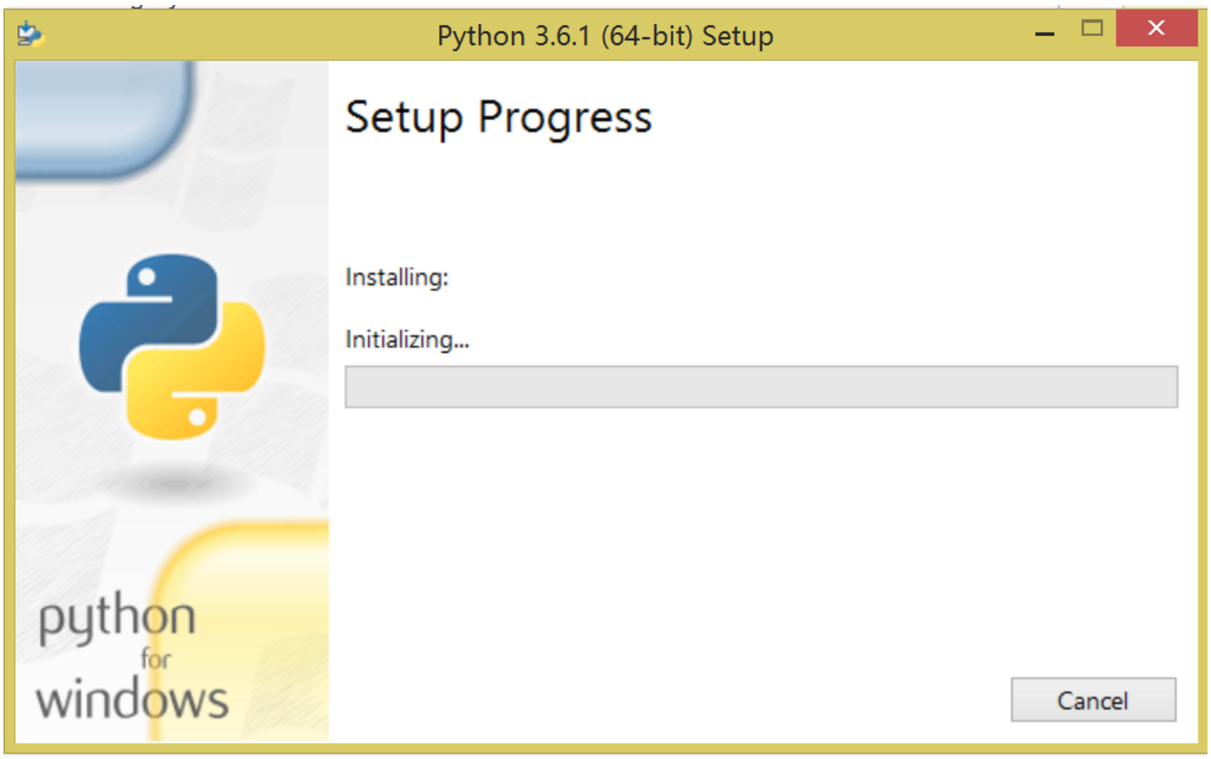
Python 3.6.0 (64-bit) Setup

Advanced Options

- [✓] Install for all users
- [✓] Associate files with Python (requires the py launcher)
- [✓] Create shortcuts for installed applications
- [✓] Add Python to environment variables
- [✓] Precompile standard library
- [ ] Download debugging symbols
- [ ] Download debug binaries (requires VS 2015 or later)

Customize install location: C:\python

STEP 6: Now, we are ready to install python-3.9.0. Let's install it

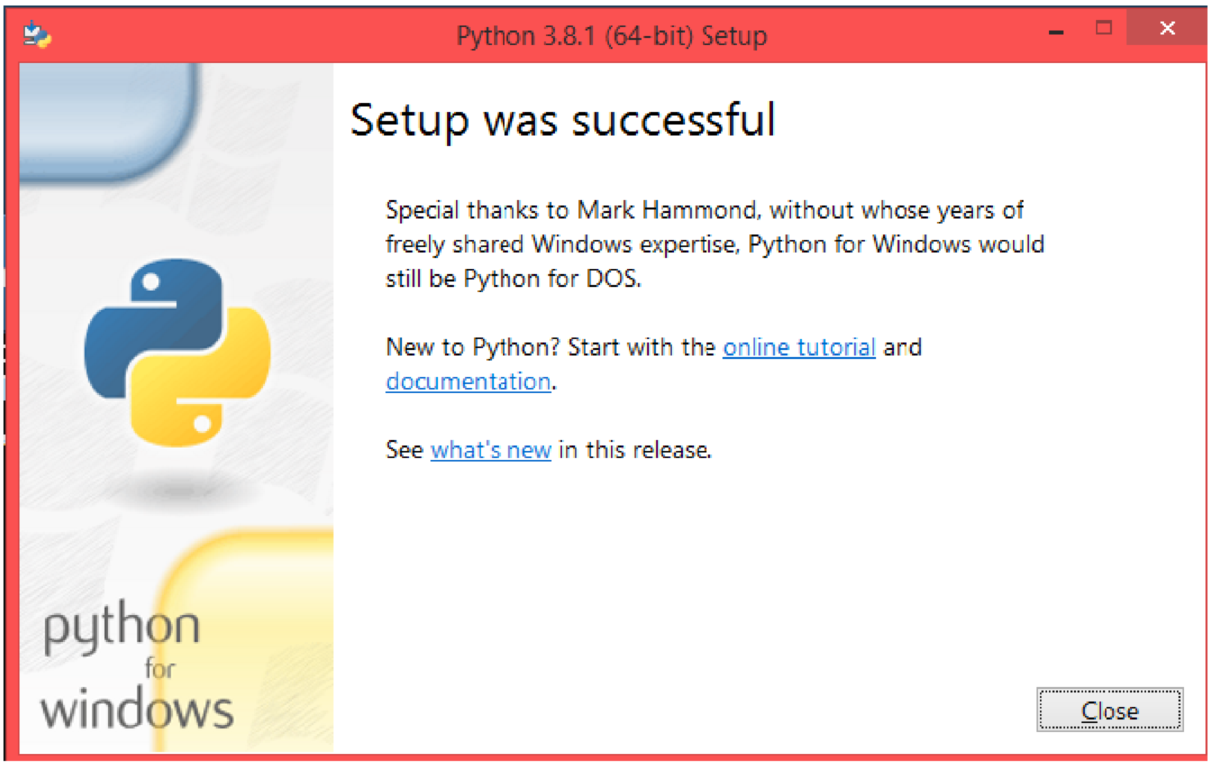

Python 3.6.1 (64-bit) Setup

Setup Progress

Installing:

Initializing...

