 ISSN: **3107-6513**

AI DRIVEN DIGITAL TWIN

Mr. Jayesh Dhuri.
Professor, Mechatronics Dept.
New Horizon Institute of Technology and Management, Thane, India - 400615
jayeshdhuri@nhitm.ac.in

Blessy Varshith
Student, Mechatronics Dept.
New Horizon Institute of Technology and Management Thane, India - 400615
blessyvarshith1105@gmail.com

Shubham Panigrahi
Student, Mechatronics Dept.
New Horizon Institute of Technology and Management Thane, India - 400615
shubhampanigrahi010@gmail.com

Dr. Sunil Bobade
Professor, Mechatronics Dept.
New Horizon Institute of Technology and Management, Thane, India - 400615

Mr. C.M Zode
Professor, Mechatronics Dept.
New Horizon Institute of Technology and Management, Thane, India - 400615

Rajas Naik
Student, Mechatronics Dept.
New Horizon Institute of Technology and Management, Thane, India - 400615

Om Pargaonkar
Student, Mechatronics Dept.
New Horizon Institute of Technology and Management Thane, India - 400615

Abstract*—The rapid evolution of modern industrial systems has highlighted the need for dynamic, real-time cyber-physical synchronisation. Traditional digital twins often lack the predictive capabilities required to autonomously adapt to complex, ever-changing environments. This paper presents an artificial intelligence-driven digital twin framework designed to enhance predictive maintenance and overall operational efficiency. By integrating machine learning algorithms with real-time sensor data, the proposed model transitions the digital twin from a passive monitoring tool into an active, forecasting system. We detail the architecture of this intelligent twin, which continuously learns from both historical and streaming data to anticipate potential system anomalies and optimise performance parameters. Experimental simulations indicate that the application of artificial intelligence significantly improves the accuracy of failure detection and reduces unexpected downtime compared to conventional monitoring methods. Ultimately, this research provides a robust, scalable foundation for deploying intelligent digital twins, offering a practical pathway toward fully autonomous and resilient operations.

Keywords- Artificial intelligence, digital twin, machine learning, predictive maintenance

I. INTRODUCTION

The concept of a "digital twin"—a virtual representation of a physical asset, system, or process—has steadily gained traction across industries ranging from manufacturing to urban planning. Initially conceived as static, three-dimensional models or basic simulations, digital twins have evolved alongside advancements in the Internet of Things (IoT). Today, they serve as crucial bridges between the physical and digital realms, enabling operators to monitor system health and performance remotely. However, as modern cyber-physical systems grow increasingly complex, the sheer volume and velocity of generated sensor data have outpaced the analytical capacities of traditional, rules-based monitoring.

This paper proposes an Artificial Intelligence (AI) driven digital twin framework designed to address these challenges. By embedding machine learning algorithms directly into the digital twin architecture, the system can continuously ingest streaming data, identify hidden patterns, and dynamically update its internal models. This integration empowers the twin to predict potential anomalies before they occur, optimise operational parameters autonomously, and facilitate highly accurate predictive maintenance schedules.

II. RELATED WORK

The development of digital twins (DT) has transitioned from simple 3D visualisations to complex, data-driven ecosystems. Current research in this field generally bifurcates into two categories: high-fidelity virtual representation and intelligent data analysis.

A. Unity as a Digital Twin Platform

Recent studies have identified game engines, specifically Unity 3D, as premier tools for industrial DT development due to their robust real-time rendering and cross-platform compatibility. Researchers have successfully utilised Unity to create immersive environments for robotic arm simulations, noting that Unity outperforms traditional simulation platforms like Gazebo in terms of latency and visual fidelity Furthermore, the integration of Unity with geospatial data through various Software Development Kits (SDKs) has enabled the creation of large-scale digital twins for smart cities and maritime ports, allowing for real-time situational awareness and the simulation of complex "what-if" scenarios.

