International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

Three.js Hand Recognition Panel Using MediaPipe and WebGL

[Rahul Sharma]
School of Computing Science and Engineering
[VIT University], [Vellore], [India]

***Abstract*-The rapid evolution of human-computer interaction (HCI) technologies has created demand for natural, contactless interfaces that transcend the limitations of traditional input devices. This paper presents the design and implementation of a real-time gesture-based interaction system that integrates Google's MediaPipe Hands framework with the Three.js WebGL rendering library. The proposed system, titled Three.js Hand Recognition Panel, captures live video from a standard webcam and detects up to two hands simultaneously, extracting 21 three-dimensional landmark points per hand. Distance-based and positional gesture recognition algorithms map these landmarks to interactive commands that drive a WebGL-rendered 3D virtual environment. Operating entirely within a web browser without specialized hardware, the system achieves 50–60 FPS on standard desktops, gesture recognition accuracy exceeding 88%, and average response latency below 70 ms. Experimental results validate the feasibility of deploying production-quality gesture interfaces through open-source web technologies. The work contributes a modular, extensible architecture suitable for applications in gaming, virtual reality, healthcare, and assistive technologies.**

Keywords—Hand gesture recognition; MediaPipe; Three.js; WebGL; Human-computer interaction; Real-time processing; Computer vision; Landmark detection; Browser-based computing

I. INTRODUCTION

In the digital era, data and interaction signals are generated from an ever-expanding array of sources. Just as the analysis of massive datasets demands new computational frameworks [1], the interpretation of human gestural input demands new perceptual pipelines capable of operating efficiently on consumer hardware. Human-computer interaction has advanced from keyboard-and-mouse paradigms toward contactless, vision-based interfaces that leverage machine learning to understand natural body language.

Big data platforms such as Apache Hadoop and Spark [1] demonstrated that complex workloads could be distributed across commodity hardware to achieve scalable, real-time analytics. An analogous philosophy motivates the present work: by distributing gesture inference, coordinate transformation, and 3D rendering across the GPU and CPU resources already present in any modern browser, a low-cost, scalable gesture interface becomes achievable without specialized sensors or proprietary software.

Gesture-based systems have historically depended on expensive hardware such as data gloves, infrared sensors, and depth cameras. These approaches impose hardware costs, calibration overhead, and deployment constraints that limit accessibility. The availability of Google's MediaPipe Hands [2]—a WebAssembly-compiled machine learning pipeline delivering 21 three-dimensional hand landmarks at real-time frame rates in the browser—removes the primary barrier to software-only gesture interaction.

Three.js [3] provides a high-level abstraction over the WebGL API, enabling hardware-accelerated 3D scene rendering within the browser without any native installation. The combination of MediaPipe and Three.js creates a uniquely accessible stack: all computation runs client-side, privacy is preserved, latency is minimised, and the result is deployable as a static web page.

This paper makes the following contributions: (i) a modular pipeline architecture integrating real-time hand tracking with interactive 3D rendering; (ii) lightweight gesture recognition algorithms tuned for browser-side execution; (iii) empirical performance evaluation across hardware configurations and lighting conditions; and (iv) a documented, extensible codebase. The remainder of the paper is structured as follows. Section II surveys related work. Section III describes the system architecture. Section IV details the gesture recognition algorithms. Section V presents experimental results. Section VI discusses findings, and Section VII concludes.

II. RELATED WORK

Gesture recognition has been studied extensively across hardware-based, vision-based, and learning-based paradigms. Early systems relied on data gloves and infrared depth cameras that, while accurate, imposed prohibitive hardware

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

costs and constrained deployment environments. Microsoft's Kinect and Intel RealSense popularised depth-based tracking but required dedicated peripherals and native drivers, precluding browser deployment.

A. Vision-Based Approaches

Vision-based systems process colour camera frames using skin-colour segmentation, edge detection, contour analysis, and background subtraction [4]. These methods are cost-effective but sensitive to illumination and cluttered backgrounds. Optical flow techniques track motion vectors between frames for trajectory-based gesture classification but impose significant computational cost.

B. Machine Learning Methods

Deep convolutional neural networks (CNNs) learned rich spatial features from raw image data, achieving human-competitive accuracy on gesture benchmarks [5]. Hybrid CNN-LSTM architectures extended recognition to dynamic gesture trajectories. Zhang et al. [6] surveyed vision-based recognition noting that accuracy and real-time performance remained in tension for software-only solutions.

