Quaternion-Based Detection and Correction of Human Posture for A Health Monitoring System

Kalyca Nathania Benedicta Manullang – 13524071[1,2]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13524071@std.stei.itb.ac.id, [2]kalycamanullang@gmail.com

***Abstract*— Poor human posture during daily activities such as studying, working, and prolonged smartphone usage can lead to musculoskeletal discomfort and long-term health problems. This paper proposes a quaternion-based method for detecting and correcting human posture using three-dimensional orientation representation. Human body segments are modeled as vectors in $\mathbb{R}^3$, whose orientations are represented by unit quaternions to avoid singularities commonly found in Euler-angle-based methods. Postural deviation is detected through the computation of a relative quaternion between the measured posture and an ideal reference posture, providing both the axis and magnitude of deviation. A simulation-based experiment is conducted to evaluate posture deviation and orientation stability before and after correction. The results demonstrate that quaternion-based orientation analysis provides a stable, compact, and mathematically robust framework for posture detection and correction, making it suitable for simple and low-complexity health monitoring systems.**

***Keywords*—health monitoring system, human posture, orientation correction, quaternions**

I. INTRODUCTION

Poor posture during daily activities such as studying, working at a desk, and prolonged smartphone usage has become a common issue in modern society. Incorrect body orientation, particularly in the upper body and spine, can lead to musculoskeletal discomfort, reduced productivity, and long-term health problems such as chronic back pain and neck strain. As awareness of ergonomic health increases, there is a growing demand for simple and accessible systems that are capable of monitoring and correcting human posture in everyday environments. Human posture can be interpreted as the spatial orientation of body segments in three-dimensional space.

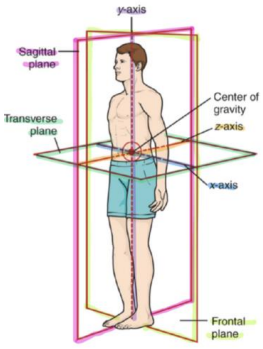


Fig. 1. Illustration of human upper-body posture represented as orientations of body segments in three-dimensional space (source: https://y-strap.com/the-y-strap-adjustment/)

Traditional posture assessment methods often rely on visual observation or simple angular measurements, which may be subjective and lack robustness. As orientation sensors and computational tools become more accessible, accurate mathematical models for three-dimensional rotation are increasingly important for reliable posture monitoring. Common representations such as Euler angles and rotation matrices have inherent limitations, including gimbal lock and computational inefficiency, which reduce their suitability for lightweight or low-cost health monitoring systems [6], [7]. However, many existing posture monitoring approaches focus primarily on detection and visualization, without providing a mathematically grounded mechanism for posture correction in three-dimensional space.

Quaternion algebra provides a mathematically robust framework for representing three-dimensional orientations. By representing rotations using four parameters, quaternions avoid the singularities associated with Euler angles and offer compact and numerically stable computations. In posture monitoring, body segment orientations can be modeled as unit quaternions in $\mathbb{R}^3$, allowing postural deviation to be detected through a relative quaternion with respect to an ideal reference posture. This relative quaternion encodes both the axis and magnitude of deviation, enabling direct mathematical correction through inverse rotation.

This paper proposes a quaternion-based method for detecting and correcting human posture within a simple health monitoring system. The main contributions of this work are: (1) modeling upper-body posture using unit quaternion orientation representation, (2) quantifying postural deviation using relative quaternion analysis, and (3) performing posture correction through inverse quaternion rotation. A simulation-based experiment is conducted to demonstrate how quaternion algebra can be applied to detect postural misalignment and compute corrective orientation adjustments. The proposed method aims to provide a stable, compact, and practical solution for posture monitoring, highlighting the relevance of quaternion algebra in everyday human-centered health applications.

II. Theoretical Foundation

A. Three-Dimensional Rotation and Orientation Representation

The orientation of a rigid body in three-dimensional space can be described as a rotation from a reference coordinate frame to a target coordinate frame. Mathematically, such a rotation belongs to the special orthogonal group SO(3). In practical applications, this rotation must be represented in a form that is both computationally efficient and numerically stable.

Common rotation representations include Euler angles and rotation matrices. Euler angles describe orientation using three sequential rotations about predefined axes, which makes them intuitive but prone to singularities such as gimbal lock. Rotation matrices represent orientation using a 3×3 orthogonal matrix with determinant equal to one. Although rotation matrices avoid gimbal lock, they require nine parameters with orthogonality constraints, resulting in higher computational cost and potential numerical drift. To address these limitations, quaternion algebra provides a compact and robust representation of three-dimensional rotations, making it suitable for continuous orientation analysis in posture monitoring systems.