STEP 7: Once python is installed we will get this window


Python 3.8.1 (64-bit) Setup

Setup was successful

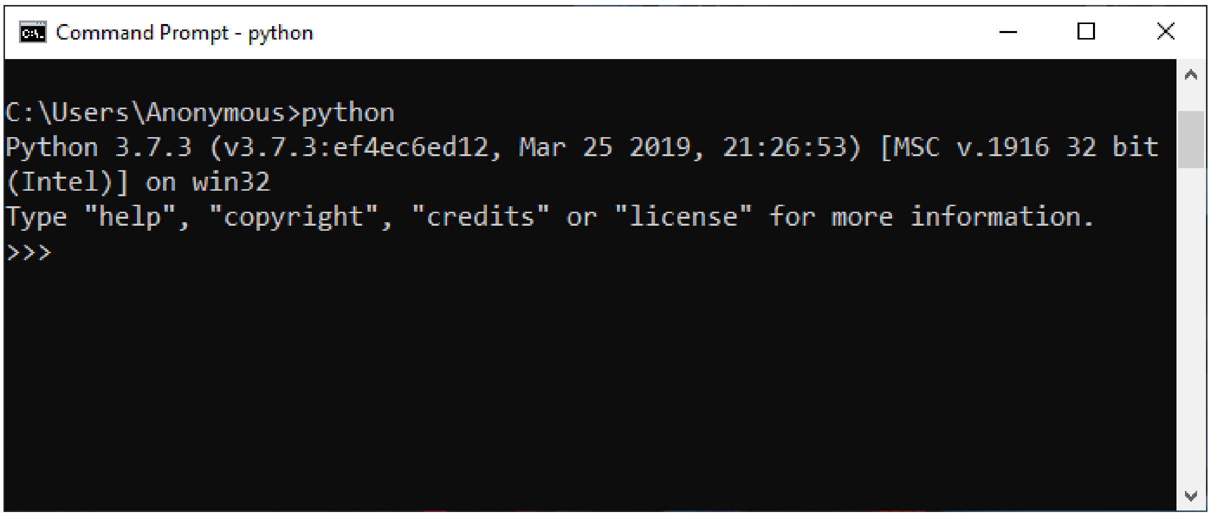
Special thanks to Mark Hammond, without whose years of freely shared Windows expertise, Python for Windows would still be Python for DOS.

New to Python? Start with the online tutorial and documentation.

See what's new in this release.

4.1.2. PIP

Pip (package manager) is a package management system used to install and manage software packages written in Python. Many packages can be found in the default source for packages and their dependencies -Python Package Index (PyPI).



```
C:\Users\Anonymous>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Python 2.7.9 and later (on the python2 series), and Python 3.4 and later include pip (pip3 for Python 3) by default. Pip is a recursive acronym for "Pip Installs Packages"

4.1.2.1. Command-line interface

Most distributions of Python come with pip preinstalled. If pip is missing, it can be installed through the system package manager or by invoking URL, a client-side data transfer tool: curl https://bootstrap-pypa.io/get-pip.py | python

One major advantage of pip is the case of its command-line interface, which makes installing Python software packages as easy as issuing a command:

Most importantly pip has a feature to manage full lists of packages and corresponding version numbers, possible through a "requirements" file. This permits the efficient re-creation of an entire group of packages in a separate environment (e.g. another computer) or virtual environment. This can be achieved with a properly formatted requirements.txt file and the following command

pip install -r requirements.txt

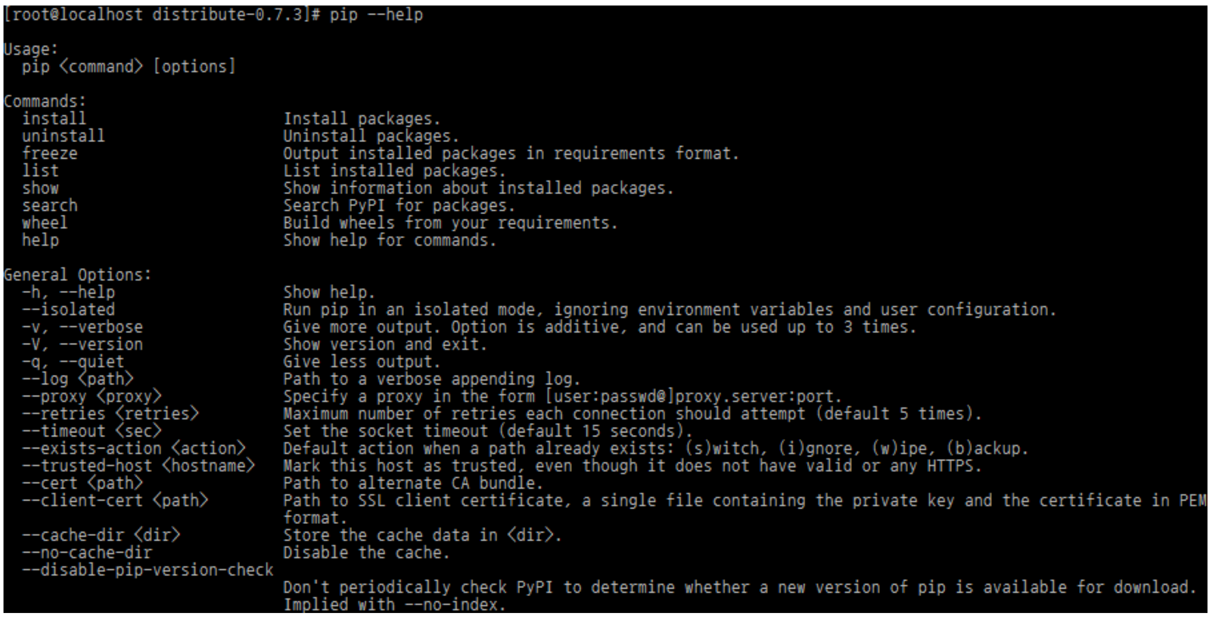
Install some package for a specific version python, where $(version) is replaced for 2, 3,3.4. etc.:

pip$ (version) install some-package-name

4.1.2.2. Installation of Pip

STEP 1: For installation of pip we need to type pip in command prompt and get the following result which show pip is installed.

STEP 2: Next we need to upgrade the pip. So, to update pip we type the the following command: pip3 install-upgrade pip.



```
[root@localhost distribute-0.7.3]# pip --help

Usage:
  pip <command> [options]

Commands:
  install                     Install packages.
  uninstall                   Uninstall packages.
  freeze                      Output installed packages in requirements format.
  list                        List installed packages.
  show                        Show information about installed packages.
  search                      Search PyPI for packages.
  wheel                       Build wheels from your requirements.
  help                        Show help for commands.

