VIRTUAL MOUSE USING AI & COMPUTER VISION

A PROJECT REPORT

In partial fulfilment of the requirements for the award of the degree

BACHELOR

OF

COMPUTER APPLICATION

Under the guidance of

JOYJIT GUHA

BY

MUGDHAMOY DASGUPTA

Future Institute of Engineering and Management

FUTURE INSTITUTE OF ENGINEERING AND MANAGEMENT

In association with

ARDENT COMPUTECH PVT. LTD.

(ISO9001:2015)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

1.	Title of the Project:	**VIRTUAL MOUSE USING AI COMPUTER VISION**
2.	Project Members:	**ABHRA PRTIM GHOSH, SOURISH GHOSH, BIJAY GHOSH, DIPANJAN DUTTA ROY, SOURAJIT BHATTACHARYA,VIVEK PAHRI**
3.	Name of the guide:	**Mr. JOYJIT GUHA**
4.	Address:	Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified) SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal, 700091

<u>Project Version Control History</u>

Version	**Primary Author**	**Description of Version**	**Date Completed**
Final	Abhra Pratim Ghosh, Sourish Ghosh, Bijay Ghosh, Sourajit Bhattacharya, Dipanjan Dutta Roy, Vivek Pahri.	Project Report	29th JUNE,2021

Signature of Team Members Signature of Approver

Date:	Date:
For Office Use Only	**MR.JOYJIT GUHA**
Approved **Not Approved**	Project Proposal Evaluator

DECLARATION

We hereby declare that the project work being presented in the project proposal entitled **"VIRTUAL MOUSE USING AI & COMPUTER VISION"** in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF COMPUTER APPLICATION** at **ARDENT COMPUTECH PVT. LTD, SALTLAKE, KOLKATA, WEST BENGAL,** is an authentic work carried out under the guidance of **MR. JOYJIT GUHA**. The matter embodied in this project work has not been submitted elsewhere for the award of any degree of our knowledge and belief.

Date: 29.06.2021

Name of the Students: Abhra Pratim Ghosh, Sourish Ghosh, Bijay Ghosh, Sourajit Bhattacharya, Dipanjan Dutta Roy, Vivek Pahri.

Signature of the students:

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

CERTIFICATE

This is to certify that this proposal of minor project entitled **"VIRTUAL MOUSE USING AI & COMPUTER VISION"** is a record of Bonafede work, carried out by **Abhra Pratim Ghosh, Sourish Ghosh, Bijay Ghosh, Sourajit Bhattacharya, Dipanjan Dutta Roy, Vivek Pahri.** under my guidance at **ARDENT COMPUTECH PVT LTD**. In my opinion, the report in its present form is in partial fulfilment of the requirements for the award of the degree of **BACHELOR OF COMPUTER APPLICATION** and as per regulations of the **ARDENT®.** To the best of my knowledge, the results embodied in this report, are original in nature and worthy of incorporation in the present version of the report.

Guide / Supervisor

MR. JOYJIT GUHA

Project Engineer

Ardent Computech Pvt. Ltd (An ISO 9001:2015 Certified)

SDF Building, Module #132, Ground Floor, Salt Lake City, GP Block, Sector V, Kolkata, West Bengal 700091

ACKNOWLEDGEMENT

Success of any project depends largely on the encouragement and guidelines of many others. I take this sincere opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project work.

I would like to show our greatest appreciation to ***Mr. JOYJIT GUHA***, Project Engineer at Ardent, Kolkata. I always feel motivated and encouraged every time by his valuable advice and constant inspiration; without his encouragement and guidance this project would not have materialized.

Words are inadequate in offering our thanks to the other trainees, project assistants and other members at Ardent Computech Pvt. Ltd. for their encouragement and cooperation in carrying out this project work. The guidance and support received from all the members and who are contributing to this project, was vital for the success of this project.

CONTENTS

OVERVIEW

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 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.

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, Small Talk, 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.

FEATURES OF PYTHON

Easy-to-learn: Python has few Keywords, simple structure and clearly defined syntax. This allows a 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 the 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 enables 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 support 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 datatypes and supports dynamic type checking.
- It supports automatic garbage collections.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA and JAVA.

ENVIRONMENT SETUP

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- UNIX (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS

BASIC SYNTAX OF PYTHON PROGRAM

Type the following text at the Python prompt and press the Enter –

```
>>> print "Hello, Python!"
```

If you are running new version of Python, then you would need to use print statement with parenthesis as in **print ("Hello, Python!");**.
However in Python version 2.4.3, this produces the following result –

```
Hello, Python!
```

Python Identifiers

A Python identifier is a name used to identify a variable, function, class, module or other object. An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
Python does not allow punctuation characters such as @, $, and % within identifiers. Python is a case sensitive programming language.

Python Keywords

The following list shows the Python keywords. These are reserved words and you cannot use them as constant or variable or any other identifier names. All the Python keywords contain lowercase letters only.

And, exec, not
Assert, finally, or
Break, for, pass
Class, from, print
continue, global, raise
def, if, return
del, import, try
elif, in, while
else, is, with
except, lambda, yield

Lines & Indentation

Python provides no braces to indicate blocks of code for class and function definitions or flow control. Blocks of code are denoted by line indentation, which is rigidly enforced.

The number of spaces in the indentation is variable, but all statements within the block must be indented the same amount. For example –

```
if True:
print "True"
else:
print "False"
```

Command Line Arguments

Many programs can be run to provide you with some basic information about how they should be run. Python enables you to do this with -h −

```
$ python-h
usage: python [option]...[-c cmd|-m mod | file |-][arg]...

Options and arguments (and corresponding environment variables):

-c cmd: program passed in as string(terminates option list)

-d      : debug output from parser (also PYTHONDEBUG=x)

-E      : ignore environment variables (such as PYTHONPATH)

-h      : print this help message and exit [ etc.]
```

VARIABLE TYPES

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Assigning Values to Variables

Python variables do not need explicit declaration to reserve memory space. The declaration happens automatically when you assign a value to a variable. The equal sign (=) is used to assign values to variables.

```
counter=10            # An integer assignment
weight=10.60          # A floating point
name="Ardent"          # A string
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously. For example −

```
a = b = c = 1
a,b,c = 1,2,"hello"
```

Standard Data Types

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has five standard data types −

- String
- List
- Tuple
- Dictionary
- Number

Data Type Conversion

Sometimes, you may need to perform conversions between the built-in types. To convert between types, you simply use the type name as a function.

There are several built-in functions to perform conversion from one data type to another.

Sr.No.	Function & Description
1	**int(x [,base])** Converts x to an integer. base specifies the base if x is a string
2	**long(x [,base])** Converts x to a long integer. base specifies the base if x is a string.
3	**float(x)** Converts x to a floating-point number.
4	**complex(real [,imag])** Creates a complex number.
5	**str(x)** Converts object x to a string representation.
6	**repr(x)** Converts object x to an expression string.
7	**eval(str)** Evaluates a string and returns an object.
8	**tuple(s)** Converts s to a tuple.
9	**list(s)** Converts s to a list.

FUNCTIONS

Defining a Function

- def functionname(parameters):
  "function_docstring"
  function_suite
  return [expression]

Pass by reference vs Pass by value

All parameters (arguments) in the Python language are passed by reference. It means if you change what a parameter refers to within a function, the change also reflects back in the calling function. For example –

```
# Function definition is here

def changeme(mylist):
        "This changes a passed list into this function"
        mylist.append([1,2,3,4]);
        print"Values inside the function: ",mylist
        return

# Now you can call changeme function

mylist=[10,20,30];
changeme(mylist);
print"Values outside the function: ",mylist
```

Here, we are maintaining reference of the passed object and appending values in the same object. So, this would produce the following result −

```
Values inside the function: [10, 20, 30, [1, 2, 3, 4]]
Values outside the function: [10, 20, 30, [1, 2, 3, 4]]
```

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope . For Example-

```
total=0;                    # This is global variable.

# Function definition is here
```

```
def sum( arg1, arg2 ):

# Add both the parameters and return them."

total= arg1 + arg2;       # Here total is local variable.
print"Inside the function local total : ", total
return total;

# Now you can call sum function

sum(10,20);
print"Outside the function global total : ", total
```

When the above code is executed, it produces the following result −

```
Inside the function local total : 30
Outside the function global total : 0
```

MODULES

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference .

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```
def print_func( par ):
    print"Hello : ", par
    return
```

The *import* Statement

You can use any Python source file as a module by executing an import statement in some other Python source file. The *import* has the following syntax –

```
import module1[, module2[,… moduleN]
```

PACKAGES

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-subpackages, and so on.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code −

```
def Pots():
	print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function Isdn()
- *Phone/G3.py* file having function G3()

Now, create one more file __init__.py in *Phone* directory −

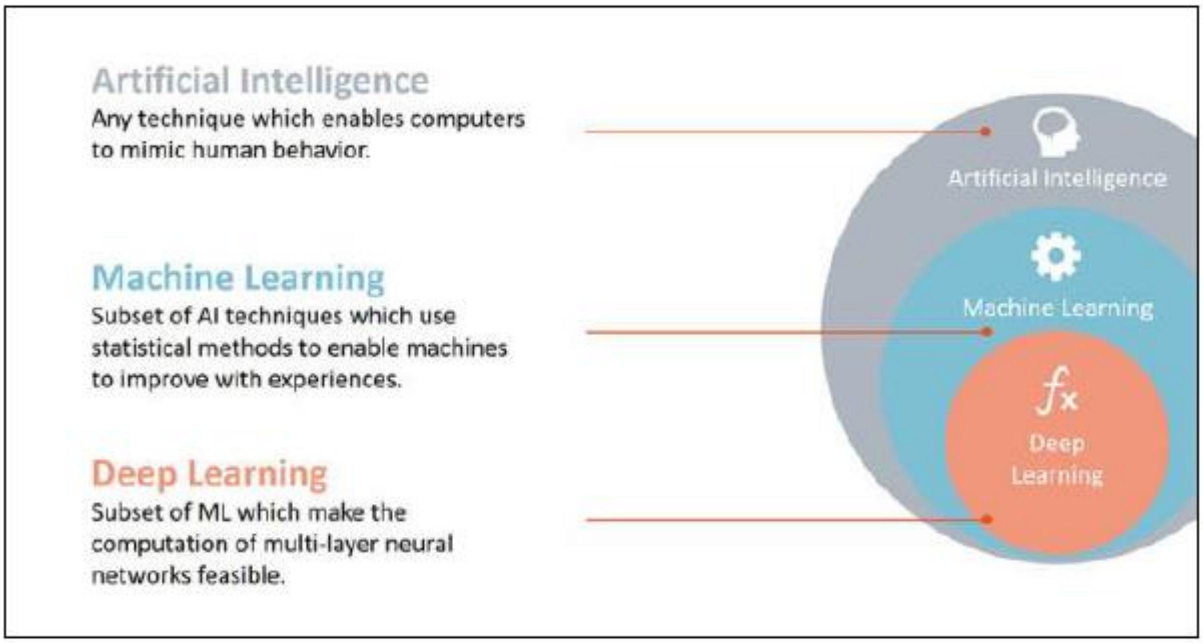
- Phone/__init__.py

To make all of your functions available when you've imported Phone, you need to put explicit import statements in __init__.py as follows −

