DEVELOPMENT OF SIGN LANGUAGE DETECTION AND CONVERSION SYSTEM USING MEDIAPIPE

Group Members:

1. **KNR21EC058 - MOHAMMED SHAFEEHE**
2. **KNR21EC046 - JASIR NUFAIZ V**
3. **KNR21EC038 - FIDA FATHIMA RM**
4. **KNR21EC068 - NIDA SALAM K**

B.Tech in Electronics and Communication Engineering

Guided By :
PROF. INDU C.
Department of Electronics and Communication Engineering
Government College of Engineering, Kannur

Contents

Objective

1. Use MediaPipe for accurate, real-time gesture recognition and conversion to speech.
2. Translate spoken language (based on ASL) into visual sign language for deaf individuals.
3. Optimize for accurate, responsive communication in real time.
4. Ensure compatibility across devices for easy adoption in various settings.

Methodology Flowchart

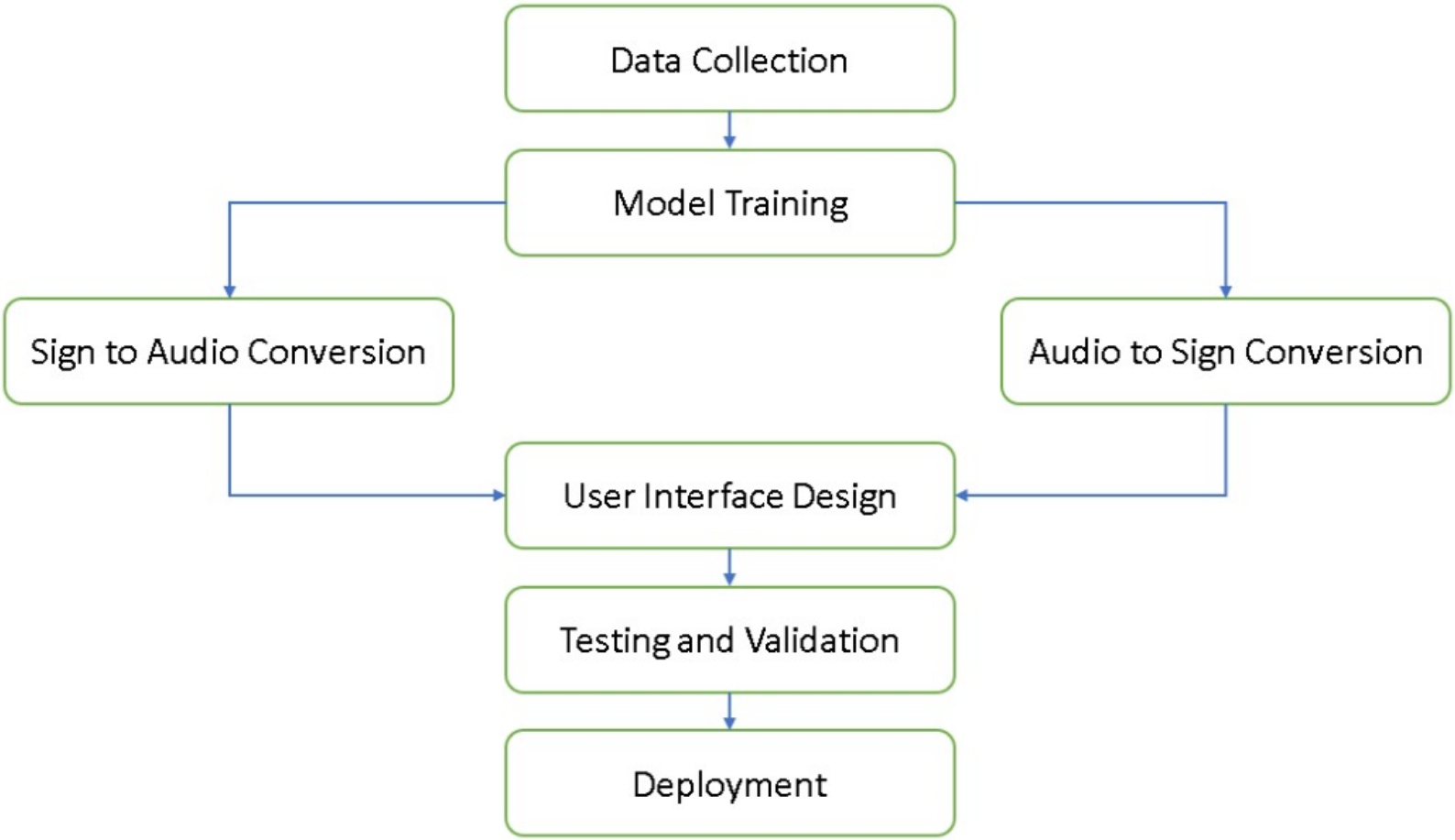

Figure: Block Diagram of Sign Language Detection-Conversion System

Software and Hardware Requirements

Category	Requirement	Use
Software	Python	Developing the machine learning models and system logic.
Python-Library file	MediaPipe	Real-time hand tracking and gesture recognition framework.
Python-Library file	OpenCV	Image and video processing for capturing and analyzing sign language gestures.
Python-Library file	PyAudio	For audio processing and conversion (used in audio-to-sign language conversion).
Python-Library file	Tkinter	For building a user-friendly interface for the system.
Hardware	Laptop with GPU	Regarding executing the machine learning model