B. Quaternion Algebra and Unit Quaternions

A quaternion is a four-dimensional hypercomplex number defined as

$$q = w + xi + yj + zk,$$

where $w \in \mathbb{R}$ is the scalar part and $(x, y, z) \in \mathbb{R}$ is the vector part [5]. The imaginary units i, j, k satisfy the relations

$$i^2 = j^2 = k^2 = ijk = -1.$$

For rotation representation, a unit quaternion is used, satisfying

$$\| q \| = \sqrt{w^2 + x^2 + y^2 + z^2} = 1.$$

Any rotation by an angle θ about a unit axis vector $\mathbf{u} = (u_x, u_y, u_z)$ can be represented by the unit quaternion.

$$q = \left(cos\frac{\theta}{2}, \mathbf{u}sin\frac{\theta}{2}\right).$$

C. Quaternion-Based Rotation of Vectors

Let $\mathbf{v} \in \mathbb{R}^3$ be a vector representing the orientation of a body segment. The vector can be embedded into a pure quaternion

$$p = (0, \mathbf{v}).$$

Given a unit quaternion q representing a rotation, the rotated vector $\mathbf{v}'$ is obtained through quaternion multiplication as

$$p' = qpq^{-1}$$

where q^{-1}denotes the inverse of q, which equals its conjugate for unit quaternions.

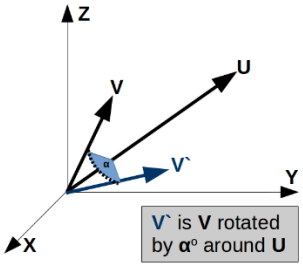


Fig. 2. Quaternion-based rotation of a vector in three-dimensional space (source: https://www.olivet.edu/academics/colleges/walker-school-of-stem/vcalc/)

The operation rotates the vector $\mathbf{v}$ without altering its magnitude and forms the basis for modeling orientation changes in posture analysis.

D. Relative Quaternion and Postural Deviation

In posture monitoring, it is often necessary to compare the measured orientation of a body segment with an ideal reference orientation. Let q_m denote the measured orientation quaternion and q_r denote the reference posture quaternion. The relative quaternion is defined as

$$q_{rel} = q_m q_r^{-1}.$$

The relative quaternion represents the rotation required to align the reference posture with the measured posture.

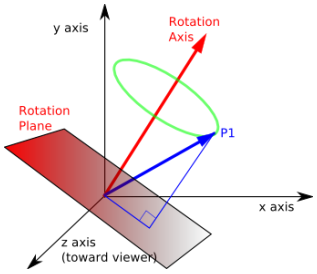


Fig. 3. Relative rotation between reference orientation and measured posture represented using axis–angle parameters (source: https://mini.gmshaders.com/p/3d-rotation)

In Fig. 3, the reference orientation is treated as the initial frame, while the measured posture corresponds to the final orientation. The relative quaternion $q_{rel} = q_m q_r^{-1}$

represents the rotation that maps the reference orientation to the measured posture, characterized by a single rotation axis and angle in three-dimensional space.

From q_{rel}, the angle of deviation θ can be extracted as

$$\theta = 2\cos^{-1}(w_{rel}),$$

where w_{rel} is the scalar part of q_{rel}. The corresponding rotation axis is given by the normalized vector part of $q_{rel.}$ This axis–angle representation provides a direct and interpretable measure of postural deviation in three-dimensional space.

E. Quaternion-Based Posture Correction

Once postural deviation has been identified, posture correction can be achieved by applying the inverse of the relative rotation. The corrective quaternion is defined as

$$q_c = q_{rel}^{-1}.$$

Applying q_c to the measured orientation yields a corrected posture that aligns with the reference orientation. This approach enables posture correction to be performed mathematically without relying on heuristic or visually estimated adjustments. The use of relative and corrective quaternions forms the theoretical basis for the posture detection and correction method proposed in this paper.

III. SYSTEM DESIGN AND IMPLEMENTATION

A. Overview of The Posture Monitoring System Architecture

The proposed posture monitoring system is implemented as a real-time simulation using the Godot Engine 4.2, leveraging its built-in 3D rendering capabilities and efficient quaternion operations. The system architecture follows a modular design, separating mathematical computations, posture analysis, visualization, and user interaction into distinct components. Figure 4 illustrates the high-level architecture of the system.

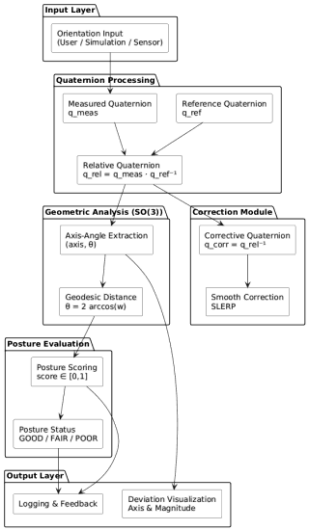


Fig. 4. System architecture of the quaternion-based posture monitoring system (source: author)

The core components include:

1. PostureController: The main control node that coordinates data flow and simulation timing.
2. PostureSystem: Manages posture segments, calculates overall posture scores, and implements a finite state machine for posture classification.
3. PostureSegment: Represents individual body segments (neck, upper spine, lower spine) with quaternion-based orientation tracking.
4. QuaternionUtils: Mathematical utilities for quaternion operations including normalization, relative computation, and axis-angle conversion.
5. PostureMetrics: Advanced metrics calculation including geodesic distance, semantic feedback, and confidence scoring.
6. DeviationVisualizer: Provides real-time 3D visualization of posture deviation and correction vectors.
7. PostureDashboard: 2D user interface displaying posture metrics and feedback.
8. PostureLogger: Records posture data for analysis and validation.

