A Minor Project Report On

"GESTURAMA – A Gesture Detection System"

Submitted in Partial Fulfilment of the Requirements

For the Award

Of

Degree of B.Tech

To

Guru Gobind Singh Indraprastha University, Delhi

Under the guidance of Ms. Upasna Joshi

Submitted By:

Gaurav Birdi (01918002718)

Prakhar Katiyar (40118002718)

Md Omer (03018002718)

Sachin Sharma (04418002718)

Affiliated to GGSIP University, New Delhi
Approved by AICTE & Council of Architecture

COMPUTER SCIENCE DEPARTMENT,

DELHI TECHNICAL CAMPUS,

KNOWLEDGE PARK-III, GREATER NOIDA

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled "GESTURAMA – A Gesture Detection System", in fulfilment of the requirement for the award of the degree Bachelor of Technology in Computer Science and Engineering, submitted in Computer Science and Engineering Department, DTC affiliated to Guru Gobind Singh Indraprastha University, New Delhi, is an authentic record of my own work carried out during my degree under the guidance of Ms. Upasna Joshi.

The work reported in this has not been submitted by me for award of any other degree or diploma.

Date: 22/12/2021
Place: Delhi Technical Campus

Gaurav Birdi (01918002718)
Md Omer (03018002718)
Prakhar Katiyar (40118002718)
Sachin Sharma (04418002718)

CERTIFICATE

This is to certify that the Project work entitled “GESTURAMA – A Gesture Detection System” submitted by Gaurav Birdi, Md Omer, Prakhar Katiyar and Sachin Sharma in fulfilment for the requirements of the award of Bachelor of Technology Degree in Computer Science and Engineering at DTC, Greater Noida is an authentic work carried out under my supervision and guidance. To the best of my knowledge, the matter embodied in the project has not been submitted to any other University / Institute for the award of any Degree.

Date: 22/12/2021

Ms. Upasna Joshi
(Asst. Prof., CSE)

ACKNOWLEDGEMENT

I express my sincere gratitude to Dr. Shipra Saraswat (HOD, CSE) and Ms. Upasna Joshi(A.P, CSE), Delhi Technical Campus, for her valuable guidance and timely suggestions during the entire duration of my dissertation work, without which this work would not have been possible. I would also like to convey my deep regards to all other faculty members of Computer Science Department, who have bestowed their great effort and guidance at appropriate times without which it would have been very difficult on my part to finish this work. Finally, I would also like to thank my friends for their advice and pointing out my mistakes.

ABSTRACT

Communications between deaf-mute and a normal person have always been a challenging task. This paper reviews a different method that we have adopted to reduce that particular barrier of communication by developing a software for providing the required help to that particular community which is in need of such help. For doing that, first we have to gather knowledge about hand gestures. Now, what hand gestures actually are?

Hand gestures are a form of nonverbal communication that can be used in several fields such as communication between deaf-mute people, robot control, human–computer interaction etc. Similarly, there is one other concept which is termed as sign language. Now, what is it?

Sign language is a physical action by using hands and eye with which we can communicate with dumb and deaf people. They can express their felling with different hand shapes and movement. While communicating with dumb and deaf peoples, those who have knowledge of sign language, can talk and hear properly. But untrained people cannot communicate with dumb and deaf people, because the person can communicate to dumb people by training sign language. Sign language to text system will be more useful for such impaired people to communicate with normal people more fluently. The Solution to tackle such problem is that to develop such a software which easily recognize the sign language and convert it into a form which untrained people can understand with ease.

What this word “Sign Language Recognitions” is that it is a breakthrough for helping deaf-mute people and has been researched for many years. Unfortunately, every research has its own limitations and are still unable to be used commercially. Some of the researches have known to be successful for recognizing sign language, but require an expensive cost to be commercialized.

The task is to convert that shape or their sign language into text or speech. Due to advancement in the field of image processing, an automatic sign language converter system can be put to good use. Also, few researchers have developed tools to help to convert sign language into text or speech.

Sensor output signals are sending to the computer for processing and analyse the gesture and convert into text or speech. In image processing, image is captured through sensory detectors. By putting all these together, the software that we desire to develop will prove out to be an essential aspect that can be used for a healthy growth of the society. The section 2 describes the efficiency of techniques used for sign language into text conversion.

Table of Contents

List of Figures

Chapter-1 Introduction

1.1 Aim and Objective of the Project

According to World Health Organization (WHO), about 285 million people are visually impaired worldwide, 466 million people have disabling hearing loss and 1 million people are dumb. These specially aided people are discriminated and mostly suffer in various sectors of our society. One of the major shortcomings of society is the social barrier that is created between disabled or handicapped persons and persons who are blessed with all their human faculties in order. Communication, which is the basis of human progress, often tends to be an obstacle for those unfortunate people who are unable to articulate their thoughts. The root cause of this gap is that while deaf and mute persons use sign language to communicate among themselves, normal people are either reluctant to learn it or are unable to comprehend the same. Further, the former relies on lip reading to comprehend what their counterparts have said. Consequently, in a conversation between a hearing and speech impaired person and a normal person the ease of communication and hence the comfort level is hampered.

Many of these people are not even exposed to sign languages and it is observed that it gives a great relief on a psychological level, when they find out about signing to connect themselves with others by expressing their love or emotions. About 5% population in world are suffering from hearing loss. Deaf and dumb people use sign language as their primary means to express their thoughts and ideas to the people around them with different hand and body gestures. There are only about 250 certified sign language interpreters in India for a deaf population of around 7 million.

Nowadays, the evolution of technology and the increasing use of computers gave the opportunity for developing new methods of education of deaf individuals and sign language interpreters.

In this project we aspire to help these specially-abled people by developing a software which could act as a communication bridge between these two sections of the society. Our project detects the signs used in sign language and convert it into something that everyone can easily understand.

1.2 Data Flow Diagram

Data flow diagram is graphical representation of flow of data in an information system. It is capable of depicting incoming data flow, outgoing data flow and stored data. The DFD does not mention anything about how data flows through the system.

There is a prominent difference between DFD and Flowchart. The flowchart depicts flow of control in program modules. DFDs depict flow of data in the system at various levels. DFD does not contain any control or branch elements.

Types of DFD :-

Data Flow Diagrams are either Logical or Physical

- Logical DFD - This type of DFD concentrates on the system process, and flow of data in the system. For example: In a Banking software system, how data is moved between different entities.