B. AI Integration for Predictive Maintenance

The shift from "Digital Shadows"—which only reflect the physical state—to "Digital Twins" requires a bidirectional data flow and predictive intelligence. Literature highlights that the integration of Artificial Intelligence, specifically deep

learning and federated learning, allows DTs to handle the high-velocity data generated by Internet of Things (IoT) sensors [2]. For instance, recent frameworks have combined Unity-based visualisation with prediction algorithms to detect failures in wind turbine components, effectively moving maintenance strategies from reactive to proactive [4].

C. Emerging Trends in Industry 4.0 and 5.0

In the context of Industry 5.0, the focus has shifted toward human-centred manufacturing and sustainable operations. These systems use synthetic data generated within virtual environments to train AI models, which are then deployed to the physical factory floor for real-time optimisation [2].

III. METHODOLOGY

This section details the software architecture and the specific technical configurations used to develop the AI-driven digital twin interface. The system is engineered to prioritise low-latency performance and accessibility, ensuring it remains operational on standard consumer-grade hardware.

PART 1 - HAND GESTURE AI

A. Software Stack and Environment

The core logic of the system is implemented in **Python**, chosen for its extensive ecosystem of computer vision and mathematical libraries. To facilitate real-time interaction between the physical user and the virtual twin, the following libraries are utilised:

1. **OpenCV:** Provides the foundational framework for video stream acquisition and image processing, handling the real-time frames from the webcam.
2. **MediaPipe:** Utilised for its high-fidelity ML-based hand tracking solution, allowing for the extraction of 21 3D hand landmarks without the need for high-end GPU acceleration.
3. **NumPy and Math:** These libraries manage the heavy numerical computations required for coordinate transformations and spatial analysis.

B. Multi-Hand Detection and Skeletal Rendering

Unlike basic gesture recognisers, this system implements a robust **Multi-hand detection** module. MediaPipe's BlazePalm model is used for initial palm detection, followed by a hand-landmark model that renders a **real-time skeletal map**. This skeletal overlay provides 21 distinct coordinate points per hand, which are mapped into the Unity environment to synchronise the movements of the user with the digital twin's control interface.

C. Spatial Computation: Tilt, Direction, and Velocity

To translate physical hand movements into actionable data for the digital twin, the system performs three primary mathematical evaluations:

- **Tilt Angle Computation:** By calculating the arc-tangent of the displacement between specific landmarks (e.g.,the wrist and the middle-finger metacarpal),the system determines the hand's orientation.
- **Directional Tracking:** The system monitors the movement of the hand centroid across successive frames to determine the vector of motion.
- **Velocity Estimation:** By measuring the Euclidean distance traveled by the hand landmarks over a fixed time interval (Δt), the system computes the velocity of the gesture, allowing for "speed-sensitive" commands within the simulation.

D. Gesture Classification and Performance

The system features a **Gesture classification display** that provides immediate visual feedback to the user. Complex hand configurations are mapped to specific control triggers—such as "Initiate Maintenance Mode" or "Rotate View." Because the implementation leverages optimised pipelines, it achieves high frame rates on a standard laptop webcam, eliminating the requirement for expensive external depth sensors or GPU-heavy processing.

Hand Landmark Detection

The foundation of gesture recognition is hand landmark detection.
The system uses MediaPipe Hands model which detects:

- 21 2D key-points per hand
- X, Y (and optionally Z) coordinates
- Landmark connectivity structure

The 21 Landmarks Include:

2–4. Thumb joints
5–8. Index finger joints
9–12. Middle finger joints
13–16. Ring finger joints
17–20. Little finger joints
Each frame from the webcam is processed and the coordinates of these landmarks are extracted.

1. Preprocessing Stage
Before gesture classification, the following preprocessing steps are performed:
A. Convert BGR frame to RGB
B. Normalize landmark coordinates
C. Store previous frame coordinates for motion tracking
D. Remove noise using smoothing (if required)
This ensures stable and accurate gesture recognition.