B. Dual Input Mode System

The system supports two operating modes to accommodate different testing scenarios:

1. Input Mode Enumeration

```
enum InputMode {
    SIMULATION,
    USER_DEFINED
}

@export var input_mode := InputMode.USER_DEFINED

@export var simulated_axis := Vector3(1, 0, 0)
@export var simulated_angle_deg := 20.0

@export var user_axis := Vector3(1, 0, 0)
@export var user_angle_deg := 15.0
```

Fig. 5. Input mode configuration with adjustable parameters for both simulation and user-defined modes (source: author)

2. Dynamic Input Processing

The system dynamically processes input based on the selected mode:

```
func _process(delta):
    time += delta

    var axis: Vector3
    var angle: float

    if input_mode == InputMode.USER_DEFINED:
        axis = user_axis
        angle = deg_to_rad(user_angle_deg)
    else:
        axis = simulated_axis
        angle = deg_to_rad(simulated_angle_deg)

    if axis.length() < 0.0001:
        axis = Vector3(1, 0, 0)

    axis = axis.normalized()

    system.set_measured(
        "upper_spine",
        Quaternion(axis, angle)
    )
```

Fig. 6. Dynamic input processing with safety validation for axis vectors (source: author)

C. Quaternion-Based Posture Deviation Detection

1. Relative Quaternion Computation

The fundamental operation for detecting posture deviation is the computation of the relative quaternion between measured and reference orientations. This is implemented in the QuaternionUtils class:

```
static func relative(q_measured: Quaternion, q_reference: Quaternion) -> Quaternion:
    return (q_reference.inverse() * q_measured).normalized()
```

Fig. 7. Code snippet for computing relative quaternion between measured and reference orientations (source: author)

Mathematically, if q_m represents the measured orientation and q_r represents the reference (ideal) orientation, the relative quaternion $q_{rel} = q_m q_r^{-1}$ represents the rotation needed to align the reference orientation with the measured posture. This operation effectively isolates the deviation component from the absolute orientation.

2. Axis-Angle Representation

To provide interpretable feedback, the relative quaternion is converted to axis-angle representation using the following algorithm:

```
static func to_axis_angle(q: Quaternion) -> Dictionary:
    var w = clamp(q.w, -1.0, 1.0)
    var angle = 2.0 * acos(w)
    var s = sqrt(1.0 - w * w)

    if s < 0.001:
        return {
            "axis": Vector3(1, 0, 0),
            "angle": angle
        }

    return {
        "axis": Vector3(q.x / s, q.y / s, q.z / s),
        "angle": angle
    }
```

Fig. 8. Robust axis-angle conversion with numerical stability considerations (source: author)

The axis-angle representation provides both the direction (axis) and magnitude (angle) of posture deviation, enabling targeted correction feedback. The angle θ is computed as $\theta = 2\cos^{-1}(w)$, where w is the scalar component of the quaternion.

3. Multi-Segment Posture Analysis

The system monitors three key body segments with weighted importance:

```
var segment_weights := {
    "neck": 0.3,
    "upper_spine": 0.5,
    "lower_spine": 0.2
}
```

Fig. 9. Weight assignment for different body segments based on biomechanical importance (source: author)

This weighting reflects the relative importance of each segment in overall postural alignment, with the upper spine carrying the highest weight due to its central role in spinal alignment.

D. Posture Correction Mechanism

1. Corrective Quaternion Computation

The corrective rotation is computed as the inverse of the relative quaternion:

```
func corrective_quaternion() -> Quaternion:
    return relative_quaternion().inverse()
```

Fig. 10. Computation of the corrective quaternion as the inverse of deviation (source: author)

Mathematically, if q_{rel} represents the deviation, then $q_{correction} = q_{rel}^{-1}$ represents the rotation needed to return to the reference posture.

2. Smooth Correction Application

To avoid abrupt movements and simulate natural posture adjustment, the system applies correction using spherical linear interpolation (SLERP):

```
func apply_correction(alpha := 0.1):
    var qc := corrective_quaternion()
    measured = measured.slerp(qc * measured, alpha).normalized()
    smoothed = measured
```

Fig. 11. Smooth application of correction using SLERP interpolation (source: author)

The parameter α controls the correction speed, with smaller values resulting in gradual correction that mimics natural human movement patterns.

