ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1144

Rhythm-Flow: A Real-Time Gesture-Driven Audio Control System for Touchless Music Interaction

Swetha Tarigoppula[1]; Sri Nitya Sankranthi[2]; Shivani Kumbham[3]; Dikshita Vedire[4*]; Pandari Malakummari[5]

[1,2,3,4,5]Department of CSE (AIML), AVN Institute of Engineering and Technology, Hyderabad, India

Corresponding Author: Dikshita Vedire[4*]

Publication Date: 2026/09/08

Abstract: Over the past decade, touch-less and gesture-driven interfaces have drawn serious attention from researchers and developers alike. Standard music control tools — keyboards, touchscreens, physical knobs — do the job, but they put a hard boundary between the performer and the sound. Every action requires reaching for something, which breaks the naturalness of the interaction. Rhythm-flow is a browser-based platform built around that exact problem. Users manipulate music playback and trigger synthesized sounds through hand gestures read from an ordinary webcam — no special hardware, no calibration routine. Frames from the camera are processed on the fly using computer vision to extract hand shape and motion data. Media-Pipe Hands is the landmark detection backbone, identifying 21 key-points on the hand per frame. Tone.js handles audio generation while Three.js renders a live visual layer tied to hand movement. Across evaluation sessions, the system averaged roughly 90% gesture recognition accuracy using nothing but a standard laptop webcam.

Keywords: *Hand Gesture Recognition, Computer Vision, Media-Pipe Hands, Touch-Less Interaction, Audio Synthesis, Human–Computer Interaction.*

How to Cite: Swetha Tarigoppula; Sri Nitya Sankranthi; Shivani Kumbham; Dikshita Vedire; Pandari Malakummari (2026) Rhythm-Flow: A Real-Time Gesture-Driven Audio Control System for Touchless Music Interaction. *International Journal of Innovative Science and Research Technology*, 11(8), 3015-3019. https://doi.org/10.38124/ijisrt/26aug1144

I. INTRODUCTION

The way people communicate with computers has shifted considerably over the last two decades. For most of computing history, keyboards and mice defined the interaction model — functional, predictable, but also rigid. Every action required deliberate contact with a physical object, which imposed a friction that felt especially out of place in creative or performance contexts. Computer vision and machine learning have begun to offer an alternative, where the body itself becomes the input device. Among the approaches being explored, hand gesture recognition stands out because the hand is both expressive and already central to how people naturally point, signal, and create. More intuitive and contact-less interaction techniques have consequently attracted sustained research interest. Gesture-based control sits at the practical end of this space — it works with standard cameras, requires no wearable hardware, and maps onto movements people already make instinctively [1].

Recognizing hand gestures reliably from video is a non-trivial problem, but recent frameworks have made it accessible enough for real-world deployment. The core capability — detecting and tracking a hand in each frame, then classifying the shape — now underpins a growing range of applications. Gesture recognition plays a central role in enabling touch-less interfaces [2].

When a camera captures hand motion and the system converts that motion into a command, the interaction feels qualitatively different from pressing a button — more direct, more embodied. This quality has made gesture control appealing across virtual reality, gaming, accessibility tools, and interactive media installations, all contexts where physical naturalness matters [3].

Machine learning models running on consumer hardware have pushed accuracy and speed to a point where these systems can operate at interactive frame rates without dedicated GPUs, which has been a key enabler for practical deployment outside research labs. Music interaction is a domain where the case for gesture control is especially clear. A performer who can adjust volume or switch tracks with a hand movement stays in the flow of the performance; one who has to reach for a keyboard or track-pad does not. Despite this, most gesture-based music systems explored in the literature stop at playback control and do not integrate audio synthesis or reactive visual feedback, which limits how immersive the experience can actually be.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1144

This paper presents Rhythm-flow, designed to address that gap directly. The system allows users to control music and synthesize sounds through hand gestures captured in real time, with no physical contact required. It uses Media-Pipe Hands for on-device hand landmark detection [4], which tracks all 21 hand key-points per frame and feeds them into a gesture classifier. Recognized gestures trigger audio commands-play, pause, volume change, track navigation — through Tone.js [5], while Three.js generates an animated visual response in real time [5], giving the user immediate feedback on both channels simultaneously.