```
from Pots import Pots
from Isdn import Isdn
from G3 import
```

ARTIFICIAL INTELLIGENCE

Introduction



Artificial Intelligence
Any technique which enables computers to mimic human behavior.
Machine Learning
Subset of AI techniques which use statistical methods to enable machines to improve with experiences.
Deep Learning
Subset of ML which make the computation of multi-layer neural networks feasible.
Artificial Intelligence
Machine Learning
f_x
Deep Learning

According to the father of Artificial Intelligence, John McCarthy, it is *"The science and engineering of making intelligent machines, especially intelligent computer programs".*

Artificial Intelligence is a way of **making a computer, a computer-controlled robot, or a software think intelligently**, in the similar manner the intelligent humans think.

AI is accomplished by studying how human brain thinks, and how humans learn, decide, and work while trying to solve a problem, and then using the outcomes of this study as a basis of developing intelligent software and systems.

The development of AI started with the intention of creating similar intelligence in machines that we find and regard high in humans.

Goals of AI

To Create Expert Systems − The systems which exhibit intelligent behaviour, learn, demonstrate, explain, and advice its users.
To Implement Human Intelligence in Machines − Creating systems that understand, think, learn, and behave like humans.

Applications of AI

AI has been dominant in various fields such as :-

Gaming − AI plays crucial role in strategic games such as chess, poker, tic-tac-toe, etc., where machine can think of large number of possible positions based on heuristic knowledge.

Natural Language Processing − It is possible to interact with the computer that understands natural language spoken by humans.

Expert Systems − There are some applications which integrate machine, software, and special information to impart reasoning and advising. They provide explanation and advice to the users.

Vision Systems − These systems understand, interpret, and comprehend visual input on the computer.

For example: A spying aeroplane takes photographs, which are used to figure out spatial information
or map of the areas.

Doctors use clinical expert system to diagnose the patient.

Police use computer software that can recognize the face of criminal with the stored
portrait made by forensic artist.

Speech Recognition − Some intelligent systems are capable of hearing and comprehending the language in terms of sentences and their meanings while a human talks to it. It can handle different accents, slang words, noise in the background, change in human's noise due to cold, etc.

Handwriting Recognition − The handwriting recognition software reads the text written on paper by a pen or on screen by a stylus. It can recognize the shapes of the letters and convert it into editable text.

Intelligent Robots − Robots are able to perform the tasks given by a human. They have sensors to detect physical data from the real world such as light, heat, temperature, movement, sound, bump, and pressure. They have efficient processors, multiple sensors and huge memory, to exhibit intelligence. In addition, they are capable of learning from their mistakes and they can adapt to the new environment.

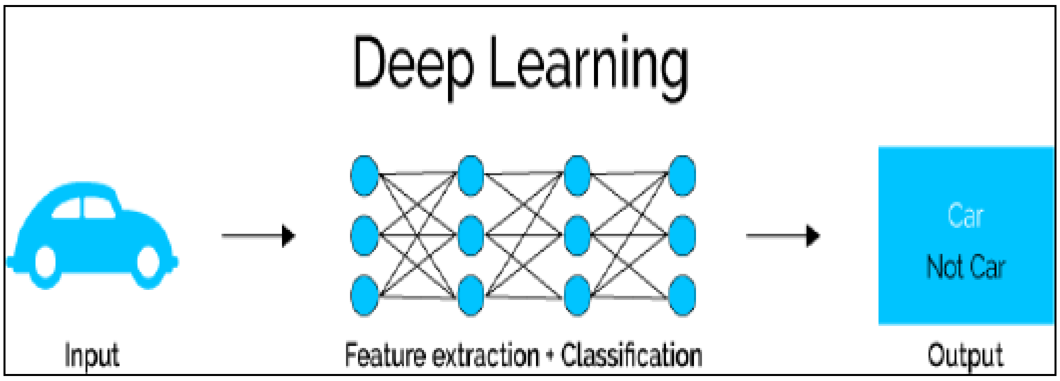
Application of AI

Deep Learning

Deep learning is a subset of machine learning. Usually, when people use the term deep learning, they are referring to deep artificial neural networks, and somewhat less frequently to deep reinforcement learning.

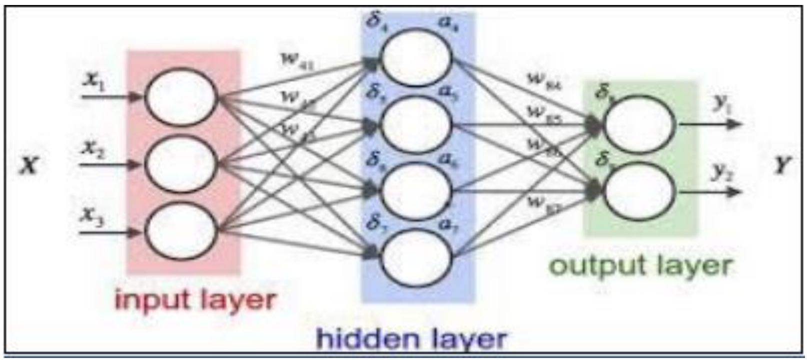
Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.
- use some form of gradient descent for training via backpropagation.



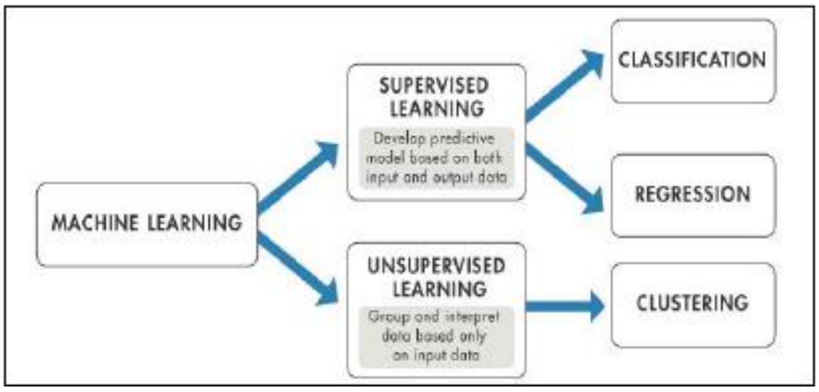
NEURAL NETWORKING

Artificial neural networks (**ANNs**) or **connectionist systems** are computing systems inspired by the biological neural networks that constitute animal brains. Such systems learn (progressively improve performance on) tasks by considering examples, generally without task-specific programming



An ANN is based on a collection of connected units or nodes called artificial neurons (analogous to biological neurons in an animal brain). Each connection between artificial neurons can transmit a signal from one to another.

MACHINE LEARNING


MACHINE LEARNING
SUPERVISED LEARNING
Develop predictive model based on both input and output data
UNSUPERVISED LEARNING
Group and interpret data based only on input data
CLASSIFICATION
REGRESSION
CLUSTERING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.
Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data.

COMPUTER VISION

Computer vision is an interdisciplinary field that deals with how computers can be made to gain high-level understanding from digital images or videos. From the perspective of engineering, it seeks to automate tasks that the human visual system can do. "Computer vision is concerned with the automatic extraction, analysis and understanding of useful information from a single image or a sequence of images. It involves the development of a theoretical and algorithmic basis to achieve automatic visual understanding." As a scientific discipline, computer vision is concerned with the theory behind artificial systems that extract information from images. The image data can take many forms, such as video sequences, views from multiple cameras, or multi-dimensional data from a medical scanner. As a technological discipline, computer vision seeks to apply its theories and models for the construction of computer vision systems.

What is computer vision?

Computer vision is a field of artificial intelligence (AI) that enables computers and systems to derive meaningful information from digital images, videos and other visual inputs — and take actions or make recommendations based on that information. If AI enables computers to think, computer vision enables them to see, observe and understand.

Computer vision works much the same as human vision, except humans have a head start. Human sight has the advantage of lifetimes of context to train how to tell objects apart, how far away they are, whether they are moving and whether there is something wrong in an image.

Computer vision trains machines to perform these functions, but it has to do it in much less time with cameras, data and algorithms rather than retinas, optic nerves and a visual cortex. Because a system trained to inspect products or watch a production asset can analyze thousands of products or processes a minute, noticing imperceptible defects or issues, it can quickly surpass human capabilities.

Computer vision is used in industries ranging from energy and utilities to manufacturing and automotive – and the market is continuing to grow. It is expected to reach USD 48.6 billion by 2022

How does computer vision work?

Computer vision needs lots of data. It runs analyses of data over and over until it discerns distinctions and ultimately recognize images. For example, to train a computer to recognize automobile tires, it needs to be fed vast quantities of tire images and tire-related items to learn the differences and recognize a tire, especially one with no defects.

Two essential technologies are used to accomplish this: a type of machine learning called deep learning and a convolutional neural network (CNN).

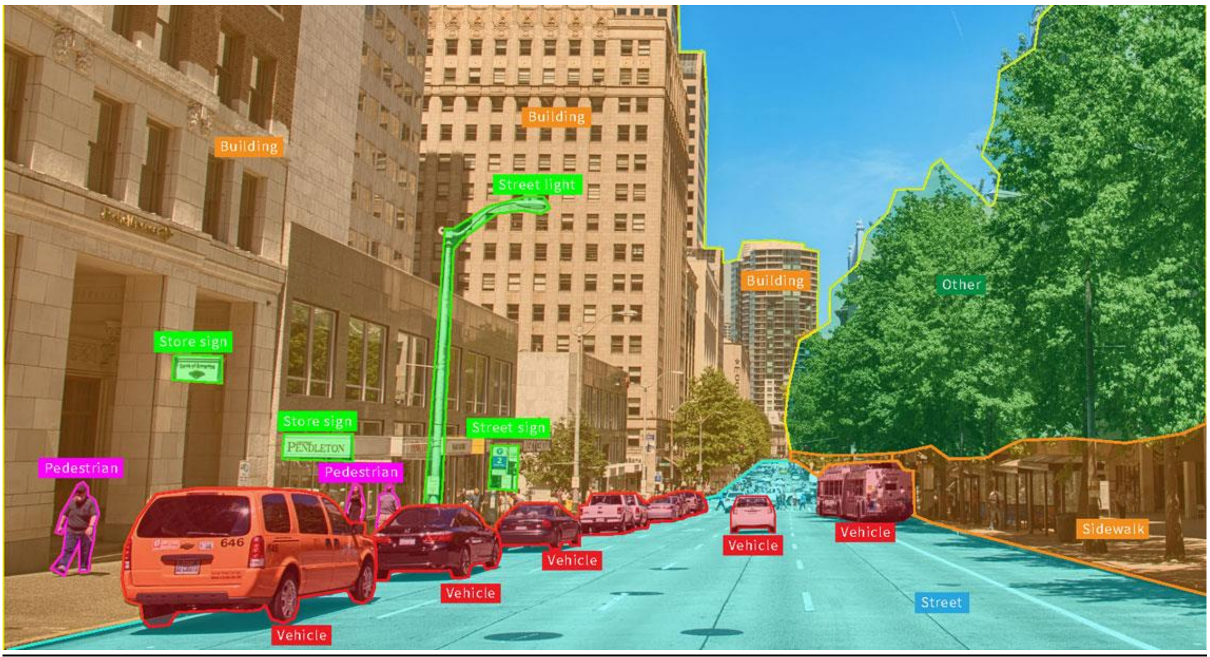
Machine learning uses algorithmic models that enable a computer to teach itself about the context of visual data. If enough data is fed through the model, the computer will “look” at the data and teach itself to tell one image from another. Algorithms enable the machine to learn by itself, rather than someone programming it to recognize an image.

A CNN helps a machine learning or deep learning model “look” by breaking images down into pixels that are given tags or labels. It uses the labels to perform convolutions (a mathematical operation on two functions to produce a third function) and makes predictions about what it is “seeing.” The neural network runs convolutions and checks the accuracy of its predictions in a series of iterations until the predictions start to come true. It is then recognizing or seeing images in a way similar to humans.

Much like a human making out an image at a distance, a CNN first discerns hard edges and simple shapes, then fills in information as it runs iterations of its predictions. A CNN is used to understand single images. A recurrent neural network (RNN) is used in a similar way for video applications to help computers understand how pictures in a series of frames are related to one another.

Application of Computer Vision:

- **Image classification** sees an image and can classify it (a dog, an apple, a person's face). More precisely, it is able to accurately predict that a given image belongs to a certain class. For example, a social media company might want to use it to automatically identify and segregate objectionable images uploaded by users.
- **Object detection** can use image classification to identify a certain class of image and then detect and tabulate their appearance in an image or video. Examples include detecting damages on an assembly line or identifying machinery that requires maintenance.
- **Object tracking** follows or tracks an object once it is detected. This task is often executed with images captured in sequence or real-time video feeds. Autonomous vehicles, for example, need to not only classify and detect objects such as pedestrians, other cars and road infrastructure, they need to track them in motion to avoid collisions and obey traffic laws.
- **Content-based image retrieval** uses computer vision to browse, search and retrieve images from large data stores, based on the content of the images rather than metadata tags associated with them. This task can incorporate automatic image annotation that replaces manual image tagging. These tasks can be used for digital asset management systems and can increase the accuracy of search and retrieval.



INTRODUCTION TO MACHINE LEARNING

Machine learning is a field of computer science that gives computers the ability to learn without being explicitly programmed.

Arthur Samuel, an American pioneer in the field of computer gaming and artificial intelligence, coined the term "Machine Learning" in 1959 while at IBM. Evolved from the study of pattern recognition and computational learning theory in artificial intelligence, machine learning explores the study and construction of algorithms that can learn from and make predictions on data

Machine learning tasks are typically classified into two broad categories, depending on whether there is a learning "signal" or "feedback" available to a learning system:-

SUPERVISED LEARNING

Supervised learning is the machine learning task of inferring a function from *labeled training data*.[1] The training data consist of a set of *training examples*. In supervised learning, each example is a *pair* consisting of an input object (typically a vector) and a desired output value.

A supervised learning algorithm analyses the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

UNSUPERVISED LEARNING

Unsupervised learning is the machine learning task of inferring a function to describe hidden structure from "unlabelled" data (a classification or categorization is not included in the observations). Since the examples given to the learner are unlabelled, there is no evaluation of the accuracy of the structure that is output by the relevant algorithm—which is one way of distinguishing unsupervised learning from supervised learning and reinforcement learning.

A central case of unsupervised learning is the problem of density estimation in statistics, though unsupervised learning encompasses many other problems (and solutions) involving summarizing and explaining key features of the data.

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 high-level mathematical functions to operate on these arrays. The ancestor of NumPy, Numeric, was originally created by Jim Hugunin.

NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms written for this version of Python often run much slower than compiled equivalents.

Using NumPy in Python gives functionality comparable to MATLAB since they are both interpreted, and they both allow the user to write fast programs as long as most operations work on arrays or matrices instead of scalars.

NUMPY ARRAY

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called *axes*. The number of axes is *rank*.

For example, the coordinates of a point in 3D space [1, 2, 1] is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

