SLIDIE

Interactive PPT

Ashutosh Sharma (A21006)
Debopriyo Mallick (A21010)
Pratyush Nayak (A21023)
Shubham Bathwal (A21031)
Gargi Goswami (A21039)

Introduction

The objective behind creating the project Slidie is to change the traditional way of managing presentation like clicking or pressing keyboard buttons. Our goal is to automate power point presentation process and managing presentations using simple gestures and voice commands. Voice commands and gestures boost the efficiency and user friendliness of the presentation as it enables presenter to control the motion of slides as per his/her own will and have more fun doing it.

For gesture commands we use several hand sings. So, in gesture commands our first objective was to read the hands using computer visions. For such purpose we use already trained library such as OpenCV and MediaPipe. Automation archived using hand gestures are:

- Presentation ON / OFF
- Next / Previous Slide
- Zoom

For voice command we used library called Pyaudio to take word inputs and based on selective predefined words we can turn them into the commands to control presentation. Objective archived using voice commands are:

- Presentation ON / OFF
- Next / Previous Slide
- Go to specific slide (using slide number)

Media Pipe

Building an application that processes perceptual in- puts involves more than running an ML model. Developers have to harness the capabilities of a wide range of devices; balance resource usage and quality of results; run multiple operations in parallel and with pipelining; and ensure that time-series data is properly synchronized. The Media-Pipe framework addresses these challenges. A developer can use Media-Pipe to easily and rapidly com- bine existing and new perception components into proto- types and advance them to polished cross-platform applications. The developer can configure an application built with Media-Pipe to manage resources efficiently (both CPU and GPU) for low latency performance, to handle synchronization of time-series data such as audio and video frames and to measure performance and resource consumption. We show that these features enable a developer to focus on the algorithm or model development and use Media-Pipe as an environment for iteratively improving their application, with results reproducible across different de- vices and platforms. Media-Pipe will be open-sourced at https://github.com/google/mediapipe.

To enable augmented reality (AR), a typical application processes sensory data such as video and audio at high frame rates to enhance the user experience. Modifying such a perception application to incorporate additional processing steps or inference models can be difficult due to excessive coupling between steps. Further, developing the same application for different platforms is time consuming as it usually involves optimizing inference and processing steps to run correctly and efficiently on a target device.

Media-Pipe addresses these challenges by abstracting and connecting individual perception models into maintainable pipelines. With Media-Pipe, a perception pipeline can be built as a graph of modular components, including, for in- stance, inference models and media processing functions.

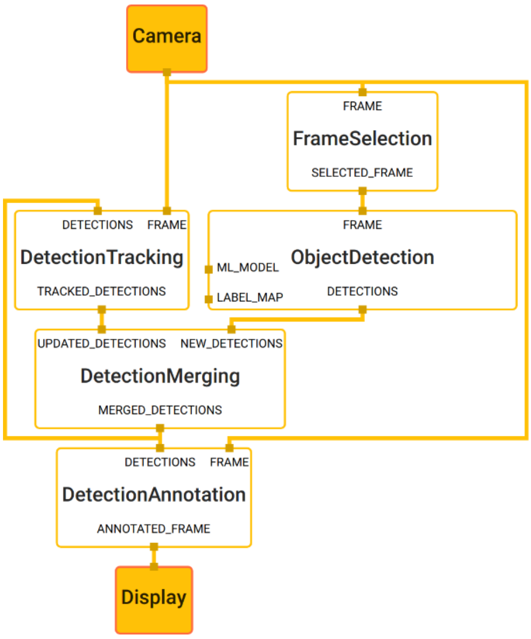


Figure 1: Object detection using Media-Pipe.

Sensory data such as audio and video streams enter the graph, and perceived descriptions such as object detection results or face landmark annotations leave the graph. An example in shown in Figure1.

Graphs of operations are used in projects such as Tensor- Flow, PyTorch, CNTK or MXNet to define a neural network model. Media-Pipe takes a complementary role: our graphs do not define the internals of a neural net- work, but instead specify larger-scale pipelines in which one or model models are embedded.

