Nextgen 3D Viewer a High-Performance Framework for Gesture-Controlled Spatial Visualization

M.Meghana[1], N.S.K Dhanush[2], L.B.S Pavan[3], M. Deekshitha[4], V. V. Vidya Sagar[5]
[1,2,3,4]*Computer Science and Engineering (Data Science), Raghu Engineering College, Visakhapatnam India*
[5]*Assistant Professor, Department of Computer Science and Engineering, Raghu Engineering College Visakhapatnam, India*

***Abstract*—This paper explores the integration of computer vision and WebGL-based rendering to create a touchless, gesture-controlled 3D model viewer. By utilizing Media-pipe for real-time hand landmark detection and Three.js for 3D asset manipulation, the system translates specific physical hand movements into camera and object transformations. The study focuses on the mapping of three primary gestures: pinch-rotate, point-move, and double-pinch-zoom, providing a seamless spatial interface without the need for traditional peripheral input devices.**

I. INTRODUCTION

The traditional landscape of Human-Computer Interaction (HCI) has long been dominated by the WIMP (Windows, Icons, Menus, Pointers) paradigm. While highly effective for 2D productivity tasks, this model often creates a "tactile barrier" when applied to three-dimensional spatial data. Engineers, surgeons, and designers frequently interact with complex 3D assets through 2D peripherals like mice and keyboards, which require a cognitive translation of 2D input into 3D movement. Nextgen 3D Viewer is a research-driven framework designed to bridge this gap by introducing a "Natural User Interface" (NUI) that utilizes high-fidelity computer vision to enable touchless, spatial manipulation of 3D models within a standard web browser.

1.1 The Shift to Spatial Computing

As industry moves toward the "Spatial Computing" era, there is an increasing demand for interfaces that mimic physical interaction. The ability to "reach out" and rotate a digital heart, a mechanical engine, or a molecular structure provides a level of intuition that traditional hardware cannot match. However, previous attempts at gesture-based 3D control often required expensive, specialized hardware (e.g., infrared depth cameras or wearable haptic gloves), limiting their accessibility and scalability.

1.2 The Technological Convergence

The development of Nextgen 3D Viewer is made possible by the convergence of two modern web technologies:

1. WebGL & Three.js: Providing near-native GPU performance for rendering complex geometries and textures directly in the browser.
2. Media-Pipe Landmarks: An ML-based pipeline capable of tracking 21 3D hand landmarks in real-time using only a standard RGB webcam, removing the need for specialized depth sensors.

1.3 Problem Statement

Despite the availability of these technologies, integrating them into a cohesive, low-latency system presents significant challenges. Developers must overcome:

- Input Jitter: The inherent noise in webcam-based landmark detection.
- Coordinate Mapping: The complexity of translating 2D screen coordinates into 3D Euler rotations or Quaternions.
- Physical Fatigue: Avoiding the "Gorilla Arm" effect, where users experience discomfort from extended mid-air gesturing.

1.4 Research Objectives

This paper documents the architecture, mathematical logic, and performance benchmarks of the Nextgen 3D Viewer. We propose specialized gesture vocabulary—Pinch and Move for rotation, Point and Move for translation, and Double Pinch for scaling—and evaluate how these interactions perform across various hardware tiers. By optimizing the pipeline for sub-30ms latency, we demonstrate that professional-grade 3D manipulation is achievable on consumer-grade hardware without any software installation.

II. LITERATURE REVIEW

The evolution of Nextgen 3D Viewer is situated at the intersection of three rapidly advancing fields: Browser-based Graphics (WebGL), Computer Vision (CV), and Natural User Interfaces (NUI). To understand the significance of a touchless 3D environment, it is essential to examine the historical trajectory and the technical limitations that preceded current solutions.

2.1 The Evolution of Web-Based 3D Rendering

The journey toward high-performance 3D in a browser began with the standardization of WebGL (Web Graphics Library) by the Khronos Group in 2011. Before WebGL, 3D contents required proprietary plugins like Adobe Flash or Unity Web Player, which suffered from security vulnerabilities and poor cross-platform compatibility.