```
[[ 1., 0., 0.],
 [ 0., 1., 2.]]
```

NumPy’s array class is called *ndarray*. It is also known by the alias.

SLICING NUMPY ARRAY

import numpy as np

a = np.array([[1,2,3],[3,4,5],[4,5,6]])

```
print 'Our array is:'
print a
print '\n'


print 'The items in the second column are:'
print a[...,1]
print '\n'


print 'The items in the second row are:'
print a[1,...]
print '\n'


print 'The items column 1 onwards are:'
print a[...,1:]
```

OUTPUT

```
Our array is:
[[1 2 3]
[3 4 5]
[4 5 6]]

The items in the second column are:
[2 4 5]

The items in the second row are:
[3 4 5]

The items column 1 onwards are:
[[2 3]
[4 5]
[5 6]]
```

SCIPY

modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

SciPy builds on the NumPy array object and is part of the NumPy stack which includes tools like Matplotlib, pandas and SymPy, and an expanding set of scientific computing libraries. This NumPy stack has similar users to other applications such as MATLAB, GNU Octave, and Scilab. The NumPy stack is also sometimes referred to as the SciPy stack.

The SciPy Library/Package

The SciPy package of key algorithms and functions core to Python's scientific computing capabilities. Available sub-packages include:

- **constants:** physical constants and conversion factors (since version 0.7.0)
- **cluster:** hierarchical clustering, vector quantization, K-means
- **fftpack:** Discrete Fourier Transform algorithms
- **integrate:** numerical integration routines
- **interpolate:** interpolation tools
- **io:** data input and output
- **lib:** Python wrappers to external libraries
- **linalg:** linear algebra routines
- **misc:** miscellaneous utilities (e.g. image reading/writing)
- **ndimage:** various functions for multi-dimensional image processing
- **optimize:** optimization algorithms including linear programming
- **signal:** signal processing tools
- **sparse:** sparse matrix and related algorithms
- **spatial:** KD-trees, nearest neighbours, distance functions
- **special:** special functions
- **stats:** statistical functions
- **weave:** tool for writing C/C++ code as Python multiline strings

Data Structures

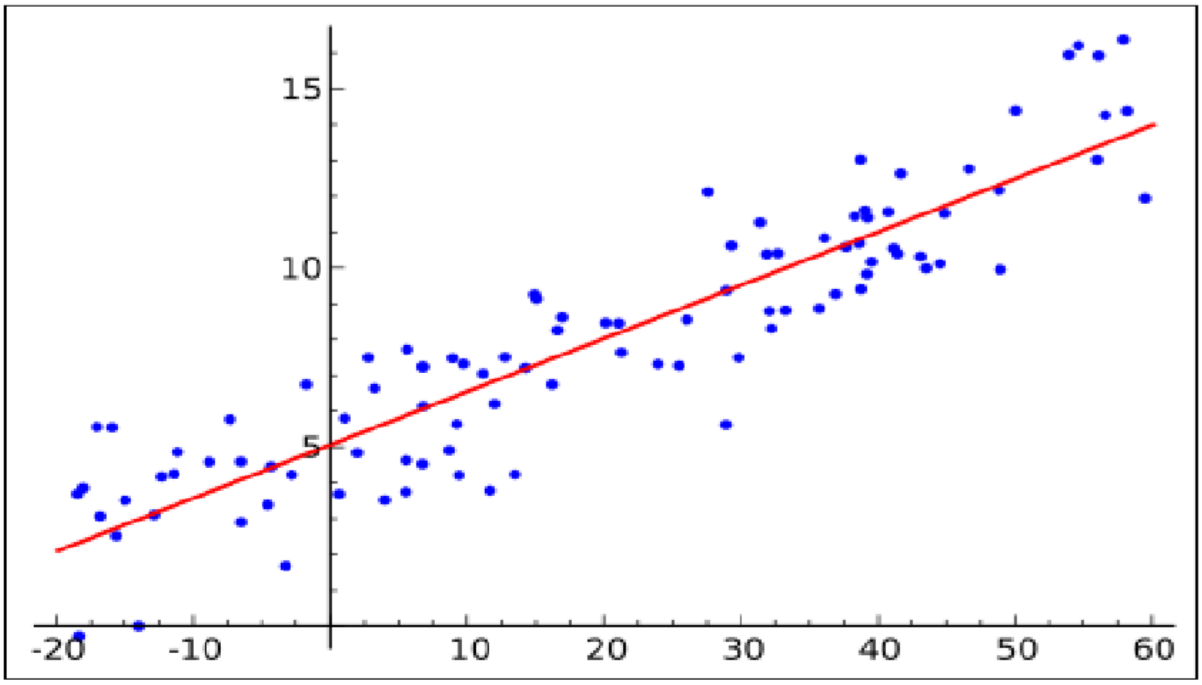
The basic data structure used by SciPy is a multidimensional array provided by the NumPy module. NumPy provides some functions for linear algebra, Fourier transforms and random number generation, but not with the generality of the equivalent functions in SciPy. NumPy can also be used as an efficient multi-dimensional container of data with arbitrary data-types. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. Older versions of SciPy used Numeric as an array type, which is now deprecated in favour of the newer NumPy array code.

SCIKIT-LEARN

Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

The scikit-learn project started as scikits.learn, a Google Summer of Code project by David Cournapeau. Its name stems from the notion that it is a "SciKit" (SciPy Toolkit), a separately-developed and distributed third-party extension to SciPy.[4] The original codebase was later rewritten by other developers. In 2010 Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort and Vincent Michel, all from INRIA took leadership of the project and made the first public release on February the 1st 2010[5]. Of the various scikits, scikit-learn as well as scikit-image were described as "well-maintained and popular" in November 2012.

REGRESSION ANALYSIS



In statistical modelling, **regression analysis** is a set of statistical processes for estimating the relationships among variables. It includes many techniques for modelling and analysing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables (or 'predictors'). More specifically, regression analysis helps one understand how the typical value of the dependent variable (or 'criterion variable') changes when any one of the independent variables is varied, while the other independent variables are held fixed.

Regression analysis is widely used for prediction and forecasting, where its use has substantial overlap with the field of machine learning. Regression analysis is also used to understand which among the independent variables are related to the dependent variable, and to explore the forms of these relationships. In restricted circumstances, regression analysis can be used to infer casual relationships between the independent and dependent variables. However this can lead to illusions or false relationships, so caution is advisable

LINEAR REGRESSION

Linear regression is a linear approach for modelling the relationship between a scalar dependent variable y and one or more explanatory variables (or independent variables) denoted X. The case of one explanatory variable is called *simple linear regression*. For more than one explanatory variable, the process is called *multiple linear regression*.

In linear regression, the relationships are modelled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called *linear models*.

LOGISTIC REGRESSION

Logistic regression, or logit regression, or logit model [1] is a regression model where the dependent variable (DV) is categorical. This article covers the case of a binary dependent variable—that is, where the output can take only two values, "0" and "1", which represent outcomes such as pass/fail, win/lose, alive/dead or healthy/sick. Cases where the dependent variable has more than two outcome categories may be analysed in multinomial logistic regression, or, if the multiple categories are ordered, in ordinal logistic regression. In the terminology of economics, logistic regression is an example of a qualitative response/discrete choice model.

POLYNOMIAL REGRESSION

Polynomial regression is a form of regression analysis in which the relationship between the independent variable x and the dependent variable y is modelled as an n^{th} degree polynomial in x.

Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y , denoted E(y | x), and has been used to describe nonlinear phenomena such as the growth rate of tissues, the distribution of carbon isotopes in lake sediments, and the progression of disease epidemics.

Although *polynomial regression* fits a nonlinear model to the data, as a statistical estimation problem it is linear, in the sense that the regression function E(y | x) is linear in the unknown parameters that are estimated from the data.

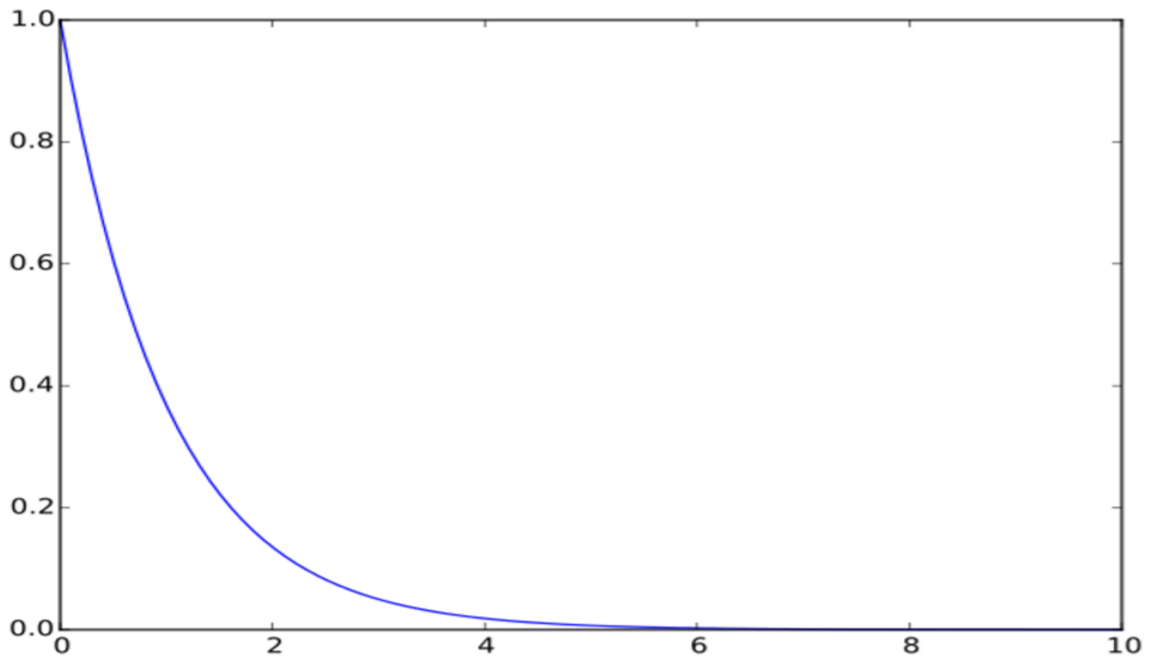
MATPLOTLIB

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter,wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged .SciPy makes use of matplotlib.

EXAMPLE

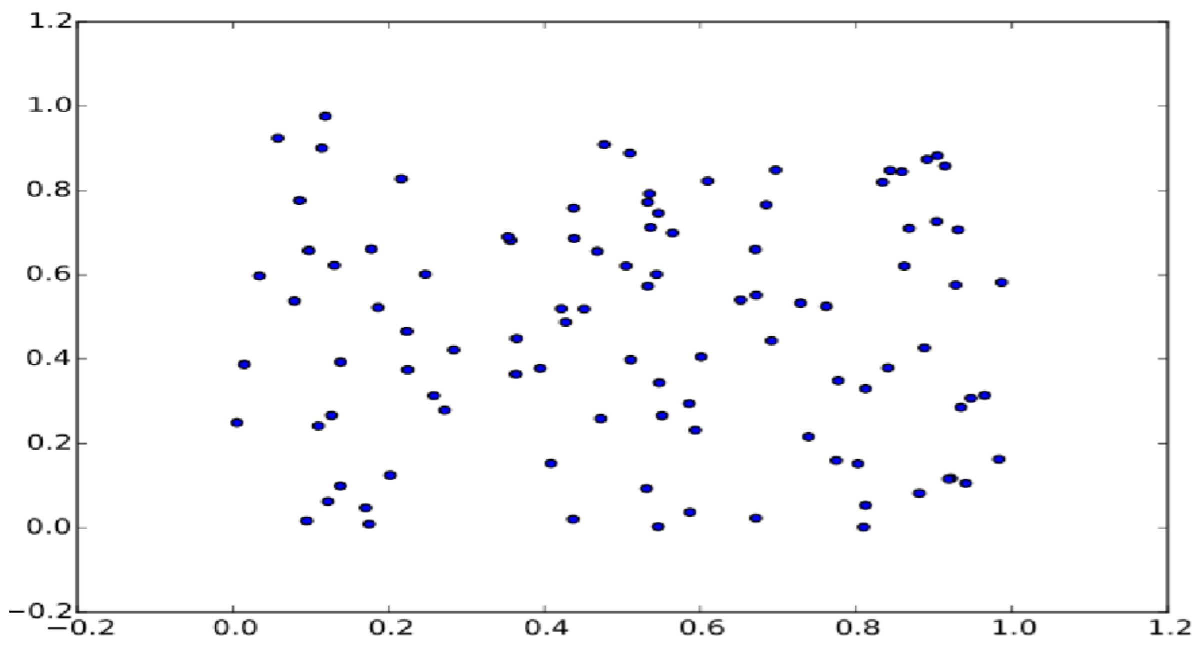
- **LINE PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>importnumpyasnp
>>> a =np.linspace(0,10,100)
>>> b =np.exp(-a)
>>>plt.plot(a,b)
>>>plt.show()
```