General Options:
  -h, --help                  Show help.
  --isolated                  Run pip in an isolated mode, ignoring environment variables and user configuration.
  -v, --verbose               Give more output. Option is additive, and can be used up to 3 times.
  -V, --version               Show version and exit.
  -q, --quiet                 Give less output.
  --log <path>                Path to a verbose appending log.
  --proxy <proxy>             Specify a proxy in the form [user:passwd@]proxy.server:port.
  --retries <retries>         Maximum number of retries each connection should attempt (default 5 times).
  --timeout <sec>             Set the socket timeout (default 15 seconds).
  --exists-action <action>    Default action when a path already exists: (s)witch, (i)gnore, (w)ipe, (b)ackup.
  --trusted-host <hostname>   Mark this host as trusted, even though it does not have valid or any HTTPS.
  --cert <path>               Path to alternate CA bundle.
  --client-cert <path>        Path to SSL client certificate, a single file containing the private key and the certificate in PEM
                              format.
  --cache-dir <dir>           Store the cache data in <dir>.
  --no-cache-dir              Disable the cache.
  --disable-pip-version-check
                              Don't periodically check PyPI to determine whether a new version of pip is available for download.
                              Implied with --no-index.
```

4.1.2. OpenCV

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed. at real time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez (which was later acquired by Intel). The library is cross platform and free for use under the open-source BSD license.

OpenCV supports the deep learning frameworks TensorFlow, Torch/PyTorch and Caffe.

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

General description

- Open source computer vision library in C/C++.
- Optimized and intended for real-time applications.
- OS/hardware/window-manager independent. Generic image/video loading, saving, and acquisition.
- Both low- and high-level API.
- Provides interface to Intel's Integrated Performance Primitives (IPP) with processor specific optimization (Intel processors).

Features

- Image data manipulation (allocation, release, copying, setting, conversion).
- Image and video I/O (file and camera based input, image/video file output).
- Matrix and vector manipulation and linear algebra routines (products, solvers,eigenvalues, SVD).
- Various dynamic data structures (lists, queues, sets, trees, graphs).
- Basic image processing (filtering, edge detection, corner detection, sampling and interpolation, color conversion, morphological operations, histograms, image pyramids). Structural analysis (connected components, contour processing, distance transform, various moments, template matching, Hough transform, polygonal approximation, line fitting, ellipse fitting, Delaunay triangulation).
- Camera calibration (finding and tracking calibration patterns, calibration,fundamental matrix estimation, homography estimation, stereo correspondence).
- Motion analysis (optical flow, motion segmentation, tracking).