OpenCV 4.0 introduced the Graph API (G-API) which allows specifications of sequences of OpenCV im- age processing operations in the form of a graph. How- ever, Media-Pipe allows operations on arbitrary data types and provides native support for streaming time-series data.

Basic Concepts

Media-Pipe consists of three main parts:

- A framework for inference from sensory data
- A set of tools for performance evaluation
- A collection of reusable inference and processing components.

A pipeline is defined as a directed graph of components, where each component is a Calculator. Developers can define custom calculators. The graph description is specified via a GraphConfig protocol buffer and then run using a Graph object.

In a graph, data flows through each calculator via data Streams. The basic data unit in Media-Pipe is a Packet. A stream carries a sequence of packets with monotonically increasing timestamps. Calculators and streams together define a data-flow graph. Each stream in a graph maintains its own queue to allow the receiving graph node to consume packets at its own pace.

Changes to the pipeline to add or remove components can be made by updating the GraphConfig file. A developer can also configure global graph-level settings to modify graph execution and resource consumption in this file. This is useful for tuning the performance of the graph on different platforms (e.g., on desktop vs. on mobile).

Using Hand Landmark Model

After the palm detection over the whole image our subsequent hand landmark model performs precise key point localization of 21 3D hand-knuckle coordinates inside the detected hand regions via regression, that is direct coordinate prediction. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions.

To obtain ground truth data, we have manually annotated ~30K real-world images with 21 3D coordinates, as shown below (we take Z-value from image depth map, if it exists per corresponding coordinate). To better cover the possible hand poses and provide additional supervision on the nature of hand geometry, we also render a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates.