➢ **SCATTER PLOT**

```
>>>importmatplotlib.pyplotasplt
>>>fromnumpy.randomimport rand
>>> a =rand(100)
>>> b =rand(100)
>>>plt.scatter(a,b)
>>>plt.show()
```



PANDAS

In computer programming, **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. "Panel data", an econometrics term for multidimensional, structured data sets.

LIBRARY FEATURES

- Data Frame object for data manipulation with integrated indexing.
- Tools for reading and writing data between in-memory data structures and different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of data sets.
- Label-based slicing, fancy indexing, and sub setting of large data sets.
- Data structure column insertion and deletion.
- Group by engine allowing split-apply-combine operations on data sets.
- Data set merging and joining.
- Hierarchical axis indexing to work with high-dimensional data in a lower-dimensional data structure.
- Time series-functionality: Date range generation.

CLUSTERING

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a **cluster**) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis, used in many fields, including machine learning, pattern recognition, image analysis, information retrieval, bioinformatics, data compression, and computer graphics.

Cluster analysis itself is not one specific algorithm, but the general task to be solved. It can be achieved by various algorithms that differ significantly in their notion of what constitutes a cluster and how to efficiently find them. Popular notions of clusters include groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. Clustering can therefore be formulated as a multi-objective optimization problem.

The appropriate clustering algorithm and parameter settings (including values such as the distance function to use, a density threshold or the number of expected clusters) depend on the individual data set and intended use of the results. Cluster analysis as such is not an automatic task, but an iterative process of knowledge discovery or interactive multi-objective optimization that involves trial and failure. It is often necessary to modify data pre-processing and model parameters until the result achieves the desired properties.

ALGORITHM

- Data Collection
- Data Formatting
- Model Selection
- Training
- Testing

Data Collection: I have collected data sets of weather from online website. I have downloaded the .csv files in which information was present.

Data Formatting: The collected data is formatted into suitable data sets. I check the collinearity with mean temperature. The data sets which have collinearity nearer to 1.0 has been selected.

Model Selection: I have selected different models to minimize the error of the predicted value. The different models used are Linear Regression Linear Model.

Training: The data sets was divided such that x_train is used to train the model with corresponding x_test values and some y_train kept reserved for testing.

Testing: The model was tested with y_train and stored in y_predict . Both y_train and y_predict was compared.

PROJECT DETAILS: VIRTUAL MOUSE USING AI & COMPUTER VISION

Abstract:

Hand Movement Identification plays a vital part in a human–machine interconnection and to interact with a computer in a most effortless way. As many modern improvements occurring in today’s world, such as Natural language processing, Bio-metric Authentication, Face detection, etc., which can be frequently seen in our Tablets, iPads, Computers and smart phones. In the same way, Hand Movement Identification was a contemporary method of Human–Machine Interconnection, that is, the mouse cursor of the system can be controlled just on appearing our figure’s before the computer’s web camera. These finger gestures are captured and controlled through a Colour Detection technique of webcam. This system allows us to direct the system pointer by using our finger bearing colour caps or tapes and the operations like dragging of files and the left click would be performed by using distinct finger gestures. It also performs the transfer of files among two PC's in a single similar network. This developed system makes use of only a less resolution webcam which acts as a sensor for tracking the user’s hands in two dimensions. This system would be developed by using a Library named Open Source Computer Vision (OpenCV) and Python Server Programming. Based upon the idea of virtual mouse implementation, the paper was introduced. In the paper, we will provide a complete description of all the methodologies along with the libraries and packages that are used for the implementation of a Virtual Mouse.

Modules Used:

1. **Open CV:**
   OpenCV (Open Source Computer Vision Library) is an open-source library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposed to the C-based OpenCV 1.x API (C API is deprecated and not tested with "C" compiler since OpenCV 2.4 releases)

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

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

- **High-level GUI** (**highgui**) - an easy-to-use interface to simple UI capabilities.
- **Video I/O** (**videoio**) - an easy-to-use interface to video capturing and video codecs.

2. Mediapipe:

MediaPipe is a cross-platform framework for building multimodal applied machine learning pipelines

MediaPipe is a framework for building multimodal (eg. video, audio, any time series data), cross platform (i.e Android, iOS, web, edge devices) applied ML pipelines. With MediaPipe, a perception pipeline can be built as a graph of modular components, including, for instance, inference models (e.g., TensorFlow, TFLite) and media processing functions.

Cutting edge ML models

- Face Detection
- Multi-hand Tracking
- Hair Segmentation
- Object Detection and Tracking
- Objectron: 3D Object Detection and Tracking
- AutoFlip: Automatic video cropping pipeline

Cross Platform ML solutions

Build once, deploy anywhere. Works optimally across mobile (iOS, Android), desktop server and the Web

Ondevice ML Acceleration

Performance optimized end-to-end ondevice inference with ML acceleration for mobile GPU & EdgeTPU compute

3. Numpy:

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance. There are several important differences between NumPy arrays and the standard Python sequences:

- NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
- The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
- NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python's built-in sequences.
- A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today's scientific/mathematical Python-based software, just knowing how to use Python's built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

The points about sequence size and speed are particularly important in scientific computing. As a simple example, consider the case of multiplying each element in a 1-D sequence with the corresponding element in another sequence of the same length. If the data are stored in two Python lists, `a` and `b`, we could iterate over each element:

```
c = []
for i in range(len(a)):
    c.append(a[i]*b[i])
```

This produces the correct answer, but if `a` and `b` each contain millions of numbers, we will pay the price for the inefficiencies of looping in Python. We could accomplish the same task much more quickly in C by writing (for clarity we neglect variable declarations and initializations, memory allocation, etc.)

```
for (i = 0; i < rows; i++): {
  c[i] = a[i]*b[i];
}
```

This saves all the overhead involved in interpreting the Python code and manipulating Python objects, but at the expense of the benefits gained from coding in Python. Furthermore, the coding work required increases with the dimensionality of our data. In the case of a 2-D array, for example, the C code (abridged as before) expands to

```
for (i = 0; i < rows; i++): {
  for (j = 0; j < columns; j++): {
    c[i][j] = a[i][j]*b[i][j];
  }
}
```

NumPy gives us the best of both worlds: element-by-element operations are the “default mode” when an *ndarray* is involved, but the element-by-element operation is speedily executed by pre-compiled C code. In NumPy

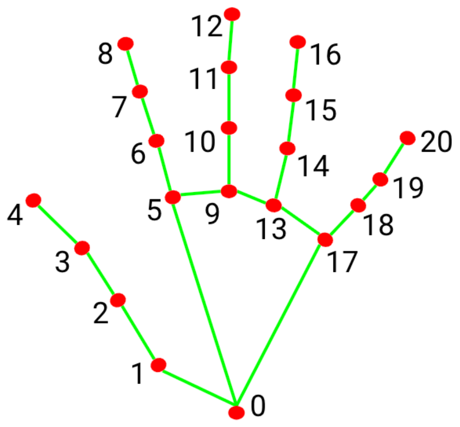
```
c = a * b
```

does what the earlier examples do, at near-C speeds, but with the code simplicity we expect from something based on Python. Indeed, the NumPy idiom is even simpler! This last example illustrates two of NumPy’s features which are the basis of much of its power: vectorization and broadcasting.

PROJECT DETAILS: VIRTUAL MOUSE USING AI & COMPUTER VISION

We have used OpenCV to capture or process images and videos to identify objects and mediapipe module to detect hand pipeline as well as landmarks.

We have used the hands module of this library to create 2 cool projects. The hands module creates a 21 points based localization of your hand. What this means is that if you supply a hand image to this module, it will return a 21 point vector showing the coordinates of 21 important landmarks present on your hand.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

Before using the mediapipe library in Python, you have to do a:

```
pip install mediapipe
```

Let’s create a utility class called HandDetector which will make our project modularized

1. Import the needed packages

```
import mediapipe as mp
import cv2
```

2. Create an instance of the hands module provided by Mediapipe followed by an instance of Mediapipe’s drawing utility. The drawing utility helps to draw those 21 landmarks and the lines connecting those landmarks on your image or frame(which you have noticed in the above video).

This step is pretty much constant and would have to be done in each of your projects.

```
mpHands = mp.solutions.hands
mpDraw = mp.solutions.drawing_utils
```

3. We then start writing our class

```
class HandDetector:
   def __init__(self, max_num_hands=2, min_detection_confidence=0.5,
min_tracking_confidence=0.5):
       self.hands = mpHands.Hands(max_num_hands=max_num_hands,
min_detection_confidence=min_detection_confidence,
                                   min_tracking_confidence=min_tracking_confidence)
```

max_num_hands: the number of hands that you want Mediapipe to detect. Mediapipe will return an array of hands and each element of the array(or a hand) would in turn have its 21 landmark points

min_detection_confidence, *min_tracking_confidence*: when the Mediapipe is first started, it detects the hands. After that, it tries to track the hands as detecting is more time-consuming than tracking. If the tracking confidence goes down the specified value then again it switches back to detection.

All these parameters are needed by the Hands() class, so we pass them to the Hands class in the next line.

4. Next we define the function findHandLandMarks() of this class *where the main stuff happens*

```
def findHandLandMarks(self, image, handNumber=0, draw=False):
   originalImage = image
   image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # mediapipe needs RGB
   results = self.hands.process(image)
   landMarkList = []
```

```
  if results.multi_hand_landmarks:  # returns None if hand is not found
     hand = results.multi_hand_landmarks[handNumber]
#results.multi_hand_landmarks returns landMarks for all the hands

     for id, landMark in enumerate(hand.landmark):
        # landMark holds x,y,z ratios of single landmark
        imgH, imgW, imgC = originalImage.shape  # height, width, channel for image
        xPos, yPos = int(landMark.x * imgW), int(landMark.y * imgH)
        landMarkList.append([id, xPos, yPos])

     if draw:
        mpDraw.draw_landmarks(originalImage, hand,
mpHands.HAND_CONNECTIONS)

  return landMarkList
```

The function arguments:

the *image* on which hand landmarks would be detected. The *handNumber* in case the image has multiple hands, so our function would return landmarks for only the specified hand number. A boolean parameter *draw* decides if we want the medapipe to draw those landmarks on our image.

The next line does everything. This small line actually is doing a lot behind the scenes and is getting all the landmarks for you

```
results = self.hands.process(image)
```

We then create an empty list *landMarkList* which would contain the final result returned from the function.