Figure: Software and Hardware Requirments

Preliminary Result

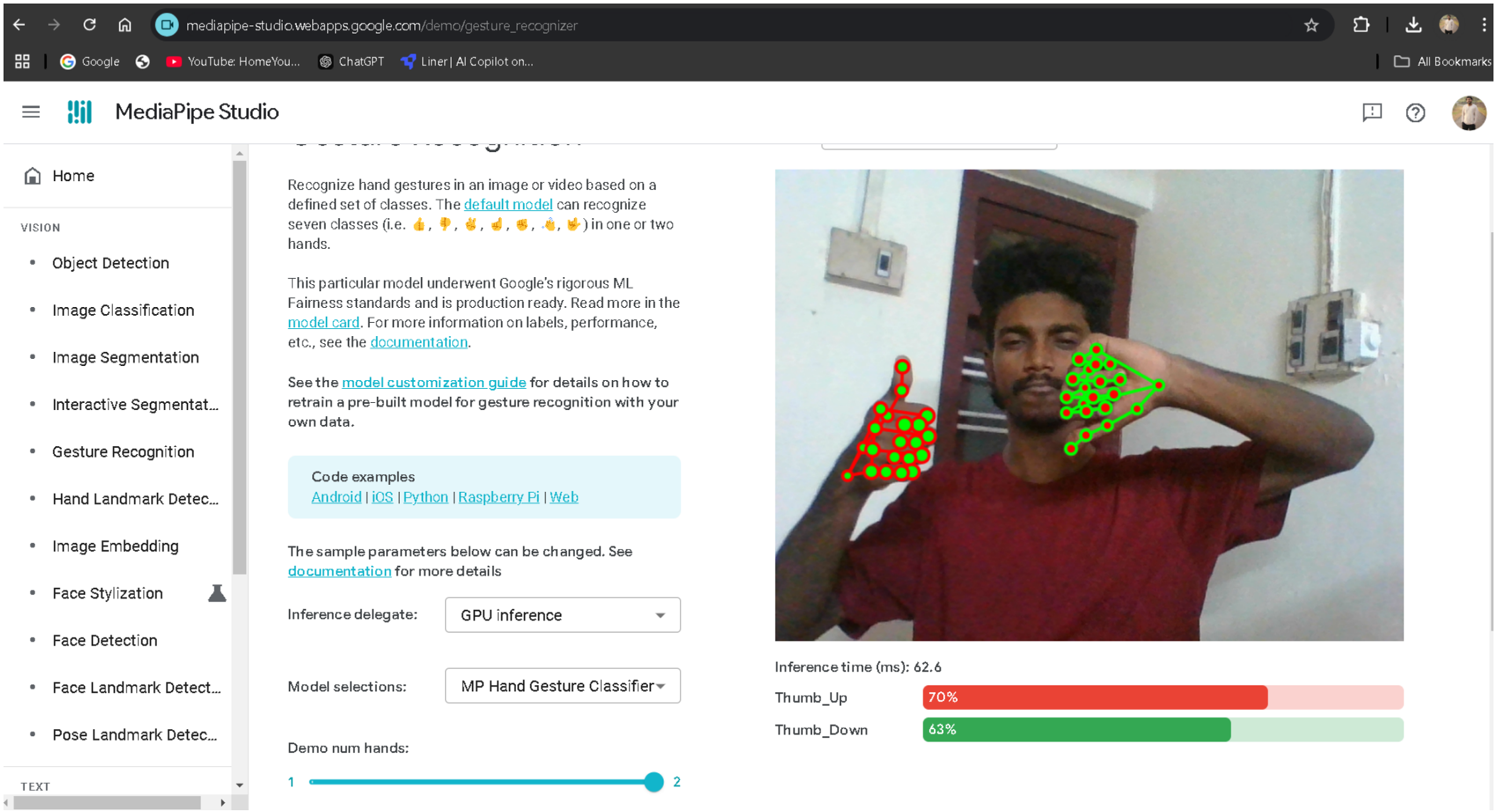

Figure: Web Demo of Basic Gesture Recognition in MediaPipe

Preliminary Result

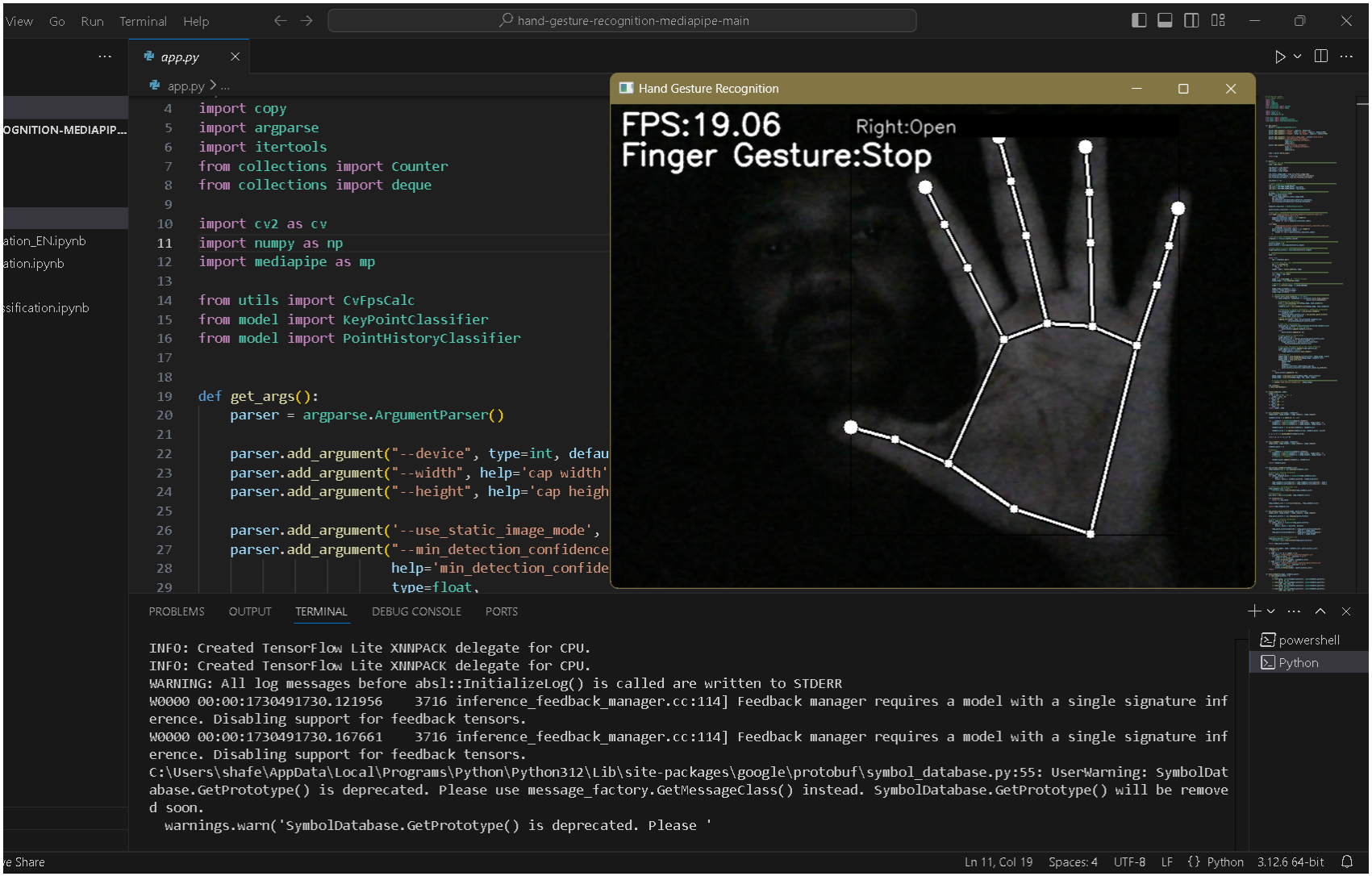

Figure: Basic Gesture Recognition Executed in MS VSCODE

PRELIMINARY RESULTS

- The storage structure of the project directory for collecting, training and experimenting the hand gesture models.

```
|  app.py
|  keypoint_classification.ipynb
|  point_history_classification.ipynb
|
├─model
|  ├─keypoint_classifier
|  |  |  keypoint.csv
|  |  |  keypoint_classifier.hdf5
|  |  |  keypoint_classifier.py
|  |  |  keypoint_classifier.tflite
|  |  └  keypoint_classifier_label.csv
|  |
|  └─point_history_classifier
|        |  point_history.csv