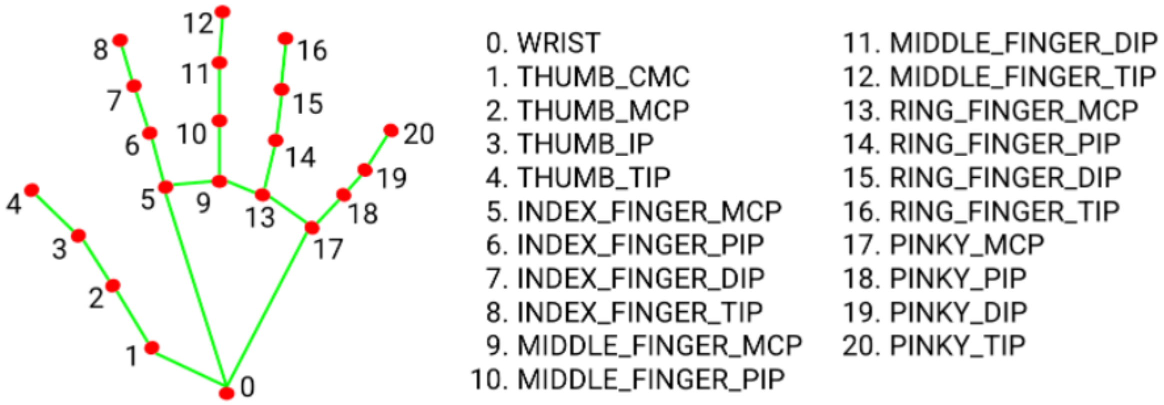


Figure 2: Hand Landmarks

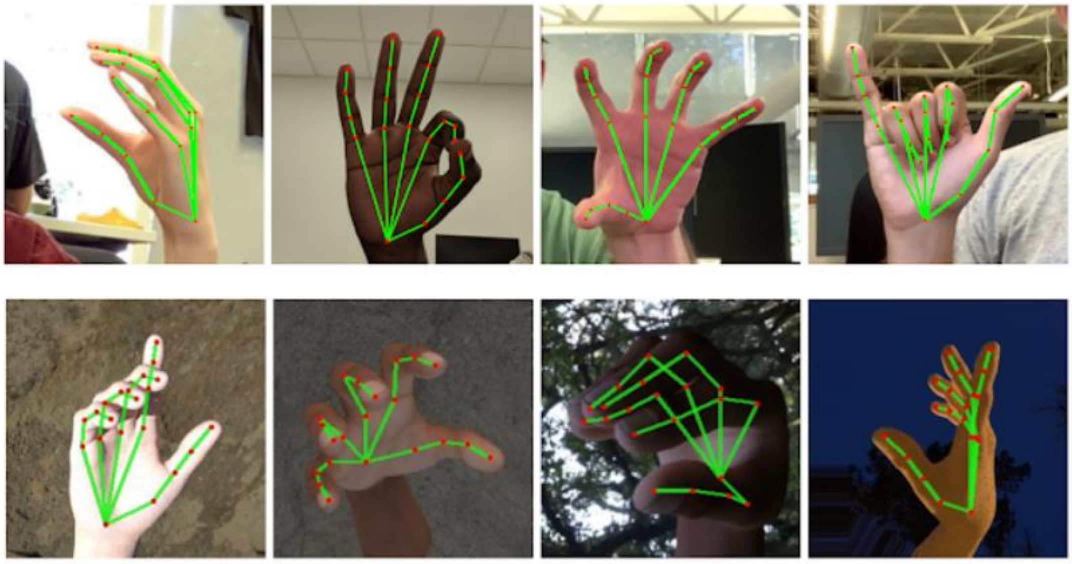


Figure 3: *Aligned hand crops passed to the tracking network with ground truth annotation. Bottom: Rendered synthetic hand images with ground truth annotation.*

Tooling and Platforms

Media-Pipe offers developer tools to inspect the run-time behavior of graphs and analyze their performance.

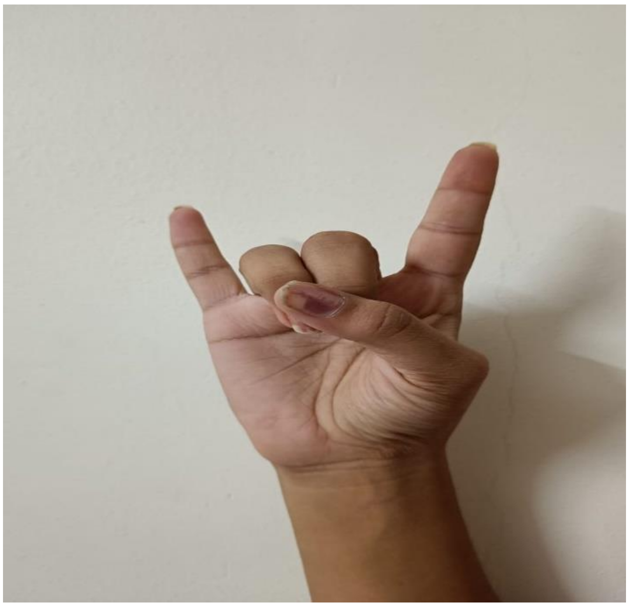
The Media-Pipe tracer module follows individual data packets across a graph and records timing events along the way. The Media-Pipe visualizer tool can help developers understand the overall behavior of their pipelines by visualizing the recorded timing events. The visualizer also offers a detailed view of the topology of the graph and lets a user observe the full state of the graph, its calculators and packets being processed, at any point in time.

Media-Pipe facilitates the deployment of perception technology into applications on a wide variety of hardware plat- forms. Media-Pipe supports GPU compute and rendering nodes, and allows combining multiple GPU nodes, as well as mixing them with CPU based nodes. GPU nodes can be built using various APIs (e.g., OpenGL ES or Metal); CPU nodes can use popular image processing nodes such as OpenCV.