- Physical DFD - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

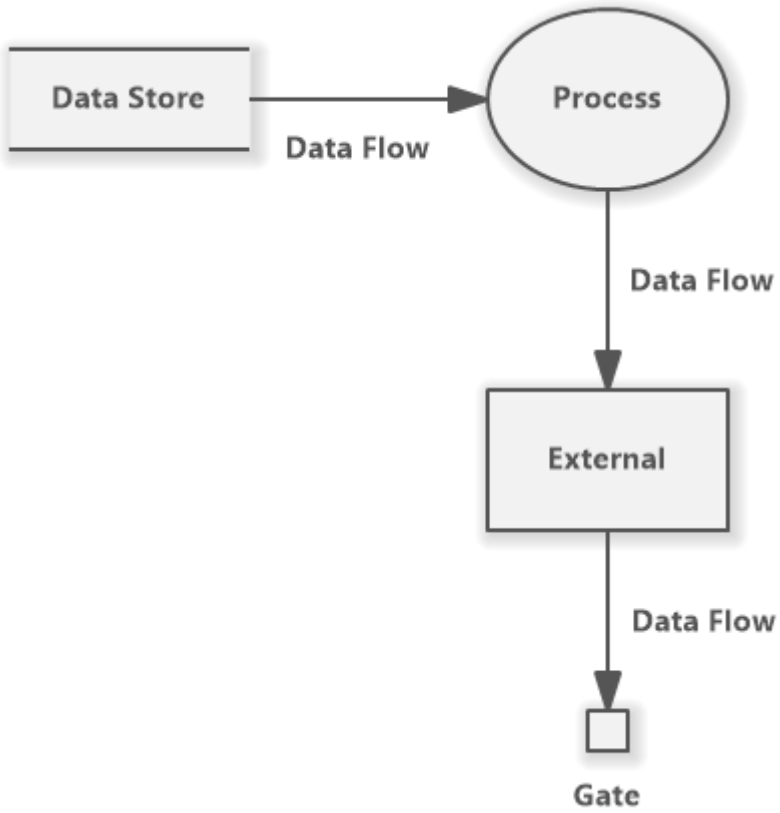


Fig 1.1

DFD Components:

DFD can represent Source, destination, storage and flow of data using the following set of components –

- **Entities** – Entities are source and destination of information data. Entities are represented by a rectangle with their respective names.
- **Process** – Activities and action taken on the data are represented by Circle or Roundedged rectangles.
- **Data Storage** – There are two variants of data storage – it can either be represented as a rectangle with absence of both smaller sides or as an open-sided rectangle with only one side missing.
- **Data Flow** – Movement of data is shown by pointed arrows. Data movement is shown from the base of arrow as its source towards head of the arrow as destination.

Levels of DFD:

- Level 0 – Highest abstraction level DFD is known as Level 0 DFD, which depicts the entire information system as one diagram concealing all the underlying details. Level 0 DFDs are also known as context level DFDs.
- Level 1 – The Level 0 DFD is broken down into more specific, Level 1 DFD. Level 1 DFD depicts basic modules in the system and flow of data among various modules. Level1 DFD also mentions basic processes and sources of information.

- Level 2 – At this level, DFD shows how data flows inside the modules mentioned in Level 1. Higher level DFDs can be transformed into more specific lower level DFDs with deeper level of understanding unless the desired level of specification is achieved.

1.3 Data Flow Diagram of the Project

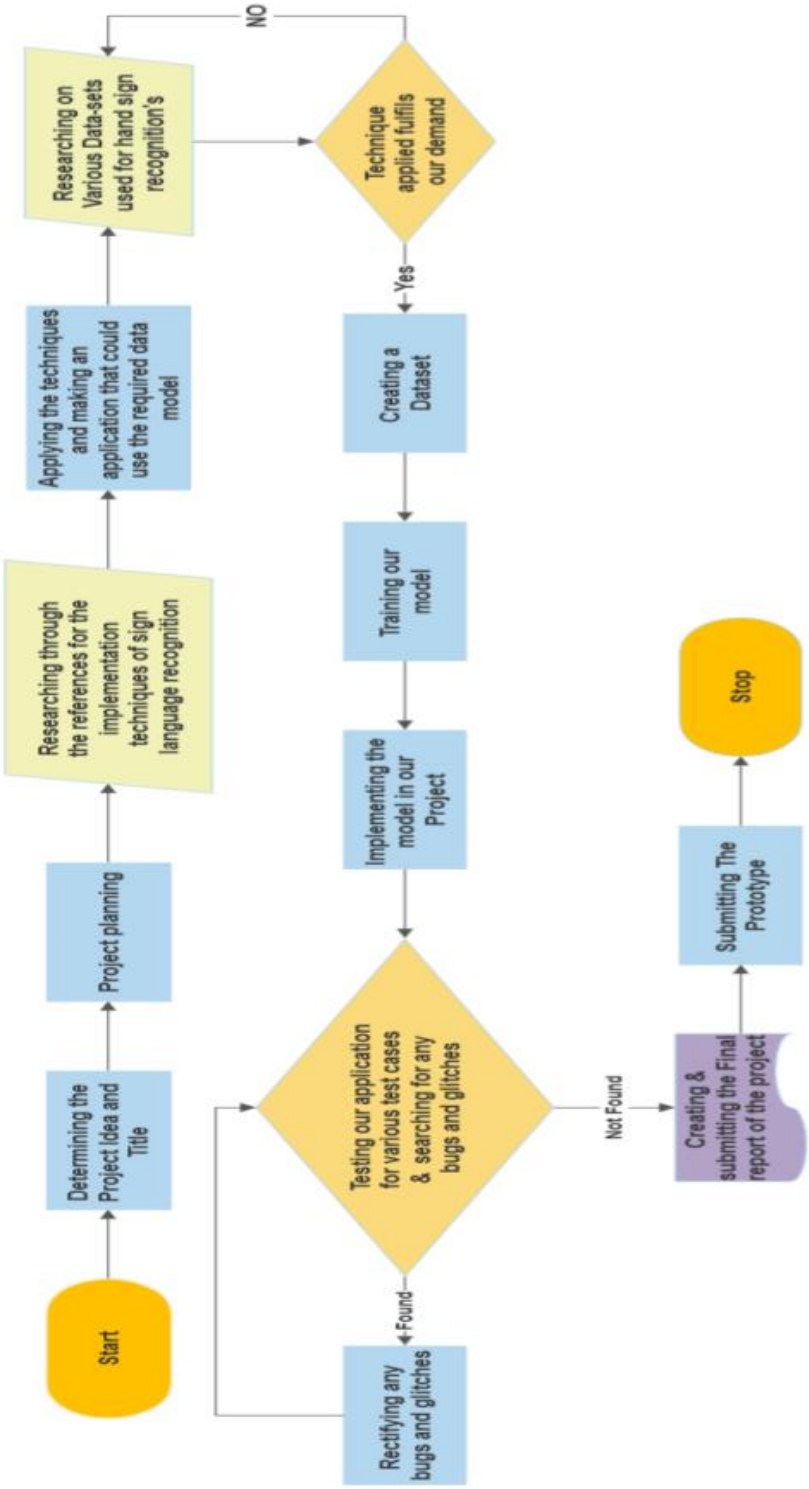


Fig 1.2

Chapter-2 Gantt Chart

2.1 Introduction to Gantt Chart

A Gantt chart, commonly used in project management, is one of the most popular and useful ways of showing activities (tasks or events) displayed against time. On the left of the chart is a list of the activities and along the top is a suitable time scale. Each activity is represented by a bar; the position and length of the bar reflects the start date, duration and end date of the activity. This allows you to see at a glance:

- What the various activities are
- When each activity begins and ends
- How long each activity is scheduled to last
- Where activities overlap with other activities, and by how much
- The start and end date of the whole project

Today, Gantt charts are most commonly used for tracking project schedules. For this it is useful to be able to show additional information about the various tasks or phases of the project, for example how the tasks relate to each other, how far each task has progressed, what resources are being used for each task and so on.