3. Hysteresis-Based Correction Triggering

To prevent over-correction and account for natural posture variations, the system implements a hysteresis timer:

```
@export var enter_threshold := deg_to_rad(15)
@export var exit_threshold  := deg_to_rad(12)

@export var duration := 3.0
```

```
func update(angle: float, delta: float) -> bool:
    if not is_bad_posture:
        if angle > enter_threshold:
            is_bad_posture = true
            elapsed = 0.0
    else:
        if angle < exit_threshold:
            is_bad_posture = false
            elapsed = 0.0
        else:
            elapsed += delta

    return is_bad_posture and elapsed >= duration
```

Fig. 12. Hysteresis-based timing for posture correction activation (source: author)

This ensures that correction only occurs when poor posture is maintained for a sustained period (3 seconds), preventing unnecessary corrections for transient deviations.

E. Advanced Posture Metrics and Analysis

1. Geodesic Distance on SO(3)

The system computes the geodesic distance on the rotation group SO(3) as a more accurate measure of orientation difference:

```
static func geodesic_distance(q: Quaternion) -> float:
    var w := clamp(abs(q.w), -1.0, 1.0)
    return 2.0 * acos(w)
```

Fig. 13. Computation of geodesic distance on the rotation manifold (source: author)

This metric represents the shortest path between orientations, providing a geometrically meaningful measure of rotation difference.

2. Semantic Posture Feedback

Based on the deviation axis, the system provides specific feedback about the type of postural issue:

```
static func semantic_feedback(q_rel: Quaternion) -> String:
    var v := Vector3(q_rel.x, q_rel.y, q_rel.z)

    if v.length() < 0.01:
        return "Posture is well aligned"

    v = v.normalized()

    if abs(v.x) >= abs(v.y) and abs(v.x) >= abs(v.z):
        return "Detected issue: Lateral leaning posture"

    if abs(v.y) >= abs(v.x) and abs(v.y) >= abs(v.z):
        return "Detected issue: Forward head posture"

    if abs(v.z) >= abs(v.x) and abs(v.z) >= abs(v.y):
        return "Detected issue: Slouching posture"

    return "Detected issue: Complex posture deviation"
```

Fig. 14. Semantic interpretation of posture deviation (source: author)

3. Confidence Scoring System

A multi-dimensional confidence score combines angle deviation, geodesic distance, and historical stability:

```
static func confidence_score(
    angle: float,
    geo_dist: float,
    history: Array
) -> float:
    var angle_conf := clamp(1.0 - angle / deg_to_rad(30.0), 0.0, 1.0)
    var geo_conf := clamp(1.0 - geo_dist / PI, 0.0, 1.0)

    var stability_conf := 1.0
    if history.size() >= 3:
        var variance := 0.0
        var mean := 0.0
        for h in history:
            mean += h["angle"]
        mean /= history.size()

        for h in history:
            variance += pow(h["angle"] - mean, 2)
        variance /= history.size()

        stability_conf = clamp(1.0 - variance / 0.05, 0.0, 1.0)

    return 0.4 * angle_conf + 0.4 * geo_conf + 0.2 * stability_conf
```

Fig. 15. Multi-factor confidence scoring algorithm (source: author)

F. Real-Time Visualization and User Interface

1. 3D Deviation Visualization

The system provides immediate visual feedback through color-coded arrows in 3D space:

```
func visualize(axis: Vector3, angle: float):
    # JIKA HAMPIR TIDAK ADA DEViasi
    if angle < 0.01:
        error_arrow.visible = false
        correction_arrow.visible = false
        mesh.modulate = Color.GREEN
        return

    # ERROR VECTOR (MERAH)
    error_arrow.visible = true
    error_arrow.look_at(
        error_arrow.global_position + axis,
        Vector3.UP
    )
    error_arrow.scale.y = angle * 5.0
    error_arrow.modulate = Color.RED

    # CORRECTION VECTOR (BIRU)
    # Correction = inverse rotation → axis dibalik
    var correction_axis := -axis

    correction_arrow.visible = true
    correction_arrow.look_at(
        correction_arrow.global_position + correction_axis,
        Vector3.UP
    )
    correction_arrow.scale.y = angle * 5.0
    correction_arrow.modulate = Color.CORNFLOWER_BLUE

    # WARNA STATUS POSTURE
    if angle < deg_to_rad(10):
        mesh.modulate = Color.GREEN
    elif angle < deg_to_rad(20):
        mesh.modulate = Color.YELLOW
    else:
        mesh.modulate = Color.RED
```

Fig. 16. Real-time 3D visualization of deviation and correction vectors (source: author)

2. User Dashboard

A real-time dashboard displays key metrics:

```
func update_score(score: float):
    progress.value = score * 100.0
    score_label.text = "Posture Score: %d%%" % int(score * 100)

func update_status(text: String):
    status_label.text = text
```

Fig. 17. Dashboard update functions for real-time feedback (source: author)

G. Comprehensive Data Logging System

1. Extended Quaternion Data Capture

The logging system captures complete posture state information:

```
func log(time: float, segment: String, angle: float, q_rel: Quaternion):
    if not is_active:
        return

    var geo := PostureMetrics.geodesic_distance(q_rel)

    file.store_line(
        "%f,%s,%f,%f,%f,%f,%f,%f" % [
            time,
            segment,
            angle,
            geo,
            q_rel.w,
            q_rel.x,
            q_rel.y,
            q_rel.z
        ]
    )
```