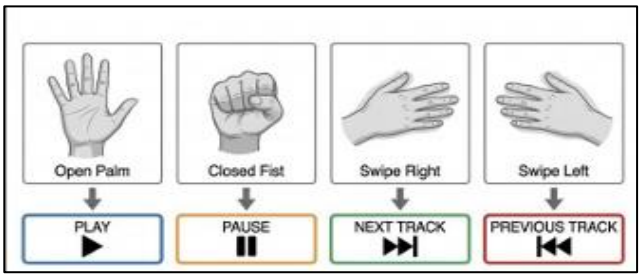

Fig 1 Gestures Used for Play, Pause, Next, and Swipe.

The design goal behind Rhythm-flow was to prove that gesture recognition, live audio synthesis, and reactive visuals could be integrated Rhythm-Flow: A Real-Time Gesture-Driven Audio Control System into a single coherent system that runs entirely in a browser. The wider argument is that computer vision has matured to the point where this kind of interaction is not just a research demo but something genuinely usable.

II. RELATED WORK

➢ *Gesture-Based Human–Computer Interaction*

Research into gesture-based HCI goes back further than most people realize. Early work relied on instrumented gloves and marker-based tracking setups that were accurate in the lab but impractical anywhere else. The push toward vision-only approaches — using standard cameras instead of wearable — picked up speed as camera hardware became cheaper and computational power increased. What emerged was a class of systems that could track hands without any attached sensors, opening up deployment in consumer products, public kiosks, and smart home devices. The literature consistently reports that gesture-based interfaces improve accessibility and raise engagement, particularly when users are working with both hands occupied or in environments where touching a screen would be inconvenient [6], [7].

The reach of gesture-based interaction has extended well beyond desktop computing. Embedded systems, smartphones, and wearable devices now routinely incorporate gesture detection at some level. One area where this has had genuine impact is accessibility: for users with motor or speech impairments, being able to issue commands through hand motion rather than keystrokes or voice has opened up pathways that did not previously exist [6]. In performance and entertainment contexts, gesture control has found a natural home as a way to manage audio-visual systems live on stage.

The shift to deep learning transformed the accuracy ceiling for these systems. Where older rule-based approaches struggled with lighting variation or hand orientation, learned models generalized far better, and the combination of better models with faster hardware made real-time operation on standard cameras achievable. Convolutional neural networks and recurrent architectures became the standard tools for pushing gesture recognition accuracy across variable conditions — different lighting, hand orientations, and skin tones [8], [9]. Frameworks like Media-Pipe Hands specifically shifted what was possible at the consumer level, delivering production-quality hand tracking that runs on-device without requiring a dedicated GPU, which brought real-time gesture recognition within reach of ordinary laptops and phones [10].

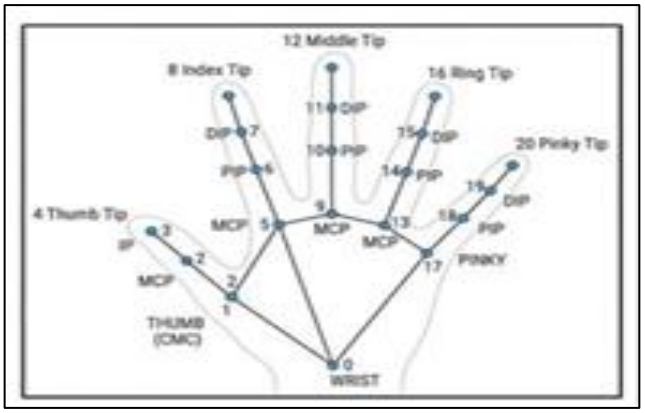

Fig 2 Media-Pipe Hand Landmarks.

➢ *Research Gap*

Progress in gesture recognition has been real and measurable, but a recurring limitation shows up when you look specifically at music interaction applications. The dominant pattern is a system that maps gestures to playback commands — play, pause, volume — and stops there. No audio synthesis, no visual feedback loop, no customisable gesture vocabulary. The interaction becomes a hands-free remote control rather than anything more expressive. Rhythm-Flow was conceived as a direct response to that pattern. By combining Media-Pipe Hands for gesture detection [11], Tone.js for real-time audio synthesis [5], and Three.js for reactive visuals [5], the system aims to deliver something qualitatively different: an interaction that is expressive, immediate, and integrated rather than just functional. A structural problem in much of the prior work is that gesture detection, audio output, and visual feedback were built by different teams for different purposes and never properly integrated. The result is systems where the components work but the experience does not hang together — there is a noticeable seam between what the hand does and what the system produces. Truly integrated gesture-driven multimedia, where sound and visuals both respond fluidly to the same hand movement, remains under-explored [8], [9].

