ITCS 6166
Computer and Communication Networks

Project Report

Facial Animation: A Real-Time 3D Face Mapping

Group 7

Saumit Chinchkhandi	Diven Ahuja	Tushar Nemade	Sayyam Gada
801305320	801310223	801257370	801307762

Introduction

Project Aim

In this project, we aim to leverage the power of MediaPipe's face detection and tracking module to build an application that can detect and track faces using a live camera. We have successfully used the Mediapipe ML model for identifying facial features and displaying a face mesh highlighting facial boundaries like eyes, nose and lips.

We aim to build a user-friendly and intuitive interface for real-time facial feature detection and tracking, which can be used for various applications such as video analytics, security systems, augmented reality etc. By using Mediapipe's robust and efficient face detection and tracking algorithm, we achieved high accuracy and performance while maintaining a low computational footprint. Additionally, the project demanded integrating WebRTC components for adding real-time features and functionalities. Streamlit WebRTC provides these features which helps in building a client-server architecture and so our group decided to use Streamlit WebRTC for making our project.

Face Mesh ML Model

Using MediaPipe Face Mesh [1], 468 3D face landmarks may be calculated in real-time. With only a single camera input, it employs machine learning (ML) to infer the 3D facial surface. The strategy provides the real-time throughput critical for live experiences by utilizing lightweight model architectures throughout the pipeline. It uses transfer learning to train a model that concurrently predicts 2D semantic contours on annotated real-world data and 3D landmark coordinates on synthetic rendered data for 3D facial landmarks. It generates 3D landmark predictions from the resulting model using both synthetic and real-world data.

A clipped video frame without any additional depth input is passed to the 3D landmark network as input. The 3D point coordinates and detects the likelihood that a face is recognized and adequately aligned in the input are both output by the model. Predicting a 2D heatmap for each landmark is a frequent alternative strategy, but it is not suitable for depth prediction and has high computational demands for such a large number of points. By iteratively bootstrapping and adjusting predictions, the accuracy and stability of the model are boosted.

WebRTC

Real-time communication between web browsers and mobile applications is made possible via the open-source WebRTC (Web Real-Time Communication) technology, which was created by Google. It enables the direct integration of audio and video communication into web pages without the need for any additional plugins or applications. To offer a smooth real-time communication experience. In order to facilitate applications like video conferencing, live streaming, and online gaming, it supports peer-to-peer connection between browsers or mobile devices.

Project Implementation

Project Progress 1 (Weeks 1-2)

In the beginning, our group sat for multiple sessions to deeply study and understand the Mediapipe ML model and its functionality and implementation. Since none of the group members had extensive previous experience with the working of Machine learning and its features, we spent time getting familiar with ML concepts. Once we did this, we began our implementation which involved setting up a Project environment for installing necessary libraries and dependencies.

Once the environment was set up, we redesigned Mediapipe code for removing unnecessary functionalities. We also gained deep insight into the crucial Key Performance Indicators (KPI’s) like min_tracking_confidence and min_detection_confidence which are model parameters used for tweaking configuration settings for facial feature detection. This project progress ended with our group developing a windows application for the model.

Project Progress 2 (Weeks 3-4)

In Project Progress 2, our group integrated Streamlit components for creating a responsive Frontend for showcasing the facemesh using the device camera. Additionally, Streamlit has been extensively used to develop our backend server for helping us implement a single server multiple clients framework for our project.

In the Frontend application developed, we have given user options to specify the number of faces to be detected while using Webcam. Min_Tracking_Confidence and Min_Detection_confidence bars which range from 0 to 100 which are seekbars which can be adjusted to modify face detection and identification rates. During execution, the server is started which can be accessed based on a shareable link which is sent to multiple devices. On opening the link, Webcam/Camera access needs to be granted and the application soon starts detecting faces visible through the device camera.

Project Progress 3 (Weeks 5-6)

In this iteration, we have converted streamlit into streamlit webrtc for implementing webrtc into streamlit. The main project goal was to implement real-time communication while running the project, which is crucial because it allows users to quickly verify what their models can do with handy video input from their own devices, such as webcams or smartphones. This key functionality was achieved with streamlit webrtc. Webrtc uses WebRTC ice server which is a Session Traversal Utilities for NAT(STUN) server which is used for relaying data.

Project Progress 4 (Weeks 7-8)

For this iteration, we began deploying our project on Streamlit Cloud which is a platform for deploying Streamlit apps. However, while doing this we experienced many version and dependencies issues which were fixed by adjusting our project to the required versions. After resolving all dependencies, we were able to successfully deploy our project on Streamlit Cloud but while running our project, the camera was not starting. On researching this issue, we got to know this is an existing issue with Streamlit WebRTC. So, we decided to host the project on our system and shared a link to other devices(mobile phones and laptops) which could access the link and try our FaceMesh project. Thus, the client-server aspect of our project was completed.

Implementation Details

Below mentioned are steps which are required for running our project. Key point to note is that MediaPipe requires Python Version 3.9 or lower so in case there is a higher python version installed on your system, please switch to and install a lower version in an environment for running the project.