|        |  point_history_classifier.hdf5
|        |  point_history_classifier.py
|        |  point_history_classifier.tflite
|        └  point_history_classifier_label.csv
|
└─utils
    └─cvfpscalc.py
```

Figure: Directory Contents for Data Collection and Processing

INTERIM RESULTS:

1. Hand Sign-to-Audio Conversion (audio conversion.py)

- Library used: pyttsx3 is used to convert the text to speech output,
- detected hand sign as text are announced as audio in real time

keypoint_classifier_label.csv | keypoint_classification_EN.ipynb | audio_conversion.py | keypoint.csv | app.py

audio_conversion.py > ...

```python
import pyttsx3
import threading
import queue

#initialize text-to-speech engine
engine=pyttsx3.init()

#create a thread-safe queue
tts_queue = queue.Queue()

#function to process the queue
def process_queue():
    while True:
        #Get the next target from the queue
        predicted_sign = tts_queue.get()
        if predicted_sign is None:  #Exit signal
            break
        engine.say(predicted_sign)
        engine.runAndWait()
        tts_queue.task_done()

#start the processing thread
tts_thread = threading.Thread(target=process_queue, daemon=True)
tts_thread.start()
```

Figure: script for audio conversion

INTERIM RESULTS:

2. Voice-to-Hand Sign Conversion (audio to handsign.py)

- Library used: speech recognition (Google Web Speech API).
- Matches the recognized text with "hand sign mapping" dictionary. Displays the corresponding hand sign image in a Tkinter window.