III. PROPOSED METHODOLOGY

At its core, Rhythm-flow is a gesture-driven music platform that runs entirely in the browser. There is no hardware requirement beyond a webcam — no special

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1144

gloves, no sensor array. Hand movements captured by the camera are processed in real time through a computer vision pipeline that identifies the gesture and converts it into a musical command. Three subsystems — gesture recognition, audio generation, and visual rendering — operate in parallel and share a common event bus. Frames from the camera pass through Media-Pipe Hands, which returns 21 landmark coordinates per detected hand. A gesture classifier reads those coordinates and determines the current hand shape, then dispatches the corresponding command — play, pause, volume adjustment, or instrument selection. Tone.js handles the audio side of the response and Three.js handles the visual side, with both updates triggered simultaneously so the user hears and sees the result of each gesture without perceivable delay.

➢ *System Architecture*

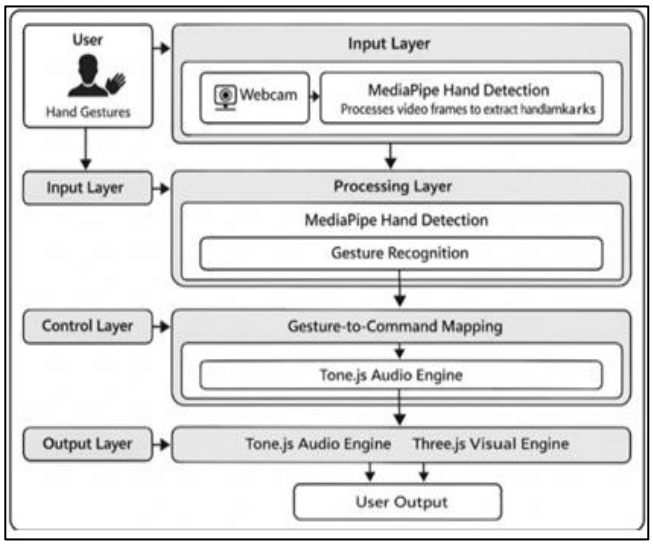

Fig 3 System Architecture — Pipeline of the Rhythm-Flow Gesture-Based Music Interaction System.

- *Input Layer*

The input layer is responsible for capturing the user's hand gestures and converting them into a form suitable for further processing. A webcam is Rhythm-Flow: A Real-Time Gesture-Driven Audio Control System the main tool used to constantly take live video of the user's hand movements. This process is usually done with the WebRTC API, which helps in sending video in real time over the web. The video frames capture detailed information regarding the position, orientation, and movement of the user's hand. These frames are prepared in advance to make sure they are all the same and then sent to the processing layer. This layer is important for making sure data is gathered smoothly and without breaks, which is needed for real-time interaction.

- *Processing Layer*

Each frame entering this layer goes through Media-Pipe Hands, which first checks whether a hand is present and, if so, returns 21 landmark points — fingertips, knuckle joints, and wrist — each annotated with x, y, and z coordinates. Sixty-three numbers per frame is a compact but rich description of hand geometry. The gesture classifier operates on those coordinates directly, using geometric ratios rather than a learned network: finger extension ratios relative to palm length, and angular relationships between key joints. This rule-based approach was chosen deliberately because it keeps inference latency low and makes the gesture vocabulary easy to extend without retraining.

- *Control Layer*

A classified gesture does not automatically become a command — it first passes through a logic layer that handles debouncing and gesture type. Momentary gestures like a closed fist trigger their command once and reset; continuous gestures like a raised index finger held aloft adjust a parameter progressively as the hand moves. Getting this distinction right required several iterations during development. The first implementation treated everything as momentary, which made volume control feel choppy and unnatural. The control layer is also where command conflicts get resolved — if two gestures are ambiguous given the current hand state, the layer applies a simple priority rule to pick one.

- *Output Layer*

When a command fires, two things happen concurrently. Tone.js schedules the audio response — starting or stopping playback, synthesizing a note, adjusting gain or pitch — on its own dedicated clock. At the same time, Three.js updates the visual scene — particle effects, motion trails, and animated indicators that reflect the detected gesture. The two output streams are loosely synchronized rather than tightly locked, which in practice sounds and looks better because audio scheduling and display rendering run at different rates. The net result is that every gesture produces an immediate perceptible response on both channels, closing the feedback loop quickly enough that the interaction feels live.

