* 1. **INTRODUCTION**

# 1. INTRODUCTION

Human computer interaction is one of the most rapidly growing technologies. Computer machines have the ability to make our daily lives much more convenient. Human Pose estimation is a computer vision task that represents the orientation of a person in a graphical format. This technique is widely applied to predict a person’s body parts or joint position. It is one of the most exciting areas of research in computer vision that has gained a lot of traction because of its abundance of applications that can benefit from such a technology.

As we all know that Natural human language communication is based on languages, poses and gestures. Keeping this in mind we have created a project which detects the pose of a human. Human Pose Estimation identifies and classifies the poses of human body parts and joints in images or videos. Essentially it is a way to capture a set of coordinates by defining the human body joints like wrist, shoulder, knees, eyes, ears, ankles, and arms, which is a key point in images and videos that can describe a pose of a person. Then, when an image or video is given to the pose estimator model as input, it identifies the coordinates of these detected body parts and joints as output and a confidence score showing precision of the estimations. We have implemented this project using computer vision and media pipe.

Human pose detection can be viewed as a way for computers to begin to understand human body language and signs, thus filling the void between computing machines and humans. The model which we have created include detecting the human body parts and tracking its landmarks which is done using media pipe. The detection include drawing the skeletal structure of a human body. The camera will be opened using cv2 module where the human is visible and the land marks are detected. The human pose detection project contains three phases. They are:

1. Pose detection for an image which is given as input where all the 33 key points are marked and a skeletal structure is formed.
2. Pose detection for a real time video where the structure formed moves and displays the position of the human.

3. Count the number of reps performed by the human and display the count as output.

## MOTIVATION

Detection of people has long been a primary center of discussion for various applications in traditional object detection. With recent developments in machine-learning algorithms, computers can now understand human body language by performing pose detection and pose tracking. The accuracy of these detections and the hardware requirements to run them has now reached a point where it has become commercially viable.

In addition, the technology’s growth is also profoundly transformed amid the coronavirus pandemic, where high-performing real-time pose detection and tracking will bring some of the most influential trends in computer vision. For instance, it can be employed for social distancing by combining human pose estimation and distance projection heuristics. It assists people in maintaining physical distance from each other in a crowded place. Human pose estimation will significantly impact various industries, including security, business intelligence, health and safety, and entertainment. One such area where this technique has already proved its existence is autonomous driving. With the help of real-time human pose detection and tracking, computers can sense and predict pedestrian behaviour much thoroughly – allowing more consistent driving.

## OBJECTIVE OF THE PROJECT

The main objective of the proposed hand gesture based keyboard is to furnish an alternative to the conventional physical keyboard that provides functions with the help of computer vision enabled computer that houses a web camera which recognizes fingers and hand gestures and processes the captured frames to execute the defined keyboard functions like hovering, click. Also we are using multiple libraries to perform this project.

## NOVELTY OF THE PROJECT

The project on human pose detection using Media Pipe uses a deep neural network to detect 33 body landmarks, including the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles. This is a significant improvement over traditional pose detection methods that may only detect a few key points on the body. The increased number of landmarks allows for more accurate detection and tracking of human movements. The Media Pipe pose detection module can be used in real-time, which is essential for applications such as fitness and sports analysis. Real-time detection enables immediate feedback and monitoring of movements, which can help improve performance and prevent injuries.

## PROBLEM DESCRIPTION

The problem that the human pose detection project aims to solve is the accurate and real-time detection and tracking of human movements. Traditional methods for detecting and tracking human movements are often limited in their accuracy and ability to operate in real-time. This can make it difficult to monitor and provide feedback on movements during activities such as fitness, sports analysis, and healthcare. Furthermore, accurate and real-time detection and tracking of human movements are essential for creating interactive experiences in the entertainment industry. The ability to detect and track the movements of the user can enhance the user experience in games, virtual reality, and augmented reality applications.

Therefore, the human pose detection project aims to solve the problem of accurate and real-time detection and tracking of human movements using the Media Pipe framework. The project aims to leverage the power of deep neural networks to detect 33 body landmarks, including the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles. The project aims to provide a solution that is accessible, customizable, and efficient, with applications in various industries such as fitness, sports analysis, healthcare, and entertainment.

* 1. **INTRODUCTION**

# LITERATURE SURVEY

## Research on recognizing required items based on open cv and machine learning, in 2022

Starting from the background of the outbreak of New Coronavirus, in order to realize the function of automatically identifying the required items by machine, the support vector machine algorithm in the neural network and the traditional computer vision algorithm opencv were used. The software developed by pycharm and python programming language was used to compile automatically a software to identify whether the required items were filled out. And on the basis of completing the software, it is connected to the embedded device high-speed clapper. It is applied to Fuzhou Customs to help the customs staff review the health form and declaration card of inbound and outbound passengers, which not only saves the time of staff and passengers, but also contributes to the prevention and control of epidemic situation to a certain extent.[1]

## Controlling computer using hand gestures, in 2022

In this system they aimed to build a real-time gesture recognition system uses hand gestures. Particularly, they used convolutional neural networks in throughout the process. This application presents a hand-gesture based system to control a computer that is performing different operations using neural network. This application is defined in five phases, Image Frame Acquisition, Hand tracking, feature extraction, recognition of gestures and classification. An image is captured from the webcam and so hand detection, hand shape feature extraction and hand gesture recognition are done.[2]

## Hand gesture keyboard using machine learning, in 2020

Gestures are the most natural expressive way for nonverbal communications between human and computers in virtual system. Hand gesture is the simplest method of non-verbal communication for human beings for its freer expressions much more other than body parts. The proposed system can recognize the hand gestures using an accelerometer and convert it to meaningful sentences using neural network architecture. Accelerometer that connected to hand will provide input to the Arduino microcontroller that programs to process the gestures and converts it to text. Recurrent Neural Network (RNN) algorithm is used here to predict the gesture to a meaningful word and also aiming to add Bluetooth support so as to make the gesture device wireless and more

flexible. The meaningful words will be communicated to PC or mobile phones through Bluetooth communication.[3]

## Glove-based hand gesture recognition sign language translator using capactive touch sensor, in 2016

We present an intelligent electronic glove system able to detect numbers of sign language in order to automate the process of communication between a deaf-mute person and others. this is done by translating the hands move sign language into an oral language. The system is inside to a glove with flex sensors in each finger that we are used to collect data that are analyzed through a methodology. The prototype recognize gestures for the numbers 0 to 9 and the 26 English alphabets, A to Z. The glove experimentally achieved, based on 1080 trials, an overall detection accuracies of over 92 %, which is comparable with current high-end counterparts. The proposed device i expected to bridge the communication gap between the hearing and speech impaired and members of the general public.[4]

## Using AEPI method for hand gesture recognition in varying background and blurred images, in 2017

In this paper, the Accurate End Point Identification method is implemented and applied on gesture images which are captured in varying background and it is also applied on blurred images containing multiple objects. The AEPI method accurately recognizes the gestures from such images and provides a new dimension to implement user interface that will help to provide more natural inputs through hand gestures. In this paper they have proposed a new method called Hand Gesture Recognition Using Accurate End Point Identification (AEPI) method. This method not only recognizes Hand Gesture in less time only but also with high percentage of accuracy.[5]

## Finger recognition and gesture based virtual keyboard, in 2020

Hand gesture recognition is very significant for human-computer interaction, in this work we present a novel real-time method for hand gesture recognition. The proposed system is vision based, which uses machine learning techniques and inputs from a computer webcam. Vision based gesture recognition tracking and gesture recognition in our framework, the hand region is extracted from the background with the background subtraction method. Finally, a rule classifier is applied to predict the labels of hand gestures. Our method shows better performance than a state-of-art method on another data set of hand gestures.[6]

## User defined custom virtual keyboard by IEEE in 2017

This paper defines new way of developing a virtual and custom keyboard, using 2D camera and other sensors available in phone, without the use of any extra input hardware. This implementation enables users to create his keyboard on any homogenous surface like table or paper.[7]