- Object recognition (eigen-methods, HMM).
- Basic GUI (display image/video, keyboard and mouse handling, scroll-bars).
- Image labeling (line, conic, polygon, text drawing)

4.1.3. NumPy

Numpy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Huguninwith contributions from several other developers. In 2005, Travis Oliphant created NumPy by incorporating features of the competing Numarray into Numeric, with extensive modifications. NumPy is open-source software and has many contributors.

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.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

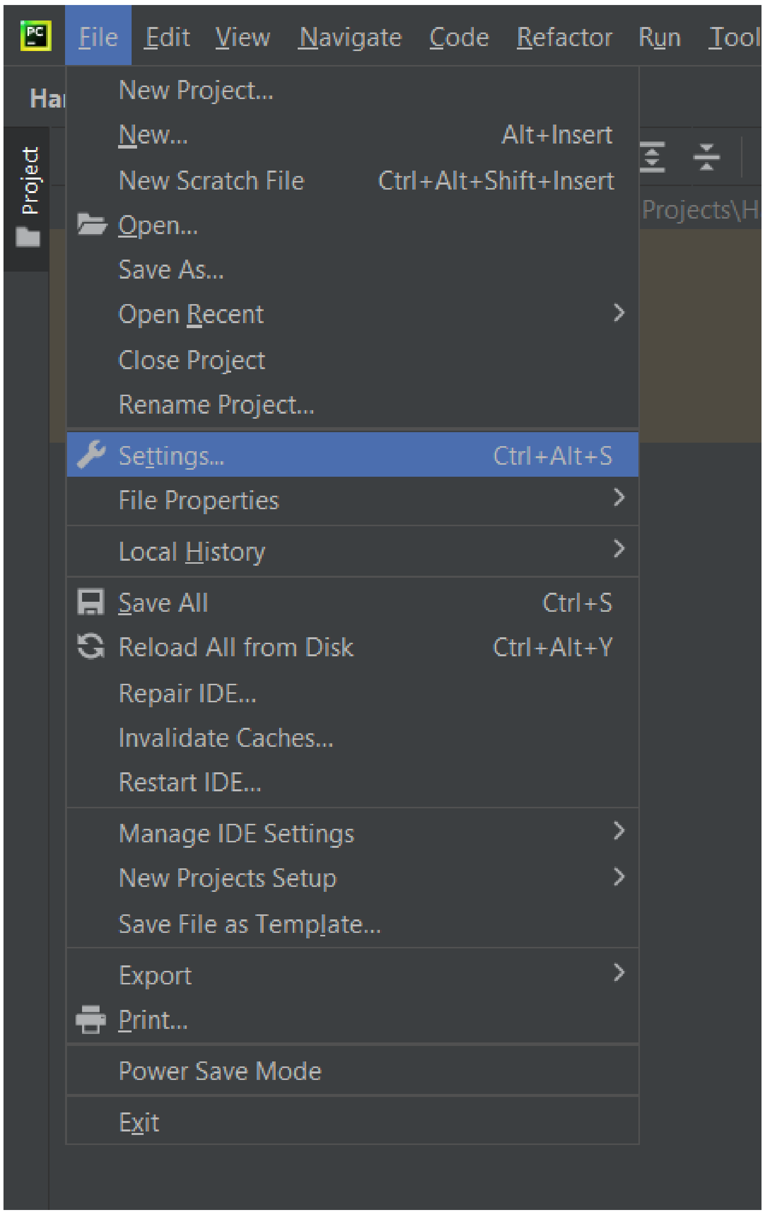
STEPS TO DOWNLOAD INSTALL AND RUN THE PROJECT

Step 1

Go to this link https://www.jetbrains.com/pycharm/download/#section=windows
And Download the community version of PyCharm

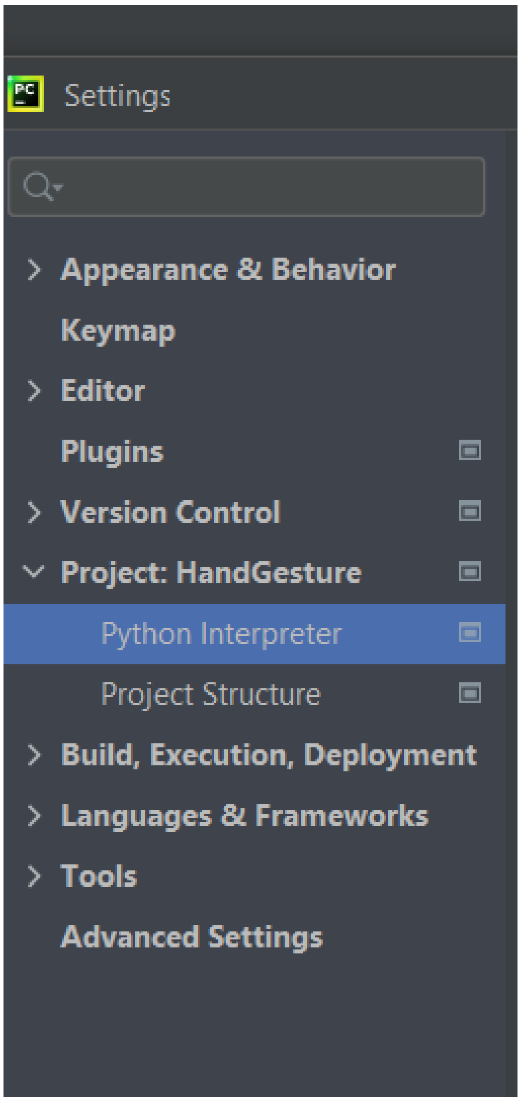
Step 2

Go to File and Click of settings



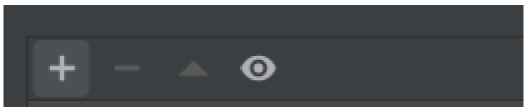
Step 3

Go to python interpreture



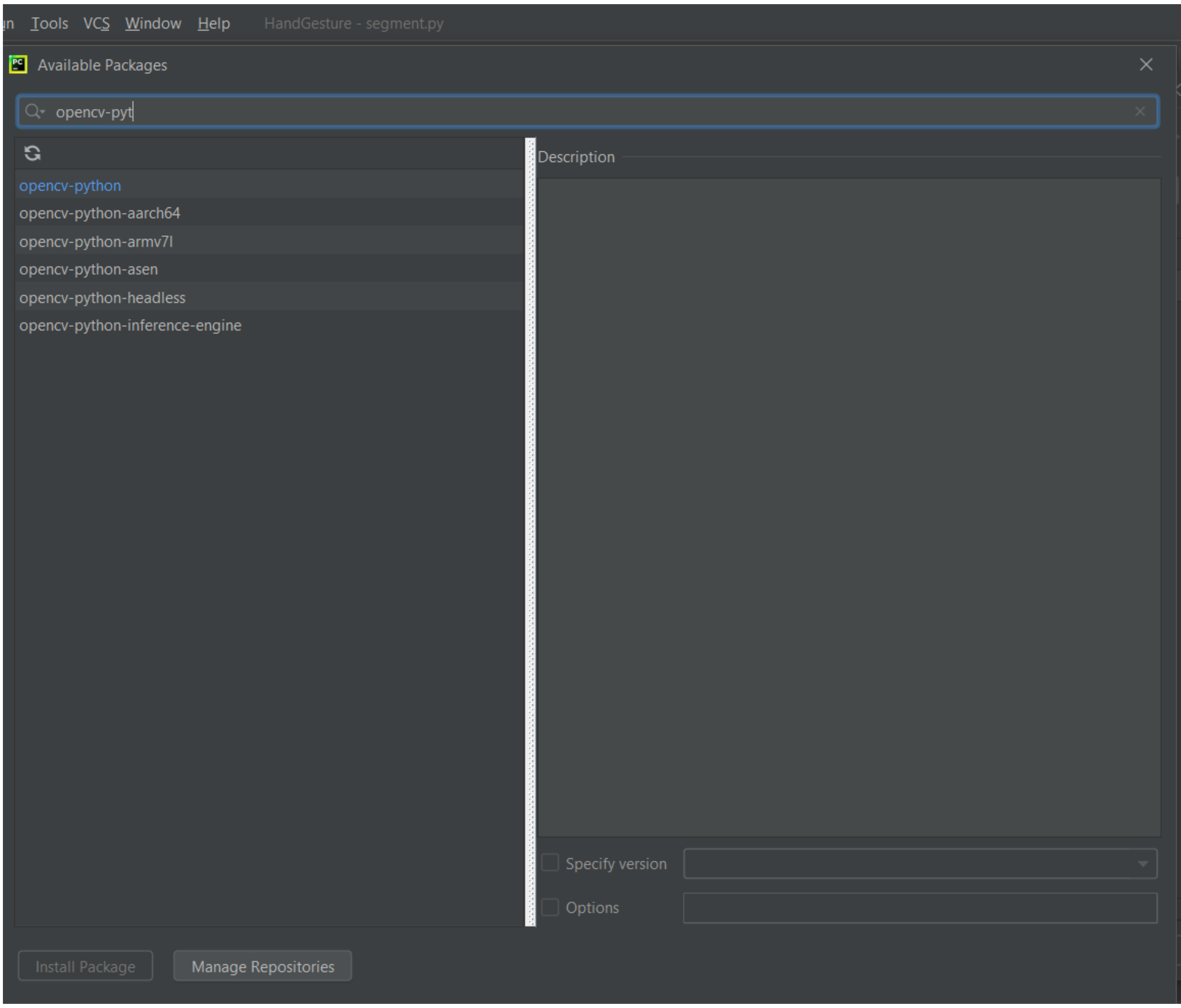
Step 4

Go to Plus(+) icon



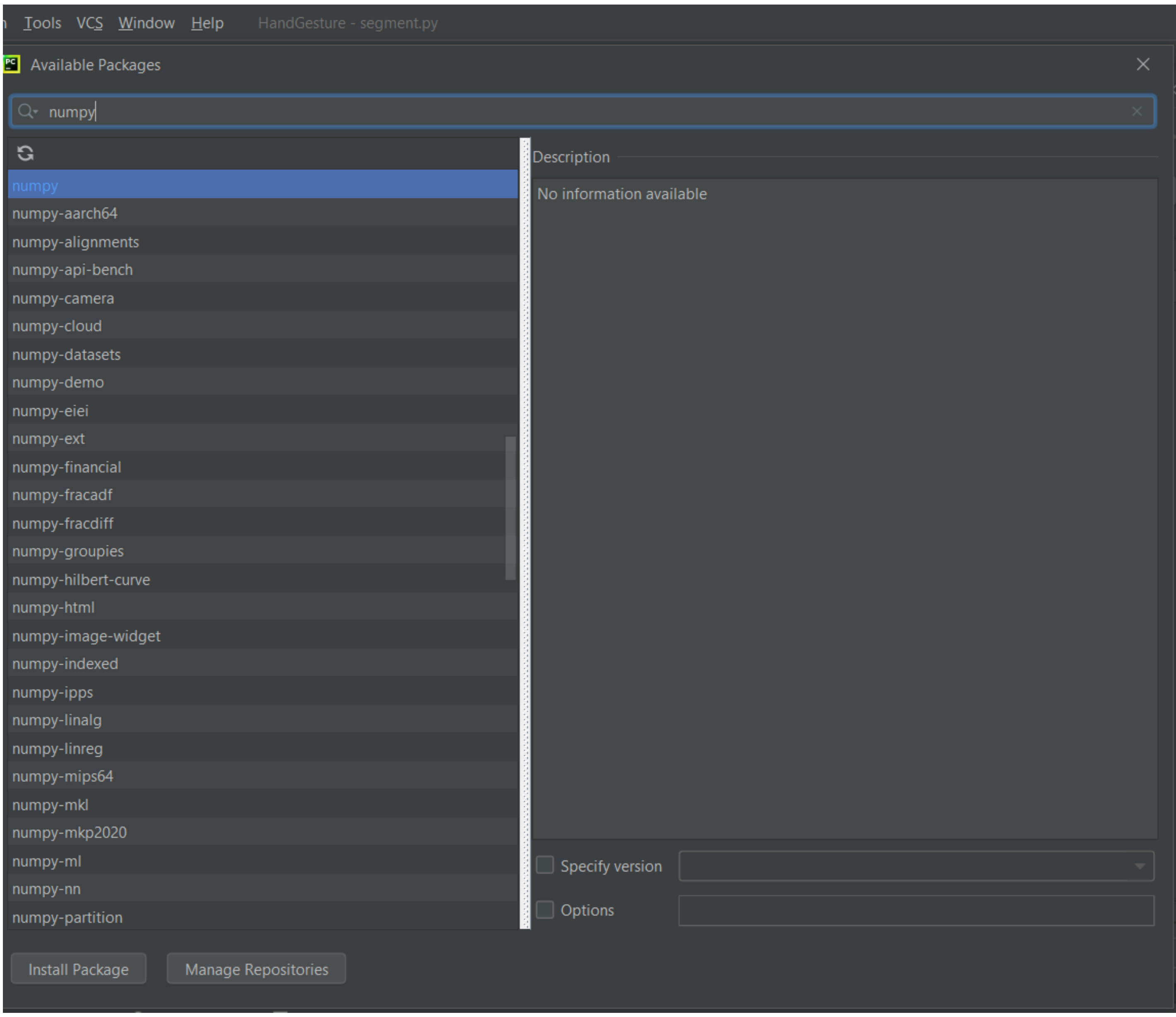
Step 5

Type OpenCv-python and select OpenCv from the drop down menu and click on install package



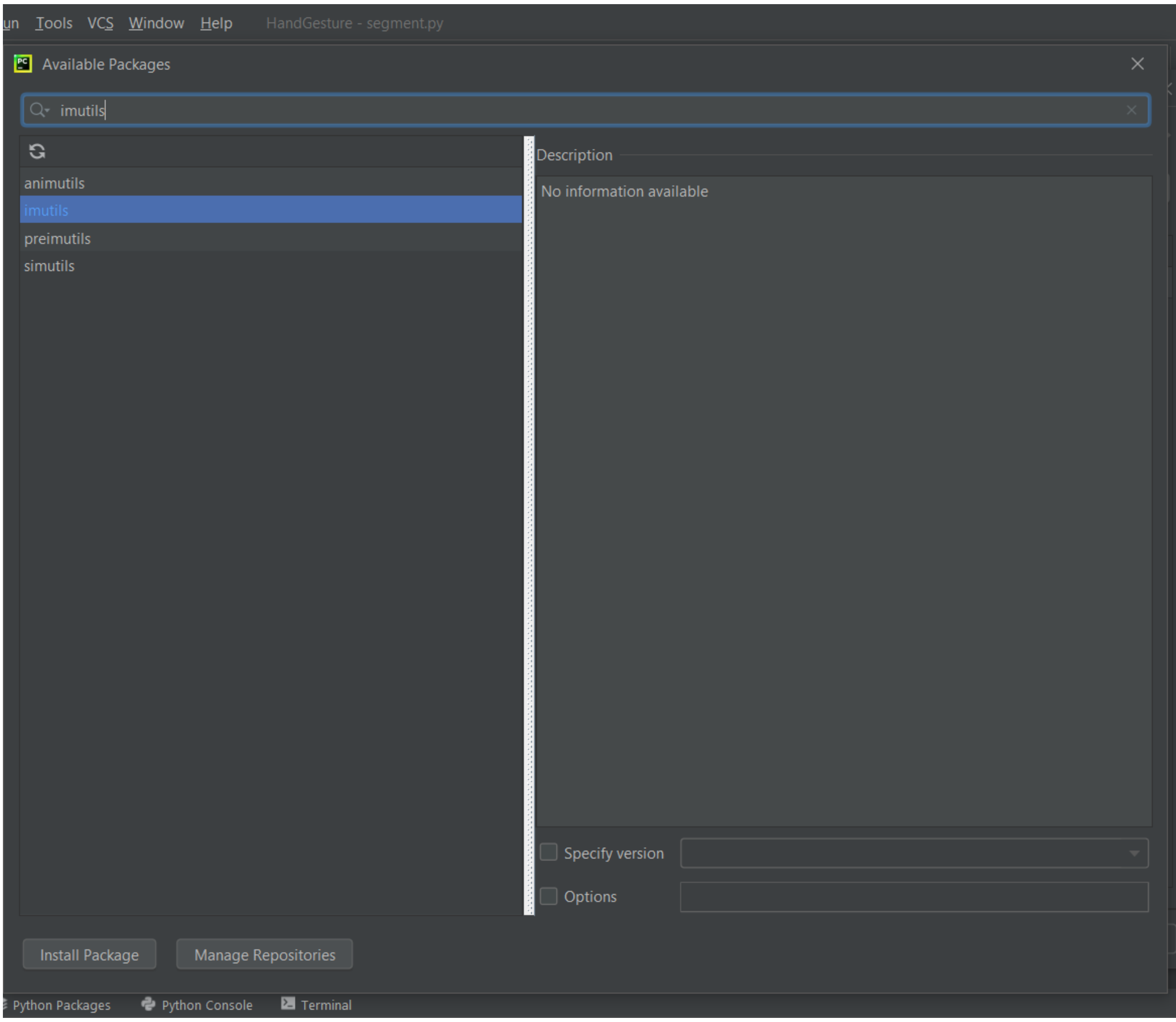
Step 6

Type NumPy and select NumPy from the drop down menu and click on install package



Step 7

Type Imutils and select Imutils from the drop down menu and click on install package



Step 8

Now Type the Source Code and then Click on run , you will get the output

4.2 SOURCE CODE

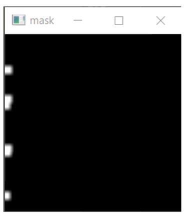
```
import traceback
import cv2
import numpy as np
import math