The *results.multi_hand_landmarks* returns None if there is no hand detected, so you should use it as a fail-safe condition.

results.multi_hand_landmarks returns landMarks for all the hands that were detected, so passing the *handNumber* to it gives you data for the correct hand.

hand.landmark gives the 21 landmarks for the selected hand. So we iterate over these 21 points where *id* holds the id for each of the landmark

Now the important point to note here is that the landmark information returned by Mediapipe is not the pixel location of the landmark. Instead, it is the ratio of the image dimensions. So to get the exact x and y coordinate of the pixel of the landmark we do this simple calculation:

```
xPos, yPos = int(landMark.x * imgW), int(landMark.y * imgH)
landMarkList.append([id, xPos, yPos])
```

We then append the id(0…21) of the landmark, and the corresponding x and y coordinate to the empty list which we had earlier created. We return this list to the calling function.

This is what the findHandLandMarks() would return:

0th index →id of landmark, 1st index →x coordinate of landmark, 2nd index →x coordinate of landmark

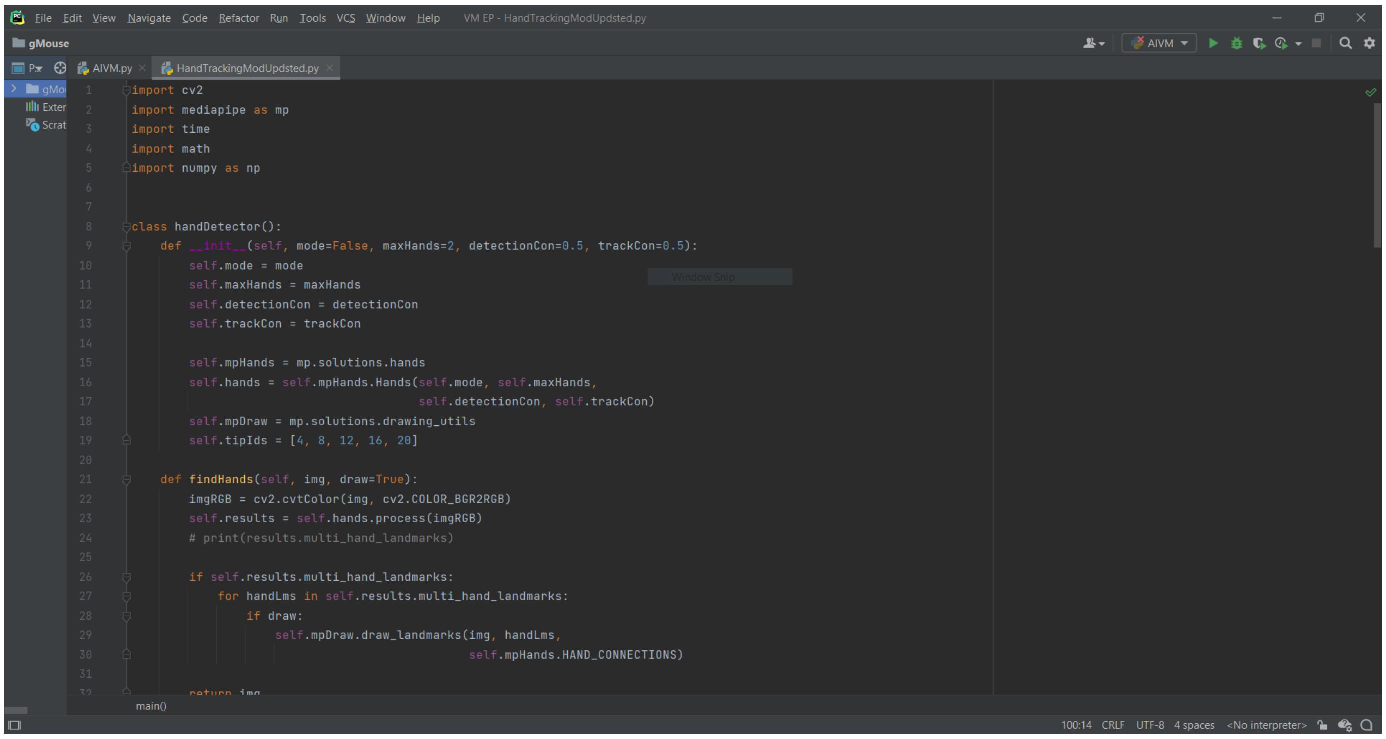
The last part draws the landmarks on the image if the boolean variable *draw* says so

```
mpDraw.draw_landmarks(originalImage, hand, mpHands.HAND_CONNECTIONS)
```

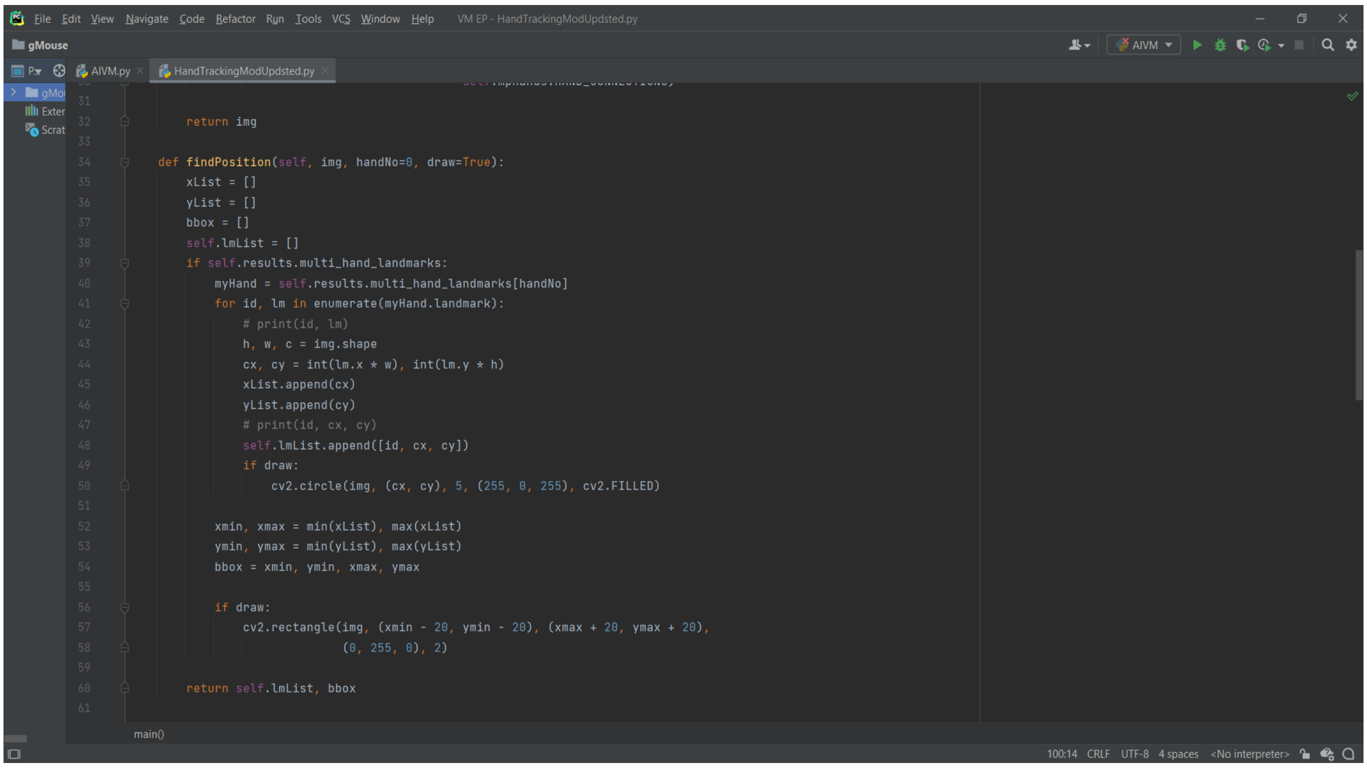
That's pretty much from our custom class. So all you need to do is pass your hand image to findHandLandMarks() and you will get the list containing information about all 21 landmarks.

Screenshots :

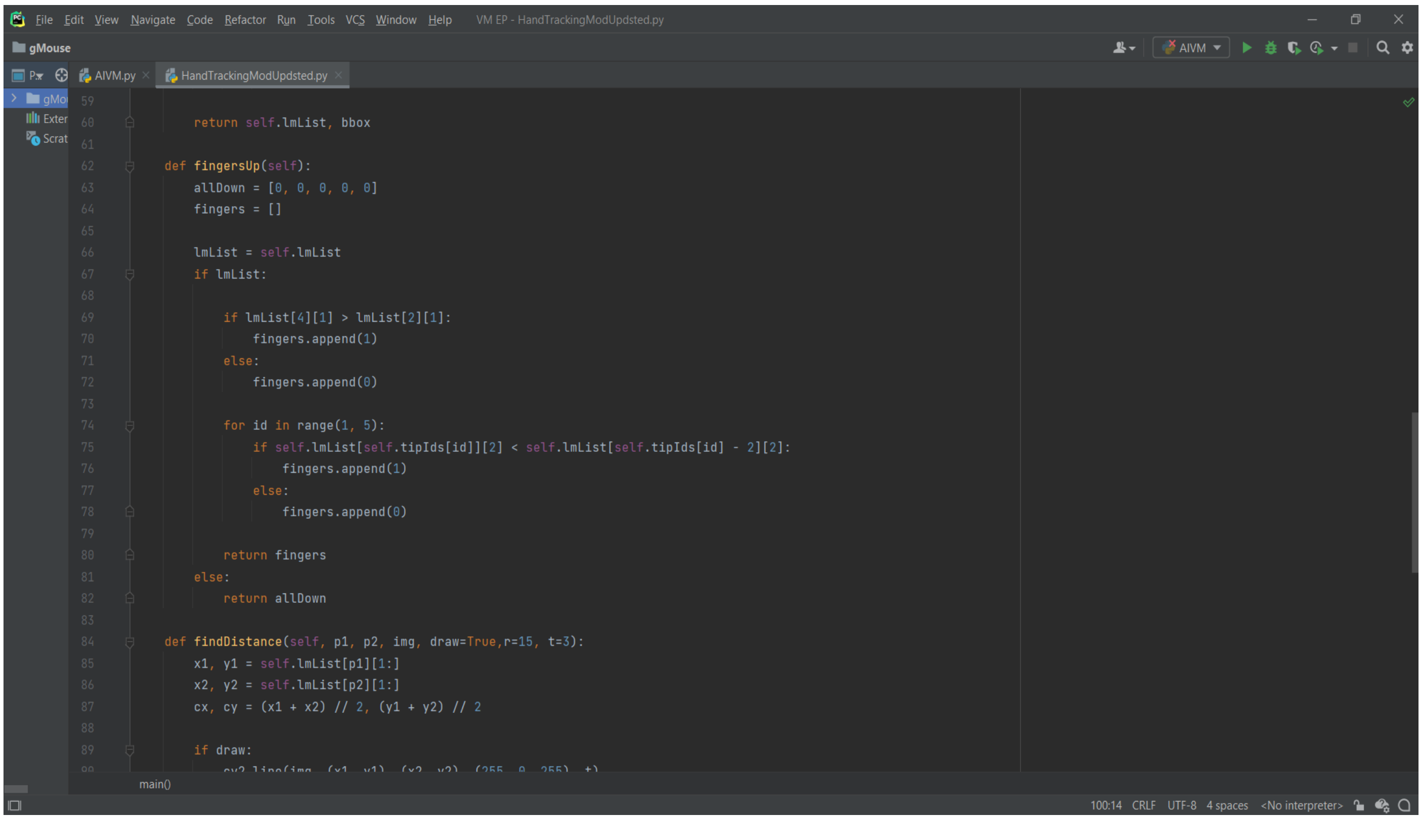
Hand Tracking Module:


File Edit View Navigate Code Refactor Run Tools VCS Window Help
VM EP - HandTrackingModUpdsted.py
gMouse
AIVM.py
HandTrackingModUpdsted.py
AIVM
main()
100:14 CRLF UTF-8 4 spaces <No interpreter>