2. Feature Extraction
Gesture recognition depends on extracting meaningful features from landmarks.
The system calculates:

3. Finger Extension State
Each finger is classified as:
• Extended
• Folded
Example logic for index finger:
If:
Index_tip_Y < Index_PIP_Y
→ Finger is extended
Else
→ Finger is folded
2. Euclidean Distance
Distance between two landmarks:
Distance = $\sqrt{((x2 - x1)^2 + (y2 - y1)^2)}$

Used for:
- Thumb-index pinch detection
- Finger spread measurement
- Palm width estimation

4. Angle Calculation
Angle between three joints is calculated using vector dot product.
If A, B, C are three points:
Vector BA = A - B
Vector BC = C - B
Angle $\theta = \cos^{-1}((BA \cdot BC) / (|BA||BC|))$
Used to:
- Determine finger bending
- Compute tilt angle
- Measure palm orientation

5. Gesture Classification Logic

The system uses rule-based classification.

Each finger is assigned a binary value:

1 → Extended
0 → Folded

Example combinations:

Thumb, Index, Middle, Ring, Pinky

6. Orientation Detection
Hand orientation is determined by:
- Wrist to middle finger vector
- Palm normal approximation

If the vertical displacement is dominant:
→ Orientation: Upright
If horizontal displacement is dominant:
→ Orientation: Tilted

7. Multi-Hand Recognition
The system:
- Detects both left and right hands
- Classifies gestures independently
- Maintains separate landmark tracking
- Labels hands as "Left" or “Right"

8. Real-Time Digital Twin Synchronisation
Once gesture is classified:
1. Landmark skeleton is drawn
2. Gesture name is displayed
3. Orientation and tilt angle are shown
4. Velocity and direction are updated
5. Digital twin representation mirrors the hand

All operations occur within milliseconds per frame

PART 2 - AERODYNAMICS

The simulation was developed within the Unity engine using a component-based architecture. Air particles are represented as individual Objects, each governed by a dedicated AirParticle MonoBehaviour that tracks physical properties — velocity, momentum, collision events, and energy dissipation — across the simulation lifetime.

Particle generation is handled by a Spawner component that emits molecules into a configurable three-dimensional volume at a fixed interval. Two concurrent coroutines run in parallel to sustain a steady flow up to a defined population ceiling. Each molecule is placed at a randomised position within the spawn area and scaled by a uniform size parameter. Particles exceeding a maximum distance from the spawner origin are automatically destroyed to bound memory usage.

Air Particle UI Controller that polls all active particles at a fixed update interval. On each cycle it accumulates speed, collision count, speed drop rate, momentum loss, energy dissipation, directional deviation, and lift forces across the population. Derived statistics — mean speeds for collided and non-collided subsets, collision rate, total energy dissipated, and average direction change — are computed and displayed via a TextMeshPro overlay panel, which can be toggled, repositioned, and scaled at runtime through keyboard input. The simulation tracked six key parameters: collision rate (percentage of particles that struck a surface), speed (mean velocity for collided and non-collided subsets, plus peak speed drop), momentum drop (average impulse lost per particle), energy dissipation (total energy absorbed across all collisions), directional change (average angular deviation from original trajectory), and lift/anti-lift forces (mean peak upward and downward forces among collided particles). Together these characterise the kinematic and energetic effects of particle-surface interactions.

PART 3 - STRESS

A.Dynamic Stress

The simulation evaluated structural response by recording maximum stress (MPa), maximum deformation (m), and factor of safety across each load case. The factor of safety acted as the primary design quality indicator, classified as good (>2.5), acceptable (>1.5), or critical (<1.5) Stress and strain were computed at the point of collision using material properties defined in the MeshDeformer component — Young's modulus, yield strength, and cross-sectional area. Permanent deformation was applied when stress exceeded the yield threshold, and failure was flagged when strain surpassed the deformation limit. Vertex-level displacements were rendered as a blue-to-red heat map to spatially locate regions of peak stress. Both individual load cases and a combined multi-force scenario were assessed, allowing comparison of isolated versus cumulative structural effects.

B.Static Stress

For the static load case, the mesh was held fixed in place with rigid body motion disabled, ensuring all applied force produced structural deformation rather than object displacement. A single force vector was applied at a defined source point on the mesh, with vertex deformation computed based on force magnitude, material stiffness, proximity to the source, and normal alignment. Stress and strain were evaluated at each vertex using Hooke's law, and the factor of safety derived from the ratio of yield strength to peak stress. The resulting deformation was displayed as an animated transition from the original to the deformed state, with a colour heat map indicating stress distribution across the mesh surface.