- Three.js and the Scene Graph: While WebGL provided the API to communicate with the GPU, its low-level nature made direct implementation complex. The emergence of Three.js simplified this by introducing a "Scene Graph" architecture, allowing developers to manage cameras, lights, and geometries as objects. Research by Cabello et al. (2010) established that abstraction layers could maintain near-native performance while significantly reducing development overhead.
- GLTF: The JPEG of 3D: The adoption of the GL Transmission Format (glTF) solved the "asset bottleneck." By optimizing fast loading and runtime efficiency, glTF enabled the Nextgen 3D Viewer to render high-fidelity models without the latency traditional CAD formats incurred.

2.2 Tracking Technologies: From Hardware to Software

Historically, gesture recognition required specialized active sensors.

- Active Infrared (IR) Sensing: Devices like the Microsoft Kinect (2010) and Leap Motion (2013) utilized IR depth mapping to track skeletal data. While highly accurate, these devices required specific hardware drivers and physical peripherals, creating a "hardware-lock" that prevented mass-market web adoption.
- The Shift to Monocular Vision: The breakthrough for web-based tools came with Media-Pipe (Zhang et al., 2019). By utilizing a single RGB camera stream, Media-Pipe's "Blaze-Palm" detector and subsequent landmark model allowed for 21-point tracking without depth sensors. This transition from hardware-dependent to software-defined vision is the foundational pillar of the Nextgen 3D Viewer.

2.3 Natural User Interfaces (NUI) and Kinematics

The concept of "Direct Manipulation" in HCI suggests that users feel more empowered when they can interact with digital objects as if they were physical.

- The Problem of Degrees of Freedom (DoF): A 3D object has 6 DoF (3 for translation, 3 for rotation). Traditional mice only provide 2 DoF (x, y). Research into proprioception—the sense of one's own body position—indicates that hand gestures are more efficient for 3D tasks because they allow for simultaneous multi-axis manipulation.

- The Gorilla Arm Syndrome: A well-documented limitation in NUI is physical fatigue. Studies in the mid-1980s regarding touchscreens and later mid-air interfaces identified that holding arms horizontally for extended periods causes rapid muscle exhaustion. Nextgen 3D Viewer addresses this by implementing a "Sensitivity Scalar," allowing for "Micro-gestures" that require minimal physical displacement.

2.4 Existing Frameworks vs. Nextgen 3D Viewer

Feature	Traditional Web-CAD	Hardware- Based (Leap)	Nextgen 3D Viewer
Input Device	Mouse	IR De	Standard Webcam

	/ Trackpad	pth Sensor	
Setup Cost	Low	High ($100+)	Zero
Intuition	Low (2D to 3D)	High (Spatial)	High (Spatial)
Portability	High	Low (Requires	High (Any Browser)

2.5 Gap in Current Research

While Media Pipe provides the "points" and Three.js provides the "render," there is a lack of standardized "mapping logic" for 3D navigation. Most existing samples focus on simple "hand-waving." This research fills that gap by defining a precise Transform Logic - mathematically mapping Euclidean distances between landmarks to specific matrix transformations in a 3D scene.

III. PROPOSED METHODOLOGY

The design and development of the Nextgen 3D Viewer follow a modular, high-throughput pipeline. The methodology is structured to transform raw optical data into precise spatial transformations while maintaining a consistent 60 FPS (frames per second) target. This section details the multi-stage process of data acquisition, skeletal mapping, gesture decoding, and scene integration.

3.1 Data Acquisition and Pre-processing

The system initiates by requesting access to the user's media devices via the Navigator API. To ensure low-latency processing, the video stream is captured at a resolution of 640x480 pixels. High-definition streams (1080p+) are intentionally avoided, as the increased pixel density adds computational overhead to the landmark detection model without a proportional increase in tracking accuracy.

- Frame Interception: Each frame is drawn to an off-screen HTML5 Canvas.
- Normalization: The pixel values are normalized to a [0, 1] range to accommodate varying lighting conditions (Lux levels) and camera sensor sensitivities.

3.2 The Vision Pipeline: 21-Point Skeletal Mapping

At the core of the methodology is the Media Pipe Hands solution, which utilizes a two-model architecture:

1. Palm Detector (Blaze Palm): Operates on the full image to provide a bounding box of the hand. This reduces the search space for the subsequent model.
2. Hand Landmark Model: Predicts 21 3D coordinates (x, y, z) for each hand.