## Gesture based motion tracking keyboard using machine learning, in 2020

This motion recognition system tracks motion made in mid-air by device in a 3D space, log its speed, angular velocity, distance covered and some other variables at real time, and in turn convert the device captured data of motion into characters of English alphabets. The device presented in this paper presents solution based on support vector algorithms.[8]

## EXISTING SYSTEM AND DISADVANTAGES OF EXISTING SYSTEM

* + - Existing systems for human pose detection typically rely on traditional computer vision techniques that involve detecting key features on the body, such as the hands or face. These systems often require complex image processing algorithms and may have limitations in their ability to detect movements accurately and in real-time.
    - One disadvantage of these traditional systems is that they may not be able to accurately detect movements in complex environments with varying lighting conditions and background clutter. Additionally, these systems may struggle to detect movements that are occluded, such as when the user's hand is behind their body.
    - Another disadvantage of traditional pose detection systems is that they may only detect a few key points on the body, which can lead to inaccurate tracking of movements. This can limit their effectiveness in applications such as sports analysis, where accurate tracking of movements is essential.
    - Furthermore, traditional pose detection systems may be difficult to customize for specific applications and may require significant development time and resources to implement.
    - In contrast, the human pose detection project using Media Pipe addresses some of the limitations of traditional systems by using a deep neural network to detect 33 body landmarks with increased accuracy and real-time performance. The system is also highly customizable and accessible, making it suitable for a wide range of applications.

## PROPOSED SYSTEM

* + - The proposed system for human pose detection uses the Media Pipe framework, which is a cross-platform solution for building real-time machine learning pipelines. The system aims to detect and track 33 body landmarks, including the nose, eyes, ears, shoulders, elbows, wrists, hips, knees, and ankles, in real-time.
    - The system leverages the power of deep neural networks to achieve high accuracy in pose detection. It is based on a pre-trained pose detection model that has been trained on a large dataset of labeled human poses. The model is optimized for real-time performance on mobile and desktop devices, making it suitable for a wide range of applications.
    - Python programming language is used for developing this project.
    - To use the system, a video or live camera feed is input into the system. The system then analyzes the frames in real-time and detects the 33 body landmarks using the pre-trained model. The detected landmarks are then used to track the movements of the user, providing real-time feedback and monitoring.
    - The model makes use of media pipe package for tracking different body parts. The media pipe takes human body image as input and produces landmarks as output.

# SYSTEM SPECIFICATIONS

## FUNCTIONAL REQUIREMENTS

In software engineering and systems engineering, a functional requirement defines a function of a system or its component, where a function is described as a specification of behaviour between outputs and inputs.

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in use cases. Functional requirements are supported by non-functional requirements (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do

<requirement>," while non-functional requirements take the form "system shall be

<requirement>." The plan for implementing functional requirements is detailed in the system design, whereas non-functional requirements are detailed in the system architecture.

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system. In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/stakeholder request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated. Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

## HARDWARE REQUIREMENTS

* + - System : i3
    - Hard disk : 40 GB
    - RAM : 4 GB
    - Webcam

## SOFTWARE REQUIREMENTS

* + - Operating system : Windows 8
    - Coding language : Python 3.7
    - IDE : PyCharm
    - Library : Mediapipe
  1. **FEASIBILTY STUDY**

# SYSTEM STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three considerations involved in feasibility analysis are:

* + Economical Feasibility
  + Technical Feasibility
  + Social Feasibility

## Economical Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

## Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

## Social Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# SYSTEM DESIGN

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

## UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing object oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

## GOALS

Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.

* + - Provide extendibility and specialization mechanisms to extend the core concepts.
    - Be independent of particular programming languages and development process.
    - Provide a formal basis for understanding the modeling language.
    - Encourage the growth of OO tools market.
    - Support higher level development concepts such as collaborations, frameworks, patterns and components.
    - Integrate best practices.

## USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioural diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

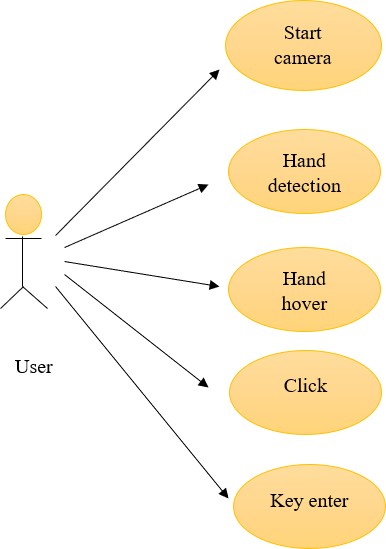


Fig 5.3.1 Use Case Diagram

## CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

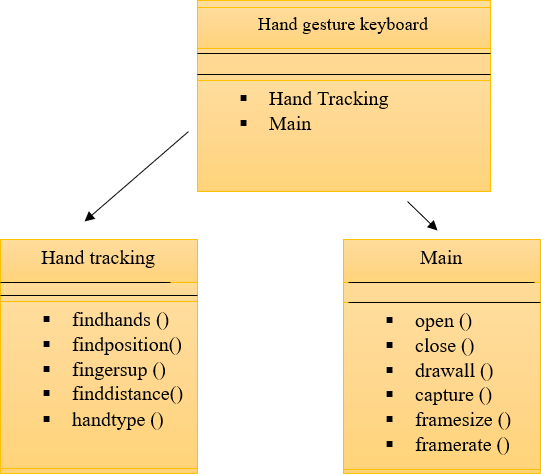


Fig 5.4.1 Class Diagram

## SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

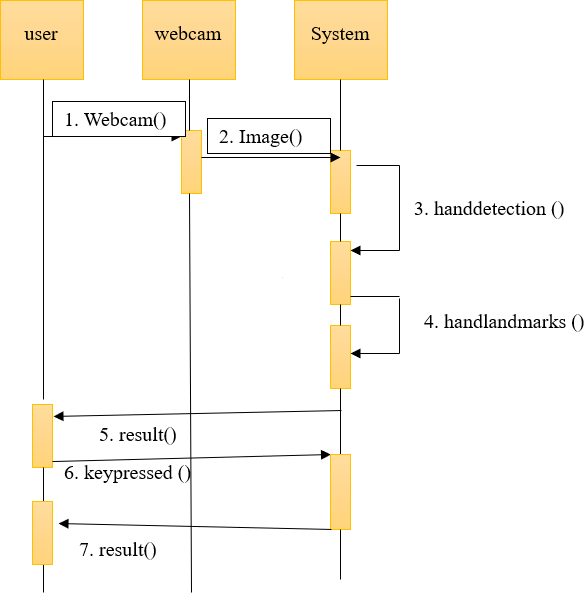


Fig 5.5.1 Sequence Diagram

## ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

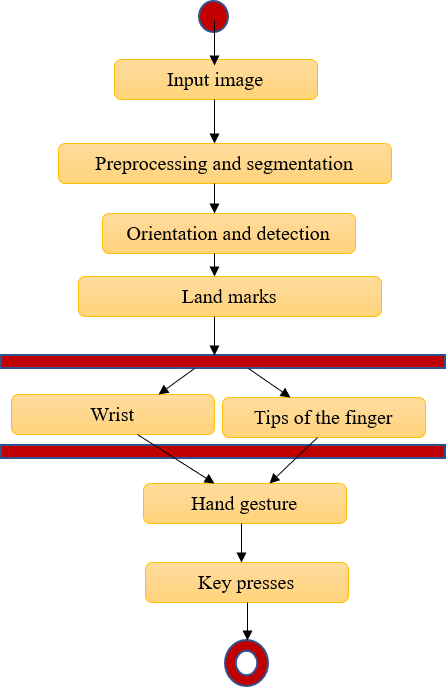


Fig 5.6.1 Activity Diagram

# DESIGN AND IMPLEMENTATION

## INTRODUCTION PROJECT EXPLANATION

The system has two main modules

* + - Hand tracking module
    - Main module

## MODULES

* + 1. **Video Capturing Modules**

With OpenCV we can capture a video from the camera. It lets us create a video capture object which is helpful to capture videos through webcam and then we can perform desired operations an that video. In this pictures can be conveyed in concealing layered with three channels red, green, blue. A video capture id is given as the id for the cam. Hand gesture based keyboard framework utilizes the instructive algorithmic rule and it changes over the coordinates of tip from camera screen to pc window full-screen.[9]

Steps to capture a video

* + - * Use cv2.VideoCapture() to get a video capture object for the camera.
      * Set up an infinite while loop and use the read() method to read the frames using the above created object.
      * Use cv2.imshow() method to show the frames in the video.
      * Breaks the loop when the user clicks a specific key.