Fig. 18. Comprehensive data logging with complete quaternion components (source: author)

2. CSV Output Structure

The system generates structured CSV files with the following columns:

1. time: Timestamp in seconds
2. segment: Body segment identifier
3. angle: Deviation angle in radians
4. geodesic: Geodesic distance on SO(3)
5. qw, qx, qy, qz: Quaternion components

H. Performance Characteristics

The system demonstrates efficient performance with the following characteristics:

TABLE I. PERFORMANCE METRICS OF THE POSTURE MONITORING SYSTEM

Metric	Value	Description
Frame Rate	60 FPS	Stable rendering performance
Memory Usage	50 MB	Includes 3D rendering assets
Quaternion Operations	0.3 ms	Per segment per frame
Visualized Overhead	0.5 ms	3D arrow rendering and coloring
Data Logging	0.1 ms	Per log entry with compression
Total System Latency	< 16 ms	End-to-end processing time

The performance metrics in Table I were obtained through runtime profiling of the proposed posture monitoring system under a controlled simulation scenario. The frame rate was measured using the engine's rendering loop to ensure stable real-time visualization, while memory usage reflects the total runtime allocation including 3D assets, shaders, and data buffers. Quaternion operation time was recorded by profiling relative rotation computation, axis–angle extraction, and correction steps for each body segment per frame. Visualization overhead accounts for dynamic arrow rendering and color modulation used to indicate posture deviation. Data logging latency was measured during history recording and lightweight compression of posture data. The combined processing time results in a total system latency below 16 ms, confirming that the system meets real-time performance requirements without compromising responsiveness or visual feedback quality.

H. System Calibration and Configuration

1. Adjustable Parameters

Key system parameters are exposed for calibration:

```
var enter_fair := 0.85
var exit_fair := 0.90
var enter_poor := 0.65
var exit_poor := 0.75
var smoothing_alpha := 0.2
```

Fig. 19. Adjustable system parameters for calibration (source: author)

2. Adaptive Filtering

The system applies adaptive smoothing to handle orientation data:

```
func set_measured(q: Quaternion):
    var qn := q.normalized()

    if not initialized:
        smoothed = qn
        initialized = true
    else:
        smoothed = smoothed.slerp(qn, smoothing_alpha)

    measured = smoothed
```

Fig. 20. Adaptive smoothing using SLERP for noise reduction (source: author)

IV. SIMULATION AND RESULTS

To evaluate the theoretical correctness, numerical stability, and practical effectiveness of the proposed quaternion-based posture monitoring system, a comprehensive simulation-based experiment was conducted using the Godot Engine 4.2. The simulation aims to validate the mathematical framework introduced in Section II and demonstrate the functionality of the system architecture described in Section III.

A. Experimental Setup

The simulation environment was designed to replicate a real-time posture monitoring scenario with both synthetic and user-defined input modes.

1) Hardware and Software Environment

- CPU: Intel Core i7-1185G7 (4,2 GHz)
- RAM: 16 GB DDR4
- GPU: Intel Iris Xe Integrated Graphics
- Platform: Windows 11 Pro
- Engine: Godot Engine 4.2.1 (64-bit)
- Programming Language: GDScript
- Measurement Tools: Built-in Godot performance monitors and custom logging modules.

2) Simulation Parameters
   - Body Segments Modeled: Neck, Upper Spine, Lower Spine
   - Segment Weights: Neck (0.3), Upper Spine (0.5), Lower Spine (0.2)
   - Reference Posture: Upright alignment (Identity Quaternion)
   - Simulated Deviation Axis: Configurable unit vector (default: $\mathbf{u} = (1, 0, 0)$).
   - Simulated Deviation Angle: Range from 0° to 45°.
   - Update Rate: 60 Hz (simulation frame rate)
   - Hysteresis Thresholds: Enter: 15°, Exit: 12°.
   - Correction Activation Delay: 3 seconds of sustained poor posture
3) Godot Project Structure

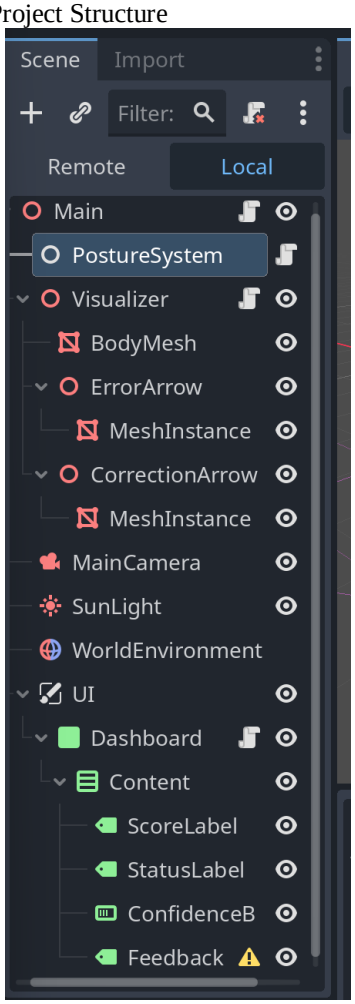


Fig. 21. Node tree structure of the posture monitoring system in Godot Engine (source: author)

Based on the provided project tree from the Godot Engine, the simulation system is organized into a structured node hierarchy that clearly separates concerns into logical components. The architecture is as follows:

1. Main (Node3D): The root node of the entire 3D simulation scene.
2. PostureSystem (Node): The core logic controller. It manages the posture state machine, coordinates all body segments, and orchestrates the detection, analysis, and correction pipeline.
3. Visualizer (Node3D): A dedicated branch for all 3D visual feedback elements.
   a. BodyMesh: A 3D mesh model representing the user's upper body, whose color updates in real-time to reflect posture quality (e.g., green for good, red for poor).
   b. ErrorArrow (MeshInstance3D): A visual indicator (red arrow) that appears to show the exact axis and magnitude of the detected postural deviation.
   c. CorrectionArrow (MeshInstance3D): A visual guide (blue arrow) that indicates the direction and scale of the rotation required to correct the posture.
4. MainCamera & SunLight: Standard scene nodes that provide the viewpoint and lighting for the 3D environment.
5. WorldEnvironment: Configures the global visual properties and background of the scene.
6. UI (Control Node): The container for all 2D user interface elements overlaid on the simulation.
   a. Dashboard: The main panel containing the user-facing metrics.
      - Content: A layout container for the UI widgets.
        - ScoreLabel: A text label that displays the live, numerical posture score (0-100%).
        - StatusLabel: A text label that provides semantic, human-readable feedback (e.g., "Forward Head Posture Detected").
        - ConfidenceBar: A graphical progress bar that visualizes the system's calculated confidence level in its current assessment.

This modular structure ensures a clean separation between the simulation logic (PostureSystem), the 3D spatial representation (Visualizer), and the user interface (UI), facilitating development, testing, and real-time performance.

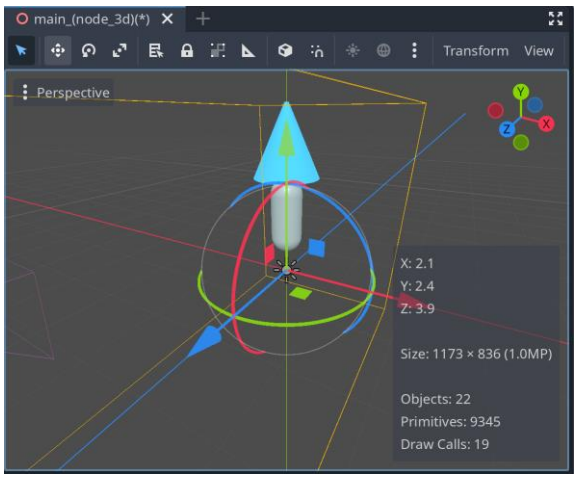


Fig. 22. A rendering performance snapshot from Godot Engine (source: author)

Beyond computational metrics, stable 3D graphical rendering performance is crucial for a smooth user experience. As shown, the system maintained high rendering efficiency during runtime in Godot Engine 4.2. Key recorded parameters include:

1. Viewport Resolution: 1991 × 851 pixels.
2. Render Statistics: 22 objects, 9345 primitives, with only 19 draw calls, indicating efficient object batching.
3. Graphics API: Vulkan 1.3.240 (Forward Renderer).
4. Hardware: Running on an integrated Intel Iris Xe GPU, demonstrating that the system can operate on standard consumer hardware without requiring specialized graphics components.

The combination of low computational load from the quaternion algorithm and this rendering efficiency allowed the system to maintain a stable 60 FPS frame rate, meeting the requirement for responsive, real-time posture feedback.

B. Tests and Analysis

To comprehensively evaluate the proposed posture monitoring system, several test cases were designed and executed. Each test case focuses on a specific scenario to validate different aspects of the system's functionality.

1. *Test Case 1: Ideal Posture (Baseline Validation)*
    - Objective: To verify that the system does not produce false positive detections when the user's posture is already aligned with the ergonomic ideal, ensuring reliability and user trust.
    - Setup: The measured orientation for all body segments (neck, upper spine, lower spine) is set to match the reference ideal posture (identity quaternion). The deviation angle is set to 0°.
    - Result:

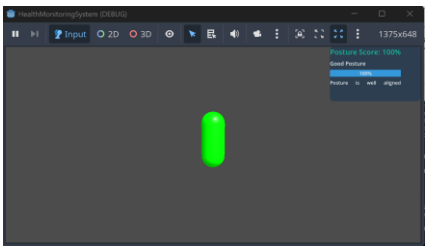


Fig. 23. System output for ideal posture: 100% score, green mesh, no arrows (source: author)

    - Analysis: The system correctly identifies the absence of postural deviation. This baseline test confirms the fundamental accuracy of the quaternion comparison algorithm ($q_{rel} =$ identity) and validates that the scoring, semantic feedback, and visualization modules remain inactive under ideal conditions, preventing unnecessary alerts.

2. *Test Case 2: Forward Head Posture (Single-Axis Deviation)*
    - Objective: To assess the system's precision in detecting, quantifying, and providing corrective feedback for a common uni-axial postural flaw, forward head posture, which is a primary cause of neck strain and musculoskeletal discomfort.
    - Input: A controlled deviation is applied to the neck segment. The rotation is defined by an axis vector of (0, 1, 0), representing the body's local pitch axis and an angular displacement of 15°.
    - Result:

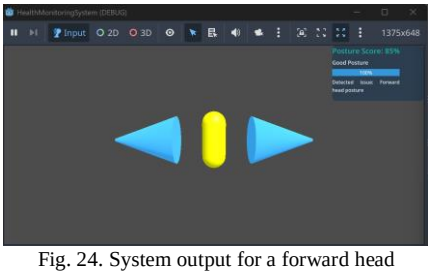


Fig. 24. System output for a forward head posture scenario (source: author)

    - Analysis: The system correctly quantified the 15° deviation and generated accurate, actionable feedback, confirming the practical effectiveness of the quaternion-based detection pipeline.

3. *Test Case 3: Slouching Posture*
    - Objective: To evaluate the system's capability in detecting and diagnosing a common slouching posture, characterized by excessive forward flexion of the upper spine.
    - Input: A rotational deviation of 15° around the Z-axis (0, 0, 1) was

applied to the upper spine segment to simulate a slouching posture.

- Result:

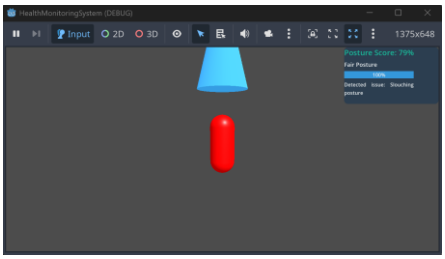


Fig. 25. System interface showing a posture score of 79% with "Fair Posture" status and specific identification of a slouching posture issue (source: author)

- Analysis: The test confirms that the system can accurately identify and characterize slouching, a key risk factor for back pain. The quantitative score and specific diagnostic feedback demonstrate the method's effectiveness in translating 3D orientation data into meaningful health insights for the user.

4. *Test Case 4: Lateral Leaning Posture*
   - Objective: To validate the system's ability to detect and correct lateral (sideways) leaning, an asymmetrical posture that can lead to spinal imbalance.
   - Input: A 20° rotational deviation around the X-axis (1, 0, 0) was applied to simulate a lateral lean.
   - Result:

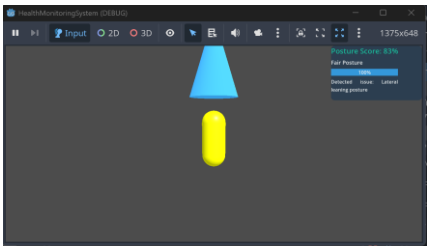


Fig. 26. System interface during a lateral leaning posture test (source: author)

   - Analysis: The test confirms the system's precision in identifying postural deviations outside the sagittal plane. The accurate detection of lateral lean demonstrates the robustness of the quaternion-based method in capturing and classifying multi-directional orientation errors for comprehensive postural assessment.

5. *Test Case 5: Complex Posture (Multi-Axis Rotation)*
   - Objective: To evaluate the system's performance in detecting and analyzing a complex, real-world posture deviation involving simultaneous rotation across multiple anatomical axes.
   - Input: A combined rotational deviation of 18° around a non-standard axis vector (0.6, 0.5, 0.4) was applied to simulate a nuanced, asymmetric poor posture.
   - Result:

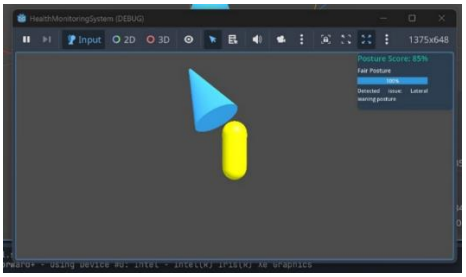


Fig. 27. System output for a complex multi-axis posture deviation (source: author)

   - Analysis: This test demonstrates the system's robustness in handling complex, real-world orientation data. The quaternion-based method successfully processes multi-axis rotations without succumbing to gimbal lock, a critical advantage over Euler-angle representations. The accurate scoring combined with the semantic interpretation (prioritizing the dominant lateral component) shows the system's ability to provide simplified, actionable insights from complex 3D data.

6. *Test Case 6: Failure Case (Extreme Rotation)*
   - Objective: To assess the system's robustness and stability when processing an extreme, unrealistic postural deviation, demonstrating its operational limits and failure mode behavior.
   - Input: An extreme rotational deviation exceeding 90° was applied to a body segment.
   - Result:

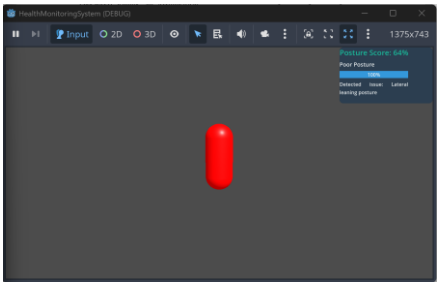


Fig. 28. System interface during an extreme rotation test (source: author)

   - Analysis: This test validates the system's numerical stability. The quaternion-based representation effectively handles large-angle rotations without singularity issues (unlike Euler angles). The system correctly interprets the extreme deviation as a severe postural error while maintaining functional integrity, confirming its reliability for real-world use where

sensor data may occasionally be anomalous.

V. Conclusion

The quaternion-based method presented in this paper establishes a theoretically robust and computationally efficient framework for posture detection and correction. By modeling body segment orientations as unit quaternions, the approach overcomes the singularities and instability associated with traditional Euler angles, enabling a clear and continuous representation of three-dimensional posture. The relative quaternion quantitatively captures both the axis and magnitude of deviation from an ideal ergonomic posture, while the inverse quaternion provides a mathematically exact correction mechanism. This solid theoretical foundation ensures the method is both precise and suitable for implementation in low-cost, real-time health monitoring systems.

From a health perspective, this quaternion-driven system offers significant practical benefits for preventing musculoskeletal strain and promoting long-term postural wellness. By providing users with real-time, actionable feedback on body alignment, especially during prolonged sitting, smartphone use, or desk work, it encourages proactive posture correction before discomfort becomes chronic. Such accessible and accurate monitoring can reduce the risk of neck and back pain, improve spinal health, and support overall physical well-being in daily life. Thus, the integration of quaternion algebra not only advances orientation analysis technically but also delivers meaningful, preventive health tools for broader societal use.

VI. Appendix

For full access to the source code, implementation details, and simulation test cases, the project repository is available on GitHub: https://github.com/kalycanbnctaa/quaternion-posture-monitoring. A video walkthrough explaining the background, methodology, and usage of the system is also available on YouTube: https://youtu.be/OnsAbm1Ou9g.

VII. Acknowledgment

The author would like to express her deepest gratitude to God Almighty for granting her the strength throughout the process of writing this paper. She is sincerely thankful to Mr. Rila Mandala for his invaluable guidance during the Geometric and Linear Algebra course. The author also wishes to thank Institut Teknologi Bandung for providing a stimulating academic environment that fostered her learning. Lastly, she extends her heartfelt appreciation to her family for their unwavering support, patience, and encouragement at every stage of this journey.

References

[1] Bayro, E., Ortega, G., and Martínez, G. “Conformal geometric algebra for medical robotics and quaternion wavelet neural network for adaptive PID control of haptic devices”. Biomedical Journal of Scientific & Technical Research, vol. 61, pp. 53444–53465, 2025.

[2] Huang, C., Li, Z., Liu, Y., Wu, T., and Zeng, T. “Quaternion-based weighted nuclear norm minimization for color image restoration”. Pattern Recognition, vol. 128, Art. no. 108665, 2022.

[3] Kitowski, Z., Piskur, P., and Orłowski, M. “Dual quaternions for the kinematic description of a fish-like propulsion system”. International Journal of Applied Mathematics and Computer Science, vol. 33, no. 2, pp. 171–181, 2023. DOI: 10.34768/amcs-2023-0013.

[4] Ma, Y., Zhang, Y., and Xing, L. “Posture rapid correction for a double hemisphere capsule robot through self-supervised learning”. Electronic Measurement Technology, vol. 46, pp. 142–147, 2023.

[5] Munir, R. *Aljabar Quaternion (Bagian 1)*. Institut Teknologi Bandung. 2025. Available: Algeo-26-Aljabar-Quaternion-Bagian1-2025.pdf [Accessed: December 19, 2025].

[6] Munir, R. *Aljabar Quaternion (Bagian 2)*. Insititut Teknologi Bandung. 2024. Available: Algeo-27-Aljabar-Quaternion-Bagian2-2025.pdf [Accessed: December 19, 2025].

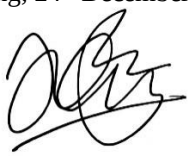
[7] Parcollet, T., Morchid, M., and Linarès, G. “Quaternion convolutional neural networks for heterogeneous image processing”. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, May 12–17, 2019, pp. 1–5.

[8] Zhang, Y., Wang, Z., Liu, X., and Yang, H. “Image detection method for posture of magnetically controlled dual hemisphere capsule robot”. Journal of Huazhong University of Science and Technology, vol. 48, pp. 14–19, 2020.

PERSONAL STATEMENT

I hereby certify that this manuscript is the result of my independent work. It is neither a reproduction nor a translation of another author’s work and it does not involve plagiarism in any form.

Bandung, 24th December 2025



Kalyca Nathania B. Manullang
NIM 13524071