2.2 Project Gantt Chart

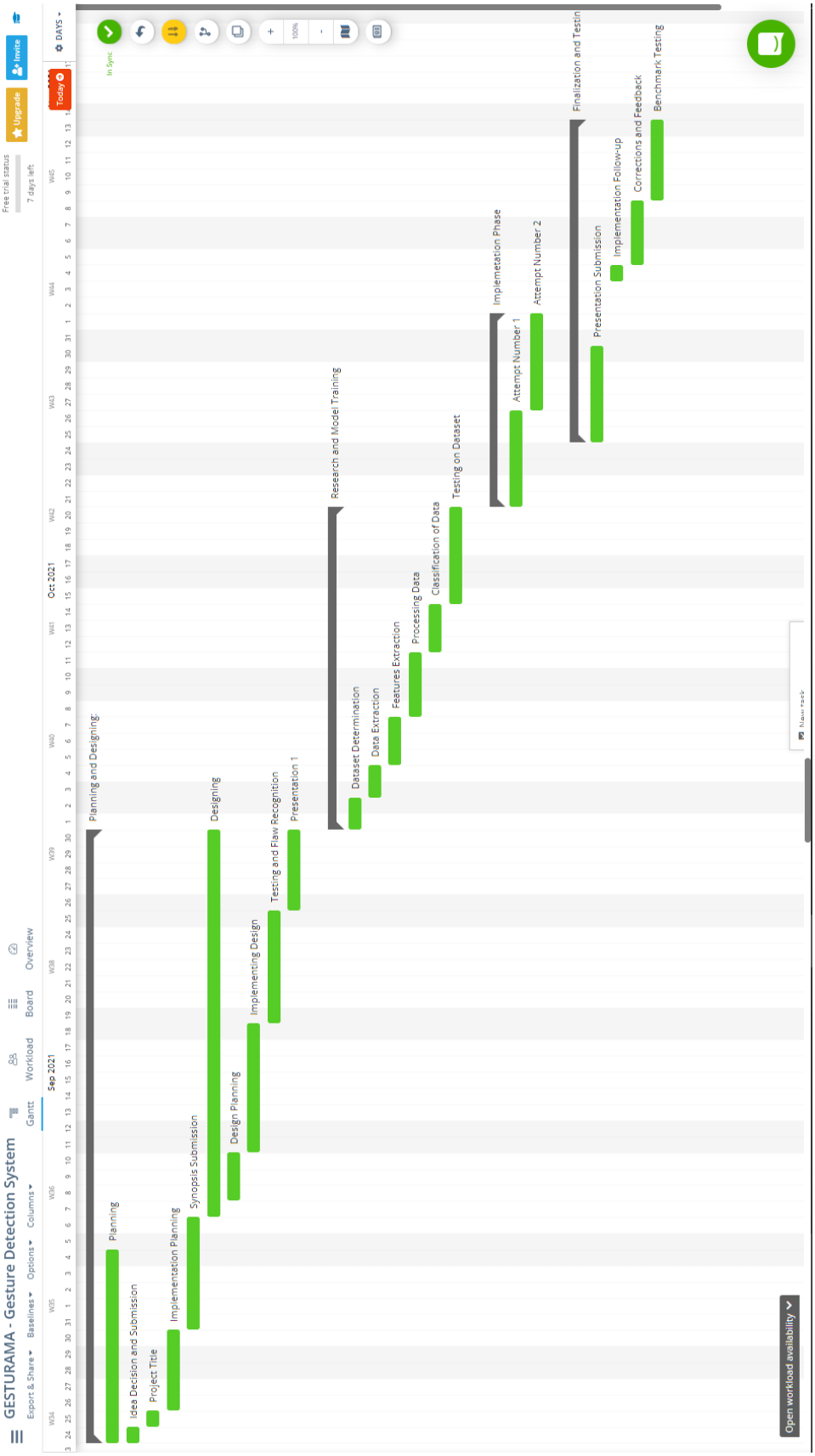


Fig 2.1

	Planning and Designing:		-	24/Aug	30/Sep	100%	
1	Planning	Unassigned	-	24/Aug	04/Sep	100%	
2	Idea Decision and Submissi...	Unassigned	-	24/Aug	24/Aug	100%	
3	Project Title	Unassigned	-	25/Aug	25/Aug	100%	
4	Implementation Planning	Unassigned	-	26/Aug	30/Aug	100%	
5	Synopsis Submission	Unassigned	-	31/Aug	06/Sep	100%	
6	Designing	Unassigned	-	07/Sep	30/Sep	100%	
7	Design Planning	Unassigned	-	08/Sep	10/Sep	100%	
8	Implementing Design	Unassigned	-	11/Sep	18/Sep	100%	
9	Testing and Flaw Recognition	Unassigned	-	19/Sep	25/Sep	100%	
10	Presentation 1	Unassigned	-	26/Sep	30/Sep	100%	
	+ Add task + Add section						
	Research and Model Training		-	01/Oct	20/Oct	100%	
13	Dataset Determination	Unassigned	-	01/Oct	02/Oct	100%	
14	Data Extraction	Unassigned	-	03/Oct	04/Oct	100%	
15	Features Extraction	Unassigned	-	05/Oct	07/Oct	100%	
16	Processing Data	Unassigned	-	08/Oct	11/Oct	100%	
17	Classification of Data	Unassigned	-	12/Oct	14/Oct	100%	
18	Testing on Dataset	Unassigned	-	15/Oct	20/Oct	100%	
	+ Add task + Add section						
	Implemetation Phase		-	21/Oct	01/Nov	100%	
21	Attempt Number 1	Unassigned	-	21/Oct	26/Oct	100%	
22	Attempt Number 2	Unassigned	-	27/Oct	01/Nov	100%	
	+ Add task + Add section						
	Finalization and Testing		-	25/Oct	13/Nov	100%	
25	Presentation Submission	Unassigned	-	25/Oct	30/Oct	100%	
26	Implementation Follow-up	Unassigned	-	04/Nov	04/Nov	100%	
27	Corrections and Feedback	Unassigned	-	05/Nov	08/Nov	100%	
28	Benchmark Testing	Unassigned	-	09/Nov	13/Nov	100%	
	+ Add task + Add section						

Fig 2.2

Chapter-3 Requirements and Tools

3.1 HARDWARE REQUIREMENTS

- PC/Laptop with Webcam connectivity
- Processor: Core 2 duo and above
- Processor Speed: 2.4 GHz and up
- System Memory: 1GB recommended
- Hard Disk: 100GB
- 1GB of RAM (Minimum)
- Monitor: SVGA Color 14”
- Keyboard: 104 US keys
- Mouse: Trackball
- USB ports for external camera attachment
- HD Webcam/External Camera

Higher configurations are also supported

3.2 SOFTWARE REQUIREMENTS

- One of these Text Editors
  - Pycharm
  - Jupyter Notebook
  - Notepad
  - PythonIDE
- Python 3.7 and following libraries –
  - OpenCV
  - TensorFlow
  - Mediapipe
  - Cvzone
- Microsoft Office.
- Keras Dataset
- Web Browser- Chrome/ Firefox
- OS- Windows 10

Chapter-4 Python

4.1 Introduction to Python

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. It is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

4.2 History of Python

Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica(CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the operating system Amoeba. Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, Benevolent Dictator For Life (BDFL).

Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release the development process was changed and became more transparent and community-backed.

Python 3.0 (initially described as Python 3000 or py3k), is a major, backward-incompatible release that was released after a long period of testing on 3 December 2008. Many of its major features have been backported to the backwards-compatible Python 2.6.x and 2.7.x version series.

4.3 Features of Python

Python is a multi-paradigm programming language: object-oriented programming and structured programming are fully supported, and many language features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.
Python uses dynamic typing and a mix of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

The design of Python offers some support for functional programming in the Lisp tradition. The language has filter(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions.[44] The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

The core philosophy of the language is summarized by the document The Zen of Python (PEP 20), which includes aphorisms such as:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated
- Readability counts

Some of the features are:

- **Simple**: Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.
- **Easy to Learn**: As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.
- **Free and Open Source**: Python is an example of a FLOSS (Free/Libré and Open Source Software). In simple terms, you can freely distribute copies of this software, read its source code, make changes to it, and use pieces of it in new free programs. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.
- **High-level Language**: When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.
- **Portable**: Due to its open-source nature, Python has been ported to (i.e. changed to make it work on) many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features. You can use Python on GNU/Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and PocketPC! You can even use a platform like Kivy to create games for your computer and for iPhone, iPad, and Android.
- **Interpreted**: This requires a bit of explanation. A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you

run the program, the linker/loader software copies the program from hard disk to memory and starts running it. Python, on the other hand, does not need compilation to binary. You just run the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

- **Object Oriented:** Python supports procedure-oriented programming as well as objectoriented programming. In procedure-oriented languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In objectoriented languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.
- **Extensible**: If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use it from your Python program.
- **Embeddable**: You can embed Python within your C/C++ programs to give scripting capabilities for your program's users.
- **Extensive Libraries**: The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, FTP, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the Batteries Included philosophy of Python. Besides the standard library, there are various other high-quality libraries which you can find at the Python Package Index.

4.4 Libraries

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, many standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.

Some parts of the standard library are covered by specifications (for example, the Web Server Gateway Interface (WSGI) implementation *wsgiref* follows PEP 333), but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of September 2017, the Python Package Index, the official repository containing third-party software for Python, contains over 117,000 packages offering a wide range of functionality.

It also includes libraries for:

- Graphical user interfaces, web frameworks, multimedia, databases, networking
- Test frameworks, automation and web scraping, documentation tools, system administration
- Scientific computing, text processing, image processing

4.5 Python Variables and Data Types

Python Variables

A variable is a location in memory used to store some data (value). They are given unique names to differentiate between different memory locations. The rules for writing a variable name is same as the rules for writing identifiers in Python.

We don't need to declare a variable before using it. In Python, we simply assign a value to a variable and it will exist. We don't even have to declare the type of the variable. This is handled internally according to the type of value we assign to the variable.

Variable assignment

We use the assignment operator (=) to assign values to a variable. Any type of value can be assigned to any valid variable.

```
a = 5    b = 3.2          c = "Hello"
```

Here, we have three assignment statements. 5 is an integer assigned to the variable a.

Similarly, 3.2 is a floating point number and "Hello" is a string (sequence of characters) assigned to the variables b and c respectively.

Multiple assignments

In Python, multiple assignments can be made in a single statement as follows:

```
a, b, c = 5, 3.2, "Hello"
```

If we want to assign the same value to multiple variables at once, we can do this as `x = y = z = "same"`

This assigns the "same" string to all the three variables.

Data types in Python

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

There are various data types in Python. Some of the important types are listed below.

Python Numbers
Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.

We can use the type() function to know which class a variable or a value belongs to and the isinstance() function to check if an object belongs to a particular class.

Integers can be of any length, it is only limited by the memory available.

A floating point number is accurate up to 15 decimal places. Integer and floating points are separated by decimal points. 1 is integer, 1.0 is floating point number.

Complex numbers are written in the form, x + yj, where x is the real part and y is the imaginary part. Here are some examples:

```
>>> a = 1234567890123456789
>>> a
1234567890123456789
>>> b = 0.1234567890123456789
>>> b
0.12345678901234568
>>> c = 1+2j
>>> c
(1+2j)
```

Notice that the float variable b got truncated.

Python List

List is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible. All the items in a list do not need to be of the same type.

Declaring a list is pretty straight forward. Items separated by commas are enclosed within brackets [ ].

```
>>> a = [1, 2.2, 'python']
```

We can use the slicing operator [ ] to extract an item or a range of items from a list. Index starts form 0 in Python.

Lists are mutable, meaning, value of elements of a list can be altered.

```
>>> a = [1,2,3]
>>> a[2]=4
>>> a
[1, 2, 4]
```

Python Tuple

Tuple is an ordered sequence of items same as list. The only difference is that tuples are immutable. Tuples once created cannot be modified. Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically. It is defined within parentheses () where items are separated by commas.

```
>>> t = (5,'program', 1+3j)
```

We can use the slicing operator [] to extract items but we cannot change its value.

Python Strings

String is sequence of Unicode characters. We can use single quotes or double quotes to represent strings. Multi-line strings can be denoted using triple quotes, ''' or """.

```
>>> s = "This is a string"
>>> s = '''a multiline
```

Like list and tuple, slicing operator [ ] can be used with string. Strings are immutable.

Python Set

Set is an unordered collection of unique items. Set is defined by values separated by comma inside braces { }. Items in a set are not ordered.

We can perform set operations like union, intersection on two sets. Set have unique values. They eliminate duplicates.

```
>>> a = {1,2,2,3,3,3}
>>> a
{1, 2, 3}
```

Since, set are unordered collection, indexing has no meaning. Hence the slicing operator [] does not work.

Python Dictionary

Dictionary is an unordered collection of key-value pairs. It is generally used when we have a huge amount of data. Dictionaries are optimized for retrieving data. We must know the key to retrieve the value. In Python, dictionaries are defined within braces {} with each item being a pair in the form key:value. Key and value can be of any type.

```
>>> d = {1:'value','key':2}
>>> type(d)
<class 'dict'>
```

We use key to retrieve the respective value. But not the other way around.
We can convert between different data types by using different type conversion functions like int(), float(), str() etc.

```
>>> float(5)
5.0
```

Conversion from float to int will truncate the value (make it closer to zero).

```
>>> int(10.6)
10
>>> int(-10.6)
-10
```

Conversion to and from string must contain compatible values.

```
>>> float('2.5')
2.5
```

```
>>> str(25)
'25'
>>> int('1p')
Traceback (most recent call last): File "<string>", line 301, in runcode
File "<interactive input>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '1p'
```

4.6 Usage

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

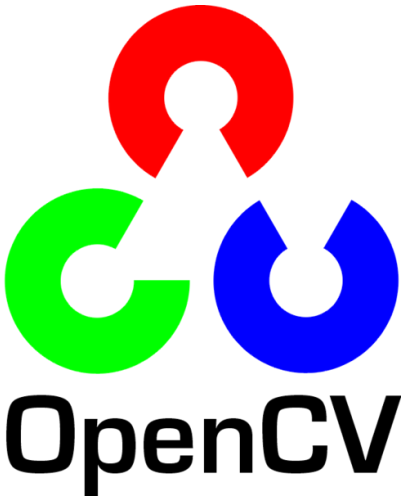
- Web development
- Game programming
- Desktop GUIs
- Scientific programming
- Network programming

Chapter-5 OpenCV

5.1 Introduction

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics. It is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage and is now maintained by Itseez. The library is cross-platform and free for use under the open-source BSD license. OpenCV supports the Deep Learning frameworks TensorFlow, Torch/PyTorch and Caffe.