## Palm Detection Module

It is used to detect initial hand locations. Detecting hands is a complex task. Our model have to work across variety of hand sizes with a large scale span relative to the image frame and be able to detect occluded and self-occluded hands. Here we first train a palm detector and next an encoder and decoder feature extract is used for bigger scene content awareness even for small objects. At last we minimize the focal loss during training to support large anchors resulting from high scale variance. Using this technique we can achieve an average precision of 95.7% in palm detection.[10]

## Hand Tracking Module

After running palm detection over the whole image, our subsequent hand landmark model performs precise landmark localization of 21 2.5D coordinates inside the detected hand regions via regression. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions. The model has three outputs

1. 21 hand landmarks consisting of x, y, and relative depth.
2. A hand flag indicating the probability of hand presence in the input image.
3. A binary classification of handedness, e.g. left or right hand.

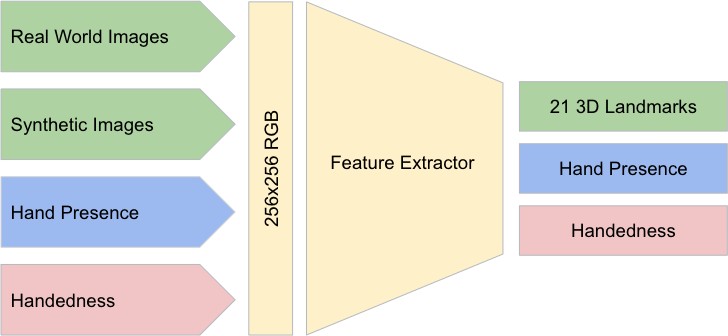
The 2D coordinates are learned from both real-world images as well as synthetic datasets as discussed below, with the relative depth with respect to the wrist point being learned only from synthetic images. To recover from tracking failure, we developed another output of the model for producing the probability of the event that a reasonably aligned hand is indeed present in the provided crop. If the score is lower than a threshold then the detector is triggered to reset tracking. Handedness is another important attribute for effective interaction using hands in AR/VR. This is especially useful for some applications where each hand is associated with a unique functionality. Thus, we developed a binary classification head to predict whether the input hand is the left or right hand. Our setup targets real-time mobile GPU inference, but we have also designed lighter and heavier versions of the model to address CPU inference on the mobile devices lacking proper GPU support and higher accuracy requirements of accuracy to run on desktop, respectively.[10][11]

Fig 6.2.3.1 Architecture Of Hand Landmark Model

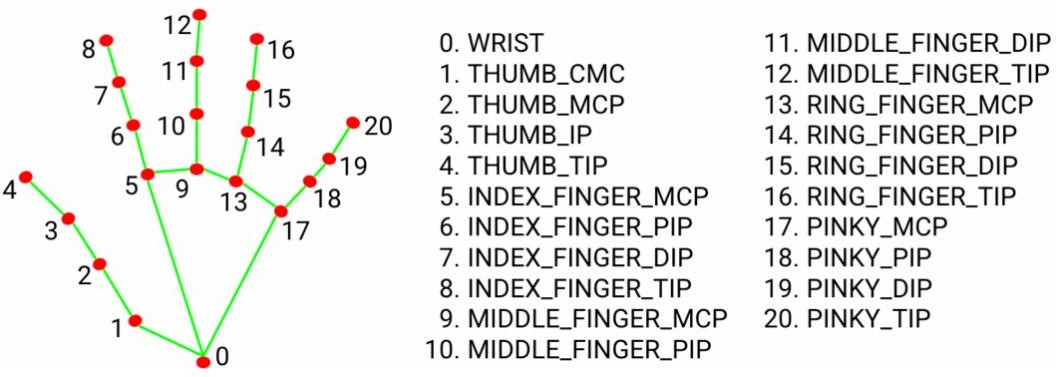


Fig 6.2.3.2 Hand Landmark Model

## REQUIREMENTS FOR THE PROJECT

* + 1. **PyCharm**

PyCharm is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used for programming in [Python.](https://en.wikipedia.org/wiki/Python_(programming_language)) It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control](https://en.wikipedia.org/wiki/Version_control) systems, and supports web development with [Django](https://en.wikipedia.org/wiki/Django_(web_framework)). PyCharm is developed by the Czech company [JetBrains.](https://en.wikipedia.org/wiki/JetBrains) It is [cross-platform,](https://en.wikipedia.org/wiki/Cross-platform_software) working on [Microsoft](https://en.wikipedia.org/wiki/Microsoft_Windows) [Windows,](https://en.wikipedia.org/wiki/Microsoft_Windows) [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux.](https://en.wikipedia.org/wiki/Linux) PyCharm has a Professional Edition, released under a [proprietary license](https://en.wikipedia.org/wiki/Proprietary_software) and a Community Edition released under the [Apache License.](https://en.wikipedia.org/wiki/Apache_License) PyCharm Community Edition is less extensive than the Professional Edition.[13]

PyCharm is available in two editions:

* + - * Community (free and [open-sourced](https://github.com/JetBrains/intellij-community/blob/master/LICENSE.txt)): for smart and intelligent Python development, including code assistance, re-factorings, visual debugging, and version control integration.
      * Professional ([paid](https://www.jetbrains.com/pycharm/buy/%23commercial?billing=yearly)) : for professional Python, web, and data science development, including code assistance, re-factorings, visual debugging, version control integration, remote configurations, deployment, support for popular web frameworks, such as Django and Flask, database support, scientific tools (including Jupyter notebook support), big data tools.

PyCharm supports the following versions of Python:

* + - * Python 2: version 2.7
      * Python 3: from the version 3.6 up to the version 3.12

PyCharm is a cross-platform IDE that works on Windows, macOS, and Linux. The minimum system requirements are:

|  |  |  |
| --- | --- | --- |
| **Requirement** | **Minimum** | **Recommended** |
| RAM | 4 GB of free RAM | 8 GB of total system RAM |
| CPU | Any modern CPU | Multi-core CPU. PyCharm supports multithreading for different operations and processes making it faster the more CPU cores it can  use. |
| Disk space | 3.5 GB | SSD drive with at least 5 GB  of free space |
| Monitor resolution | 1024×768 | 1920×1080 |
| Operating system | Officially released 64-bit versions of the following:   * Microsoft Windows 10 1809 or later Windows Server 2019 or later * macOS 10.15 or later * Any Linux distribution that supports Gnome, KDE , or Unity DE.   PyCharm is not available for the Linux distributions that do not include [GLIBC](https://ftp.gnu.org/gnu/libc/) 2.27 or later. | Latest 64-bit version of Windows, macOS, or Linux (for example, Debian, Ubuntu, or RHEL) |

Table 6.3.1.1 Supported Platforms

Features of pycharm [12]

## Intelligent Code Editor:

* It helps us write high-quality codes!
* It consists of color schemes for keywords, classes, and functions. This helps increase the readability and understanding of the code.
* It helps identify errors easily.
* It provides the autocomplete feature and instructions for the completion of the code.

## Code Navigation:

* It helps developers in editing and enhancing the code with less effort and time.
* With code navigation, a developer can easily navigate to a function, class, or file.
* A programmer can locate an element, a symbol, or a variable in the source code within no time.
* Using the lens mode, further, a developer can thoroughly inspect and debug the entire source code.

## Refactoring

* It has the advantage of making efficient and quick changes to both local and global variables.
* Refactoring in PyCharm enables developers to improve the internal structure without changing the external performance of the code.
* It also helps split up more extended classes and functions with the help of the extract method.

## Assistance for Many Other Web Technologies

* It helps developers create web applications in Python.
* It supports popular web technologies such as HTML, CSS, and JavaScript.
* Developers have the choice of live editing with this IDE. At the same time, they can preview the created/updated web page.
* The developers can follow the changes directly on a web browser.
* PyCharm also supports AnglularJS and NodeJS for developing web applications.