 ISSN: **3107-6513**

PART 4 - IMAGE TO 3D MODEL

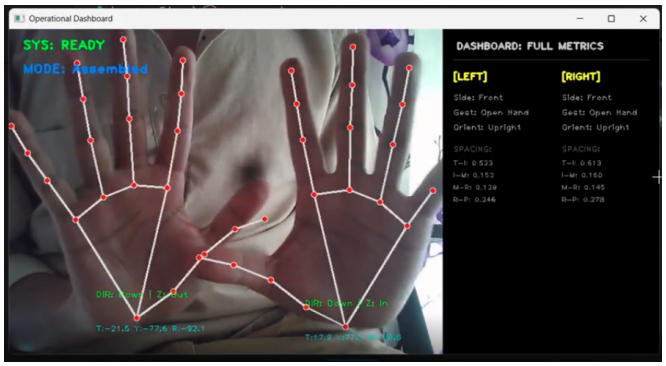
The image-to-3D model methodology using Python involves capturing one or more 2D images of an object, preprocessing them (resizing, noise removal, feature enhancement), and extracting key features using computer vision techniques such as edge detection and feature matching with libraries like Open CV. These features are then used to estimate depth and reconstruct the object's geometry through methods like Structure-from-Motion or deep learning models (e.g., neural radiance fields), often implemented with PyTorch or TensorFlow. The generated point cloud is converted into a mesh, refined with smoothing and texture mapping, and finally exported as a 3D model file (e.g., OBJ or STL) for visualisation or further processing.

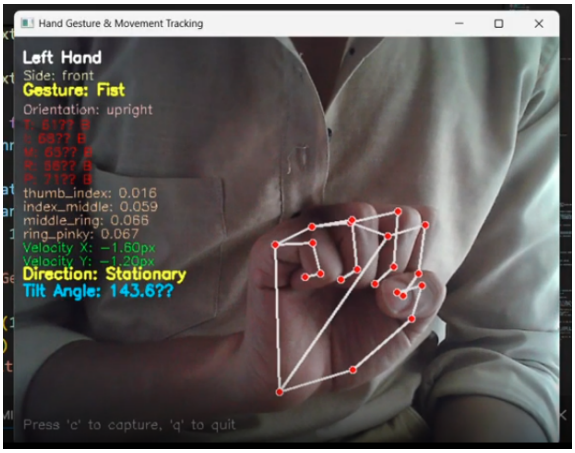
PART 5 - HEAT SIMULATION

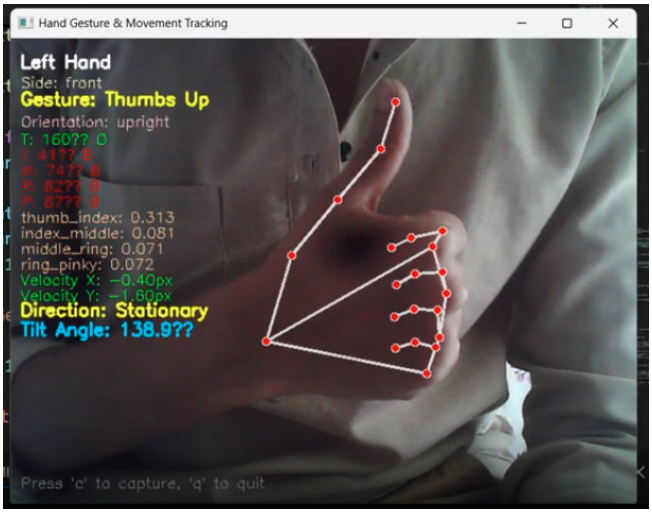
Heat simulation using Python is typically implemented by modelling heat transfer equations (such as the heat diffusion equation) and solving them numerically using methods like finite difference or finite element techniques. Libraries like NumPy and SciPy are used to compute temperature changes over a grid, while visualisation tools such as Matplotlib help display heat distribution over time.