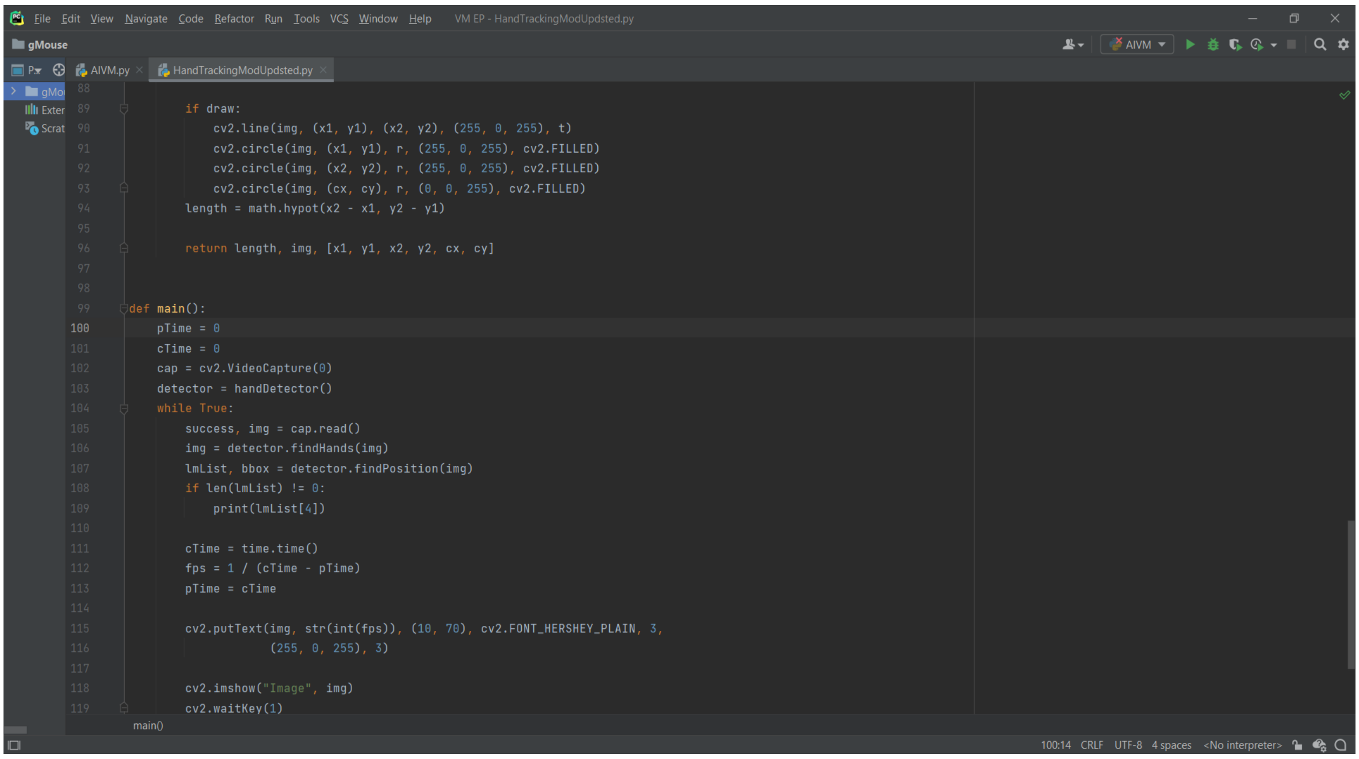
```python
import cv2
import mediapipe as mp
import time
import math
import numpy as np


class handDetector():
    def __init__(self, mode=False, maxHands=2, detectionCon=0.5, trackCon=0.5):
        self.mode = mode
        self.maxHands = maxHands
        self.detectionCon = detectionCon
        self.trackCon = trackCon

        self.mpHands = mp.solutions.hands
        self.hands = self.mpHands.Hands(self.mode, self.maxHands,
                                        self.detectionCon, self.trackCon)
        self.mpDraw = mp.solutions.drawing_utils
        self.tipIds = [4, 8, 12, 16, 20]

    def findHands(self, img, draw=True):
        imgRGB = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
        self.results = self.hands.process(imgRGB)
        # print(results.multi_hand_landmarks)

        if self.results.multi_hand_landmarks:
            for handLms in self.results.multi_hand_landmarks:
                if draw:
                    self.mpDraw.draw_landmarks(img, handLms,
                                               self.mpHands.HAND_CONNECTIONS)

        return img
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
VM EP - HandTrackingModUpdsted.py
gMouse
AIVM.py
HandTrackingModUpdsted.py
AIVM
main()
100:14 CRLF UTF-8 4 spaces <No interpreter>

```python

        return img

    def findPosition(self, img, handNo=0, draw=True):
        xList = []
        yList = []
        bbox = []
        self.lmList = []
        if self.results.multi_hand_landmarks:
            myHand = self.results.multi_hand_landmarks[handNo]
            for id, lm in enumerate(myHand.landmark):
                # print(id, lm)
                h, w, c = img.shape
                cx, cy = int(lm.x * w), int(lm.y * h)
                xList.append(cx)
                yList.append(cy)
                # print(id, cx, cy)
                self.lmList.append([id, cx, cy])
                if draw:
                    cv2.circle(img, (cx, cy), 5, (255, 0, 255), cv2.FILLED)

            xmin, xmax = min(xList), max(xList)
            ymin, ymax = min(yList), max(yList)
            bbox = xmin, ymin, xmax, ymax

            if draw:
                cv2.rectangle(img, (xmin - 20, ymin - 20), (xmax + 20, ymax + 20),
                              (0, 255, 0), 2)

        return self.lmList, bbox

```

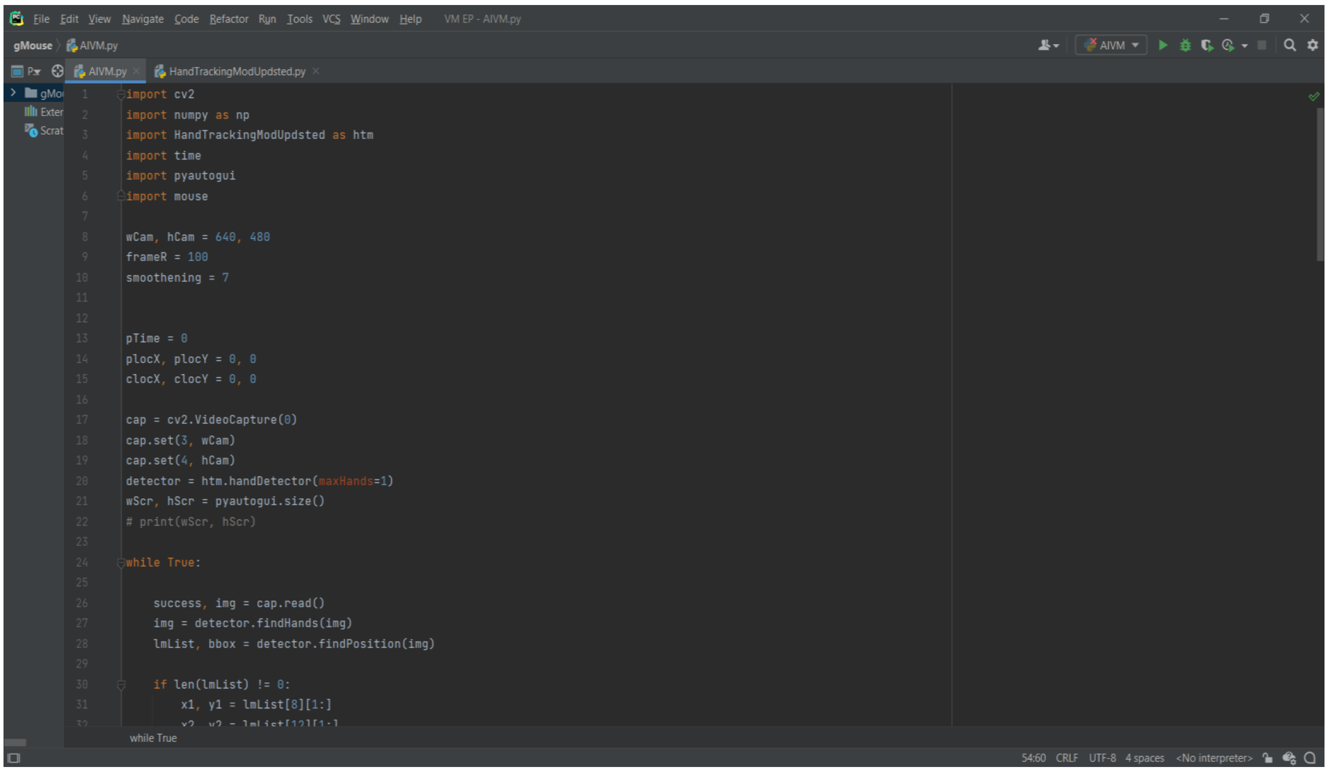

File Edit View Navigate Code Refactor Run Tools VCS Window Help VM EP - HandTrackingModUpdsted.py
gMouse
AIVM.py
HandTrackingModUpdsted.py
main()
100:14 CRLF UTF-8 4 spaces <No interpreter>

```python
        return self.lmList, bbox

    def fingersUp(self):
        allDown = [0, 0, 0, 0, 0]
        fingers = []

        lmList = self.lmList
        if lmList:

            if lmList[4][1] > lmList[2][1]:
                fingers.append(1)
            else:
                fingers.append(0)

            for id in range(1, 5):
                if self.lmList[self.tipIds[id]][2] < self.lmList[self.tipIds[id] - 2][2]:
                    fingers.append(1)
                else:
                    fingers.append(0)

            return fingers
        else:
            return allDown

    def findDistance(self, p1, p2, img, draw=True,r=15, t=3):
        x1, y1 = self.lmList[p1][1:]
        x2, y2 = self.lmList[p2][1:]
        cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

        if draw:
```

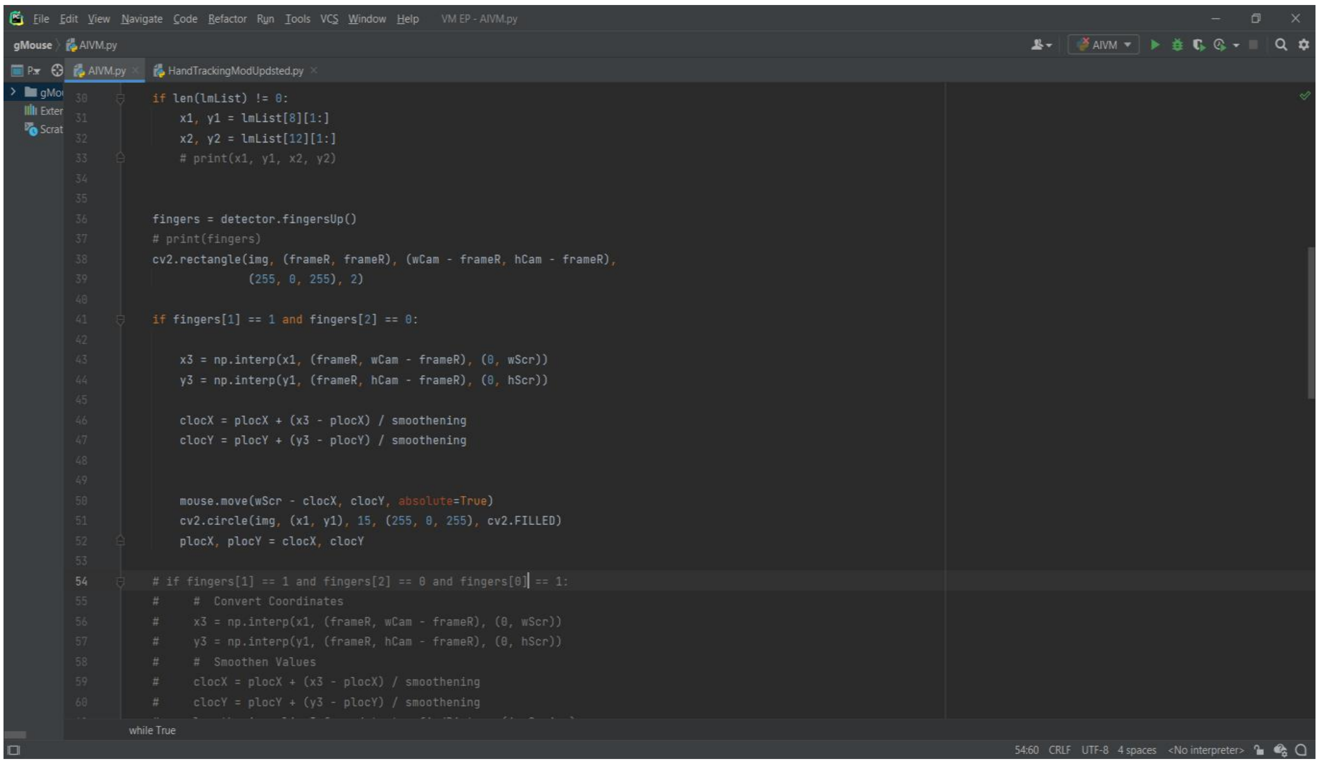

File Edit View Navigate Code Refactor Run Tools VCS Window Help VM EP - HandTrackingModUpdsted.py
gMouse
AIVM.py
HandTrackingModUpdsted.py
main()
100:14 CRLF UTF-8 4 spaces <No interpreter>