## Support for Popular Python Web Frameworks

* PyCharm supports web frameworks such as Django.
* It provides the autocomplete feature and suggestions for the parameters of Django.
* It helps in debugging the codes of Django.
* It also assists web2py and Pyramid, the other popular web frameworks.

## Assistance for Python Scientific Libraries

* PyCharm supports Python’s scientific libraries such as [Matplotlib,](https://intellipaat.com/blog/tutorial/python-tutorial/python-matplotlib/) NumPy, and Anaconda.
* These scientific libraries help in building projects of Data Science and Machine Learning.
* It consists of interactive graphs that help developers understand data.
* It is capable of integrating with various tools such as IPython, Django, and Pytest. This integration helps innovate unique solutions.

## Python Programming Language

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language include NASA, Google, YouTube, BitTorrent, etc. Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science. Python had deep focus on code readability & this class will teach you python from basics.

## Python Programming Characteristics

* + - * It provides rich data types and easier to read syntax than any other programming languages.
      * It is a platform independent scripted language with full access to operating system API's.
      * Compared to other programming languages, it allows more run-time flexibility.
      * It includes the basic text manipulation facilities of Perl and Awk.
      * A module in Python may have one or more classes and free functions.
      * Libraries in Pythons are cross-platform compatible with Linux, Macintosh, and Windows.
      * For building large applications, Python can be compiled to byte-code.
      * Python supports functional and structured programming as well as OOP.
      * It supports interactive mode that allows interacting Testing and debugging of snippets of code.
      * In Python, since there is no compilation step, editing, debugging and testing is fast.

## Applications of Python Programming Web Applications

You can create scalable Web Apps using frameworks and CMS (Content Management

System) that are built on Python. Some of the popular platforms for creating Web Apps are: Django, Flask, Pyramid, Plone, Django CMS. Sites like Mozilla, Reddit, Instagram and PBS are written in Python.

## Scientific and Numeric Computing

There are numerous libraries available in Python for scientific and numeric computing. There are libraries like: SciPy and NumPy that are used in general purpose computing. And, there are specific libraries like: EarthPy for earth science, AstroPy for Astronomy and so on. Also, the language is heavily used in machine learning, data mining and deep learning.

## Creating software Prototypes

Python is slow compared to compiled languages like C++ and Java. It might not be a good choice if resources are limited and efficiency is a must. However, Python is a great language for creating prototypes. For example: You can use Pygame (library for creating games) to create your game's prototype first. If you like the prototype, you can use language like C++ to create the actual game.

## Good Language to Teach Programming

Python is used by many companies to teach programming to kids and newbies. It is a good language with a lot of features and capabilities. Yet, it's one of the easiest language to learn because of its simple easy-to-use syntax.

## OpenCV

OpenCV is an open-source library for computer vision, machine learning and image processing. It supports a wide variety of programming languages like Python, C++, Java etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as NumPy which is a highly optimized library for numerical operations, then the number of weapons increases in our Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. It contains imageprocessing algorithms for object detection and real-time computer vision applications can be developed by using the computer vision library. The OpenCV library is used in image and video processing and also analysis such as face detection and object detection. The OpenCV is used to capture the images through webcam in this project and for which we will be importing cv2.

Compared to languages like C/C++, Python is slower. That said, Python can be easily extended with C/C++, which allows us to write computationally intensive code in C/C++ and create Python wrappers that can be used as Python modules. This gives us two advantages: first, the code is as fast as the original C/C++ code (since it is the actual C++ code working in background) and second, it easier to code in Python than C/C++. OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

OpenCV-Python makes use of Numpy, which is a highly optimized library for numerical operations with a MATLAB-style syntax. All the OpenCV array structures are converted to and from Numpy arrays. This also makes it easier to integrate with other libraries that use Numpy such as SciPy and Matplotlib.

The term Computer Vision (CV) is used and heard very often in artificial intelligence (AI) and deep learning (DL) applications. The term essentially means giving a computer the ability to see the world as we humans do.

Computer Vision is a field of study which enables computers to replicate the human visual system. As already mentioned above, It’s a subset of artificial intelligence which collects information from digital images or videos and processes them to define the attributes. The entire process involves image acquiring, screening, analysing, identifying and extracting information. This extensive processing helps computers to understand any visual content and act on it accordingly.

Computer vision projects translate digital visual content into explicit descriptions to gather multi-dimensional data. This data is then turned into a computer-readable language to aid the decision-making process. The main objective of this branch of artificial intelligence is to teach machines to collect information from pixels.

A digital image is an image composed of picture elements, also known as pixels, each with finite, discrete quantities of numeric representation for its intensity or grey level. So the computer sees an image as numerical values of these pixels and in order to recognise a certain image, it has to recognise the patterns and regularities in this numerical data.

Here is a hypothetical example of how pixels form an image. The darker pixels are represented by a number closer to the zero and lighter pixels are represented by numbers approaching one. All other colours are represented by the numbers between 0 and 1.

But usually, you will find that for any colour image, there are 3 primary channels – Red, green and blue and the value of each channel varies from 0-255. In more simpler terms we can say that a digital image is actually formed by the combination of three basic colour channels Red, green, and blue whereas for a grayscale image we have only one channel whose values also vary from 0-255.

Computer Vision overlaps significantly with the following fields

* Image Processing − It focuses on image manipulation.
* Pattern Recognition − It explains various techniques to classify patterns.
* Photogrammetry − It is concerned with obtaining accurate measurements from images.

Image processing deals with image-to-image transformation. The input and output of image processing are both images. Computer vision is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

Features of OpenCV Library

* Read and write images
* Capture and save videos
* Process images (filter, transform)
* Perform feature detection
* Detect specific objects such as faces, eyes, cars, in the videos or images.
* Analyse the video, i.e., estimate the motion in it, subtract the background, and track objects in it.[14]

## Mediapipe

MediaPipe is a Framework for building machine learning pipelines for processing timeseries data like video, audio, etc. This cross-platform Framework works on Desktop/Server, Android, iOS, and embedded devices like Raspberry Pi and Jetson Nano.

MediaPipe powers revolutionary products and services we use daily. Unlike powerhungry machine learning Frameworks, MediaPipe requires minimal resources. It is so tiny and efficient that even embedded IoT devices can run it. In 2019, MediaPipe opened up a new world of opportunity for researchers and developers following its public release.

The MediaPipe perception pipeline is called a Graph. Let us take the example of the first solution, Hands. We feed a stream of images as input which comes out with hand landmarks rendered on the images.

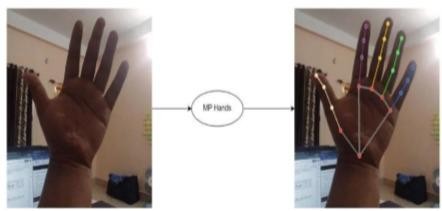


Fig 6.3.4.1 Mediapipe Hands

The flow chart below represents the MP (Abbr. MediaPipe) hand solution graph.

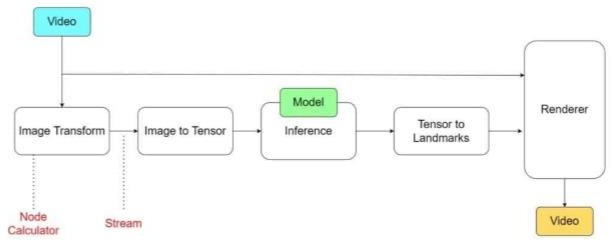


Fig 6.3.4.2 Hand Solution Graph

In computer science jargon, a graph consists of Nodes connected by Edges. Inside the MediaPipe Graph, the nodes are called Calculators, and the edges are called Streams. Every stream carries a sequence of Packets that have ascending time stamps. In the image above, we have represented Calculators with rectangular blocks and Streams using arrows.

Solutions are open-source pre-built examples based on a specific pre-trained TensorFlow or TFLite model. You can check Solution specific models here. MediaPipe Solutions are built on top of the Framework.