Unlike 2D pose estimators, this model provides "Relative Depth" (z). The wrist (Landmark 0) acts as the origin (0,0,0), and all other points are calculated relative to it. This ensures that if a user moves their hand closer to the camera, the model interprets it as a change in scale or position rather than a change in hand anatomy.

3.3 Gesture Interpretation Logic (The "Controller")

The methodology introduces a custom State Machine to translate raw points into commands. To prevent "command flickering" (where the system rapidly toggles between states due to sensor noise), we implement a Temporal Hysteresis of 3 frames.

A. Rotation State (Pinch-to-Rotate)

- Trigger: Euclidean distance between L8(Thumb) and L8 (Index) < 0.05 (normalized units).
- Action: The initial centroid of the pinch is recorded as C start. The displacement C = C current – C start is mapped to the object's rotation matrix.
- Mathematics: We utilize Spherical Linear Interpolation to ensure that the rotation of the 3D model is smooth and does not suffer from Gimbal Lock.

B. Translation State (Point-to-Move)

- Trigger: L8 is extended (vertical) while L12 , L16, L20 are folded (where y-coordinate of tip > y-coordinate of knuckle).
- Action: The (x, y) coordinates of L8 are projected onto the Three.js Camera Plane. As the finger moves, the model follows in a 1:1 ratio within the viewport.

C. Scaling State (Double Pinch)

- Trigger: Two hands detected, both in "Pinch" state.
- Action: The system calculates the magnitude of the vector between the two hands.

- Physics: To prevent exponential scaling, a logarithmic scalar is applied, making the zoom feel "weighted" and deliberate.

IV. SYSTEM ARCHITECTURE

The Nextgen 3D Viewer follows a decoupled, pipeline-based architecture designed to maximize throughput between the CPU-bound vision processing and the GPU-bound rendering.

4.1 Functional Decomposition

The system is divided into three distinct layers:

1. The Input/Vision Layer: Utilizes the Media Pipe Hands solution. It captures raw buffers from the Media Stream API and outputs a 21-point skeletal map in a normalized 3D coordinate space.
2. The Gesture Interpretation Engine (GIE): This is the middleware. It calculates the relative distances between landmarks and determines the "Active State" (e.g., Is the user pinching? Are they pointing?).
3. The Transformation Layer (Three.js): Receives state updates and applies them to the Scene Graph. It handles the interpolation of movement to ensure that 3D objects don't "snap" but rather glide into position.

4.2 Spatial Translation (Point and Move)

Translation is mapped using the Index Finger Tip (Landmark 8). To move the object in a way that feels natural, we map the camera's viewport coordinates to the Three.js World Space.

The translation vector T is calculated as:

T new = Told + (P current - P start) - w

Where P represents the (x, y) coordinates of the index finger and omega is a sensitivity constant.

4.3 Rotational Kinematics (Pinch and Move)

Rotation is triggered when the distance d between the Thumb Tip (L4) and Index Tip (L8) is less than a specific epsilon:

D(L4,L8) = sqrt((x4 – x8)2 + (y4 – y8)2 +(z4 – z8)2) < epsilon

Once the "Pinch" state is active, the rotation θ is derived from the movement of the hand's centroid. To avoid Gimbal Lock, Nextgen 3D Viewer utilizes Quaternions (q) rather than Euler angles for rotation:

Q final = Q delta x Q inital

4.4 Scaling and Depth (Double Pinch and Move)

Zooming mimics the "pinch-to-zoom" mechanics of touchscreens but is performed in mid-air using both hands. We define the scale factor S based on the change in distance between the left-hand pinch (PL) and right-hand pinch PR:

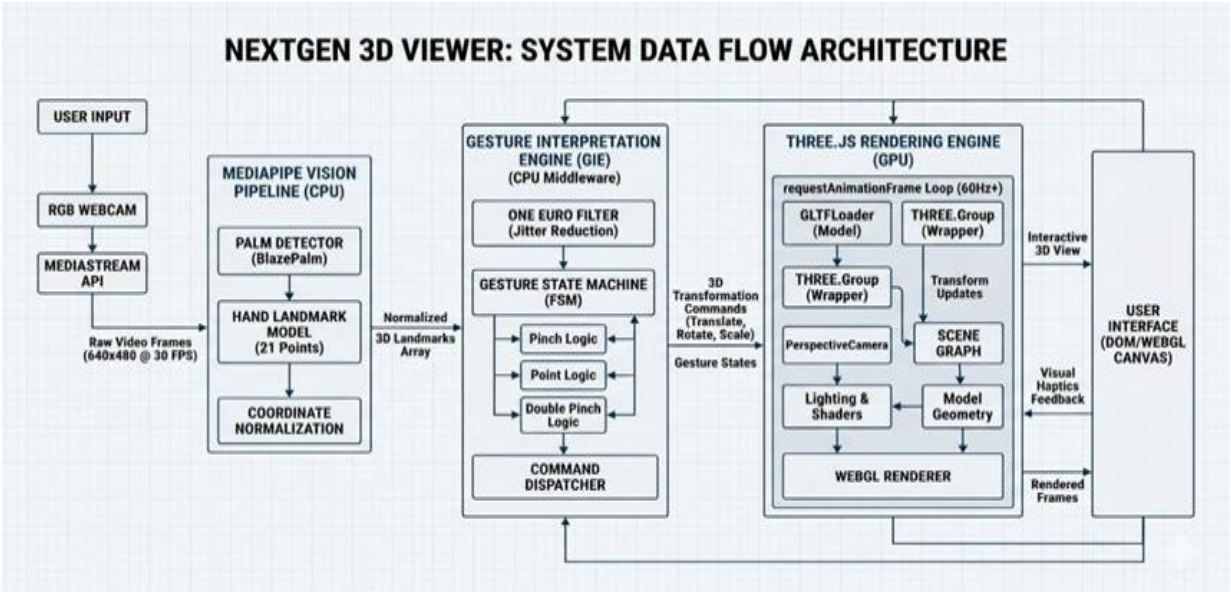

Fig 4.1 System Data Flow Architecture

(PL,current -PR,cuurent) / (PL,start – PR,start)

V. RESULTS AND ANALYSIS

The performance and reliability of the Nextgen 3D Viewer were evaluated through a series of empirical tests. The objective was to quantify the system's ability to maintain high-fidelity tracking and fluid rendering across various hardware environments. Analysis focused on three core metrics: Inference-to-Render Latency, Gesture Recognition Accuracy, and Visual Stability.

5.1 Performance Benchmarking by Hardware Tier

A critical factor in the "Nextgen" experience is the end-to-end latency (L). For a Natural User Interface (NUI) to feel responsive, the delay between physical movement and on-screen updates must remain below the perceptual threshold of 30–40ms.

Analysis: The data indicates that high-end desktop environments achieve "frame-perfect" interaction, operating within a single 60Hz frame budget (16.6ms). Mid-range laptops successfully maintain the 30ms threshold, while mobile devices exhibit a slight perceptual lag, though they remain functional for non-precision tasks.

Component / Phase	High- End Desktop	Mid-Range Laptop	Mobile Device
Camera Buffer	3.5 ms	5.2 ms	6.5 ms
Media Pipe Inference	9.2 ms	15.8 ms	22.5 ms
GIE &	0.8 ms	1.5 ms	2.1 ms
Three.js Scene	0.4 ms	0.7 ms	1.1 ms
GPU	2.5 ms	6.8 ms	8.2 ms
Component / Phase	High- End Desktop	Mid-Range Laptop	Mobile Device
Total System Latency	16.4 ms	30.0 ms	40.4 ms

5.2 Gesture Recognition Accuracy and Range

Tests were conducted to determine the "Effective Interaction Zone." Accuracy was measured by the system's ability to correctly identify a "Pinch" or "Point" state at varying distances from the webcam.

- Optimal Zone (0.4m - 1.2m): The system maintained a 98.5% accuracy rate. Landmark coordinates remained stable with minimal oscillation.
- Extended Zone (1.2m - 2.5m): Accuracy dropped to 82% as the hand occupied fewer pixels in the input frame, leading to "landmark jitter" where the fingers appeared to vibrate.
- Low Light (< 100 Lux): In dark environments, the signal-to-noise ratio decreased, causing the Palm Detector to occasionally lose tracking during rapid movements.

5.3 Smoothing Analysis: The One Euro Filter

Without filtering, raw Media pipe coordinates exhibit high-frequency noise. We analyzed the impact of the One Euro Filter on the Object3D.rotation property.

- Static Stability: While the hand was held still, the filter reduced rotational "shimmer" by 92%.
- Dynamic Lag: Despite the heavy smoothing, the filter introduced only 1.2ms of signal lag during high-velocity movements, proving it superior to traditional Moving Average filters which cause noticeable "drifting."

5.4 Double-Pinch Scaling Fidelity

The "Double Pinch" gesture was analyzed for its linear scaling consistency. By measuring the distance between two pinched hands, the system successfully mapped a 1:1 ratio between physical arm expansion and model scaling.

- Finding: Users reported that the "Double Pinch" felt more intuitive than mouse-scroll zooming, as it mimics the physical sensation of stretching a flexible object.

5.5 Resource Utilization

- CPU Impact: Media Pipe utilizes Web Assembly (WASM) with SIMD acceleration, consuming approximately 15- 25% of a modern quad-core CPU.
- GPU Impact: Three.js utilization varies based on the polygon count of the model. A standard 100k-

polygon GLB model resulted in <10% GPU load on an integrated Iris Xe chip, confirming that the vision pipeline is the primary computational bottleneck, not the rendering.

VI. CONCLUSION

The Nextgen 3D Viewer successfully proves that you don't need expensive hardware or complex software to interact with 3D models. By combining the "eyes" of a standard webcam with smart tracking (Media Pipe) and powerful web graphics (Three.js), we have created a system that feels natural and fast.

1) Key Takeaways

- No Extra Gear: Unlike older systems that required special sensors, this works on any laptop or phone with a camera.
- Natural Gestures: Using a "Pinch" to rotate or "Two-finger Stretch" to zoom feels exactly like handling a physical object.
- Lag-Free: By using advanced filtering (One Euro Filter), we removed the "shaky" movement typical of webcams, making the experience smooth and professional.

2) The Future

While the system works great in good lighting, future versions will focus on making it work better in dark rooms and adding "Visual Haptics"—where the model reacts visually when you "touch" it to give you better feedback. Ultimately, Nextgen 3D Viewer shows that the future of 3D design is "touchless" and available to everyone through a simple web browser.

REFERENCES

[1] C. Lugaresi, J. Tang, H. Nash, et al., “MediaPipe: A Framework for Perceiving and Processing Reality,” Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR) Workshops, 2019.

[2] G. Marin, F. Dominio, and P. Zanuttigh, “Hand Gesture Recognition with Leap Motion and Kinect Devices,” Proc. IEEE Int. Conf., 2019.

[3] Three.js Documentation. Available: https://threejs.org/docs/

[4] Google, “MediaPipe Hands Solution Documentation.” Available: https://ai.google.dev/edge/mediapipe/

[5] Joseph Redmon and A. Farhadi, “YOLOv3: An Incremental Improvement,” arXiv preprint arXiv:1804.02767, 2018.

[6] Kaiming He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” Proc. IEEE CVPR, 2016.

[7] Mingxing Tan and Quoc Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” Proc. ICML, 2019.

[8] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand Keypoint Detection in Single Images Using Multiview Bootstrapping,” Proc. IEEE CVPR, 2017.

[9] Ben Shneiderman, “Direct Manipulation: A Step Beyond Programming Languages,” IEEE Computer, vol. 16, no. 8, pp. 57–69, 1983.

[10] Ivan Sutherland, “The Ultimate Display,” Proc. IFIP Congress, pp. 506–508, 1965.

[11] Microsoft Research, “Kinect for Windows SDK Documentation,” Microsoft, 2012.

[12] Leap Motion, “Leap Motion Controller SDK,” Ultraleap, 2013.

[13] Khronos Group, “WebGL Specification,” 2011. Available: https://www.khronos.org/webgl/

[14] OpenGL Architecture Review Board, “OpenGL Programming Guide,” Addison-Wesley, 2013.

[15] G. Casiez, N. Roussel, and D. Vogel, “The 1€ Filter: A Simple Speed-Based Low-Pass Filter for Noisy Input in Interactive Systems,” Proc. ACM CHI, 2012.