```python
        if draw:
            cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), t)
            cv2.circle(img, (x1, y1), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (x2, y2), r, (255, 0, 255), cv2.FILLED)
            cv2.circle(img, (cx, cy), r, (0, 0, 255), cv2.FILLED)
        length = math.hypot(x2 - x1, y2 - y1)

        return length, img, [x1, y1, x2, y2, cx, cy]


def main():
    pTime = 0
    cTime = 0
    cap = cv2.VideoCapture(0)
    detector = handDetector()
    while True:
        success, img = cap.read()
        img = detector.findHands(img)
        lmList, bbox = detector.findPosition(img)
        if len(lmList) != 0:
            print(lmList[4])

        cTime = time.time()
        fps = 1 / (cTime - pTime)
        pTime = cTime

        cv2.putText(img, str(int(fps)), (10, 70), cv2.FONT_HERSHEY_PLAIN, 3,
                    (255, 0, 255), 3)

        cv2.imshow("Image", img)
        cv2.waitKey(1)
```

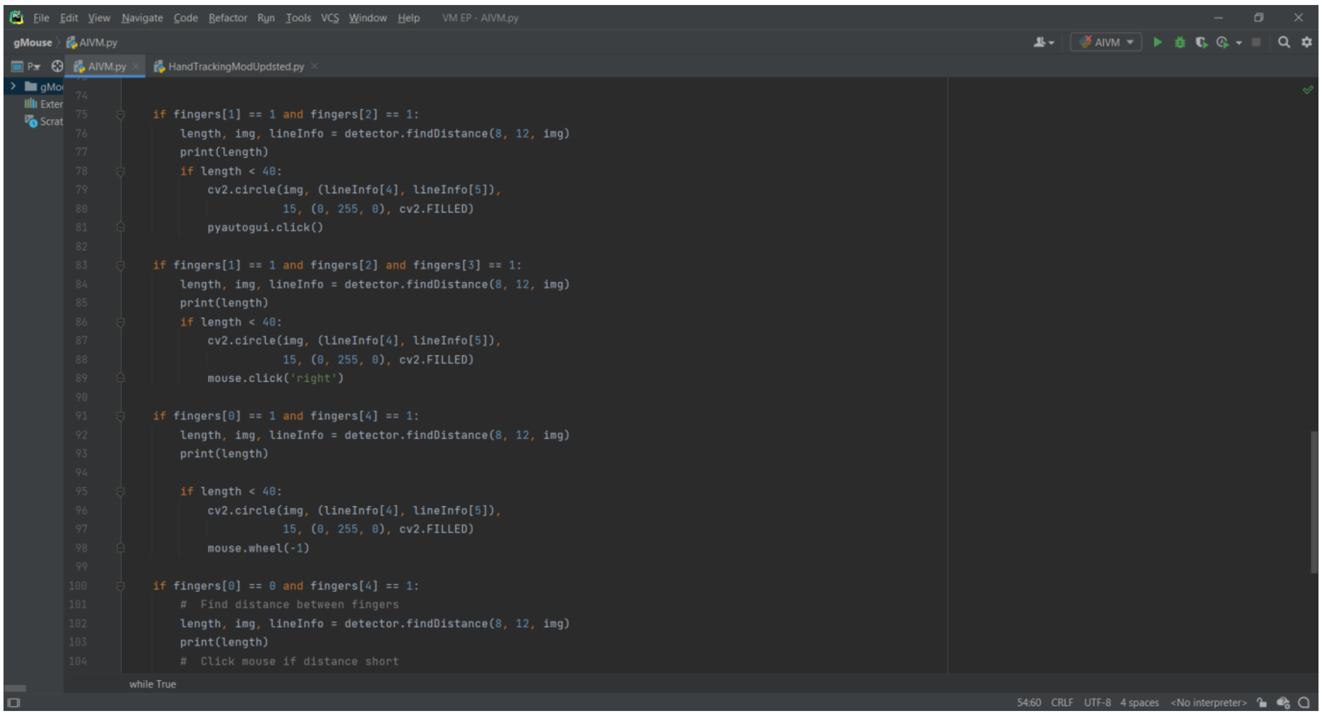
Main:


File Edit View Navigate Code Refactor Run Tools VCS Window Help
VM EP - AIVM.py
gMouse
AIVM.py
HandTrackingModUpdsted.py
AIVM
while True
54:60 CRLF UTF-8 4 spaces <No interpreter>

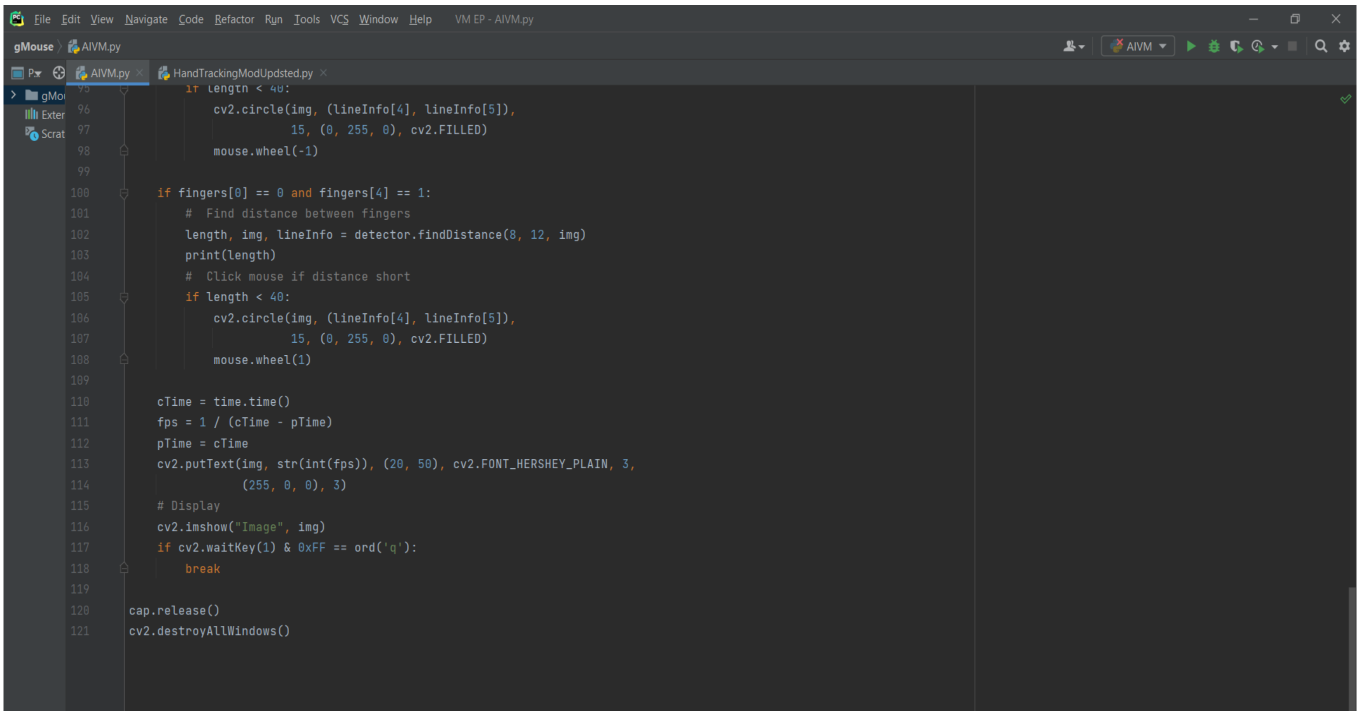
```python
import cv2
import numpy as np
import HandTrackingModUpdsted as htm
import time
import pyautogui
import mouse

wCam, hCam = 640, 480
frameR = 100
smoothening = 7


pTime = 0
plocX, plocY = 0, 0
clocX, clocY = 0, 0

cap = cv2.VideoCapture(0)
cap.set(3, wCam)
cap.set(4, hCam)
detector = htm.handDetector(maxHands=1)
wScr, hScr = pyautogui.size()
# print(wScr, hScr)

while True:

    success, img = cap.read()
    img = detector.findHands(img)
    lmList, bbox = detector.findPosition(img)

    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]
```


File Edit View Navigate Code Refactor Run Tools VCS Window Help
VM EP - AIVM.py
gMouse
AIVM.py
HandTrackingModUpdsted.py
AIVM
while True
54:60 CRLF UTF-8 4 spaces <No interpreter>

```python
    if len(lmList) != 0:
        x1, y1 = lmList[8][1:]
        x2, y2 = lmList[12][1:]
        # print(x1, y1, x2, y2)


    fingers = detector.fingersUp()
    # print(fingers)
    cv2.rectangle(img, (frameR, frameR), (wCam - frameR, hCam - frameR),
                  (255, 0, 255), 2)

    if fingers[1] == 1 and fingers[2] == 0:

        x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
        y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))

        clocX = plocX + (x3 - plocX) / smoothening
        clocY = plocY + (y3 - plocY) / smoothening


        mouse.move(wScr - clocX, clocY, absolute=True)
        cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)
        plocX, plocY = clocX, clocY

    # if fingers[1] == 1 and fingers[2] == 0 and fingers[0] == 1:
    #     #  Convert Coordinates
    #     x3 = np.interp(x1, (frameR, wCam - frameR), (0, wScr))
    #     y3 = np.interp(y1, (frameR, hCam - frameR), (0, hScr))
    #     #  Smoothen Values
    #     clocX = plocX + (x3 - plocX) / smoothening
    #     clocY = plocY + (y3 - plocY) / smoothening
```



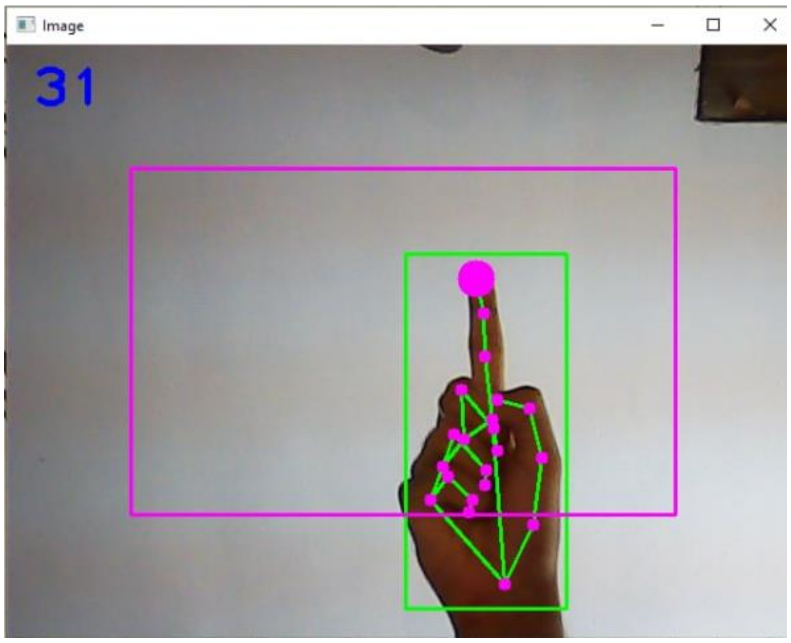
```python
    if fingers[1] == 1 and fingers[2] == 1:
        length, img, lineInfo = detector.findDistance(8, 12, img)
        print(length)
        if length < 40:
            cv2.circle(img, (lineInfo[4], lineInfo[5]),
                       15, (0, 255, 0), cv2.FILLED)
            pyautogui.click()

    if fingers[1] == 1 and fingers[2] and fingers[3] == 1:
        length, img, lineInfo = detector.findDistance(8, 12, img)
        print(length)
        if length < 40:
            cv2.circle(img, (lineInfo[4], lineInfo[5]),
                       15, (0, 255, 0), cv2.FILLED)
            mouse.click('right')

    if fingers[0] == 1 and fingers[4] == 1:
        length, img, lineInfo = detector.findDistance(8, 12, img)
        print(length)

        if length < 40:
            cv2.circle(img, (lineInfo[4], lineInfo[5]),
                       15, (0, 255, 0), cv2.FILLED)
            mouse.wheel(-1)

