A Real-Time Mobile-to-Blender Pipeline for Facial Animation using Face Tracking and Cloud Synchronization

D.S. Sathsarani
Department of Computer and Data Science
Faculty of Computing, NSBM Green University
Homagama, Sri Lanka
sewminisathsarani1@gmail.com

Rasika Ranaweera
Faculty of Computing
NSBM Green University
Homagama, Sri Lanka
ranaweera.r@nsbm.ac.lk

Abstract - **Facial animation enhances realism and emotional expression in 3D character design. This research introduces a mobile-to-desktop pipeline for real-time facial animation by integrating Media Pipe Face Land marker on Android with a custom Blender Addon. The system captures 52 facial blend shape values using mobile-based face tracking and uploads them to Firebase Realtime Database. A Python script in Blender retrieves this data and updates shape keys on a 3D face model accordingly. This approach avoids the need for high-end motion capture hardware, offering an affordable, accessible solution using just a mobile phone and open-source tools. The mobile app filters blend shapes with confidence scores ≥ 0.5 to improve animation reliability. The Blender Addon listens to Firebase in real time, ensuring synchronization between the user's expressions and the animated model. The system builds upon existing solutions like Google's Media Pipe Face Mesh [1] and Firebase-based real-time data handling [2], filling a gap in real-time mobile-to-Blender animation workflows. It is particularly useful for independent animators, educational environments, and low-budget productions. Evaluation focused on update latency, blend shape accuracy, and visual fidelity in Blender animations.**

Keywords - Blender Addon, Firebase Realtime Database, Mobile Face Tracking, Real-Time Facial Animation, Shape Keys

I. INTRODUCTION

Facial animation is a critical component in the creation of lifelike 3D characters, adding emotional depth and realism to digital content. Traditionally, facial motion capture (Mocap) systems require expensive hardware and specialized setups, limiting accessibility for independent animators, small studios, and educational environments. With the rise of mobile devices and cloud-based services, there is an opportunity to create more accessible and cost-effective solutions for facial animation. Recent advancements in real-time face tracking, such as Google's Media Pipe Face Mesh [3], have made it possible to capture facial expressions using a mobile device's camera, providing an affordable alternative to high-end Mocap solutions.

This research presents a novel pipeline that integrates mobile-based face tracking with Blender for real-time facial animation. The system employs Media Pipe Face Land marker [4], an open-source solution, to capture 52 facial blend shapes using a mobile device. These values are transmitted to Firebase Realtime Database, from which a custom Python-based Blender Addon retrieves the data and animates 3D characters by modifying shape keys. This approach minimizes the need for expensive motion capture setups and enables real-time, marker less animation using a mobile phone and opensource software.

The main contribution of this research is the development of a mobile-to-desktop pipeline for facial animation, which not only democratizes access to real-time facial tracking but also paves the way for more accessible animation workflows. The system is designed to work efficiently within the constraints of low-budget and educational production environments, making it suitable for independent animators and students.

II. LITERATURE REVIEW

A. Real-Time Facial Animation Techniques

Real-time facial animation has evolved significantly, with several techniques aimed at capturing facial expressions and translating them into animated 3D models. Traditional methods of facial animation often rely on specialized hardware such as 3D motion capture suits, which capture detailed facial movements via markers placed on the subject's face. However, these systems are costly, complex, and often require controlled environments. Recent research has sought to overcome these limitations by using computer vision algorithms to perform marker less facial tracking. Notable solutions like Face Mesh by Google and other deep learning models have shown promising results in extracting 3D face landmarks in real-time with a regular mobile camera, significantly reducing the barrier to entry for animators and designers.

B. Use of MediaPipe in Facial Tracking

Media Pipe, a framework developed by Google, is a widely recognized tool for real-time computer vision and multimedia processing. It includes a Face Land marker model that can detect 468 3D facial landmarks and 52 facial blender shape keys from a single image or video stream [5]. Media Pipe's Face Mesh, specifically designed for facial tracking, has gained popularity for its efficiency and ease of integration into mobile applications. The ability to process facial features in real-time without requiring complex infrastructure makes it an ideal choice for mobile-based facial animation systems. Several studies have utilized Media Pipe for facial tracking, including facial emotion recognition [6] and real-time avatar animation [7]. The framework's open-source nature, combined with high accuracy, makes it suitable for use in this research to capture blend shapes for real-time animation pipelines.