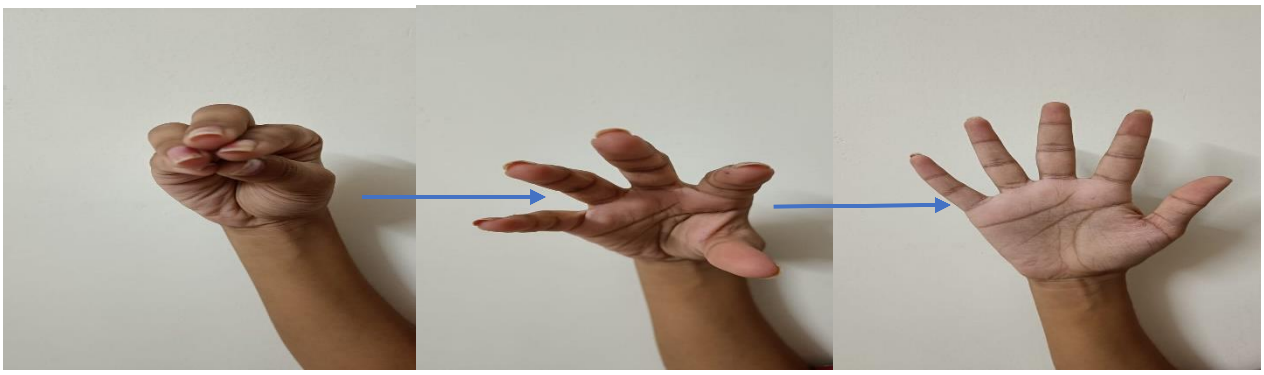
Gestures

After reading hands using Mediapipe we required to model certain hand gestures to run different functionalities of PPT. These gestures are based on the points on our fingers their position on frame and the distance between them. For the project the frame size used is of 640*480, it is important to for the user to have minimum resolution of devise mentioned above.
Besides Mediapipe other libraries such as Pynput has also been used to access the keyboard keys. It has vital role to play in both Gesture model as well as Speech recognition model.
Now that we learn prerequisites for gesture modelling let’s see the different gestures used for functionalities.

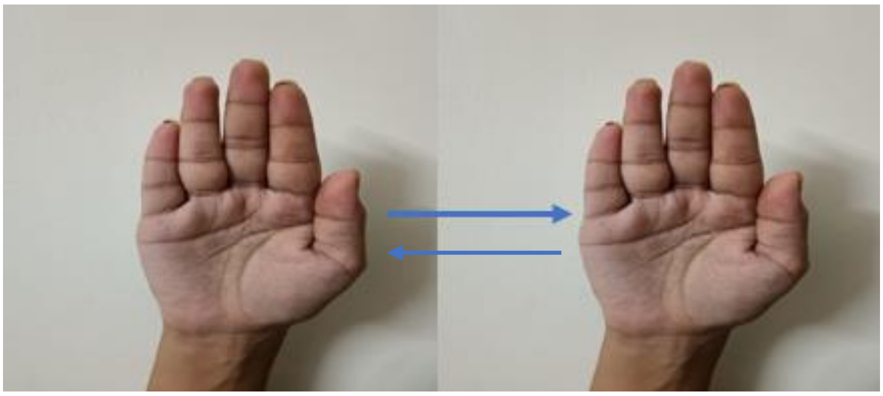
- **Gesture ON & OFF**
  The first symbol is to start and stop the gesture reading while using the application one can start and stop the gesture reading at will. One of the reasons behind this function is that many people use hands while explain the presentation slide which can be taken as command so to avoid such situation one can turn off the gesture commands.



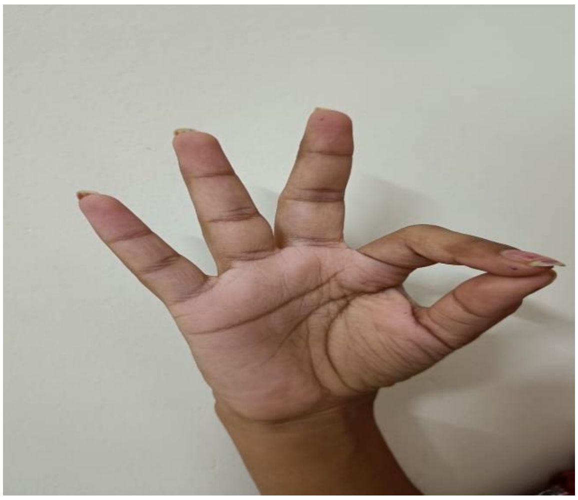
- **Presentation ON & OFF**
  Second, functionality is to turn on or off the presentation. For this one has to bring all fingertips together and then expand it eventually to turn on presentation and vice versa when turning off the presentation.