cap = cv2.VideoCapture(0)

while (1):

    try:  # an error comes if it does not find anything in window as it cannot find contour of max area
        # therefore this try error statement

        ret, frame = cap.read()
        frame = cv2.flip(frame, 1)
        kernel = np.ones((3, 3), np.uint8)

        # define region of interest
        roi = frame[100:300, 100:300]

        cv2.rectangle(frame, (100, 100), (300, 300), (0, 255, 0), 0)
        hsv = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)

        # define range of skin color in HSV
        lower_skin = np.array([0, 20, 70], dtype=np.uint8)
        upper_skin = np.array([20, 255, 255], dtype=np.uint8)

        # extract skin colur imagw
        mask = cv2.inRange(hsv, lower_skin, upper_skin)

        # extrapolate the hand to fill dark spots within
        mask = cv2.dilate(mask, kernel, iterations=4)

        # blur the image
        mask = cv2.GaussianBlur(mask, (5, 5), 100)

        # find contours
        contours, hierarchy = cv2.findContours(mask, cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)
        print(contours)
        print(hierarchy)
        # find contour of max area(hand)
        cnt = max(contours, key=lambda x: cv2.contourArea(x))

        # approx the contour a little
        epsilon = 0.0005 * cv2.arcLength(cnt, True)
        approx = cv2.approxPolyDP(cnt, epsilon, True)

        # make convex hull around hand
        hull = cv2.convexHull(cnt)

        # define area of hull and area of hand
        areahull = cv2.contourArea(hull)
        areacnt = cv2.contourArea(cnt)

        # find the percentage of area not covered by hand in convex hull
        arearatio = ((areahull - areacnt) / areacnt) * 100

        # find the defects in convex hull with respect to hand
        hull = cv2.convexHull(approx, returnPoints=False)
        defects = cv2.convexityDefects(approx, hull)

        # l = no. of defects
        l = 0

        # code for finding no. of defects due to fingers
        for i in range(defects.shape[0]):
            s, e, f, d = defects[i, 0]
            start = tuple(approx[s][0])
            end = tuple(approx[e][0])
            far = tuple(approx[f][0])
            pt = (100, 180)

            # find length of all sides of triangle
            a = math.sqrt((end[0] - start[0]) ** 2 + (end[1] - start[1]) ** 2)
            b = math.sqrt((far[0] - start[0]) ** 2 + (far[1] - start[1]) ** 2)
            c = math.sqrt((end[0] - far[0]) ** 2 + (end[1] - far[1]) ** 2)
            s = (a + b + c) / 2
            ar = math.sqrt(s * (s - a) * (s - b) * (s - c))

            # distance between point and convex hull
            d = (2 * ar) / a

            # apply cosine rule here
            angle = math.acos((b ** 2 + c ** 2 - a ** 2) / (2 * b * c)) * 57

            # ignore angles > 90 and ignore points very close to convex hull(they generally come due to
noise)
            if angle <= 90 and d > 30:
                l += 1
                cv2.circle(roi, far, 3, [255, 0, 0], -1)

            # draw lines around hand
            cv2.line(roi, start, end, [0, 255, 0], 2)

        l += 1

        # print corresponding gestures which are in their ranges
        font = cv2.FONT_HERSHEY_SIMPLEX
        if l == 1:
            if areacnt < 2000:
```

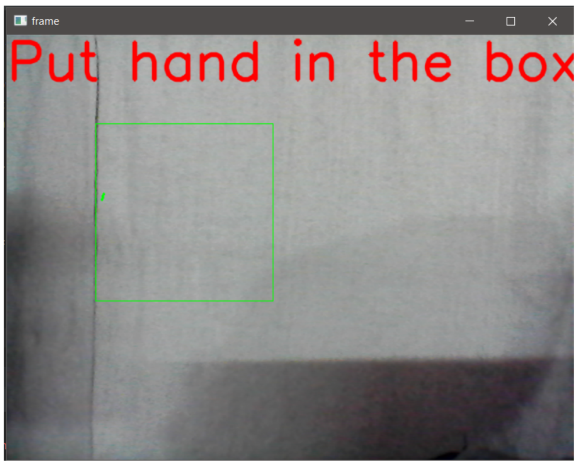
```
            cv2.putText(frame, 'Put hand in the box', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
        else:
            if arearatio < 12:
                cv2.putText(frame, '0', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
            elif arearatio < 17.5:
                cv2.putText(frame, 'Best of luck', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

            else:
                cv2.putText(frame, '1', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    elif l == 2:
        cv2.putText(frame, '2', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    elif l == 3:

        if arearatio < 27:
            cv2.putText(frame, '3', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)
        else:
            cv2.putText(frame, 'ok', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    elif l == 4:
        cv2.putText(frame, '4', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    elif l == 5:
        cv2.putText(frame, '5', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    elif l == 6:
        cv2.putText(frame, 'reposition', (0, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    else:
        cv2.putText(frame, 'reposition', (10, 50), font, 2, (0, 0, 255), 3, cv2.LINE_AA)