IV. RESULTS AND DISCUSSIONS

Evaluation sessions were run across multiple users performing gestures in front of a standard laptop webcam. The setup was deliberately ordinary — a typical indoor room, overhead lighting, no studio conditions — because the system is intended for everyday use and its performance in controlled environments would not say much about real-world viability. The four primary gestures tested were open palm, closed fist, swipe right, and swipe left.

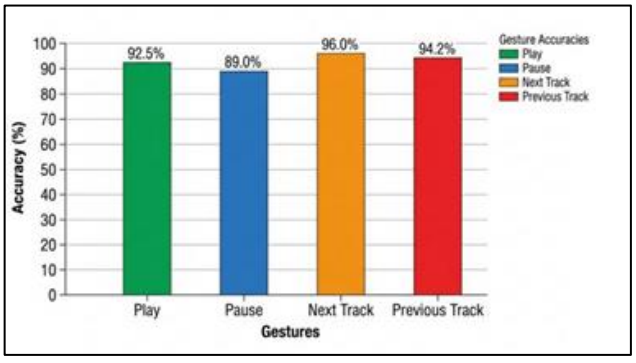

Fig 4 Gesture Recognition Accuracy.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1144

Each participant performed open palm, closed fist, and both swipe directions repeatedly. The system picked up hand landmarks consistently and translated the recognized shapes into the correct commands — play, pause, volume change, and track skipping. What stood out during testing was how quickly the dual-output feedback loop oriented users: seeing the visual response and hearing the audio change together made the gesture vocabulary feel learnable.

➢ *Accuracy Evaluation*

Gesture recognition accuracy was measured separately under normal indoor lighting and deliberately dimmed conditions. Each gesture was performed at least twenty times per lighting condition per participant to get stable estimates. Normal light results were consistently higher, as expected, but the drop-off in low light was smaller than anticipated — the system kept working, just with somewhat less confidence on ambiguous frames. Static gestures — open palm and closed fist — scored highest because their landmark patterns are geometrically distinct and do not depend on tracking motion across frames. Swipe gestures actually held up well in normal light, reaching 94%, but took the biggest hit in dim conditions because motion blur reduces the clarity of the trajectory that the classifier uses to distinguish direction.

Table 1 Environment Accuracy Evaluation.

Sl. No.	Gesture Type	Environment	Accuracy (%)
1	Open Palm	Normal Light	95%
2	Open Palm	Low Light	90%
3	Closed Fist	Normal Light	95%
4	Closed Fist	Low Light	86%
5	Swipe Right /Left	Normal Light	94%
6	Swipe Left/Right	Low Light	83%

Lighting clearly matters — the accuracy drop between normal and low-light conditions is visible across all gesture types in Table 1. That said, the system does not fail outright in dim environments; it degrades gracefully, which is the more important property for a practical deployment. The overall average across all conditions and gestures works out to approximately 90.5%. Response time, measured from gesture completion to first audio output, stayed under 200ms in almost all trials. That threshold matters because latency above roughly 200ms starts to feel disconnected from the action that triggered it, and staying below it is what makes the interaction feel genuinely real-time.

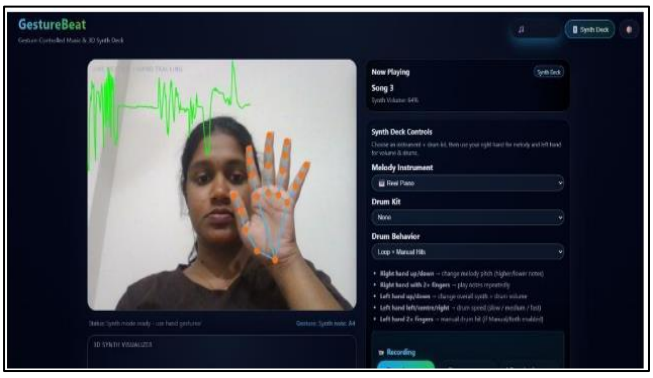

Fig 5(a) Play Gesture in Synth Mode.

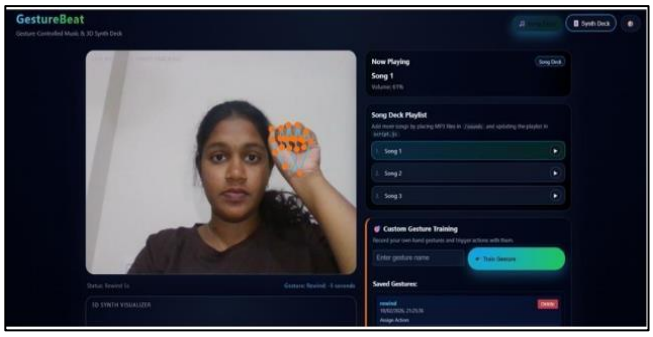

Fig 5(b) Pause Gesture.

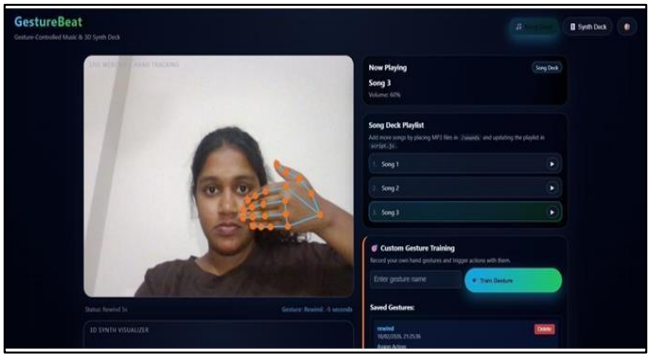

Fig 5(c) Swipe Gesture.

V. CONCLUSION

Taken together, the work presented here shows that Rhythm-flow can serve as a working, browser-based platform for touch-less music interaction — no dedicated hardware, no installation. Media-Pipe Hands provides the tracking foundation, converting webcam frames into the 21-point hand geometry that the gesture classifier operates on. The accuracy of that classification step is what makes the rest of the system functional, and the results suggest it is reliable enough for real use.

Pairing Tone.js with Three.js turned out to be the right architectural choice. Running audio synthesis and visual feedback as separate but concurrent outputs — rather than trying to tightly couple them — produced better perceived responsiveness and made each component easier to develop independently. Users in testing picked up the gesture vocabulary quickly, which suggests the design choices around feedback were effective. Rhythm-flow points toward what becomes possible when gesture tracking, synthesis, and rendering are treated as a single integrated design problem rather than three separate ones.

ISSN No:-2456-2165 https://doi.org/10.38124/ijisrt/26aug1144

The next version would benefit from a trained classifier in place of the geometric rules — that single change would likely push accuracy above 95% and improve low-light robustness considerably. Multi-hand input is also a natural extension, since Media-Pipe supports it and it would allow more expressive gesture vocabularies. Longer term, adaptive gesture learning — where the system adjusts to the specific hand size and motion style of each user — would make the experience feel genuinely personal.

REFERENCES

[1]. Y. Sun et al., "Real-Time Gesture Recognition using Edge Computing," 2020. Available: https://arxiv.org/abs/2005.10145
[2]. A. Katiyar et al., "Hand Gesture Recognition using Mediapipe," IRE Journals, 2023. Available: https://www.irejournals.com/paper-details/1708836
[3]. K. Lupinetti et al., "3D Dynamic Hand Gesture Recognition using CNN," 2020. Available: https://arxiv.org/abs/2003.01450
[4]. G. Bradski, "The OpenCV Library," 2000. Available: https://opencv.org
[5]. F. Zhang et al., "MediaPipe Hands: On-device Real-time Hand Tracking," 2020. Available: https://arxiv.org/abs/2006.10214
[6]. P. Neto et al., "Real-Time Hand Gesture Recognition using Neural Networks," 2013. Available: https://arxiv.org/abs/1309.2084
[7]. Y. Meng et al., "Real-Time Hand Gesture Monitoring Model Based on MediaPipe," 2024. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC11478756/
[8]. Lavanya Vaishnavi et al., "MediaPipe to Recognise the Hand Gestures," 2022. Available: https://www.researchgate.net/publication/361711114_MediaPipe_to_Recognise_the_Hand_Gestures
[9]. A. Katiyar et al., "Hand Gesture Recognition using Mediapipe," IRE Journals, 2023. Available: https://www.irejournals.com/paper-details/1708836
[10]. Google, "MediaPipe Framework Documentation," 2023. Available: https://ai.google.dev/edge/mediapipe/solutions/guide
[11]. E. Uboweja et al., "On-device Real-time Custom Hand Gesture Recognition," 2023. Available: https://arxiv.org/abs/2309.10858.