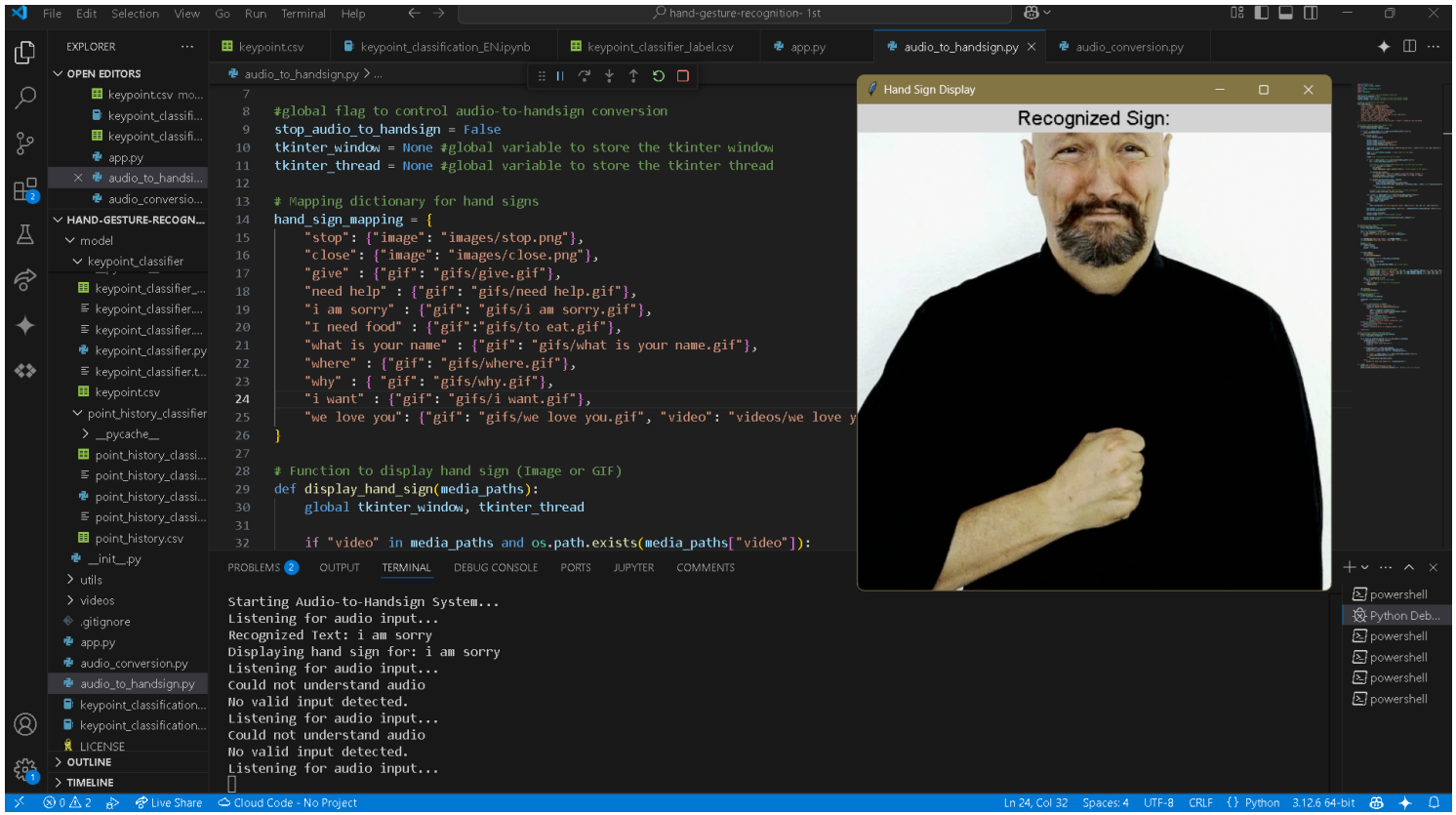

Figure: Running the Voice-to-Handsign Conversion script

INTERIM RESULTS:

- **Current Features:**

  (i) Hand Sign Recognition based on ASL:

  - Utilizes Mediapipe for real-time hand landmark detection. Recognizes 15 specific hand signs:
    Open (ID = 0), Close (ID=1), where is (ID=2), Ok (ID=3), I (ID=4), Want(ID=5), You(ID=6), To eat(Food)(ID=7), give(ID=8), What(ID=9), name(ID=10) I'm Sorry(ID=11) why(ID=12) How(ID=13) I need help(ID=14)

  (ii)Audio Conversions(annoucing and Displaying hand gesture):

  - Detected hand signs are converted into corresponding audio output using pyttsx3.
  - The captured voice is recognised using Google Web Speech API to display resp. hand sign.

INTERIM RESULTS:

(iii)Real-Time Performance:

- Processes webcam input and overlays bounding boxes, hand landmarks, and the detected hand sign label on the video feed.

(iv)Modular Structure:

- The project code is organized into separate modules:
  The "audio conversion.py" handles text-to-speech conversion.
  "app.py" serves as the main script for hand sign detection and system integration.

Challenges: Errors and Solutions during Interim Phase

1. FPS Drop and Unnecessary Delay :

2. RuntimeError in "audio conversion.py":queue-based system to handle TTS requests
3. Repeated Audio Output for the Same Gesture: solved by introducing TTS delay(TTS queue cleared).
4. Offline incompatibility of Google Web Speech API, used in Speech Recognition Library(VOSK, whisper by OpenAI)

Challenges: Developments after Interim Phase

1. Confidence threshold based recognition feature :
   - Cause: previousky Opencv was detecting the hand sign which are most close to its own understanding, increasing false positives.
   - Solution: Set a default value for min. confidence threshold.
     If it doesn't meet min.threshold, displays "UNKNOWN" but doesn't announce.

Challenges: Developments After Interim Phase

2. Introduced "Gesture Speed Adaption and Tracking" Algorithm:
   - The system detect signing speed and dynamically adjust the confidence threshold. Faster gestures require a higher confidence threshold, reducing false positives.
     How It Works:
   - Track the position of key landmarks (e.g., wrist, fingers) over time.
   - Calculate speed based on distance moved per frame.(Euclidean Distance formula)
   - Adjust confidence thresholds or frame capture rate dynamically based on speed.(slow,normal,fast)

Challenges: Developments After Interim Phase

3. Introduced Sentence Formation using Buffer:

- Added sign sentence buffer at the beginning of main().
- Modified gesture recognition to store words in the buffer.
- Added logic to announce full sentences after a delay of 3 seconds.
- Added display of sentence buffer on the OpenCV window.

4. Added toggling ON/OFF for TTS announcement.

5.Extended to video and GIF mapping for the "audio to handsign.py" from only image mapping.

6. Improved UI tkinter window, more user friendly, with control buttons

Challenges: Developments After Interim phase

7. Customly Trained the system with American Sign Language(ASL).

- Used a Two-Key Combination for logging keypoints of Class IDs upto 99.
- allows you to log up to 100 unique gestures (class IDs 0–99) by pressing two keys in sequence (e.g., 1 + 0 for class ID 10).

- Since we have trained 15 ASL hand gestures, The total number of sentences that can be formed by the '15 ASL gestures' that I customly trained is around '3000' random sentences!

  Examples: "I want you," "I need help," "I'm sorry." "What is your name?", "Why?", "How?", "You need help.", "I want you to eat.", "Why need help?".. and so on.

Controls and Key Bindings

- k = to log the keypoints
- h = to log the point history
- n = Toggle audio to handsign conversion mode
- Pressing 0 to 9 and upto 99(Class IDs) and = to logging the keypoints to class IDs certain hand landmarks
- c = to clear the sentence buffer
- s = to terminate creating sentences
- a = to toggle TTS announcement

Conclusion

- Overcame Technical Challenges :
  Fixed FPS drop, TTS errors, and offline incompatibility issues. Introduced confidence thresholds and Gesture Speed Tracking for better accuracy.
- Enhanced User Experience :
  Added sentence formation, TTS toggling, and video/GIF support. Improved UI with user-friendly controls.
- Custom Hand gestures trained based on ASL :
  Trained 15 ASL gestures for now, enabling 3000+ sentence combinations. Examples: "I want you," "I need help," "What is your name?"

Work Distribution

SHAFEEHE	System Design and Methodology, Setting algorithm for gesture detection and audio conversions.
FIDA	Data Collection, Prepare Datasets and Model Training for Gesture Recognition.
NIDA	Integrate MediaPipe and TTS components, Design User Interface, Conducting system tests.
JASIR	Validating model accuracy, Project Coordination, User feedback.
ALL MEMBERS	Literature Review, Documentation and Presentation.

References

- P Sharmila, V Bri and R Jeg. (2023), 'IoT-Oriented Gesture Automation with Mesh Detection through OpenCV and Pyfirmata Protocol using ResNet Mediapipe', Innovations in Power and Advanced Computing Technologies (i-PACT), 2023 IEEE — DOI: 10.1109/I-PACT58649.2023.10434872
- Abraham, A. and Rohini, V. (2018) 'Real-time conversion of sign language to speech and prediction of gestures using artificial neural network', Procedia Computer Science, 143, pp. 587–594. DOI: 10.1016/j.procs.2018.10.435.
- Athira, P.K., Sruthi, C.J., and Lijiya, A. (2022) 'A signer independent sign language recognition with co-articulation elimination from live videos: An Indian scenario', Journal of King Saud University – Computer and Information Sciences, 34, pp. 771–781. DOI: 10.1016/j.jksuci.2019.05.002.
- Sharma, S. and Singh, S. (2021) 'Vision-based hand gesture recognition using deep learning for the interpretation of sign language', Expert Systems with Applications, 162, pp. 113778. DOI: 10.1016/j.eswa.2020.113778 .

THANK YOU