1. **Upgrade pip.**

   ```
   python -m pip install --upgrade pip
   ```

2. **Upgrade pip dependencies.**

   ```
   pip install --upgrade pip setuptools wheel
   ```

3. **To install all the dependencies.**

   ```
   pip install streamlit mediapipe streamlit_webrtc
   opencv-python matplotlib itertools numpy
   ```

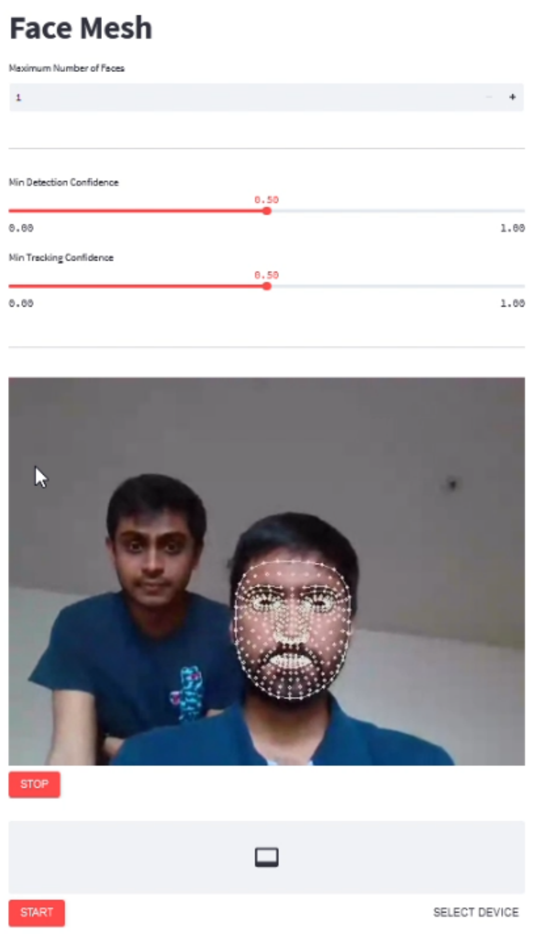
4. **Execute the python file.**

   ```
   streamlit run face_mesh.py
   ```

By following the above steps in an ordered manner, you would be able to set up and run the project successfully.

Screenshots

In the left screenshot, when we set the Maximum number of faces to 1 but in the camera there are 2 faces, the model still detects 1 face which is how it should be displayed. However, in the right screenshot, the maximum number of faces is set to 2 and our model correctly showcases facemesh on both the faces.

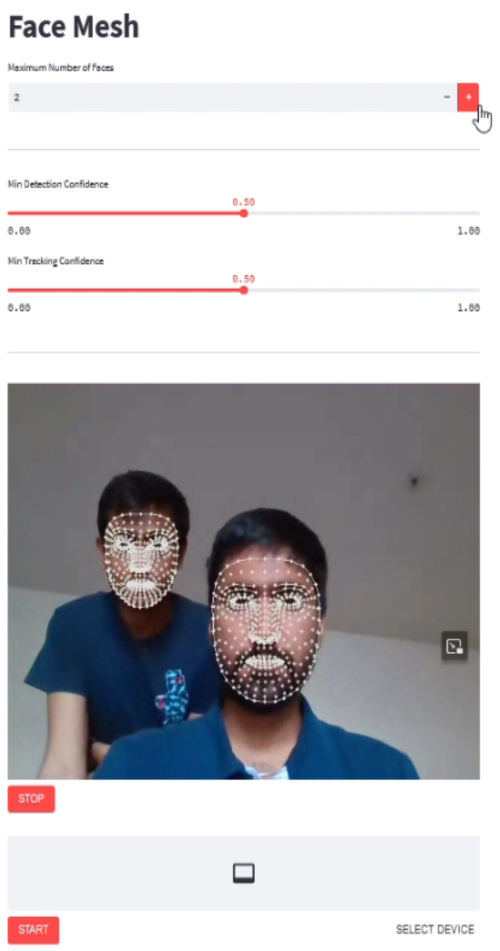


Face Mesh

Maximum Number of Faces: 1

Min Detection Confidence: 0.50 (0.00 – 1.00)

Min Tracking Confidence: 0.50 (0.00 – 1.00)



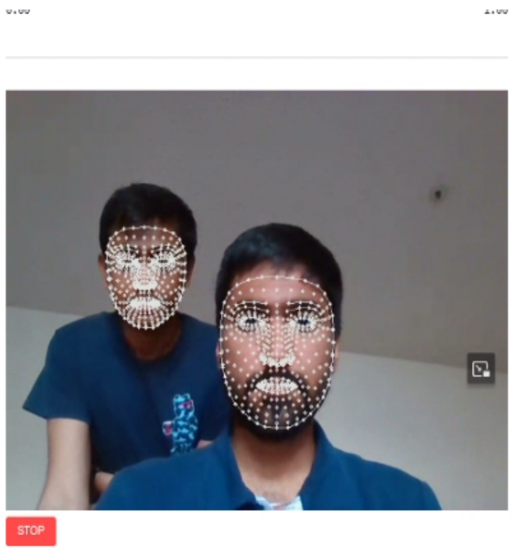
Face Mesh

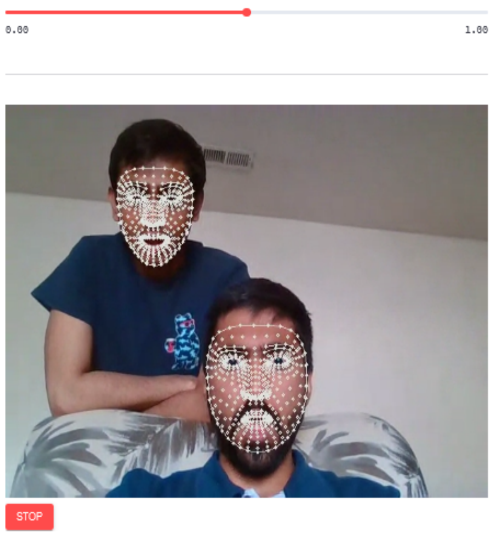
Maximum Number of Faces: 2

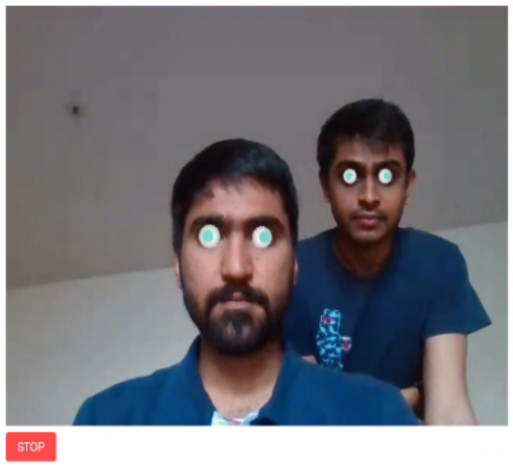
Min Detection Confidence: 0.50 (0.00 – 1.00)

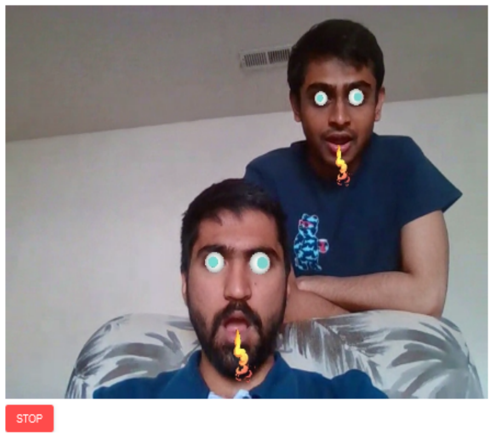
Min Tracking Confidence: 0.50 (0.00 – 1.00)

In the below screenshots, we also have added another filter which correctly identifies eyes and mouth and superimposes white eyes and on opening mouth shows a snake split tongue. This additional filter was an additional feature added by our group along with the facemesh detection.









Challenges Faced

While working on this project, our team faced numerous challenges which ranged from minor fixes to sometimes researching long hours trying to implement a solution. Below we have listed some of the challenges faced during project implementation:

1. **Setting up project environment:** We faced many dependencies and version issues while implementing the Mediapipe model. These issues were resolved by upgrading pip and installing necessary libraries.
2. **Mediapipe ML Model:** Due to minimal background in Machine learning, our group had multiple group sessions for understanding the working of Mediapipe and its functionalities. We also had to understand its implementation and solve any issues that arose during the implementation.
3. **Integration of ML model with Streamlit**: Streamlit frontend application was crashing while integrating our ML model with it. After tweaking and adjusting the model and Streamlit code, we were able to achieve stability and reliability with the Frontend application.
4. **Deploying Project to Streamlit Cloud:** Our team faced issues while deploying the project to cloud with the camera bugging and shutting off repeatedly. So, we decided to host the project on our system and shared a link to other devices(mobile phones and laptops). Thus one system could act as a server and others as clients who could access our Facemesh project using the shareable link.

These were the main issues or roadblocks our team experienced during the implementation. However, we were able to come up with quick alternatives and solutions for getting through these issues and complete the project.

Conclusion

The ability of MediaPipe's face detection and tracking module was effectively used in this project to create a real-time facial feature detection and tracking application. The project achieved great accuracy and performance with a small computational footprint by utilizing Mediapipe's ML model for recognizing facial characteristics and presenting a face mesh highlighting facial boundaries like eyes, noses, and lips.

The project's goal was to design an easy-to-use interface for a variety of applications, including augmented reality, security systems, and video analytics. Real-time features and functions were made available by integrating WebRTC components with Streamlit, which also assisted in the development of a client-server architecture.

Overall, the project has applications in security, entertainment, education, and healthcare, among other fields. This application's accurate and effective face detection and tracking capabilities can be utilized to improve safety and security protocols, user experiences, and data analytics.

References

1. https://github.com/google/mediapipe/blob/master/docs/solutions/face_mesh.md
2. https://github.com/whitphx/streamlit-webrtc