OpenCV is written in C++ and its primary interface is in C++, but it still retains a less comprehensive though extensive older C interface. There are bindings in Python, Java and MATLAB/OCTAVE. The API for these interfaces can be found in the online documentation. Wrappers in other languages such as C#, Perl, Ch, Haskell and Ruby have been developed to encourage adoption by a wider audience. All of the new developments and algorithms in OpenCV are now developed in the C++ interface.



5.2 History

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

- Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.

- Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
- Advance vision-based commercial applications by making portable, performanceoptimized code available for free—with a license that did not require code to be open or free itself.

The first alpha version of OpenCV was released to the public at the IEEE Conference on Computer Vision and Pattern Recognition in 2000, and five betas were released between 2001 and 2005. The first 1.0 version was released in 2006. A version 1.1 "pre-release" was released in October 2008. The second major release of the OpenCV was in October 2009. OpenCV 2 includes major changes to the C++ interface, aiming at easier, more type-safe patterns, new functions, and better implementations for existing ones in terms of performance (especially on multi-core systems). Official releases now occur every six months and development is now done by an independent Russian team supported by commercial corporations. In August 2012, support for OpenCV was taken over by a non-profit foundation OpenCV.org, which maintains a developer and user site.

5.3 OpenCV-Python

OpenCV-Python is the Python API of OpenCV. It combines the best qualities of OpenCV C++ API and Python language.

Python is a general purpose programming language started by Guido van Rossum, which became very popular in short time mainly because of its simplicity and code readability. It enables the programmer to express his ideas in fewer lines of code without reducing any readability. Compared to other languages like C/C++, Python is slower. But another important feature of Python is that it can be easily extended with C/C++. This feature helps us to write computationally intensive codes in C/C++ and create a Python wrapper for it so that we can use these wrappers as Python modules. This gives us two advantages: first, our code is as fast as original C/C++ code (since it is the actual C++ code working in background) and second, it is very easy to code in Python. This is how OpenCV-Python works, it is a Python wrapper around original C++ implementation. And the support of Numpy makes the task easier. Numpy is a highly optimized library for numerical operations. It gives a MATLAB-style syntax. All the OpenCV array structures are converted to-and-from Numpy arrays. So whatever operations you can do in Numpy, you can combine it with OpenCV, which increases number of weapons in your arsenal. Besides that, several other libraries like SciPy, Matplotlib which supports Numpy can be used with this. So OpenCV-Python is an appropriate tool for fast prototyping of computer vision problems.

5.4 OpenCV-Python bindings

In OpenCV, all algorithms are implemented in C++. But these algorithms can be used from different languages like Python, Java etc. This is made possible by the bindings generators. These generators create a bridge between C++ and Python which enables users to call C++ functions from Python. To get a complete picture of what is happening in background, a good knowledge of Python/C API is required. A simple example on extending C++ functions to Python can be found in official Python documentation. So extending all functions in OpenCV to Python by writing their wrapper functions manually is a time-consuming task. So OpenCV does it in a more intelligent way. OpenCV generates these wrapper functions automatically from the C++ headers using some Python scripts which are located in modules/python/src2. We will look into what they do. First, modules/python/CMakeFiles.txt is a CMake script which checks the modules to be extended to Python. It will automatically check all the modules to be extended and grab their header files. These header files contain list of all classes, functions, constants etc for those particular modules.

Second, these header files are passed to a Python script, modules/python/src2/gen2.py. This is the Python bindings generator script. It calls another Python script modules/python/src2/hdr_parser.py. This is the header parser script. This header parser splits the complete header file into small Python lists. So these lists contain all details about a particular function, class etc. For example, a function will be parsed to get a list containing function name, return type, input arguments, argument types etc. Final list contains details of all the functions, structs, classes etc. in that header file.

But header parser doesn't parse all the functions/classes in the header file. The developer has to specify which functions should be exported to Python. For that, there are certain macros added to the beginning of these declarations which enables the header parser to identify functions to be parsed. These macros are added by the developer who programs the particular function. In short, the developer decides which functions should be extended to Python and which are not. Details of those macros will be given in next session. So header parser returns a final big list of parsed functions. Our generator script (gen2.py) will create wrapper functions for all the functions/classes/enums/structs parsed by header parser (You can find these header files during compilation in the build/modules/python/ folder as pyopencv_generated_*.h files). But there may be some basic OpenCV datatypes like Mat, Vec4i, Size. They need to be extended manually. For example, a Mat type should be extended to Numpy array, Size should be extended to a tuple of two integers etc. Similarly, there may be some complex structs/classes/functions etc. which need to be extended manually. All such manual wrapper functions are placed in modules/python/src2/pycv2.hpp.

So now only thing left is the compilation of these wrapper files which gives us cv2 module. So when you call a function, say res = equalizeHist(img1,img2) in Python, you pass two numpy arrays and you expect another numpy array as the output. So these numpy arrays are converted to cv::Mat and then calls the equalizeHist() function in C++. Final result, res will be converted back into a Numpy array. So in short, almost all operations are done in C++ which gives us almost same speed as that of C++.

So this is the basic version of how OpenCV-Python bindings are generated.

5.5 Applications

OpenCV's application areas include:

- 2D and 3D feature toolkit
- Facial recognition system
- Gesture recognition
- Human–computer interaction (HCI)
- Mobile robotics
- Motion understanding and Motion tracking
- Object identification
- Segmentation and recognition
- Stereopsis stereo vision: depth perception from 2 cameras
- Structure from motion (SFM)
- Augmented reality

To support some of the above areas, OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning

- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbour algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

Chapter-6 TensorFlow

6.1 Introduction

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. .The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as JavaScript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

6.2 History Of TensorFlow

TensorFlow is Google Brain's second-generation system. Version 1.0.0 was released on February 11, 2017. While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA and SYCL extensions for general-purpose computing on graphics processing units). TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.

Its flexible architecture allows for the easy deployment of computation across a variety of platforms (CPUs, GPUs, TPUs), and from desktops to clusters of servers to mobile and edge devices.

TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays, which are referred to as tensors. During the Google I/O Conference in June 2016, Jeff Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

In December 2017, developers from Google, Cisco, RedHat, CoreOS, and CaiCloud introduced Kubeflow at a conference. Kubeflow allows operation and deployment of TensorFlow on Kubernetes.

In March 2018, Google announced TensorFlow.js version 1.0 for machine learning in JavaScript.

In Jan 2019, Google announced TensorFlow 2.0.[20] It became officially available in Sep 2019.

In May 2019, Google announced TensorFlow Graphics for deep learning in computer graphics.

6.3 Features

AutoDifferentiation

AutoDifferentiation is the process of automatically calculating the gradient vector of a model with respect to each of its parameters. With this feature, TensorFlow can automatically compute the gradients for the parameters in a model, which is useful to algorithms such as backpropagation which require gradients to optimize performance. To do so, the framework must keep track of the order of operations done to the input Tensors in a model, and then compute the gradients with respect to the appropriate parameters.