The solutions are available in C++, Python, JavaScript, Android, iOS, and Coral. As of now, the majority of the solutions are available only in C++ (except KNIFT and IMT) followed by Android, with Python not too far behind.

The other wrapper languages, too, are growing fast with a very active development state. As you can see, even though MediaPipe Framework is cross-platform, that does not imply the same for the solutions. MediaPipe is currently at alpha version 0.7. We can expect the solutions to get more support with the beta releases.

MediaPipe supports multimodal graphs. To speed up the processing, different calculators run in separate threads. For performance optimization, many built-in calculators come with options for GPU acceleration. Working with time series data must be in proper synchronization; otherwise, the system will break. The graph ensures this so that flow is handled correctly according to the timestamps of packets. The Framework handles synchronization, context sharing, and inter-operations with CPU calculators.

MediaPipe depends on OpenCV for video and FFMPEG for audio data handling. It also has other dependencies like OpenGL/Metal, Tensorflow, Eigen [1], etc.

MediaPipe solutions are straightforward, and you can cover them in a day or two.On the other hand, the learning curve can be pretty steep for the C++ MediaPipe Framework. Don’t worry; we will get there by taking baby steps.

Overall, it is a beautiful, fast-growing library that delivers promising results. Implementing MediaPipe in projects nullifies most of the hassles we usually face while working on an ML project. No need to worry about synchronization and cumbersome setups. It allows you to focus on the actual development part.

In the upcoming posts, we will show how to build interesting Augmented Reality filters using the MediaPipe Face solution. Later in this series, we will cover customizing calculators of pre-built MediaPipe solutions and building custom graphs.

## Features

* **Input image processing** - Processing includes image rotation, resizing, normalization, and color space conversion.
* **Score threshold** - Filter results based on prediction scores.

|  |  |
| --- | --- |
| **Task inputs** | **Task outputs** |
| The Hand Landmarker accepts an input of one of the following data types:   * Still images. * Decoded video frames. * Live video feed. | The Hand Landmarker outputs the following results:   * Handedness of detected hands. * Landmarks of detected hands in image coordinates. * Landmarks of detected hands in world coordinates. |

Table 6.3.4.1 Features Of Mediapipe

## Solution API’s in mediapipe static\_image\_mode

If it is set to false the solution treats the input images as video stream. At first it tries to detect hands and upon a successful detection further localizes the hand landmarks. In subsequent images, once all max\_num\_hands are detected and the corresponding hand landmarks are localized. It simply tracks those landmarks without invoking another detection until it looses track of any of the hands. This reduces latency and is ideal for processing video frames. If set to true, hand detection runs on every input image, ideal for processing a batch of static. It is by default set to false.

## max\_num\_hands

It specifies the maximum number of hands to detect at a time. By default set to 2.

## model\_complexity

It specifies the complexity of the hand landmark model. It specifies the value as either 0 or 1. Landmark accuracy is directly proportional to model\_complexity. Default value is 1.

## minimum\_detection\_confidence

The minimum confidence score for the hand detection to be considered successful in palm detection model. It ranges between 0.0-1.0. By default the value is 0.5.

## minimum\_tracking\_confidence

The minimum confidence score for the hand tracking to be considered successful. This is the bounding box IoU threshold between hands in the current frame and the last frame. In Video mode and Stream mode of Hand Landmarker, if the tracking fails, Hand Landmarker triggers hand detection. Otherwise, it skips the hand detection. It ranges between 0.0-1.0. By default the value is 0.5.

## Output functions

**multi\_hand\_landmarks**

It is collection of detected or tracked hands, where each hand is represented as a list of 21 hand landmarks. Each landmark is composed of x, y, z. here x and y are normalized to [0.0- 0.1] by the image width and height respectively. z represents the landmark depth with the depth at the wrist begin the origin. The smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x.

## multi\_hand\_world\_landmarks

It is collection of detected or tracked hands, where each hand is represented as a list of 21 hand landmarks in world coordinates. Each landmark is composed of x, y, z in real world 3D coordinates in metres with the origin at the hands approximate geometric centre.

## multi\_handedness

It is collection of handedness of detected or tracked hands. Each hand is composed of label and score. Label is a string of value either left or right. Score is the estimated probability of the predicted handedness and is always greater than or equal to 0.5, handedness is detected by assuming the input image is mirrored i.e taken with a front facing camera with images flipped horizontally.[10]

## Time

The Python time module provides many ways of representing time in code, such as objects, numbers, and strings. It also provides functionality other than representing time, like waiting during code execution and measuring the efficiency of your code. This article will walk you through the most commonly used functions and objects in time.

First, time.time() returns the number of seconds that have passed since the epoch. The return value is a floating point number to account for fractional seconds:

>>> from time import time

>>> time() 1551143536.9323719

The number you get on your machine may be very different because the reference point considered to be the epoch may be very different.

Measuring time in seconds is useful for a number of reasons:You can use a float to calculate the difference between two points in time.A float is easily serializable, meaning that it can be stored for data transfer and come out intact on the other side.Sometimes, however, you may want to see the current time represented as a string. To do so, you can pass the number of seconds you get from time() into time.ctime().

As you saw before, you may want to convert the Python time, represented as the number of elapsed seconds since the epoch, to a string. You can do so using ctime():

>>> from time import time, ctime

>>> t = time()

>>> ctime(t)

'Tue Apr 19 19:11:56 2023'

Here, you’ve recorded the current time in seconds into the variable t, then passed t as an argument to ctime(), which returns a string representation of that same time. The representation of time dependent on your physical location is called local time and makes use of a concept called time zones.[15]

## Math

The methods in this module accepts int, float, and complex numbers. It even accepts Python objects that has a \_complex() or float\_() method. The methods in this module almost always return a complex number. If the return value can be expressed as a real number, the return value has an imaginary part of 0.

The cmath module has a set of methods and constants. This module provides access to the mathematical functions defined by the C standard. These functions cannot be used with complex numbers; use the functions of the same name from the cmath module if you require support for complex numbers. The distinction between functions which support complex numbers and those which don’t is made since most users do not want to learn quite as much mathematics as required to understand complex numbers. Receiving an exception instead of a complex result allows earlier detection of the unexpected complex number used as a parameter, so that the programmer can determine how and why it was generated in the first place. The following functions are provided by this module. Except when explicitly noted otherwise, all return values are floats.

## 6.3.6 Numpy

Numpy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.

Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists. The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

Numpy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behaviour is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

## Pynput

The pynput library allows us to control and monitor input devices. It contains subpackages for each type of input device supported:

* + - * pynput.mouse**-**Contains classes for controlling and monitoring a mouse or trackpad.
      * pynput.keyboard**-**Contains classes for controlling and monitoring the keyboard.

The package pynput.keyboard contains classes for controlling and monitoring the keyboard.[16]

A keyboard listener is a threading. Thread, and all callbacks will be invoked from the thread. Call pynput.keyboard.Listener.stop from anywhere, raise StopException or return False from a callback to stop the listener. The key parameter passed to callbacks is a pynput.keyboard.Key, for special keys, a pynput.keyboard. KeyCode for normal alphanumeric keys, or just None for unknown keys. When using the non-blocking version above, the current thread will continue executing. This might be necessary when integrating with other GUI frameworks that incorporate a main-loop, but when run from a script, this will cause the program to terminate immediately.

The keyboard listener thread

The listener callbacks are invoked directly from an operating thread on some platforms, notably Windows. This means that long running procedures and blocking operations should not be invoked from the callback, as this risks freezing input for all processes. A possible workaround is to just dispatch incoming messages to a queue, and let a separate thread handle them.

Handling keyboard listener errors

If a callback handler raises an exception, the listener will be stopped. Since callbacks run in a dedicated thread, the exceptions will not automatically be reraised. To be notified about callback errors, call Thread.join on the listener instance.

Toggling event listening for the keyboard listener

Once pynput.keyboard.Listener.stop has been called, the listener cannot be restarted, since listeners are instances of threading.Thread. If your application requires toggling listening events, you must either add an internal flag to ignore events when not required, or create a new listener when resuming listening.

Synchronous event listening for the keyboard listener

To simplify scripting, synchronous event listening is supported through the utility class pynput.keyboard. Events. This class supports reading single events in a non-blocking fashion, as well as iterating over all events.

Keyboard controller class functions

class pynput.keyboard.Controller A controller for sending virtual keyboard events to the system.

* exception InvalidCharacterException - The exception raised when an invalid character is encountered in the string passed to Controller.type(). Its first argument is the index of the character in the string, and the second the character.
* exception InvalidKeyException - The exception raised when an invalid key parameter is passed to either Controller.press() or Controller.release(). Its first argument is the key parameter.
* alt\_gr\_pressed - Whether altgr is pressed.
* alt\_pressed - Whether any alt key is pressed.
* ctrl\_pressed - Whether any ctrl key is pressed.
* modifiers - The currently pressed modifier keys. Only the generic modifiers will be set; when pressing either Key.shift\_l, Key.shift\_r or Key. shift, only Key.shift will be present.
* press(key) - Presses a key. A key may be either a string of length 1, one of the Key members or a KeyCode. Strings will be transformed to KeyCode using KeyCode.char(). Members of Key will be translated to their value().

Parameters key – The key to press. Raises

* + InvalidKeyException – if the key is invalid
  + ValueError – if key is a string, but its length is not 1
* pressed(\*args) - Executes a block with some keys pressed. Parameters keys – The keys to keep pressed.
* release(key) - Releases a key. A key may be either a string of length 1, one of the Key members or a KeyCode. Strings will be transformed to KeyCode using KeyCode.char(). Members of Key will be translated to their value().

Parameters key – The key to release. If this is a string, it is passed to touches() and the returned releases are used.

Raises

* + InvalidKeyException – if the key is invalid.
  + ValueError – if key is a string, but its length is not 1.
* shift\_pressed - Whether any shift key is pressed, or caps lock is toggled.
* tap(key) - Presses and releases a key. Parameters key – The key to press. Raises
  + InvalidKeyException – if the key is invalid.
  + ValueError – if key is a string, but its length is not 1.
* touch(key, is\_press) - Calls either press() or release() depending on the value of is\_press.

Parameters

* + key – The key to press or release.
  + is\_press (bool) – Whether to press the key.

Raises InvalidKeyException – if the key is invalid.

* type(string) - Types a string. This method will send all key presses and releases necessary to type all characters in the string.

Parameters string (str) – The string to type.

Raises InvalidCharacterException – if an untypable character is encountered.[17]

## FLOWCHART

Hand tracking model

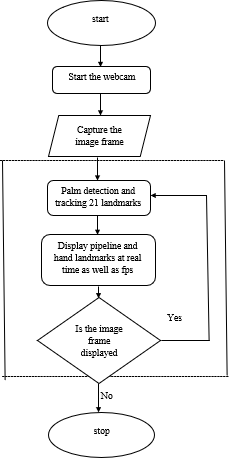


Fig 6.4.1 Hand Tracking Model

* + - In order to start the process we first need to take hand input
    - For capturing of the hand we need to import cvzone. With the help of videocapture we will get access to the camera.
    - After placing the hand in front of camera at a certain distance with the help of mediapipe it detects a maximum of 2 hands from the video, after that 21 landmark projections are displayed on each hand
    - Pipelines are used to join 21 landmarks.
    - If no hand is present in-front of the camera the landmarks won’t be detected.

Main model

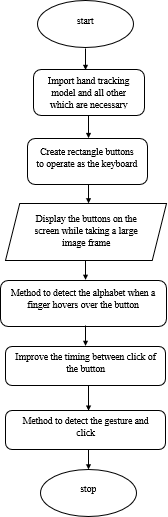


Fig 6.4.2 Main Model

* + - After creating hand landmarks next we need to design our keyboard.
    - We have created rectangle boxes by specifying the height, width, color for our keyboard.
    - After that we have to arrange the alphabets, special characters and remaining keys in the rectangle boxes which we have created earlier.
    - Next we have to import handtracking model. After importing the handtracking model when we place hands in-front of the screen the landmarks will be displayed
    - Whenever we place our finger on a particular key the respective key box colour changes and the size of the key increases
    - When the tip of index finger and middle finger are combined then the particular key will be entered at that time the colour of the particular key box changes to green.

## SOURCE CODE

* + 1. **Handtracking Module**

import cv2

import mediapipe as mp import math

class HandDetector:

def init (self, mode=False, maxHands=2, detectionCon=0.5, minTrackCon=0.5): self.mode = mode

self.maxHands = maxHands self.detectionCon = detectionCon self.minTrackCon = minTrackCon self.mpHands = mp.solutions.hands

self.hands = self.mpHands.Hands(self.mode, self.maxHands,

self.detectionCon, self.minTrackCon) self.mpDraw = mp.solutions.drawing\_utils

self.tipIds = [4, 8, 12, 16, 20] self.fingers = []

self.lmList = []

def findHands(self, img, draw=True):

imgRGB = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB) self.results = self.hands.process(imgRGB)

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

def findPosition(self, img, handNo=0, draw=True): xList = []

yList = [] bbox = [] bboxInfo =[]

self.lmList = []

if self.results.multi\_hand\_landmarks:

myHand = self.results.multi\_hand\_landmarks[handNo] for id, lm in enumerate(myHand.landmark):

h, w, c = img.shape

px, py = int(lm.x \* w), int(lm.y \* h) xList.append(px)

yList.append(py) self.lmList.append([px, py]) if draw:

cv2.circle(img, (px, py), 5, (255, 0, 255), cv2.FILLED) xmin, xmax = min(xList), max(xList)

ymin, ymax = min(yList), max(yList) boxW, boxH = xmax - xmin, ymax - ymin bbox = xmin, ymin, boxW, boxH

cx, cy = bbox[0] + (bbox[2] // 2), \ bbox[1] + (bbox[3] // 2)

bboxInfo = {"id": id, "bbox": bbox,"center": (cx, cy)} if draw:

cv2.rectangle(img, (bbox[0] - 20, bbox[1] - 20),

(bbox[0] + bbox[2] + 20, bbox[1] + bbox[3] + 20), (0, 255, 0), 2)

return self.lmList, bboxInfo

def fingersUp(self):

if self.results.multi\_hand\_landmarks: myHandType = self.handType() fingers = []

# Thumb

if myHandType == "Right":

if self.lmList[self.tipIds[0]][0] > self.lmList[self.tipIds[0] - 1][0]: fingers.append(1)

else:

fingers.append(0)

else:

if self.lmList[self.tipIds[0]][0] < self.lmList[self.tipIds[0] - 1][0]: fingers.append(1)

else:

fingers.append(0)

for id in range(1, 5):

if self.lmList[self.tipIds[id]][1] < self.lmList[self.tipIds[id] - 2][1]: fingers.append(1)

else:

fingers.append(0) return fingers

def findDistance(self, p1, p2, img, draw=True): if self.results.multi\_hand\_landmarks:

x1, y1 = self.lmList[p1][0], self.lmList[p1][1] x2, y2 = self.lmList[p2][0], self.lmList[p2][1] cx, cy = (x1 + x2) // 2, (y1 + y2) // 2

if draw:

cv2.circle(img, (x1, y1), 15, (255, 0, 255), cv2.FILLED)

cv2.circle(img, (x2, y2), 15, (255, 0, 255), cv2.FILLED)

cv2.line(img, (x1, y1), (x2, y2), (255, 0, 255), 3)

cv2.circle(img, (cx, cy), 15, (255, 0, 255), cv2.FILLED)

length = math.hypot(x2 - x1, y2 - y1) return length, img, [x1, y1, x2, y2, cx, cy]

def handType(self):

if self.results.multi\_hand\_landmarks:

if self.lmList[17][0] < self.lmList[5][0]: return "Right"

else:

return "Left"

def main():

cap = cv2.VideoCapture(0)

detector = HandDetector(detectionCon=0.8, maxHands=1) while True:

# Get image frame success, img = cap.read()

# Find the hand and its landmarks img = detector.findHands(img)

lmList, bboxInfo = detector.findPosition(img) print(detector.handType()) cv2.imshow("Image", img)

cv2.waitKey(1)

if name == " main ": main()

## Main Module

import cv2

from cvzone.HandTrackingModule import HandDetector from time import sleep

import numpy as np import cvzone

from pynput.keyboard import Controller cap = cv2.VideoCapture(0)

cap.set(3, 1280)

cap.set(4, 720)

detector = HandDetector(detectionCon=0.5)

keys = [["1", "2", "3", "4", "5", "6", "7", "8", "9", "0", "\_", "="],

["Q", "W", "E", "R", "T", "Y", "U", "I", "O", "P", "[", "]"],

["A", "S", "D", "F", "G", "H", "J", "K", "L", ";", "'", ":"],

["Z", "X", "C", "V", "B", "N", "M", ",", ".", "/", "\"", "|"],

["!", "@", "#", "$", "%", "^", "&", "\*", "(", ")", "\\", "`"],

["~", "<", ">", "?", "-", "+", "\*", "{", "}", "?", " "]]

finalText = ""

keyboard = Controller()

def drawAll(img, buttonList): for button in buttonList:

x, y = button.pos

w, h = button.size

cvzone.cornerRect(img, (button.pos[0], button.pos[1], button.size[0], button.size[1]), 20,

rt=0)

cv2.rectangle(img, button.pos, (x + w, y + h), (255, 0, 255), cv2.FILLED)

cv2.putText(img, button.text, (x + 10, y + 25), cv2.FONT\_HERSHEY\_PLAIN, 2, (255,

255, 255), 2)

return img class Button():

def init (self, pos, text, size=[40, 40]): self.pos = pos

self.size = size self.text = text

buttonList = []

for i in range(len(keys)):

for j, key in enumerate(keys[i]): buttonList.append(Button([55 \* j , 60 \* i + 20 ], key))

while True:

success, img = cap.read()

Vimg = detector.findHands(img)

lmList, bboxInfo = detector.findPosition(img) img = drawAll(img, buttonList)

if lmList:

for button in buttonList: x, y = button.pos

w, h = button.size

if x < lmList[8][0] < x + w and y < lmList[8][1] < y + h:

cv2.rectangle(img, (x - 2, y - 2), (x + w + 2, y + h + 2), (175, 0, 175), cv2.FILLED) cv2.putText(img, button.text, (x+10 , y+30 ), cv2.FONT\_HERSHEY\_PLAIN, 3,

(255, 255, 255), 3)

l, \_, \_ = detector.findDistance(8, 12, img, draw=False) print(l)

## when clicked if l < 30:

keyboard.press(button.text)

cv2.rectangle(img, button.pos, (x + w, y + h), (0, 255, 0), cv2.FILLED) cv2.putText(img, button.text, (x+10 , y+30 ), cv2.FONT\_HERSHEY\_PLAIN, 3,

(255, 255, 255), 3)

finalText += button.text sleep(0.2)

cv2.rectangle(img, (50, 400), (700, 550), (175, 0, 175), cv2.FILLED)

cv2.putText(img, finalText, (60, 450), cv2.FONT\_HERSHEY\_PLAIN, 3, (255, 255, 255),

3)