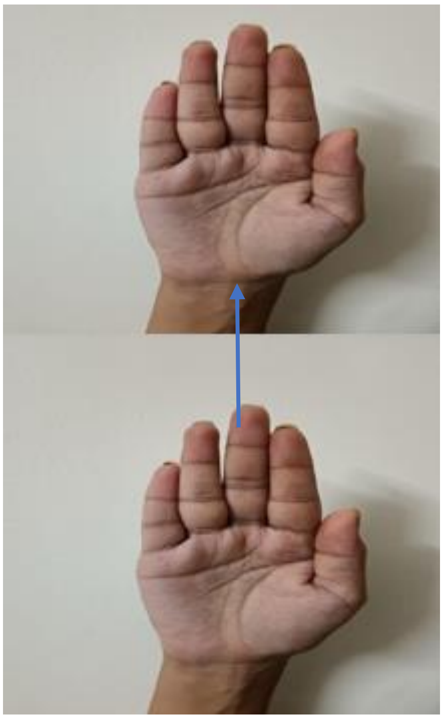
- **Next & Previous**
  To go to the next or previous slide in presentation one need to slide his/her hands from left to right for next and right to left for previous. The distance we took is 70 pixels on x-axis

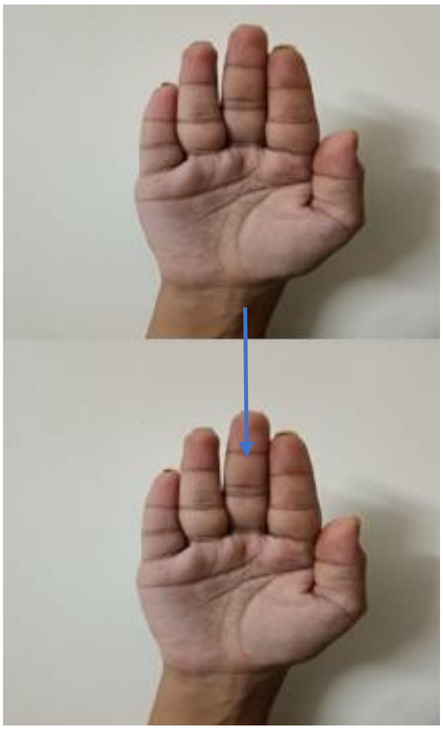


- **Zoom IN & Zoom OUT**
  To start the Zoom functionalities the gesture is given as

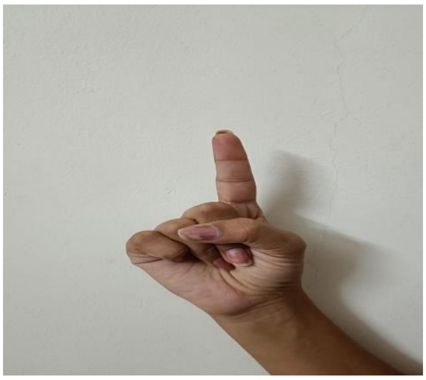


For zooming we first divided the screen into two halves at the center of the screen (at 320 pixels on x-axis). On left side if we go upward, basically 30 pixels upward, we can zoom in and if we go downward in on right side we can zoom out.

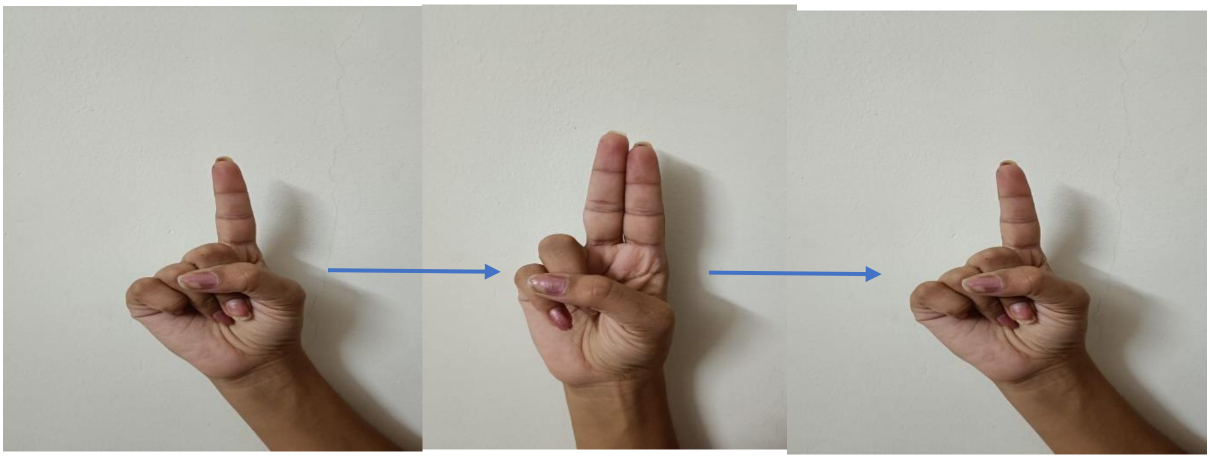




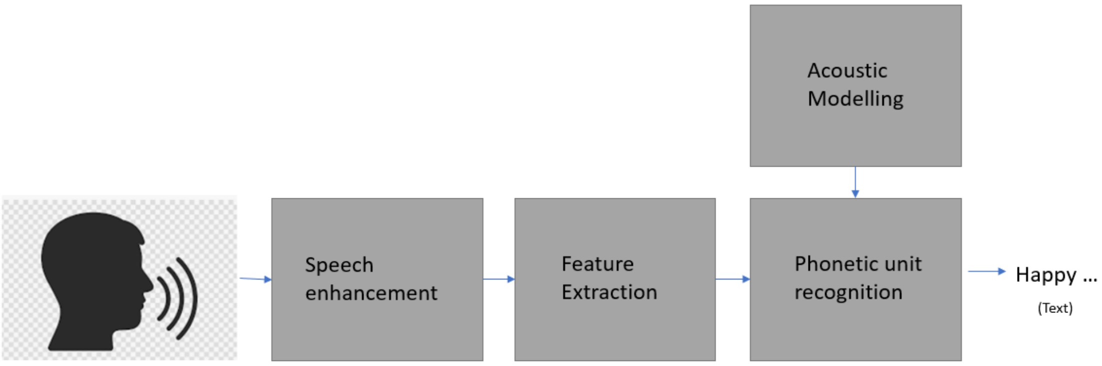
- **Moving after Zooming IN**
  After zooming in one can move the mouse pointer using the gesture shown below



To hold and release the left mouse button, so that one can hold and move while Zoomed IN, one must flicker the middle finger next to index one & do the same to release the hold.



Speech Recognition



Speech Recognition

Speech recognition software works by breaking down the audio of a speech recording into individual sounds, analyzing each sound, using algorithms to find the most probable word fit in that language, and transcribing those sounds into text.

i) A microphone converts the vibrations of a person's voice into a wave like electrical signal.
ii) This signal is converted to digital signal by the computer's sound card.
iii) A preprocessing unit enhances the speech signal and reduces noise
iv) The speech recognition software analyzes the digital signal to register phonemes.
v) The phenomes are reconstructed into words.
vi) To pick the correct word, the program must rely on context cues, accomplished through trigram analysis. This method relies on a database of frequent three-word clusters in which probabilities are assigned that any two words will be followed by a given third word.

Automatic speech recognition systems make use of probabilistic approach to compute a score to match spoken words with a speech signal. A speech signal corresponds to any word or sequence of words in the vocabulary with a probability value. The score is calculated from phonemes in the acoustic model knowing which words can follow other words through linguistic knowledge. The word sequence with the highest score gets chosen as the recognition result.

Speech signals are slowly timed varying signals, and their characteristics are fairly stationary when examined over a short period of time (5-100 ms). Therefore, in the feature extraction step, acoustic observations are extracted in frames of typically 25 ms. For the acoustic samples in that frame, a multi-dimensional vector is calculated, and a fast Fourier transformation is performed on the vector, to transform a function of time, e.g. a signal in this case, into their frequencies. A common feature extraction step is cepstral mean subtraction (CMS) which is used to normalize differences between channels, microphones, and speakers.

In the decoding process, calculation is done to find which sequence of words is the most probable match to the feature vectors. For decoding process to work fine, three things have to be present: an acoustic model with a hidden Markov model (HMM) for each unit (phoneme or word), a dictionary containing possible words and their phoneme sequences and a language model with words or word sequences likelihoods.

In decoding the SR systems tries to find the word or sequence of words w* best matching the observation X using p(w) from the language model and p(X|w) from the phoneme sequence in the vocabulary.

$$w* = argmaxw(p(X|w)p(w)) \qquad (i)$$

$$p(X|w) = argmaxs(\prod j\ (p(x|sj)p(sj))) \qquad (ii)$$

SR systems usually, in the post-processing step, attempts to re-score this list, e.g. by using a higher-order language model and/or pronouncing models.
Mostly speech recognition systems use hidden Markov models (HMMs) to deal with the temporal variability of speech and Gaussian mixture models (GMMs) to determine how well each state of each HMM fits a frame or a short window of frames of coefficients that represents the acoustic input. Another way to evaluate the fit is to use a feed-forward neural network that takes several frames of coefficients as input and produces posterior probabilities over HMM states as output. Deep neural networks (DNNs) that have many hidden layers and are trained using new methods have been shown to outperform GMMs on a variety of speech recognition benchmarks, sometimes by a large margin. Google's current speech recognition system is speaker-independent and is built on deep neural networks together with hidden Markov models (DNN-HMM).
Evaluation and accuracy of speech recognition can vary depending upon:

i) Recording conditions- Performance is affected by background noise, echoes, type of microphone (e.g. close-speaking, telephone or omnidirectional), limited frequency bandwidth (for example telephone transmissions) and manner of speaking (shouting, speaking quickly, etc.).

ii) Spontaneous vs. read speech - Read speech from a text is easy to understand compared to spontaneous speech where words like" uh" and" um", stuttering, coughing and laughter can occur.

iii) Smaller vocabulary size and isolated, discontinuous speeches are easier to understand.

Voice controlled presentation

Here, we have used Happy as the trigger word which triggers the speech detection process. Microphone is the input device which listens to the voice command and google audio acts as speech recognition system which recognizes speech input from the microphone accurately and transcripts the voice to text using an API powered by Google's AI technologies.

Command	**Action**
Present	will start the presentation in presentation mode
Next	will move to the next slide (move one slide forward)
Go back	will move to the previous slide (move one slide backward)
Number (any number below 99)	will direct the presentation to go to that particular slide number
End of Slide	will end the presentation

Web Application

Flask has been used to deploy the model on web. Flask is a Python-based microweb framework. Because it does not require any tools or libraries, it is categorized as a microframework. It lacks a database abstraction layer, form validation, or any other components for which third-party libraries already exist. Extensions, on the other hand, may be used to add application functionalities as if they were built into Flask itself. Object-relational mappers, form validation, upload handling, different open authentication protocols, and other framework-related utilities all have extensions.

The whole frontend infrastructure was built in HTML, using CSS style and JavaScript function execution. The site was created to be user-friendly and simple to navigate. There are interactive buttons on the website to help you navigate around it, and you can also download the report straight from the site.

The various modules utilized in the project are listed, along with a link to obtain instructions on how to use and operate the program using gesture and speech. Because PowerPoint is an important aspect of our project, we've included a quick introduction to the software as well as a link to download it immediately.

The user may utilize one of the two capabilities, gesture, or voice command, independently or both at the same time. A new tab appears when the gesture button is pushed, allowing the user to use gesture commands.

Furthermore, the user may share his or her screen and use Picture in Picture mode to overlay the video feed on the presentation. This feature makes user interaction with the presentation smooth since the user can see the various instructions in real time.

When you hit the Voice button, a new tab appears that allows users to utilize voice commands.