Eager Execution

TensorFlow includes an "eager execution" mode, which means that operations are evaluated immediately as opposed to being added to a computational graph which is executed later. Code executed eagerly can be examined step-by step-through a debugger, since data is augmented at each line of code rather than later in a computational graph. This execution paradigm is considered to be easier to debug because of its step by step transparency.

Distribute

In both eager and graph executions, TensorFlow provides an API for distributing computation across multiple devices with various distribution strategies. This distributed computing can often speed up the execution of training and evaluating of TensorFlow models and is a common practice in the field of AI.

Losses

To train and assess models, TensorFlow provides a set of loss functions (also known as cost functions). Some popular examples include mean squared error (MSE) and binary cross entropy (BCE). These loss functions compute the "error" or "difference" between a model's output and the expected output (more broadly, the difference between two tensors). For different datasets and models, different losses are used to prioritize certain aspects of performance.

Metrics

In order to assess the performance of machine learning models, TensorFlow gives API access to commonly used metrics. Examples include various accuracy metrics (binary, categorical, sparse categorical) along with other metrics such as Precision, Recall, and Intersection-over-Union (IoU).

TF.nn

TensorFlow.nn is a module for executing primitive neural network operations on models. Some of these operations include variations of convolutions (1/2/3D, Atrous, depthwise), activation functions (Softmax, RELU, GELU, Sigmoid, etc.) and their variations, and other Tensor operations (max-pooling, bias-add, etc.).

Optimizers

TensorFlow offers a set of optimizers for training neural networks, including ADAM, ADAGRAD, and Stochastic Gradient Descent (SGD).When training a model, different optimizers offer different modes of parameter tuning, often affecting a model's convergence and performance.

6.4 Applications

Medical

GE Healthcare used TensorFlow to increase the speed and accuracy of MRIs in identifying specific body parts. Google used TensorFlow to create DermAssist, a free mobile application that allows users to take pictures of their skin and identify potential health complications. Sinovation Ventures used TensorFlow to identify and classify eye diseases from optical coherence tomography (OCT) scans.

Social media

Twitter implemented TensorFlow to rank tweets by importance for a given user, and changed their platform to show tweets in order of this ranking. Previously, tweets were simply shown in reverse chronological order. The photo sharing app VSCO used TensorFlow to help suggest custom filters for photos.

Search Engine

Google officially released RankBrain on October 26, 2015, backed by TensorFlow.

Education

InSpace, a virtual learning platform, used TensorFlow to filter out toxic chat messages in classrooms. Liulishuo, an online English learning platform, utilized TensorFlow to create an adaptive curriculum for each student. TensorFlow was used to accurately assess a student's current abilities, and also helped decide the best future content to show based on those capabilities.

Retail

The e-commerce platform Carousell used TensorFlow to provide personalized recommendations for customers. The cosmetics company ModiFace used TensorFlow to create an augmented reality experience for customers to test various shades of make-up on their face.

Chapter-7 Image Processing

7.1 Introduction

To work on videos, one needs to work with images first. After all, a video is a stack of images moving too fast. Python processes images using Open CV. Images can not only be resized, grayed out but also read for face detection and written onto files.

7.2 Applications of Digital Image Processing

Some of the major fields in which image processing is widely used are mentioned below

- Image sharpening and restoration
- Medical field
- Remote sensing
- Transmission and encoding
- Machine/Robot vision
- Color processing
- Pattern recognition
- Video processing
- Microscopic Imaging

Image sharpening and restoration: It refers here to process images that have been captured from the modern camera to make them a better image or to manipulate those images in way to achieve desired result. It refers to do what Photoshop usually does. This includes Zooming, blurring, sharpening, grayscale to color conversion, detecting edges and vice versa, Image retrieval and Image recognition.

Medical field: The common applications of DIP in the field of medical is

- Gamma ray imaging
- PET scan
- X Ray Imaging
- Medical CT
- UV imaging

Remote Sensing: In the field of remote sensing, the area of the earth is scanned by a satellite or from a very high ground and then it is analyzed to obtain information about it. One particular application of digital image processing in the field of remote sensing is to detect infrastructure damages caused by an earthquake. As it takes longer time to grasp damage, even if serious damages are focused on. Since the area effected by the earthquake is sometimes so wide, that it not possible to examine it with human eye in order to estimate damages. Even if it is, then it is very hectic and time consuming procedure. So, a solution to this is found in digital image processing. An image of the affected area is captured from the above ground and then it is analyzed to detect the various types of damage done by the earthquake.

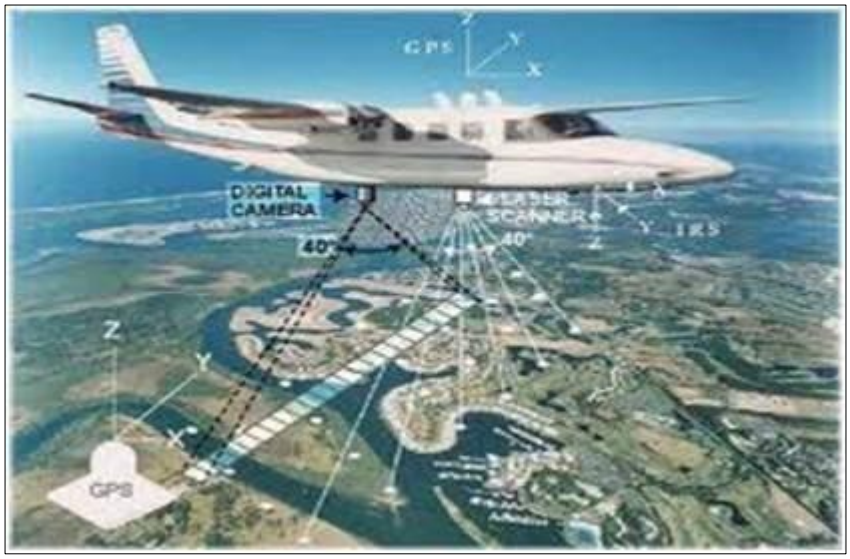


Fig 7.1

The key steps include in the analysis are

- The extraction of edges
- Analysis and enhancement of various types of edges

Transmission and encoding: The very first image that has been transmitted over the wire was from London to New York via a submarine cable. The picture that was sent is shown below.

Fig 7.2

The picture that was sent took three hours to reach from one place to another. Now just imagine, that today we are able to see live video feed, or live CCTV footage from one continent to another with just a delay of seconds. It means that a lot of work has been done in this field too. This field does not only focus on transmission, but also on encoding. Many different formats have been developed for high or low bandwidth to encode photos and then stream it over the internet or etc.

Machine/Robot vision: Apart from the many challenges that a robot face today, one of the biggest challenge still is to increase the vision of the robot. Make robot able to see things, identify them, identify the hurdles etc. Much work has been contributed by this field and a complete other field of computer vision has been introduced to work on it.

Hurdle detection: Hurdle detection is one of the common task that has been done through image processing, by identifying different type of objects in the image and then calculating the distance between robot and hurdles.

Fig 7.3

Line follower Robot: Most of the robots today work by following the line and thus are called line follower robots. This help a robot to move on its path and perform some tasks. This has also been achieved through image processing.