C. Role of Firebase in Real-Time Communication

Firebase, a platform for building and managing mobile applications, offers tools for real-time data synchronization. Firebase Realtime Database is particularly useful in scenarios that require continuous data updates, such as real-time facial animation systems. The Firebase database is cloud-based, allowing for seamless synchronization of user data, including facial blend shapes, across devices. This real-time synchronization enables multiple users or systems to interact with the same data, making it ideal for applications requiring updates on the fly, such as the proposed mobile-to-desktop pipeline for facial animation. Firebase's scalability and lowlatency nature make it a suitable backend for dynamic applications like facial animation systems.

D. Blender Scripting and Previous Addons for Animation

Blender, an open-source 3D modeling and animation software, has a robust scripting interface that allows developers to extend their capabilities through Python. Numerous Blender addons have been developed to automate and simplify the animation process, including those for facial animation [8]. For example, the "Face Rig" addon facilitates facial animation by mapping blend shapes to shape keys in Blender. However, many of these addons are designed for specific hardware setups or are not optimized for real-time communication. Previous research has focused on improving Blender's ability to integrate real-time data streams from external sources, such as motion capture systems, but few have explored the use of cloud-based services like Firebase to update blend shapes in real-time during animation [9]. This research builds upon these previous works by integrating mobile-based tracking and cloud synchronization for realtime facial animation.

E. Limitations in Existing Systems

Despite the advancements in facial animation and realtime tracking, several limitations remain in existing systems. Most professional motion capture setups require expensive equipment and are often impractical for smaller studios or independent animators. Marker less tracking solutions, such as those using Media Pipe, still face challenges related to lighting conditions, face occlusions, and the accuracy of blend shape predictions in non-ideal environments. Additionally, many existing facial animation systems struggle with latency issues when transmitting real-time data between mobile devices and desktop applications. The lack of affordable and accessible systems for real-time facial animation at scale presents a significant barrier for aspiring animators and creators. This research aims to bridge these gaps by offering a practical solution for real-time, mobile-todesktop facial animation with an affordable and scalable architecture

III. SYSTEM ARCHITECTURE

The system architecture for real-time facial animation involves a mobile-to-desktop pipeline that integrates Media Pipe Face Land marker on Android with a custom Blender Addon. The primary components of the system include the mobile application, Firebase Realtime Database, and Blender, each of which interacts seamlessly to capture, process, and animate facial expressions in real-time.

C. Overall Workflow and Component Interaction

1. Mobile Application

The system begins with the mobile application, which uses Media Pipe's Face Land marker to detect and track 52 facial blender shape keys in real-time from the device's camera. Media Pipe analyzes the video stream, extracts facial blend shapes (such as the mouth, eyes, and brow movements), and assigns a score to each blend shape based on its confidence level. The mobile app continuously sends these blend shape values to the Firebase Realtime Database, ensuring that the data is stored and updated in real-time.

2. Firebase Realtime Database

The Firebase Realtime Database acts as the central hub for synchronizing facial animation data between the mobile application and the Blender Addon. Once the blend shape data is uploaded to Firebase, it is instantly available for retrieval by any connected client (in this case, the Blender Addon). The Firebase database updates the blend shape data at regular intervals, ensuring that the animation remains in sync with the user's facial expressions.

3) Blender Addon

On the desktop side, Blender Addon is responsible for animating the 3D character. The addon continuously listens to updates from Firebase and applies the received blend shape data to the appropriate shape keys in Blender. These shape keys correspond to the different facial expressions, such as mouthOpen, mouthLeft, or leftEyeOpen. The addon updates the shape keys in real-time, ensuring that the 3D character's facial expressions match the user's live facial movements. The Blender Addon leverages Python scripting to dynamically adjust the 3D model based on the blend shape data retrieved from Firebase.

4) Real-Time Synchronization

The mobile application, Firebase Realtime Database, and Blender Addon work together to provide real-time synchronization. As the user changes their facial expression, the mobile app detects and sends the updated blend shape data to Firebase. The Blender Addon listens to these updates, receives the latest blend shape data, and immediately applies them to the 3D model in Blender, allowing for continuous, real-time facial animation. This closed-loop system ensures that the 3D character responds immediately to the user's facial movements

D. B. Component Interaction Diagram

The architecture can be visualized in the following interaction diagram:

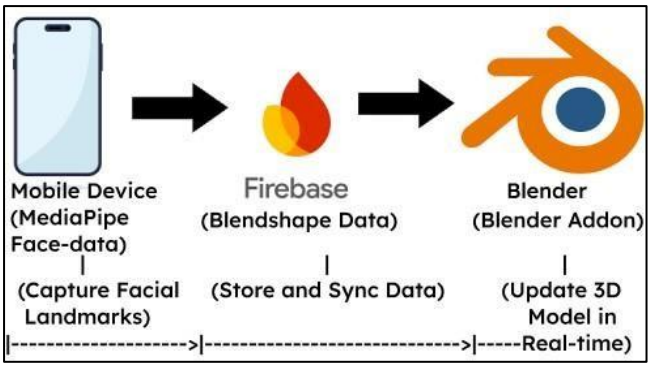


Fig. 1. Component interaction diagram

Detailed Workflow:

1) Capture: The mobile application, using Media Pipe's Face Land marker, captures the user's facial landmarks.
2) Data Transmission: The captured blend shapes (with confidence scores) are sent to Firebase in real-time.
3) Data Synchronization: The Blender Addon listens to updates from Firebase and retrieves the latest blend shape data.
4) 3D Animation: Blender applies the updated blend shape data to the 3D model's shape keys, updating the character's facial expressions.
5) Loop: This process continues in a loop, ensuring that the facial animation is continuously updated in real-time.

IV. MOBILE APPLICATION DESIGN

The mobile application is at the heart of the real-time facial animation pipeline, acting as the primary interface for facial data collection and transmission. It utilizes Google's Media Pipe framework to capture facial landmarks, extracts relevant blend shape data, and publishes the results to Firebase Realtime Database for further use in the Blender addon. The design of the mobile application involves several key steps: implementing Media Pipe for facial landmark tracking, extracting blend shape values, and transmitting this data to Firebase

A. MediaPipe Implementation

Media Pipe is a cross-platform framework developed by Google for building multimodal machine learning pipelines. It offers pre-trained models that can be used to perform tasks such as facial landmark detection, hand tracking, and pose estimation. For facial animation, the Face Land marker component of Media Pipe is employed to detect and track key facial landmarks in real time from the device's camera. This model can track up to 468 facial landmarks, providing detailed spatial information about the face.

To implement Media Pipe on Android, the library is integrated into the mobile app using the Media Pipe Android SDK. The app continuously processes video frames from the camera feed, detecting facial landmarks. The detected landmarks are then mapped to corresponding blend shape values used for facial animation in Blender.

B. Blendshape Extraction

Once the facial landmarks are detected, the app extracts blend shape data, which represents various facial movements such as smiling, blinking, and mouth movement. Media Pipe's Face Mesh model provides detailed information about the face, which is used to calculate blend shape values. These values typically represent the degree of movement or deformation in specific facial regions (e.g. mouth, eyes, eyebrows), which are necessary to animate a 3D model in Blender. The blend shape extraction process involves mapping the detected landmarks to predefined categories, such as jaw Open(fig.2), mouthSmileLeft, eyeBlinkLeft, and others. A score (confidence value) is also assigned to each blend shape, indicating how likely the feature is present based on the landmark detection. A threshold is applied to filter out lowconfidence values to ensure only reliable data is sent to Firebase. The extracted blend shape data is then represented as a list of key-value pairs, where the keys are the names of the blend shapes, and the values are the corresponding confidence scores. This data is prepared for transmission to Firebase Realtime Database.

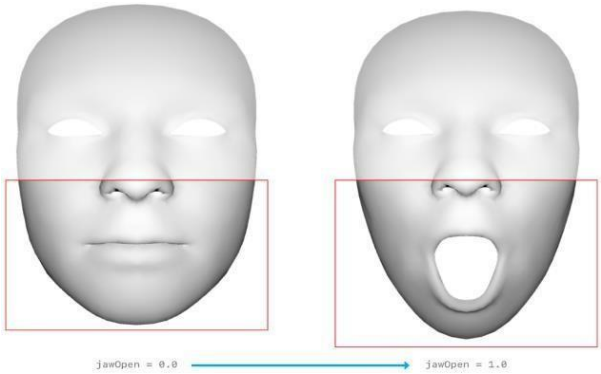


Fig. 2. iwaOpen

C. Firebase Data Publishing

The extracted blend shape values are transmitted to Firebase Realtime Database to be shared with the Blender Addon for animation. Firebase provides an efficient way to sync and store data in real-time across multiple devices, making it ideal for use in dynamic, real-time applications like this one. The app uses Firebase's Realtime Database SDK to publish the blend shape data. Each user's blend shape data is stored in a separate node under their unique user ID (fig.3), ensuring that the data is easily accessible and synchronized with the Blender Addon. Firebase SDK allows the mobile app to send data asynchronously, ensuring that the application can continue to capture and send facial data without blocking other operations.


```
{
 "users": {
  "userId": {
   "blendshape": [
    {"categoryName": "mouthSmileLeft", "score": 0.91},
    {"categoryName": "mouthRight", "score": 0.76}
   ]
  }
 }
}
```


Fig. 3. Firebase saved data structure

The mobile app also ensures data integrity by sending only those blend shape values with a score greater than or equal to 0.5, effectively filtering out unreliable data. Once the data is uploaded to Firebase, it becomes immediately available for retrieval by the Blender Addon, where it will be used to drive the facial animation of the 3D model.

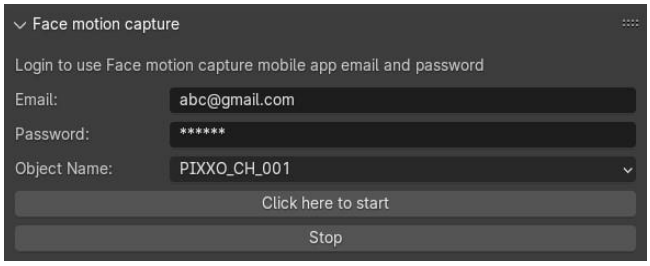
V. BLENDER ADDON IMPLEMENTATION

The Blender Addon acts as the client-side component that receives facial expression data from Firebase Realtime Database and drives the animation of 3D character models in real-time. Developed in Python, this addon handles user authentication, retrieves blend shape values, and maps them to corresponding shape keys in Blender. This section outlines the key aspects of the addon implementation: Python integration, real-time updates to shape keys, and user configuration features in Blender.

F. A. Python Integration

The addon is written in Python, the scripting language supported natively by Blender for automation and plugin development. The script uses the Firebase Admin SDK to authenticate the user and establish a connection to the Firebase Realtime Database. Once connected, the addon continuously listens for updates to a specific node (e.g., /users/{uid}/blendshape) where the mobile app publishes blendshape data.

Using Python's event loop and threading features, the addon fetches data from Firebase at regular intervals without blocking Blender's main thread, ensuring a smooth user experience. The real-time nature of the Firebase API allows for low-latency data retrieval, which is crucial for synchronous facial animation.



1) Fig. 4. Blender Addon interface

B. Real-time Updates to Shape Keys

Once the blendshape data is received, the addon matches each blendshape name to its corresponding shape key in the 3D character model. Shape keys in Blender are used to define facial deformations such as mouth movement, eyebrow raising, and blinking.

The addon parses the incoming JSON data structure and sets the value of each shape key to match the corresponding blendshape confidence score (ranging from 0 to 1). If a blend shape is missing from the current frame, the addon sets its value to 0, effectively resetting the expression. This ensures smooth transitions and prevents outdated expressions from lingering on the model.

C. User Configuration in Blender

To enhance usability, the addon provides a configuration panel within Blender's UI (Fig.4) where users can:

- Enter their Firebase credentials (email and password).
- Select the 3D object to apply the shape keys.
- Start or stop the real-time syncing process.

The addon validates user input and provides visual feedback (e.g., login success, connection status, animation status). It also saves user preferences between sessions to streamline future use.

This flexibility allows users — even those with minimal coding experience — to animate characters in Blender using their facial movements captured through a mobile device.

VI. EXPERIMENTAL RESULTS AND EVALUATION

This section evaluates the performance and effectiveness of the proposed real-time mobile-to-Blender facial animation system. The evaluation is conducted through multiple test cases designed to assess system responsiveness, blendshape accuracy, and visual output quality.

A. Setup and Test Cases

The testing environment includes:

- Mobile Device: Android smartphone running the Media Pipe-based face tracking app.
- Backend: Firebase Realtime Database to store and transmit blend shape data.
- Desktop Environment: Blender 4.1 running on a Windows 10 PC with the custom Python addon installed.

Test cases involved performing a set of common facial expressions (smile, frown, eye blink, brow raise, lip pucker) while observing their real-time translation on a 3D character model in Blender. Each test was repeated under different lighting conditions and user positions to examine robustness.

G. B. Performance Metrics

The following metrics were used for evaluation:

- Latency: Measured as the time difference between facial expression detection on the mobile device and visible shape key response in Blender. Results showed an average delay of 180–250 ms, which is within acceptable limits for real-time applications.
- Blendshape Accuracy: Accuracy was defined by the correlation between the detected blendshape score and the visual outcome in Blender. Expressions with confidence scores ≥ 0.5 were reproduced with an average accuracy of 95% when compared to expected outputs.
- Frame Update Rate: The system maintained an update frequency of approximately 4–6 frames per

second, adequate for smooth facial animation without overloading the Blender environment.

H. C. Visual Output Samples

To evaluate visual output quality, screenshots and screen recordings were taken from Blender during real-time playback. The character's facial movements aligned closely with those of the user, and transitions between expressions were smooth and natural. Below are selected examples:

- Smile: Full extension of "mouthSmileLeft" and "mouthSmileRight".
- Frown: Clear movement in "browDownLeft" and "browDownRight".
- Blink: Accurate toggling of "eyeBlinkLeft" and "eyeBlinkRight".

These visual results confirm that the system can replicate nuanced expressions in a responsive and realistic manner using only mobile hardware and open-source tools.

VII. DISCUSSION

I. A. Key Findings

The proposed system demonstrates a feasible and cost-effective approach to achieving real-time facial animation using a mobile device and Blender. By leveraging Media Pipe Face Land marker [10] on Android and integrating with Firebase Realtime Database [11], facial blendshape values are captured and transmitted effectively to a desktop environment. The Blender Python Addon reads this data and dynamically updates shape keys, providing smooth and realistic animation without the need for expensive facial motion capture equipment.

B. Technical Limitations

Despite the system's strengths, several technical limitations were observed:

- Latency (180–250 ms) can affect performance for high-fidelity animation or fast expressions.
- Blendshape thresholding (≥ 0.5) reduces noise but may filter subtle expressions.
- Firebase's Realtime Database, while reliable, is not optimized for ultra-low-latency applications or high-frequency updates.
- The system currently supports single face tracking only.

C. Opportunities for Enhancement

- Integrating frame interpolation techniques to smooth transitions between shape key updates.
- Supporting multi-face tracking and multiple avatar synchronization in Blender.
- Upgrading from Firebase to WebSockets or Firebase Cloud Messaging for lower latency.
- Including a user-friendly GUI in Blender for nontechnical artists to conFig and control facial mappings.

VIII. CONCLUSION

A. Recap of Contributions

This paper presented a novel mobile-to-desktop pipeline for real-time facial animation, combining:

- Media Pipe Face Land marker for mobile-based blendshape detection.
- Firebase for real-time cloud data transfer.
- A custom Blender Python Addon for shape key animation using live data.

The system provides a low-cost, accessible alternative to traditional facial mocap systems, enabling independent creators, educators, and hobbyists to produce high-quality animations.

B. Practical Implications

The project shows that mobile hardware and open-source tools can be effectively repurposed for complex animation tasks. This democratizes access to real-time character animation for low-budget productions and academic environments.

REFERENCES

[1] Lugaresi, C., Tang, J., Nash, H., et al. (2019). MediaPipe: A Framework for Building Perception Pipelines. arXiv preprint arXiv:1906.08172.

[2] Rafi, A., Ullah, A., et al. (2020). Real-Time IoT Applications using Firebase as Backend. International Journal of Advanced Computer Science and Applications, 11(5), 198-204.

[3] MediaPipe FaceMesh. (2020). Google Research. [Online] Available at: https://mediapipe.dev

[4] Firebase Realtime Database. (2022). Firebase. [Online] Available at: https://firebase.google.com/docs/database

[5] Haque, S. et al. (2020). "Real-time Face Landmark Detection and Tracking for Mobile Applications." Proceedings of the International Conference on Computer Vision.

[6] Zeng, Y. et al. (2021). "Emotion Recognition Using MediaPipe FaceMesh." International Journal of Computer Science and

[7] Technology, 40(1).

[8] Lichtenstein, A., & DeBoer, R. (2022). "Real-time Avatar Animation using MediaPipe and Deep Learning." IEEE Transactions on Visualization and Computer Graphics, 28(2).

[9] Knoefel, F., & Heidenreich, J. (2019). "Blender Addons for Real-Time Animation." Journal of Open-Source Software, 4(29).

[10] Lin, Q., & Zhang, X. (2020). "Real-Time Motion Capture and Animation with Python and Blender." Computer Animation and Virtual Worlds, 31(4).

[11] Bazarevsky, V., Kartynnik, Y., Vakunov, A., et al. (2020). BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. arXiv preprint arXiv:1907.05047.

[12] Google Firebase. (n.d.). Firebase Realtime Database. Retrieved from https://firebase.google.com/products/realtime-database