MediaPipe Hands [2] resolved this tension through a two-stage pipeline: a palm detector localising hand bounding boxes in full-resolution frames, followed by a landmark regressor returning 21 three-dimensional keypoints from cropped regions. Compiled to WebAssembly, the pipeline achieves real-time inference on mobile CPUs, making it the enabling technology for the present system.

C. Web-Based 3D Interaction

Three.js [3] abstracts the WebGL API, providing scene graphs, camera primitives, and lighting models that enable sophisticated 3D applications in any WebGL-capable browser. Wachs et al. [7] demonstrated the breadth of gesture-controlled interfaces across domains from surgery to gaming; browser deployment further broadens accessibility by eliminating installation barriers.

D. Big Data and Scalable Architectures

The architectural principles underlying scalable big data systems inform the design of the present work. Acharjya and Ahmed [1] classify big data challenges into data storage and analysis, knowledge discovery, scalability, and information security. The present system addresses analogous concerns at interaction scale: the gesture pipeline must process high-velocity landmark streams with low latency, maintaining throughput across a range of hardware configurations. The modular, pipeline architecture of tools such as Apache Spark [1]—separating ingestion, processing, and output stages—directly inspired the layered system design described in Section III.

III. SYSTEM ARCHITECTURE

The Three.js Hand Recognition Panel follows a four-stage pipeline modelled on the streaming data architectures employed in big data analytics [1]. Each stage has a well-defined input–output contract, enabling independent optimisation and replacement.

A. Input Acquisition Layer

The browser's MediaDevices.getUserMedia API acquires a continuous 640×480 video stream from the device camera. Frames are rendered into a hidden HTML video element that serves as the MediaPipe input source. The requestAnimationFrame scheduling API drives the processing loop, synchronising frame acquisition with the browser's rendering cycle and capping throughput at the display refresh rate.

B. Hand Tracking Layer

MediaPipe Hands processes each video frame asynchronously via WebAssembly-compiled inference. Configuration parameters include maxNumHands = 2, modelComplexity = 1, minDetectionConfidence = 0.7, and minTrackingConfidence = 0.5, values tuned empirically to balance accuracy against processing speed. The onResults callback returns an array of NormalizedLandmark objects, each carrying normalised (x, y, z) coordinates for all 21 anatomical keypoints per detected hand.

C. Gesture Recognition Layer

Normalised landmark coordinates are transformed to Three.js world space by scaling and axis-inversion operations. Exponential moving average smoothing (alpha = 0.6) suppresses per-frame jitter without introducing perceptible lag. Gesture recognition functions then evaluate geometric conditions on the smoothed landmarks as detailed in Section IV.

D. Rendering Layer

Three.js manages the scene graph, camera, and WebGL renderer. A pool of SphereGeometry meshes represents the 21 landmarks per hand; LineSegments primitives connect adjacent joints. Mesh positions are updated each frame from the landmark buffer before the renderer redraws the scene. Gesture-triggered actions—camera translation, object creation, scene reset—are applied to the scene graph prior to rendering.

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

Table I summarises the key architectural parameters of the pipeline.

TABLE I. *System Pipeline Parameters*

Parameter	Value
Input Resolution	640 × 480 px
Max Hands	2
Landmarks / hand	21 (3-D)
Detection Confidence	0.70
Tracking Confidence	0.50
Smoothing α	0.60
Rendering API	WebGL 2.0 / Three.js

IV. GESTURE RECOGNITION ALGORITHMS

Gesture recognition algorithms operate on the smoothed, world-space landmark buffer produced by the pipeline's third stage. Two computational primitives underpin all implemented gestures: Euclidean distance between landmark pairs and relative positional comparison between landmarks.

A. Pinch Gesture

The pinch gesture is identified when the Euclidean distance d between the thumb tip (landmark 4) and index fingertip (landmark 8) satisfies $d < \theta\_pinch$, where $\theta\_pinch = 0.05$ world units, calibrated against the expected hand size in the detection field of view. A hysteresis band ($\theta\_release = 0.08$) prevents rapid toggling when the distance hovers near the threshold. State transitions emit pinch-start, pinch-continue (carrying a delta-position vector), and pinch-end events consumed by the rendering layer for camera translation.

B. Rotation Gesture

Rotation is detected by tracking the instantaneous angle φ of the vector from the wrist (landmark 0) to the middle-finger tip (landmark 12) in the scene plane. An angular velocity $|\Delta\varphi / \Delta t|$ exceeding $45°\cdot s^{-1}$, sustained for at least three consecutive frames, triggers a rotation event. The signed angular velocity is passed to the camera orbit controller, rotating the scene about its origin.

C. Multi-Hand Interaction

When two hands are simultaneously detected, the inter-wrist distance is computed. A rapid reduction in this distance below a scene-scale threshold triggers an object-spawn event; a rapid increase triggers a scene-reset event. State debouncing with a 300 ms cooldown prevents repeated triggering from a single physical gesture.

D. Algorithmic Complexity

All recognition functions execute in O(1) time per frame with respect to landmark count, as they evaluate fixed-landmark-index expressions. The dominant computational cost is the MediaPipe inference, which is offloaded to WebAssembly and executes asynchronously, leaving the JavaScript event loop free for rendering updates. This architecture mirrors the separation of ingestion and processing stages in stream analytics platforms such as Apache Storm [1].

V. EXPERIMENTAL EVALUATION

A. Setup

Testing was conducted on two systems: System A (Intel Core i7-11th Gen, NVIDIA GTX 1660, 16 GB RAM, Chrome 120) and System B (Intel Core i5-10th Gen, Intel Iris Xe, 8 GB RAM, Chrome 120). Three lighting conditions were evaluated: good lighting (>300 lux), moderate (100–300 lux), and poor (<100 lux). Each test session comprised 60 seconds of continuous operation; gesture accuracy was measured over 100 intentional performances per gesture type.

B. Frame Rate

Table II reports average frame rates. System A sustained 55–60 FPS under good lighting, degrading to 50–55 FPS under poor lighting as the MediaPipe model reinitiates palm detection more frequently. System B maintained 30–36 FPS, above the 25 FPS threshold generally considered acceptable for real-time interaction.

TABLE II. *Average Frame Rate (FPS)*

System	Good Lighting	Poor Lighting
System A	57.4 FPS	51.8 FPS
System B	33.6 FPS	27.2 FPS

C. Detection and Gesture Accuracy

Table III presents hand detection accuracy and gesture recognition accuracy across conditions. Detection accuracy peaks at 95% under good lighting. Gesture recognition accuracy ranges from 88% (rotation) to 92% (pinch) under optimal conditions, consistent with prior benchmarks for webcam-based systems [6]. Multi-hand accuracy (85%) is

Received: 20-04-2026 | Accepted: 28-05-2026 | Published: 04-06-2026

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

lower due to increased occlusion when both hands are in frame.

TABLE III. *Accuracy Results*

Metric	Good	Poor
Hand Detection	95%	65%
Pinch Gesture	92%	78%
Rotation Gesture	88%	73%
Multi-Hand Gesture	85%	69%

D. Response Latency

End-to-end latency from gesture completion to visible scene update was measured using a 120 FPS reference camera. The mean latency was 68 ms (min 33 ms, max 112 ms, 95th percentile 98 ms). All measurements fall below the 100–150 ms perceptual immediacy threshold established in the HCI literature [7], confirming the system as genuinely real-time.

E. Comparative Analysis

Table IV compares the proposed system against traditional hardware-based gesture systems across key dimensions. The proposed system sacrifices some accuracy under adverse lighting—an inherent cost of monocular RGB tracking versus depth sensing—but delivers substantial advantages in cost, accessibility, and deployment ease.

TABLE IV. *Comparison with Traditional Systems*

Feature	Traditional	Proposed
Hardware Cost	High	Low
Installation	Required	None
Real-Time FPS	30–60	27–60
Lighting Sensitivity	Low	Moderate
Cross-Platform	Limited	Full

VI. DISCUSSION

The experimental results confirm that the Three.js Hand Recognition Panel achieves real-time, browser-native gesture interaction on consumer hardware without specialised peripherals. The system's layered pipeline architecture—separating acquisition, inference, recognition, and rendering—mirrors the modular design philosophy of scalable big data platforms [1], yielding a system that is independently optimisable at each stage.

The primary limitation is the sensitivity of MediaPipe's palm detection stage to adverse lighting. Accuracy drops from 95% under good lighting to 65% in poor conditions. Image pre-processing techniques such as adaptive histogram equalisation or neural network-based illumination normalisation represent promising mitigations. Background complexity similarly challenges the palm detector; incorporating a lightweight background subtraction step could improve robustness.