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

## OUTPUT SCREENS

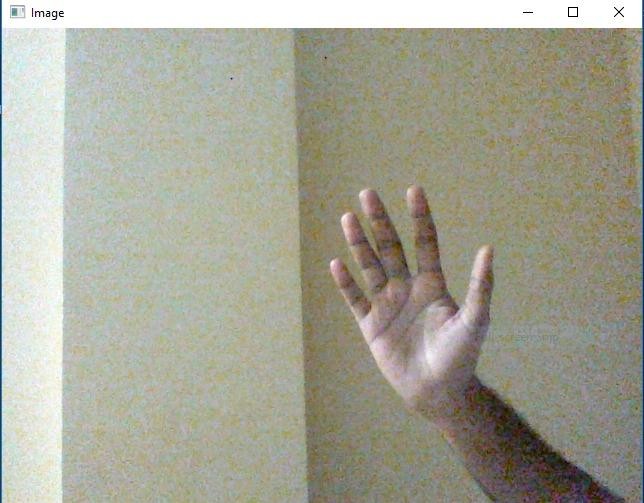


Fig 6.6.1 Hand with no landmarks

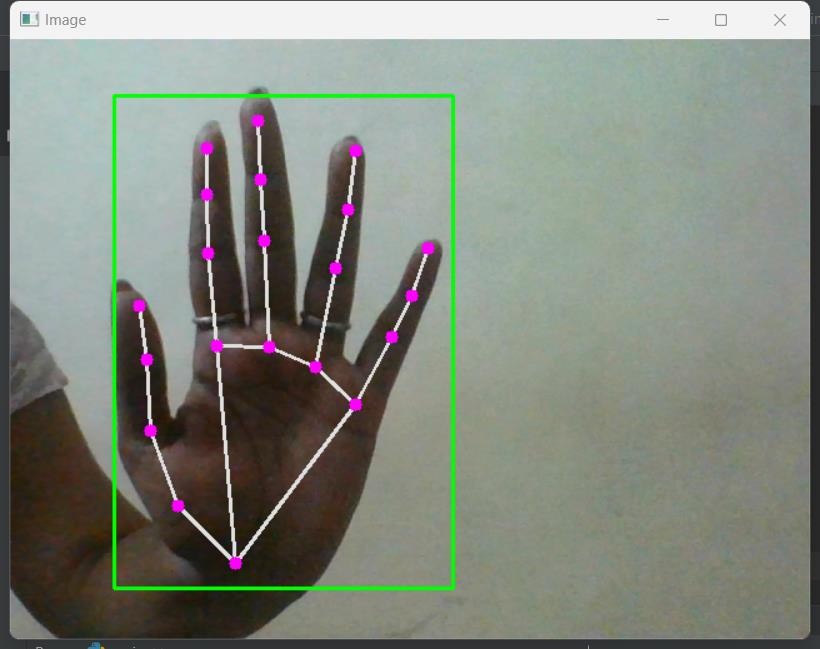


Fig 6.6.2 Hand with landmarks

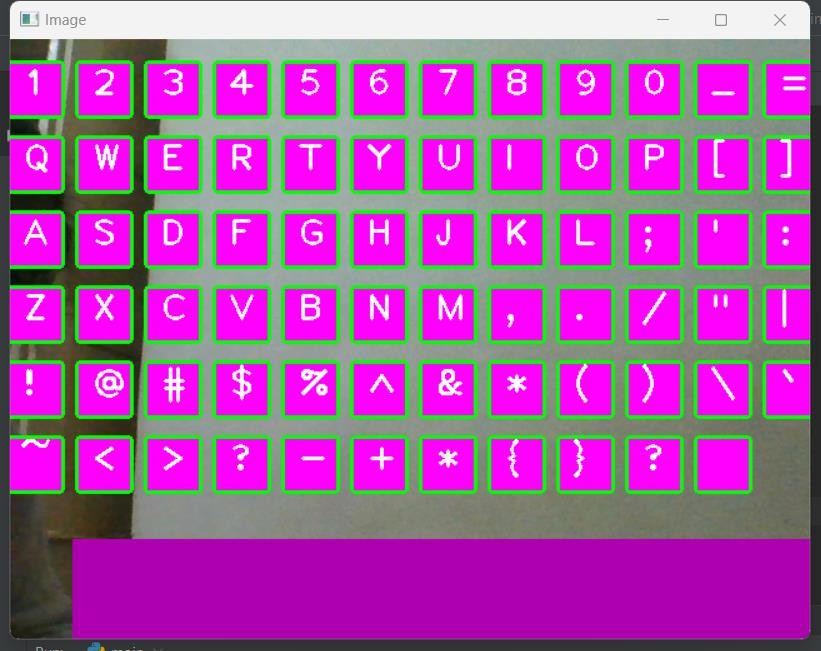


Fig 6.6.3 Keyboard

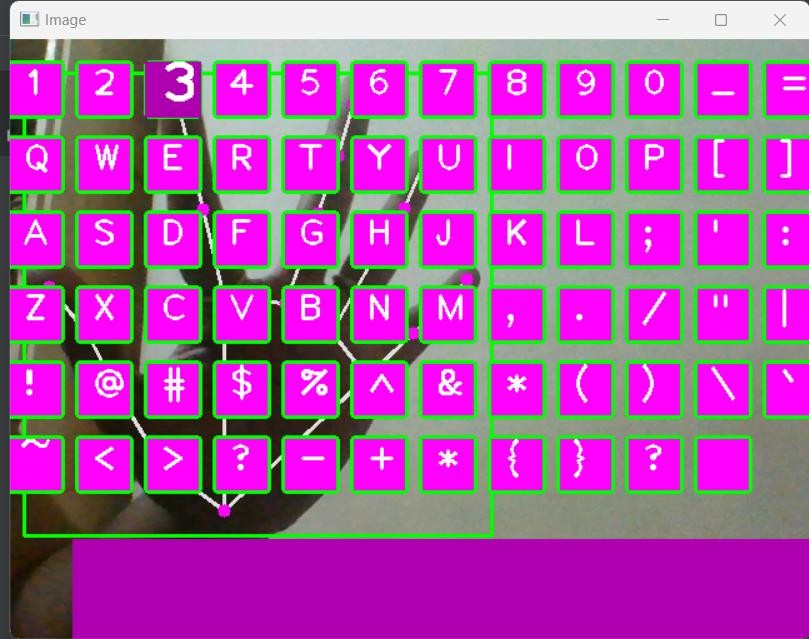


Fig 6.6.4 Key hovering when finger placed on it

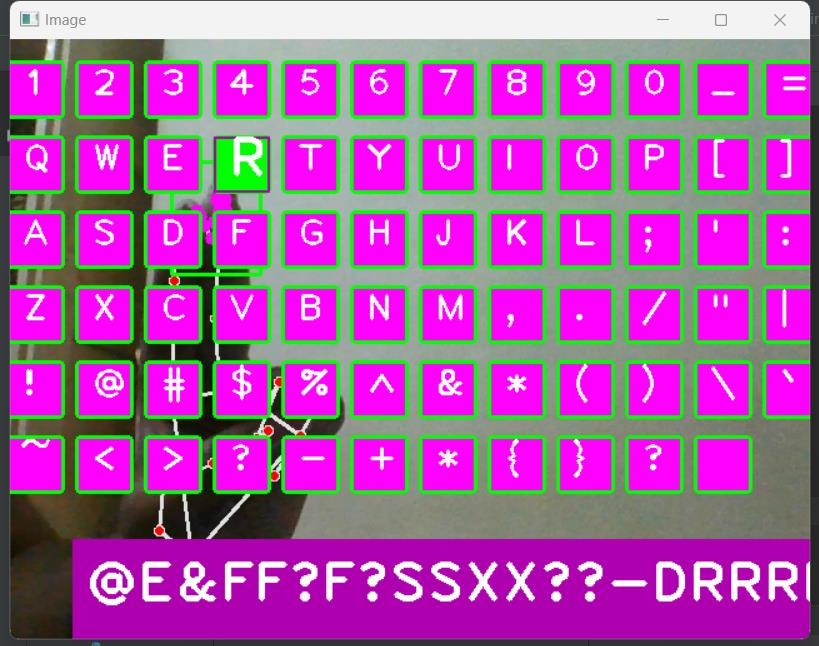


Fig 6.6.5 Keyboard when key pressed

## SOFTWARE TESTING

Testing involves operation of a system or application under controlled conditions and evaluating the results. The controlled conditions should include both normal and abnormal conditions. Testing should intentionally attempt to make things go wrong to determine if things happen when they shouldn't or things don't happen when they should. It is oriented to 'detection'.

## Unit Testing

Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. This testing mode is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

Unit tests are written from a programmer's perspective. They ensure that a particular method of a class successfully performs a set of specific tasks. Each test confirms that a method produces the expected output when given a known input.

## Performance Testing

Performance testing is the process of determining the speed or effectiveness of a computer, network, software program or device. This process can involve quantitative tests done in a lab, such as measuring the response time or the number of MIPS (millions of instructions per second) at which a system functions. Qualitative attributes such as Reliability, scalability and interoperability may also be evaluated. Performance testing is often done in conjunction with stress testing.

Performance testing can verify that a system meets the specifications claimed by its manufacturer or vendor. The process can compare two or more devices or programs in terms of parameters such as speed, data transfer rate, bandwidth, throughput, efficiency or reliability.

Performance testing can also be used as a diagnostic aid in locating communications bottlenecks. Often a system will work much better if a problem is resolved at a single point or in a single component.

## Integration Testing

Integration testing, also known as integration and testing (I&T), is a software development process which program units are combined and tested as groups in multiple ways. In this context, a unit is defined as the smallest testable part of an application. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

## Test Cases

Test case 1

|  |  |  |
| --- | --- | --- |
| Test no | Tester | Mismatched keys |
| 1 | A | Z |
| 2 | B | G |
| 3 | D | T |
| 4 | K | A |
| 5 | S | W |
| 6 | V | Y |

Table 6.7.4.1 Test Case 1

Test case 2

|  |  |  |
| --- | --- | --- |
| Test no | Tester | Mismatched keys |
| 1 | A | N/A |
| 2 | B | N/A |
| 3 | D | N/A |
| 4 | 6 | N/A |
| 5 | $ | N/A |
| 6 | \* | N/A |

Table 6.7.4.2 Test Case 2

* 1. **CONCLUSION**

# CONCLUSION

The hand gesture based keyboard was developed in the python language, using the OpenCV library. The system was able to control the keyboard by tracking the user hands with the help of real time camera. The keyboard is simulated as an actual keyboard as well as the alphabets are well aligned. The result includes key press and key functionalities. The system designed can be worked in all conditions just requirement is a bright light and hand as well as a webcam for fast detection and better result. The system has the potential of being a viable replacement for the computer keyboard. The system efficiency will help us in saving space at work, reduce plastic waste by eliminating the keyboard.

## FUTURE SCOPE

System is quite autonomous. It’s one of the best features is that it is easy to use. Future work will be focused on algorithm improvement by merging the models created and making an entire virtual system handler such that it has functionalities of virtual mouse and keyboard as well as it can control the volume, brightness and other functionalities. It also includes improvement in keyboard. We plan to add a caps-lock button. Also, a button that switches the keyboard characters to special characters so that they can be used when they are needed. It can even be implemented in gaming industry.

# REFERENCES

1. Research on recognizing required items based on opencv and machine learning by Researchgate (2022).
2. Pradnya Kedari, Shubhangi Kadam, Rajesh Prasad,” Controlling Computer using Hand Gestures”, (IRJET) 2022.
3. Sam Sunny, Sindhu V,” Hand Gesture Based Keyboard Using Machine Learnung”, in 2020
4. K. S. Abhishek, “ Glove-based hand gesture recognition sign language translator using capacitive touch sensor”, in 2016
5. A. V. Dehankar, Sanjeev Jain, “ Using AEPI method for hand gesture recognition in varying background and blurred images”, in 2017.
6. Ashok, M. A., Revathi, M. P., & Nirmal, B. “FINGER RECONGITION AND GESTURE BASED VIRTUAL KEYBOARD.” Turkish Journal of Computer and Mathematics Education (TURCOMAT), 11(3), 2237-2245, in 2020
7. Rishikesh Kumar, Poonam Chaudhary,” User defined custom virtual keyboard”, in IEEE 2017
8. Pravin Kshirsagar, “ Gesture based motion tracking keyboard using machine learning”, in 2020
9. <https://www.geeksforgeeks.org/python-play-a-video-using-opencv/>
10. https://developers.google.com/mediapipe/solutions/vision/hand\_landmarker
11. <https://www.arxiv-vanity.com/papers/2006.10214/>
12. <https://intellipaat.com/blog/what-is-pycharm/>
13. <https://www.jetbrains.com/pycharm/>
14. Open CV [Online]. Available: <https://opencv.org/>
15. <https://realpython.com/python-time>module
16. <https://pynput.readthedocs.io/en/latest/>
17. <https://buildmedia.readthedocs.org/media/pdf/pynput/latest/pynput.pdf>