    if fingers[0] == 0 and fingers[4] == 1:
        #  Find distance between fingers
        length, img, lineInfo = detector.findDistance(8, 12, img)
        print(length)
        #  Click mouse if distance short
```



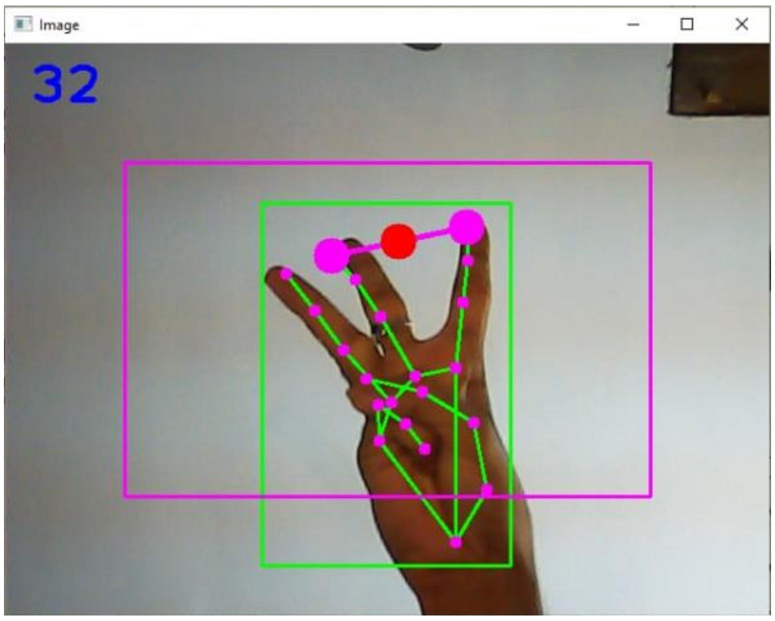
```python
        if length < 40:
            cv2.circle(img, (lineInfo[4], lineInfo[5]),
                       15, (0, 255, 0), cv2.FILLED)
            mouse.wheel(-1)

    if fingers[0] == 0 and fingers[4] == 1:
        #  Find distance between fingers
        length, img, lineInfo = detector.findDistance(8, 12, img)
        print(length)
        #  Click mouse if distance short
        if length < 40:
            cv2.circle(img, (lineInfo[4], lineInfo[5]),
                       15, (0, 255, 0), cv2.FILLED)
            mouse.wheel(1)

    cTime = time.time()
    fps = 1 / (cTime - pTime)
    pTime = cTime
    cv2.putText(img, str(int(fps)), (20, 50), cv2.FONT_HERSHEY_PLAIN, 3,
                (255, 0, 0), 3)
    # Display
    cv2.imshow("Image", img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

Outputs Screenshots:

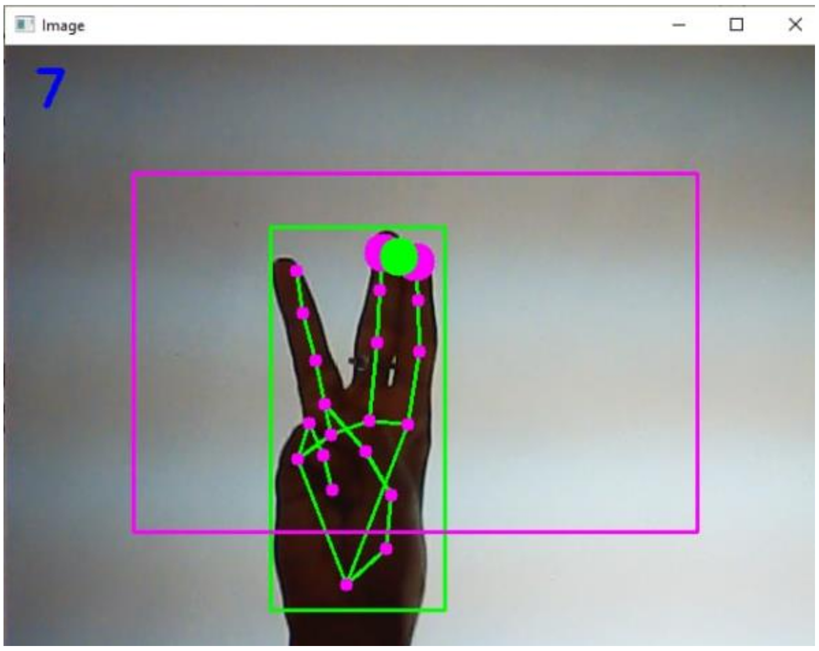
Mouse Pointer Mode:



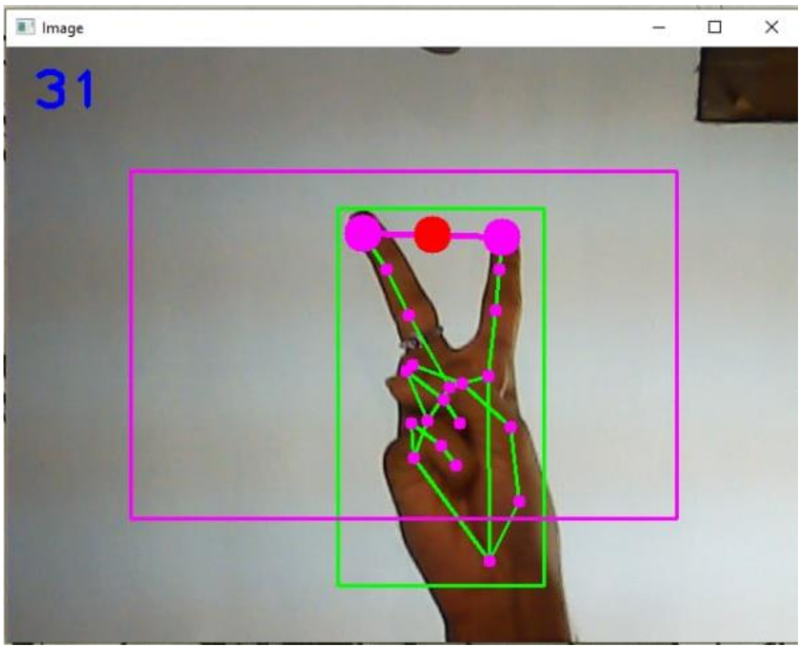
Right Click Mode:



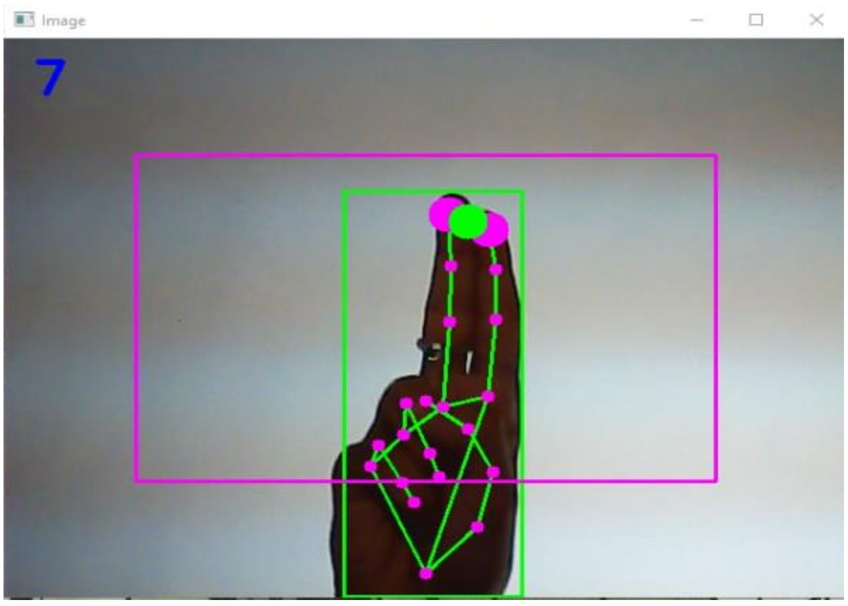
Right Click :



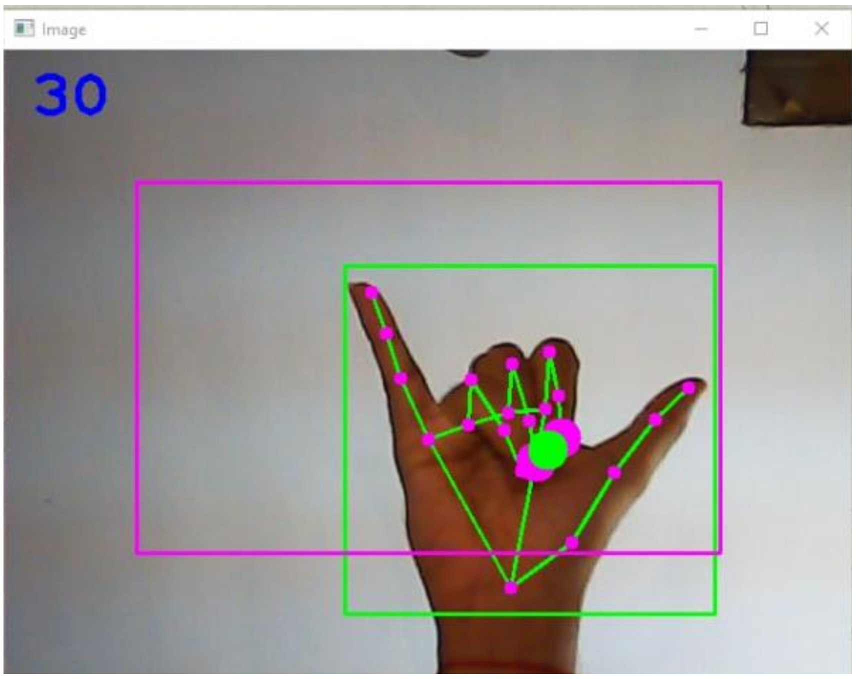
Left Click Mode:



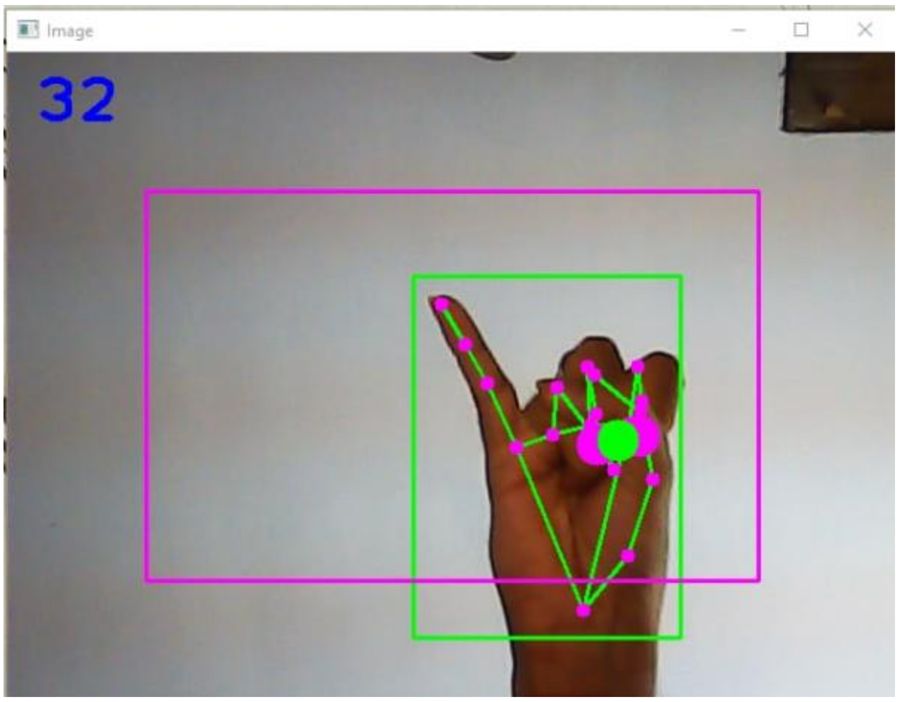
Left Click:



Scroll Down:



Scroll Up:



CONCLUSION

VIRTUAL MOUSE is an idea of implementing an adaptable, multi- functional navigation/interaction tool that overcomes physical barriers

- We are developing a system to control the mouse cursor using a real-time camera.
- This system is based on computer vision algorithms and can do all mouse tasks.
- However, it is difficult to get stable results because of the variety of lighting and skin colours of human races.
- This system could be useful in presentations and to reduce work space.
- Features such as enlarging and shrinking windows, closing window, etc. by using the palm and multiple fingers.
- The system will be ‘real’ enough to not affect the interaction much.

FUTURE SCOPE

In future, we plan to add more features such as enlarging and shrinking windows, volume control zoom-in or zoom-out for pictures etc, by using the palm and multiple fingers to make a fully functional Virtual Mouse by the help of gestures.

THANK YOU