The implemented gesture vocabulary is intentionally small to maintain recognition reliability. Scaling to a larger gesture set would benefit from a data-driven classifier trained on MediaPipe landmark sequences, analogous to the machine learning pipeline approach advocated for big data analytics [1]. An LSTM-based sequence classifier operating on sliding windows of landmark frames could recognise dynamic gestures while remaining computationally light enough for real-time browser execution.

The IoT perspective articulated by Acharjya and Ahmed [1] is relevant to future extensions: treating each webcam as an IoT sensor generating a continuous stream of landmark data, and aggregating streams from multiple users, would enable collaborative multi-user gesture environments. Cloud-side processing of aggregated streams could support richer analytics while maintaining low-latency local feedback through a split-processing architecture.

Security and privacy considerations, identified as a major big data challenge [1], apply to gesture systems that transmit video or landmark data to servers. The present system's entirely client-side architecture avoids this issue entirely: no video or biometric data leaves the user's device.

VII. CONCLUSION

This paper presented the Three.js Hand Recognition Panel, a real-time, browser-based gesture interaction system integrating MediaPipe Hands and the Three.js WebGL library. The system detects up to two hands simultaneously, extracts 21 three-dimensional landmarks per hand, recognises pinch, rotation, and multi-hand gestures through efficient geometric algorithms, and renders interactive 3D feedback at 27–60 FPS across

International Journal of
AI ELECTRICAL CIVIL AND MECHANICAL ENGINEERING
Peer Reviewed, Referred & Indexed Journal
ISSN: 3070-0434 www.zesterapublications.com Original Research Paper

consumer hardware configurations. End-to-end gesture response latency of 68 ms falls within the perceptual immediacy threshold, confirming genuine real-time interactivity.

The modular, pipeline architecture draws on scalable big data design principles [1], yielding a system that is extensible, maintainable, and adaptable to emerging inference models and rendering APIs. The zero-installation, cross-platform deployment model democratises access to advanced gesture interaction in a manner consistent with the accessibility objectives of open-source big data tools such as Apache Spark and Mahout [1].

Future work will address lighting robustness through pre-processing enhancements, expand the gesture vocabulary via sequence-based machine learning classifiers, explore multi-user collaborative environments leveraging IoT-style landmark stream aggregation, and investigate WebGPU-accelerated rendering for improved performance on next-generation browsers.

ACKNOWLEDGMENT

The author thanks the project supervisor for guidance and the Department of Computer Science and Engineering for providing the computational resources used in this work.

REFERENCES

[1] D. P. Acharjya and Kauser Ahmed P, “A Survey on Big Data Analytics: Challenges, Open Research Issues and Tools,” International Journal of Advanced Computer Science and Applications, vol. 7, no. 2, pp. 511–518, 2016.

[2] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C. L. Chang and M. Grundmann, “MediaPipe Hands: On-device Real-time Hand Tracking,” arXiv:2006.10214, 2020.

[3] R. Cabello, Three.js Documentation, 2023. [Online]. Available: https://threejs.org/docs/

[4] S. Mitra and T. Acharya, “Gesture recognition: A survey,” IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 37, no. 3, pp. 311–324, 2007.

[5] O. Y. Al-Jarrah, P. D. Yoo, S. Muhaidat, G. K. Karagiannidis and K. Taha, “Efficient machine learning for big data: A review,” Big Data Research, vol. 2, no. 3, pp. 87–93, 2015.

[6] Z. Zhang, “Hand gesture recognition based on vision: A survey,” IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 42, no. 6, pp. 987–1003, 2012.

[7] J. P. Wachs, M. Kolsch, H. Stern and Y. Edan, “Vision-based hand-gesture applications,” Communications of the ACM, vol. 54, no. 2, pp. 60–71, 2011.

[8] A. Gandomi and M. Haider, “Beyond the hype: Big data concepts, methods, and analytics,” International Journal of Information Management, vol. 35, no. 2, pp. 137–144, 2015.

[9] X. Jin, B. W. Wah, X. Cheng and Y. Wang, “Significance and challenges of big data research,” Big Data Research, vol. 2, no. 2, pp. 59–64, 2015.

[10] D. P. Acharjya, S. Dehuri and S. Sanyal, Computational Intelligence for Big Data Analysis, Springer International Publishing, Switzerland, ISBN 978-3-319-16597-4, 2015.

[11] Khronos Group, WebGL 2.0 Specification, 2023. [Online]. Available: https://registry.khronos.org/webgl/specs/latest/2.0/

[12] Mozilla Foundation, “MediaDevices.getUserMedia() — Web APIs,” MDN Web Docs, 2023.