Fig 7.4

Color processing: Color processing includes processing of colored images and different color spaces that are used. For example: RGB color model, YCbCr, HSV. It also involves studying transmission, storage, and encoding of these color images.

Pattern recognition: Pattern recognition involves study from image processing and from various other fields that includes machine learning (a branch of artificial intelligence). In pattern recognition, image processing is used for identifying the objects in an image and then machine learning is used to train the system for the change in pattern. Pattern recognition is used in computer aided diagnosis, recognition of handwriting, recognition of images etc.

- **Importing The Necessary Packages:**

```
import cv2
import numpy as np
import mediapipe as mp
import tensorflow as tf
from tensorflow.keras.models import load_model
```

Fig 8.1

Here, we import numpy, mediapipe, tensorflow and opencv packages. The model used here is Keras.

- **Initializing The Models:**

```
# initialize mediapipe
mpHands = mp.solutions.hands
hands = mpHands.Hands(max_num_hands=1, min_detection_confidence=0.7)
mpDraw = mp.solutions.drawing_utils
```

Fig 8.2

Mp.solution.hands module performs the hand recognition algorithm. So, we create the object and store it in mpHands.
Using mpHands.Hands method we configured the model. The first argument is max_num_hands, that means the maximum number of hands will be detected by the model in a single frame. MediaPipe can detect multiple hands in a single frame, but we'll detect only one hand at a time in this project.
Mp.solutions.drawing_utils will draw the detected key points for us so that we don't have to draw them manually.

```
14  # Load the gesture recognizer model
15  model = load_model('mp_hand_gesture')
16
17  # Load class names
18  f = open('gesture.names', 'r')
19  classNames = f.read().split('\n')
20  f.close()
21  print(classNames)
```

Fig 8.3

Using the load_model function we load the TensorFlow pre-trained model.

Gesture.names file contains the name of the gesture classes. So first, we **open** the file using python's inbuilt open function and then read the file.
After that, we read the file using the read() function.

- **Capturing Frames From Webcam Using OpenCV:**

```
23
24  # Initialize the webcam
25  cap = cv2.VideoCapture(0)
26
27  while True:
28      # Read each frame from the webcam
29      _, frame = cap.read()
30
31      x, y, c = frame.shape
32
33      # Flip the frame vertically
34      frame = cv2.flip(frame, 1)
35      framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

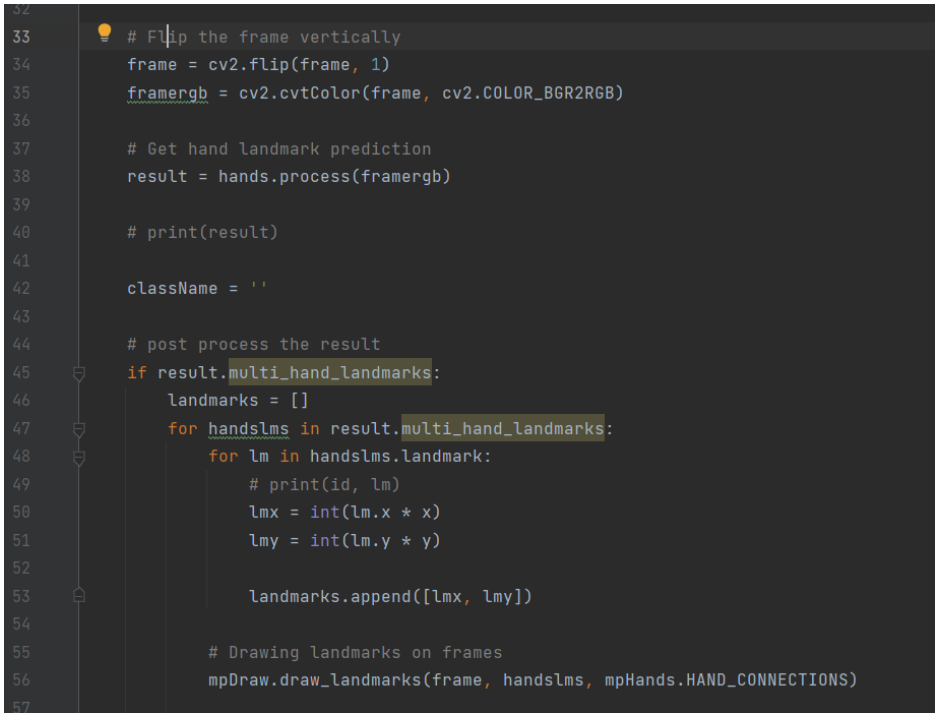
Fig 8.4

```
68      # Show the final output
69      cv2.imshow("Output", frame)
70
71      if cv2.waitKey(1) == ord('q'):
72          break
73
74  # release the webcam and destroy all active windows
75  cap.release()
76
77  cv2.destroyAllWindows()
```

Fig 8.5

We create a VideoCapture object and pass an argument '0'. It is the camera ID of the system. In this case, we have 1 webcam connected with the system. If you have multiple webcams then change the argument according to your camera ID. Otherwise, leave it default.
The cap.read() function reads each frame from the webcam.
cv2.flip() function flips the frame.
cv2.imshow() shows frame on a new openCV window.
The cv2.waitKey() function keeps the window open until the key 'q' is pressed.

- **Detecting Hand Key points:**



```
# Flip the frame vertically
frame = cv2.flip(frame, 1)
framergb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

# Get hand landmark prediction
result = hands.process(framergb)

# print(result)

className = ''

# post process the result
if result.multi_hand_landmarks:
    landmarks = []
    for handslms in result.multi_hand_landmarks:
        for lm in handslms.landmark:
            # print(id, lm)
            lmx = int(lm.x * x)
            lmy = int(lm.y * y)

            landmarks.append([lmx, lmy])

        # Drawing landmarks on frames
        mpDraw.draw_landmarks(frame, handslms, mpHands.HAND_CONNECTIONS)
```

Fig 8.6

MediaPipe works with RGB images but OpenCV reads images in BGR format. So, using cv2.cvtCOLOR() function we convert the frame to RGB format.
The process function takes an RGB frame and returns a result class.
Then we check if any hand is detected or not, using result.multi_hand_landmarks method.
After that, we loop through each detection and store the coordinate on a list called landmarks.
Here image height (y) and image width(x) are multiplied with the result because the model returns a normalized result. This means each value in the result is between 0 and 1.
And finally using mpDraw.draw_landmarks() function we draw all the landmarks in the frame.

- **Recognize Hand Gestures:**

```
            # Predict gesture
            prediction = model.predict([landmarks])
            # print(prediction)
            classID = np.argmax(prediction)
            className = classNames[classID]

    # show the prediction on the frame
    cv2.putText(frame, className, (10, 50), cv2.FONT_HERSHEY_SIMPLEX,
                1, (0,0,255), 2, cv2.LINE_AA)