    # show the windows
    cv2.imshow('mask', mask)
    cv2.imshow('frame', frame)
  except Exception:
    traceback.print_exc()
    pass
  # break

  k = cv2.waitKey(5) & 0xFF
  if k == 27:
    break

cv2.destroyAllWindows()
cap.release()
```

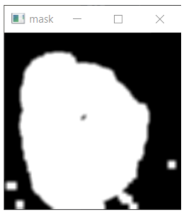
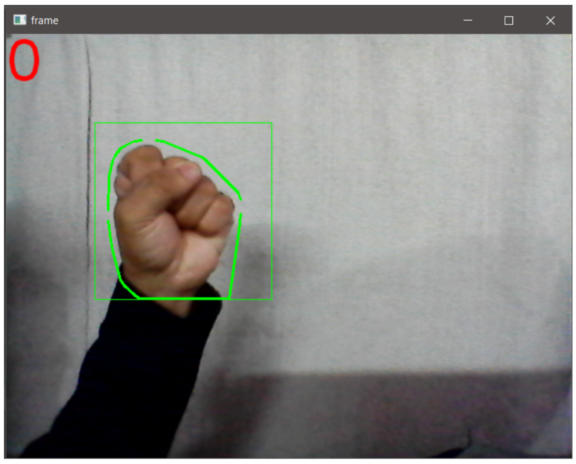
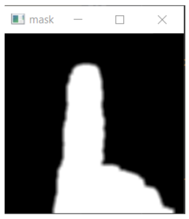
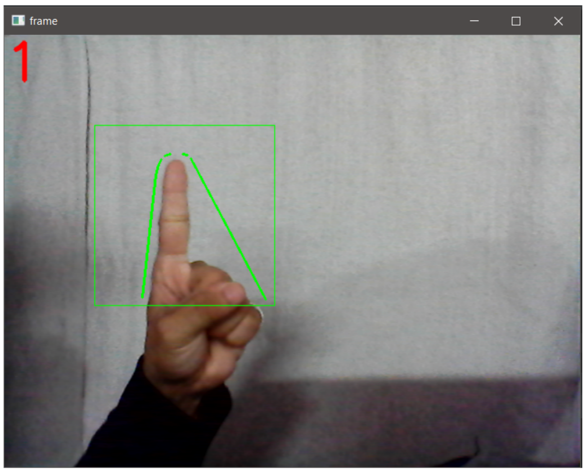
CHAPTER 5

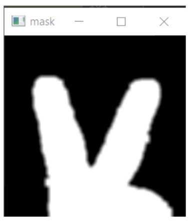
RESULT AND DISCUSSION

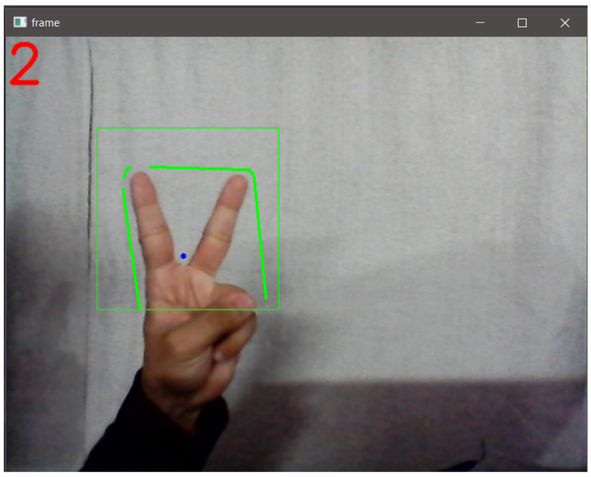
- By Gaussian blurring, we create smooth transition from one color to another and reduce the edge content. Then we use thresholding to create binary images from grayscale images.
- We now need to find out the hand contour from the binary image we created before and detect fingers (or in other words, recognize gestures)
- Below figure shows the number of finger in the region

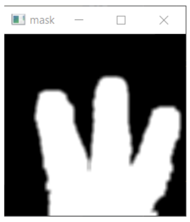


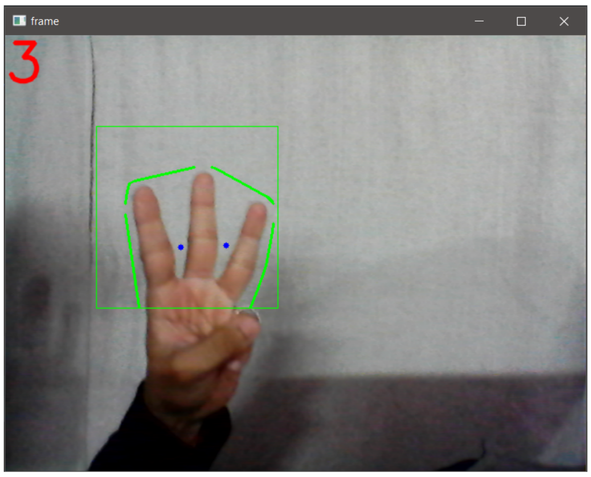


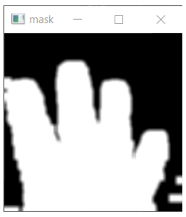

mask

frame
0

mask

frame
1

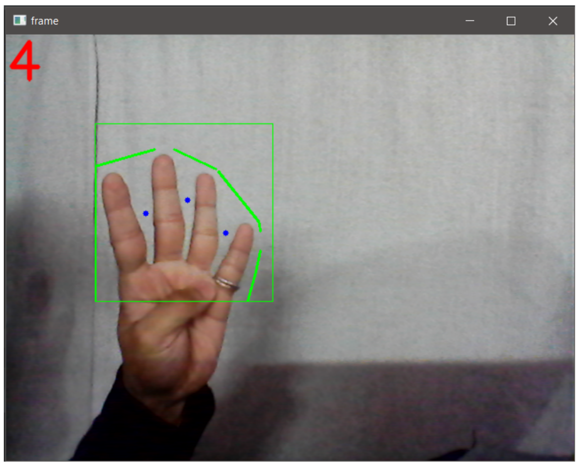

mask

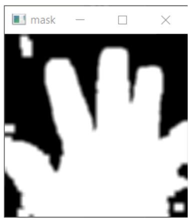

frame
2

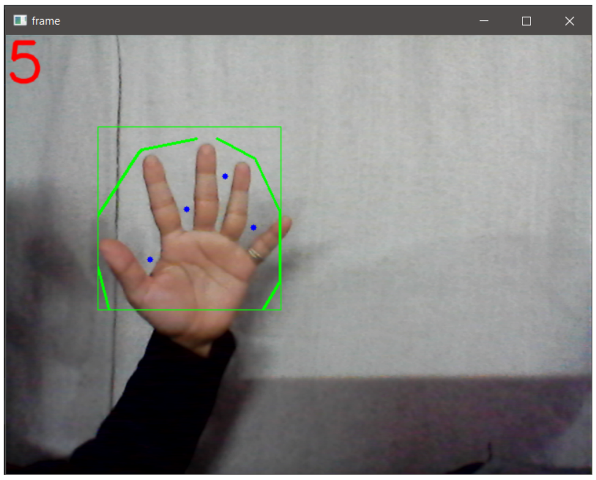

mask

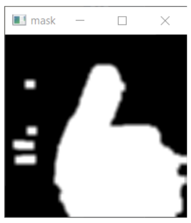

frame
3

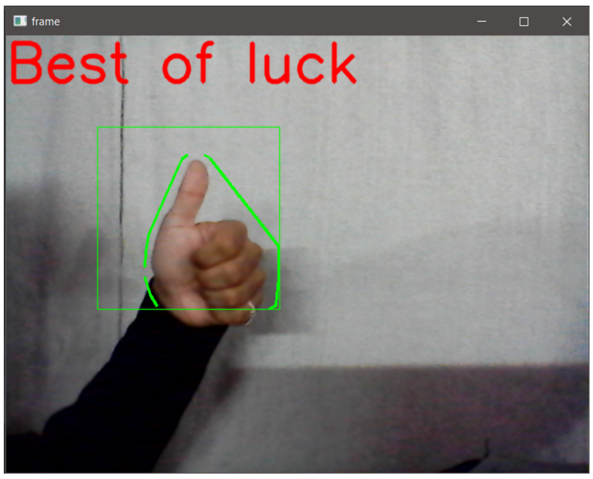

mask

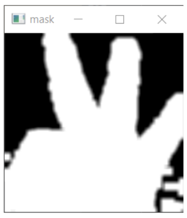

frame
4

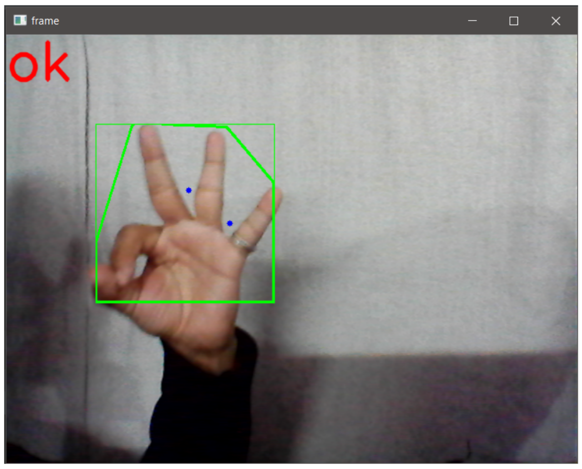

mask


frame
5


mask


frame
Best of luck


mask


frame
ok

CHAPTER 6

CONCLUSION AND FUTURE WORKS

CONCLUSION AND FUTURE WORKS

In recent years a lot of research has been conducted in gesture recognition. The aim of this project was to develop an offline Gesture recognition system. We have shown in this project that offline gesture recognition system can be designed using SVM. It is determined that contour is very important feature and can be used for discrimination between two gesture. The processing steps to classify a gesture included gesture acquisition, segmentation, morphological filtering, contour representation and classification using different technique.

In this project firstly the original image is been processed by contours for finding the number of fingers using background deletion and taking only the feasible region of our interest, then the feasible part is made white and rest part is all black using the binary masking technique. After this segmentation is done and hence the number of fingers is counted as output.

Future work of the project is that it can be used in gaming consoles, it can be used for controlling the applications of laptop desktop and other devices. It can also be used in smart Tv for controlling the volume and other functions using the hand gestures. The other field where this can be used is in controlling the robots for working according to our commands.

REFERENCES

1. https://gogul09.github.io/software/hand-gesture-recognition -pl

2. https://gogul09.github.io/software/hand-gesture-recognition -p2

3. https://medium.com/@muchler.v/simple-hand-gesture-recognition-using-openev
and-javascript-eb3d6ced28a0

4. http://www.cs.cornell.edu/courses/cs4670/2010fa/projects/final/resul
ts/group_of_jah477_rhs229/Writeup.pdf

5.https://www.youtube.com/watchv=7JhjINPwfYQ&list-PLEIEAq2VkUULYYgi13YHU
WmRePqiu8Ddy

6.https://pdfs.semanticscholar.org/fe14/7092dbfb0c5ddb6a956cd98c13fcacc56alf.pd
f

7. https://www.academia.edu/17775220/Hand Gesture_Recognition A_Literature
Review

8. https://web.stanford.edu/class/cs231a/prev_projects 2016/CS231A Project
Final.pdf
9.https://pdfs.semanticscholar.org/4651/b6a9fc9d4dba7dac347ee76274ab6c05c8e6.
pdf

10.http://www.ijfresce.org/download/browse/Volume_4/June_18Volume_4_Issue_6/1
530176523 28-06-2018 .pdf

11. https://github.com/rpmcginty/hand-gesture-recognition project/blob/master/Hand
Gesture Recognition Project Report.pdf

12. https://www.diva-portal.org/smash/get/diva2:519237/

FULLTEXTO1.pdf

13.
https://pdfs.semanticscholar.org/01fa/619ef31e45c5838d06651539128ef254b030
.pdf

14. http://www.massey.ac.nz/- albarcza ResearchFiles/Maria_Abastillas
Project_2011.pdf

15. Amiraj Dhawan, Vipul Honraolimplementation of hand detection based
techniques for human computer interactionl, in International Journal of Computer
Applications (0975-8887)

16. L. R. Rabiner, A tutorial on hidden Markov models and selected applications in
speech recognition, Proceedings of The IEEE 77 (2), 1989, pp.257-285.

17. OpenCV usage documentation http://docs.opencv .org

18. http://inside.mines.edu/~whoff/courses/EENG512/projects/2015/Rodriguez .pdf
19. Hand gesture recognition system by Prof. Praveen D. Hasalkar1, Rohit S. Chougule2,Vrushabh B. Madake3, Vishal S. Magdum4,2015

20. Vision Based Hand Gesture Recognition for Human Computer Interaction by Radhika Bhatt, Nikita Fernandes, Archana Dhage, 2013

21. Implentation of Hand Gesture Recognition Technique for HCI Using OpenCV

Nayana P, Sanjeev Kubakaddi 2014

22. REAL TIME HAND GESTURE RECOGNITIONSYSTEM FOR DYNAMIC

APPLICATIONS. Siddharth S. Rautarayl, Anupam Agrawal2

23. Gesture recognition -A Review by miss kawade Sonam PI, Prof VS. Ubale.

24. Yikai Fang. "A real time Hand Gesture method

25. M.Correra, "Real time Hand gesture method for Human Robot Interaction.

26. V.L. Pavlovic, R. Sharma, T.S. Huang, Visual interpretation of hand gestures for human-computer interaction, A Review, IEEE Transactions on Patter Analysis and Machine Intelligence 19(7): 677-695, 1997.

27. M.Young, The Technical Writer'sHandbook MillValley,CA:University Science, 1989.

28. Stammer, T. and Pentland. Real-Time American Sign Language Recognition from Video Using Hidden Markov Models, TR-375,MITMedialab, 1995.