IV. LIST OF FIGURES

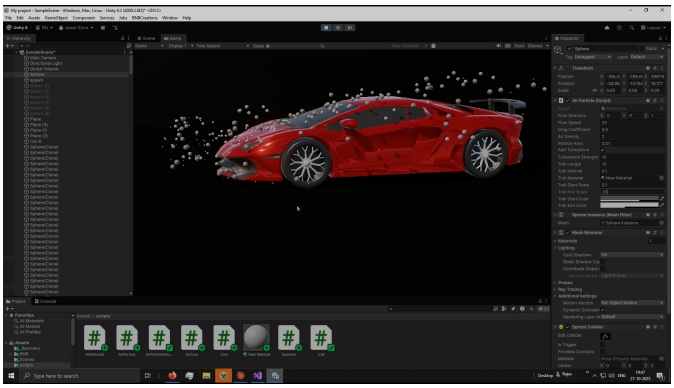
Hand Gestures:

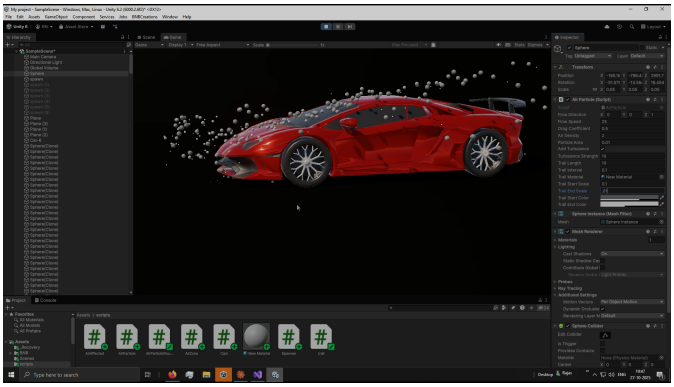






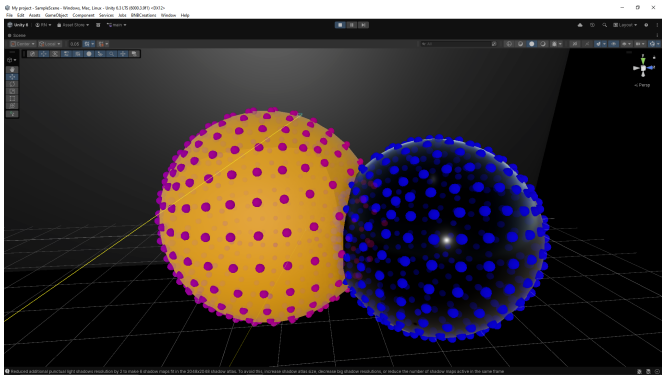
Aerodynamics:



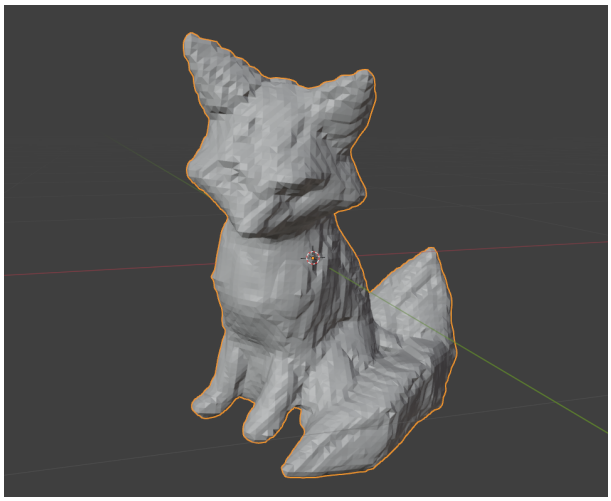


 ISSN: **3107-6513**

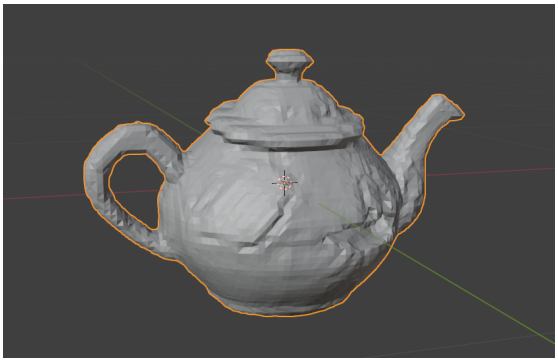
Stress:

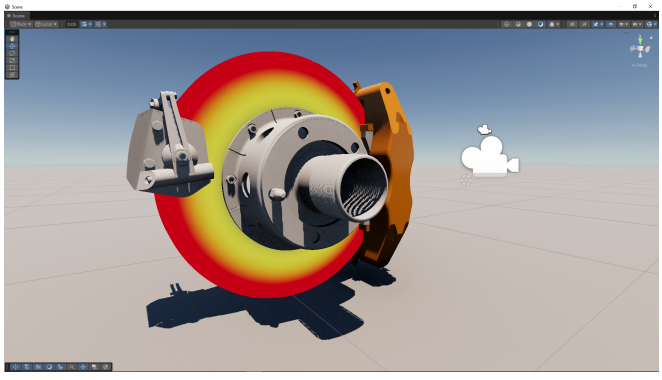


3D MODEL:



Heat Simulation:





ISSN: **3107-6513**

V. DISCUSSION

- This project shows how an AI-driven digital twin can act like a virtual copy of a real system, helping us understand its behaviour without physically testing it every time.

  One important observation is that real-time data plays a major role — the more accurate the data, the better the results from the digital twin.

- The use of AI makes the system smarter, as it can predict problems in advance instead of just reacting after they happen.
- During the development, it was clear that proper training of the model is necessary; otherwise, the predictions may not be reliable.
- The simulation environment helped in testing different conditions safely, which reduces risk and saves time in real-world operations.
- Another key point is that the system can improve efficiency by identifying performance issues and suggesting optimisations.
- However, the project also showed some challenges, such as handling large data, maintaining system accuracy, and computational requirements.
- Overall, the project proves that AI-driven digital twins are very useful in modern industries, especially for automation, monitoring, and predictive maintenance.

VI. CONCLUSION

This paper about AI-driven digital twin developed in this project successfully demonstrates the potential of integrating artificial intelligence with real-time system modelling. The approach enables accurate monitoring, analysis, and prediction of system behaviour, allowing better decision-making and improved operational efficiency. By creating a virtual representation of the physical system, it becomes possible to test scenarios, detect faults early, and reduce downtime without affecting actual operations. Although challenges such as data dependency, computational complexity, and model accuracy were observed, the overall results highlight the effectiveness of this technology. The project confirms that AI-driven digital twins can play a significant role in advancing smart manufacturing and Industry 4.0 applications, paving the way for more intelligent and autonomous systems in the future

REFERENCES

[1] A. Fuller, Z. Fan, C. Day, and C. Barlow, “Digital Twin: Enabling Technologies, Challenges and Open Research,” *IEEE Access*, vol. 7, pp. 108952–108971, 2019.

[2] F. Tao, H. Zhang, A. Liu, and A. Y. C. Nee, “Digital Twin in Industry: State-of-the-Art,” *IEEE Transactions on Industrial Informatics*, vol. 15, no. 4, pp. 2405–2415, 2019.

[3] X. Liu and I. David, “AI Simulation by Digital Twins: Systematic Survey and Reference Framework,” *arXiv preprint arXiv:2506.06580*, 2025.

[4] R. Zhou et al., “Digital Twin AI: Opportunities and Challenges from Large Language Models to World Models,” *arXiv preprint arXiv:2601.01321*, 2026.

[5] S. Chen et al., “AI-Enhanced Digital Twins in Predictive Maintenance: A Review,” *Elsevier*, 2025.

[6] D. Woo, “Open-Data-Driven Unity Digital Twin Pipeline,” *Applied Sciences*, MDPI, 2025.

[7] “Comparative Study of Digital Twin Developed in Unity and Gazebo,” *ResearchGate*, 2025.