    # Show the final output
    cv2.imshow("Output", frame)

    if cv2.waitKey(1) == ord('q'):
        break

# release the webcam and destroy all active windows
cap.release()

cv2.destroyAllWindows()
```

Fig 8.7

The model.predict() function takes a list of landmarks and returns an array contains 10 prediction classes for each landmark.
The output looks like this-
[[2.0691623e-18 1.9585415e-27 9.9990010e-01 9.7559416e-05
1.6617223e-06 1.0814080e-18 1.1070732e-27 4.4744065e-16 6.6466129e-07 4.9615162e-21]]
Np.argmax() returns the index of the maximum value in the list.
After getting the index we can simply take the class name from the classNames list.
Then using the cv2.putText function we show the detected gesture into the frame.

Chapter 9 : Final Execution

Fig 9.1

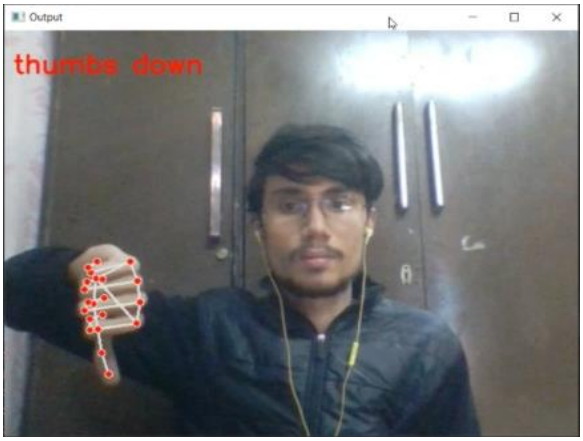


Fig 9.2

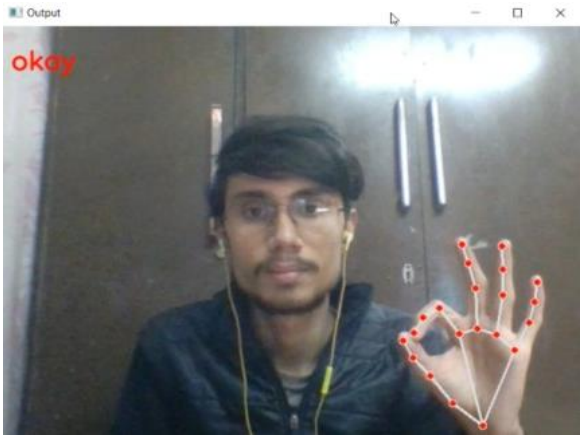


Fig 9.3

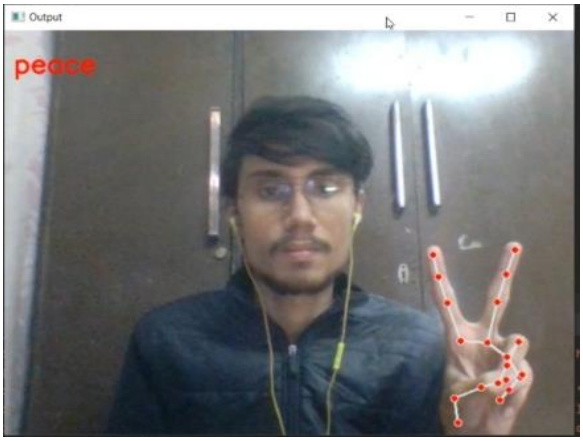


Fig 9.4

Fig 9.5

Fig 9.6

Fig 9.7

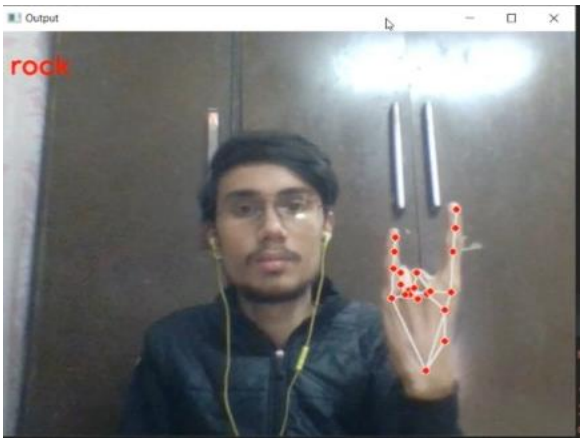


Fig 9.8

Fig 9.9

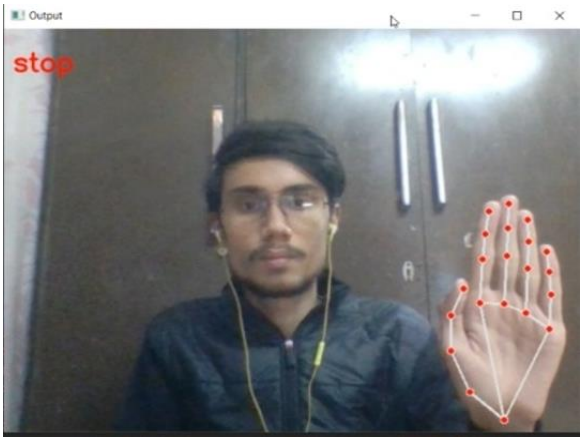


Fig 9.10

Chapter-10 Conclusion

10.1 Conclusion

By the end of this project, we all can agree with one thing that the field of deep learning (Computer Vision) is beneficial for the growth of society. Our project is considered as the first step towards achieving a bigger goal of creating a fully functional sign language recognition system consisting of both hand gestures, sign language and alphabets. This system would not only be able to recognise the hand gestures but can further be used for helping those with difficulty in hearing or speaking. With the help of this project a lot of possibilities have been opened and a number of problems can be tackled in a similar manner. This would not only be restricted to only one type of problems but can further be manipulated and converted into different forms according to the requirement. For example, instead of using hand gestures for sign languages these could be used in an automated car to understand the road signs, etc. This project could be very useful with the advancement of technology and could also be the first step towards robots learning to recognise various daily slangs or hand gestures.

Chapter-11 Future Scope

11.1 Future Scope of the Project

- **Automated Cars with Road Sign Recognition**

  As the technological advancement taking such a great leap in the past few years the future with automated cars running on the roads is not that far away. But these cars would need to understand road signs and especially in highly populated countries like India, China, etc. these cars should be able to recognise common hand gestures used by drivers in day-to-day life.

- **Sign Language Recognition System**

  With some slight modifications to this project this system could easily be converted into a sign language recognition system for the specially abled people who find it difficult to convey their thoughts to people having some to no knowledge of the sign language.

- **Enhancing Teaching System in Schools for Specially Abled Children**

  This system could be used in the schools for specially abled children enabling the teachers to teach the kids more effectively.

- **Music Player Control System with the help of Hand Gestures**

  Gestures can be used to control various functionalities of a music player like volume control, play/pause, previous/next, etc.

References

1. https://www.tensorflow.org/learn
2. https://google.github.io/mediapipe/getting_started/python.html
3. https://www.geeksforgeeks.org/python-programming-language/
4. https://keras.io/api/datasets/
5. https://www.jetbrains.com/pycharm/
6. http://pythonhow.com/
7. http://opencv.org/

Declaration by Student

I certify that I have properly verified all the items in the checklist and ensure that the report is in proper format as specified in the course handout.

Name: Gaurav Birdi,
Md Omer,
Prakhar Katiyar
Sachin Sharma
Place: Delhi Technical Campus
Date: 22-12-2021
Signature of the Student:

Verification by Faculty Project Guide

I have duly verified all the items in the checklist and ensured that the report is in proper format.

Name: Ms. Upasna Joshi
Place: Delhi Technical Campus
Date: 22-12-2021
Signature of the Project Guide: