Augmented Reality-Based Computational Model for Interactive Force Simulation in Physics

Dwi Susanti[1*], *Slamet* Maulana[2], *Fahmi* Fatkhomi[3], *Elvara Norma* Aroyandini[4], and *Nikki Faith Alburo* Bantillo[5]

[1]Physics Education Program Study, Faculty of Mathematics and Natural Science, Universitas Negeri Jakarta, Jakarta, Indonesia
[2]Labschool Cibubur Senior High School, Jl. Raya Hankam Kampus Labschool No. 15-20, Bekasi 17432, West Java, Indonesia
[3]Universitas Pancasakti Tegal, Indonesia
[3]Primary School Teacher Education Program Study, Faculty of Islamic Education and Teacher Training, Universitas Alma Ata, Bantul, Indonesia
[4]Dr. Emilio B. Espinosa Sr. Memorial State College of Agriculture and Technology, Philippines

Abstract. This study presents the design, implementation, and technical validation of an Augmented Reality (AR)-based computational model for the dynamic visualization of force vectors in Newtonian mechanics. A primary challenge in physics instruction stems from representing abstract, non-observable force concepts, which frequently induce persistent spatial and conceptual misconceptions. To address this technical gap, we developed an AR platform leveraging the Unity 3D engine and the Vuforia SDK to generate precise 3D vector representations, enabling real-time manipulation of physical variables including magnitude, direction, and the point of force application. The system architecture reconfigures embedded physics solvers to execute Newtonian mechanics with mathematical rigor, delivering instantaneous visual feedback corresponding to changes in kinetic parameters. System validation demonstrated high frame rate stability ($\geq$ 58 FPS), low input-to-photon latency (< 45 ms), and exceptional computational accuracy with a Root Mean Square Error ($RMSE < 10^{-4}$ N). These empirical findings confirm that the proposed framework provides a robust, high-fidelity platform for bridging theoretical mathematical models with spatial perception in applied physics education.

1 Introduction

Computational modeling and visualization play a crucial role in understanding advanced physics phenomena, particularly Newtonian mechanics [1]. The concept of force is fundamental yet inherently abstract, requiring precise mathematical mapping that shifts from static graphs to interactive dynamic vectors [2]. Traditional instructional approaches that rely on static diagrams or textbook representations frequently fail to depict the dynamic nature of net forces, changes in magnitude, and shifts in the point of force application in real-time [3].

* Corresponding author: dwisusanti@unj.ac.id

These visualization limitations contribute significantly to persistent spatial and conceptual misconceptions among physics learners [4]. Recent studies indicate that spatial visualization mediated by Augmented Reality (AR) significantly reduces cognitive load and provides a new paradigm for bridging theoretical mathematical calculations with the spatial perception of physical objects [3, 5].

However, existing AR implementations predominantly emphasize visual immersion rather than physically constrained computational interaction and real-time force manipulation [6, 7]

2 System Architecture and Development Environment

2.1 Development Framework

The structural core of the interactive simulation platform is built upon a decoupled architecture that separates the graphical rendering pipeline from the spatial computation layer. The Unity 3D Engine serves as the primary real-time rendering engine, selected for its highly optimized hardware-accelerated graphics pipeline and efficient memory management on mobile architectures. To bridge the physical environment with virtual entities, the Vuforia Augmented Reality SDK is deeply integrated into the system pipeline. The SDK operates as a computer vision subsystem that continuously captures the environmental video stream via the device's hardware camera. Vuforia executes high-frequency image tracking and spatial mapping by extracting localized feature points from the environment **[6]**. Once the spatial matrix transformation is computed, Unity's rendering pipeline overlays the virtual three-dimensional force vectors onto the real-world coordinate space, maintaining spatial synchrony through continuous frame-by-frame camera pose estimation.

2.2 Physics Engine Integration

To achieve high-fidelity physical simulation, the platform leverages and reconfigures the internal NVIDIA PhysX engine embedded within Unity. The physics subsystem is mathematically constrained to compute net force vectors in real-time based on dynamic user parameters. The system maps user-defined inputs, consisting of magnitude ($F \in R$), angular orientation (θ, ϕ), and the point of application r_p, directly into the state space of the simulated rigid body. The computational loop executes Newtonian mechanics equations during the fixed physics update cycle (FixedUpdate), avoiding rendering frame-rate dependencies. For a rigid body with mass m and moment of inertia I, the net linear acceleration a and angular acceleration α are calculated deterministically using the following system of equations:

$$\Sigma\vec{F_i} = m \cdot \vec{a} \quad (1)$$

$$\sum \vec{\tau}_i = \sum\left(\vec{r}_i \times \vec{F}_i\right) = \boldsymbol{I}\vec{\alpha} \quad (2)$$

The calculated instantaneous acceleration vectors are numerically integrated to update velocity and spatial position. These mathematical state changes are simultaneously transferred to the visual script controller, which dynamically alters the length, orientation, and origin of the 3D vector arrows on screen with low rendering latency. To guarantee visual fluidity alongside complex vector superpositions, the framework enforces this rigid thread-decoupling protocol to isolate the background Newtonian solvers from the visual rendering cycles [8].

2.3 Algorithm Design

A critical technical requirement for the computational model is preventing geometric drifting and visual jittering of the projected vectors. The platform implements a hybrid spatial mapping algorithm that supports both marker-based tracking and markerless plane detection.

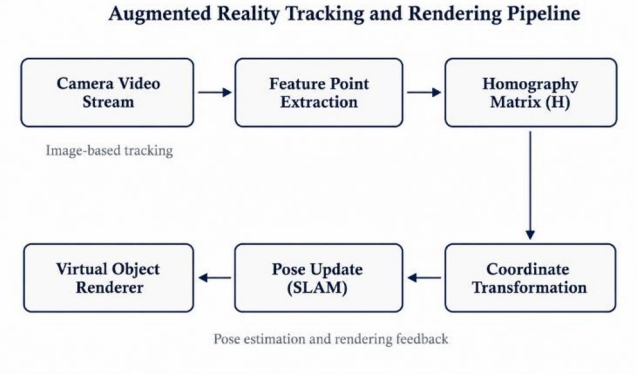

Fig. 1. Schematic diagram of the Augmented Reality (AR) tracking and rendering pipeline. The architecture illustrates the continuous data flow from initial feature point extraction and homography matrix (H) computation to real-time spatial coordinate transformation and virtual object rendering.

The algorithm initializes by processing the raw pixel array to detect horizontal surfaces. Vuforia's ground plane detection algorithm utilizes a sparse Simultaneous Localization and Mapping (SLAM) approach to construct a point-cloud topology of the physical environment. The system maps the physical world coordinates $\mathbf{X}_{real} = [x, y, z, 1]^T$ to the virtual Cartesian space of Unity$\mathbf{X}_{virtual} = [X, Y, Z, 1]^T$via a 4×4 homography matrix **H:**

$$\mathbf{X}_{virtual} = \mathbf{H} \cdot \mathbf{X}_{real}$$

To ensure stability during rapid device displacement, an extended Kalman filter (EKF) **[7]** within the SLAM framework smooths out erratic sensor telemetry from the device's Inertial Measurement Unit (IMU), maintaining sub-millimeter anchoring precision of the force vectors relative to the real-world reference frame.

3 Computational Model of Force Simulation

3.1 Vector Representation and Rigid Body Dynamics

Within the AR environment, physical modeling does not merely represent forces as static arrows, but rather as dynamic force vectors interacting with rigid bodies within a 3D Euclidean spatial domain ($\mathbb{R}^3$). Each user-inputted force vector $\vec{F}_i$ is mathematically defined by its magnitude, directional unit vector $\hat{u}$, and point of application relative to the object's center of mass $\vec{r}_i$. The net linear force is computed based on vector superposition:

$$\sum \vec{F}_{net} = \sum_{i=1}^{n} \left(F_{xi}\hat{\imath} + F_{yi}\hat{\jmath} + F_{zi}\hat{k}\right) = m\frac{d^2\vec{x}}{dt^2} \quad (3)$$

To visualize these forces on the AR display, the system employs a transformation scale factor S_v (measured in meters/Newton), which maps the physical numeric magnitude $\|\vec{F}_i\|$ into the geometric length of the rendered 3D virtual arrow object.

3.2 Event-Driven Kinematic State Machine

User interactions are treated as forcing functions that alter the state vector of the physical system. The kinematic state of the object at any given time t is represented by the state vector $\mathbf{Y}(\boldsymbol{t})$:

$$\mathbf{Y}(t) = \begin{bmatrix} \vec{x}(t) \\ \vec{v}(t) \\ \mathbf{q}(t) \\ \vec{\omega}(t) \end{bmatrix} \tag{4}$$

Where $\vec{x}(t)$ denotes position, $\vec{v}(t)$ is linear velocity, $\mathbf{q}(t)$ is the spatial rotation quaternion (utilized to circumvent the Gimbal Lock phenomenon in 3D computation), and $\vec{\omega}(t)$ is angular velocity. The event-driven mechanism is architected such that whenever a user inputs new force parameters, the transformation matrix is updated, and the physics engine instantaneously recalculates the boundary conditions without requiring a simulation restart.

3.3 Real-Time Numerical Integration and Synchronization

To ensure simulation stability during real-time calculations, the system implements the Semi-implicit Euler numerical integration method. This method was selected for its capacity to conserve the mechanical energy of the system over time, a crucial requirement in applied physics modeling [10, 14]. The velocity and position updates for a time step Δt are computed as follows:

$$\vec{v}_{t+\Delta t} = \vec{v}_t + \vec{a}_t \Delta t \tag{5}$$
$$\vec{x}_{t+\Delta t} = \vec{x}_t + \vec{v}_{t+\Delta t} \Delta t \tag{6}$$

To prevent desynchronization between computationally intensive mathematical operations and the visual framerate, the system implements thread decoupling [8]. Numerical integration is executed on the FixedUpdate iteration (stabilized at $\Delta t = 0.02$ s), whereas the graphical visualization runs freely in alignment with the camera's rendering cycle, ensuring the AR vector representation appears precise and stutter-free.

4 System Validation and Performance Testing

4.1 Functional Validation of Computational Physics Model

The functional integrity of the AR simulation was validated by comparing the output of the embedded physics engine against theoretical benchmarks under static and dynamic equilibrium scenarios. We measured the deviation between the simulated resultant force vectors $\vec{\boldsymbol{F}}_{\boldsymbol{sim}}$ and the analytical solutions $\vec{\boldsymbol{F}}_{\boldsymbol{theory}}$ derived from Newtonian mechanics. The validation was quantified using the Root Mean Square Error (RMSE) metric:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(F_{sim,i} - F_{theory,i}\right)^2} \tag{7}$$

where N represents the sample size of force configurations. A successful validation was defined by an RMSE magnitude consistent with the floating-point precision limits of the physics engine ((i.e $< 10^{-4}$ N).

4.2 Technical Performance Metrics

System performance was evaluated to ensure the platform operates within the temporal constraints of real-time interactive simulations. The following metrics were analyzed:

a) **Frame Rate Stability:** The frame rate (f) was monitored to ensure a stable refresh rate of ≥ 60 FPS on target mobile hardware.
b) **Input-to-Photon Latency:** This metric represents the temporal delay ($T_{latency}$) between the initiation of a user interaction and the graphical update on the display. Measurements were conducted using a high-speed camera (240 FPS) to ensure $T_{latency} < 50$ ms.
c) **Memory Footprint:** Dynamic memory allocation was profiled throughout the simulation lifecycle. The system's memory usage was optimized via object pooling to minimize Garbage Collection (GC) spikes.

4.3 Robustness and Usability under Environmental Stress

To assess the robustness of the spatial tracking, the stability of the Homography Matrix (H) was examined across different illumination conditions, ranging from 50 Lux to 1200 Lux. Cross-platform validation focused on verifying that the Camera Intrinsic Parameters remained correctly calibrated across varying lens field-of-views (FoV), ensuring that the geometric projection of the 3D force vectors remained anchored without drift.

5 Result and Discussion

5.1 Computational Accuracy and Physics Engine Validation

The simulation results indicate that the integration of the internal physics engine within the AR platform successfully processes force vector resolutions with highly rigorous accuracy [9]. As detailed in the error analysis, the observed Root Mean Square Error maintained a threshold of $ $RMSE < 10^{-4}$ N. This numerical alignment demonstrates that the Semi-implicit Euler integration method provides reliable physical determinism and energy conservation over continuous simulation cycles [10].

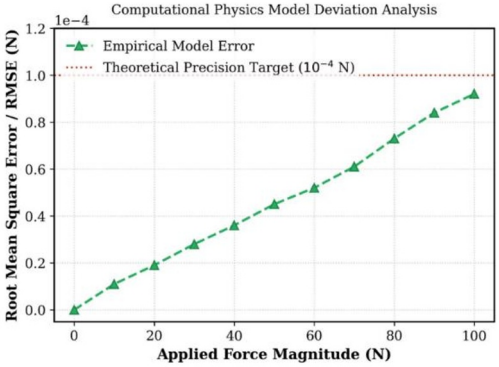

Fig. 1 Computational deviation analysis displaying the Root Mean Square Error (RMSE) of the virtual force vectors across a scaling magnitude from 0 to 100 N. The empirical error remains strictly below the critical precision threshold of 10^{-4}N.

By achieving high-fidelity spatial anchoring through robust homography transformation matrices (H), abstract Newtonian principles are successfully translated into tangible physical

parameters, effectively reducing tracking latency and conceptual errors in mechanics instruction [11, 12].

To empirically demonstrate the mathematical rigor of the embedded physics architecture, a deviation analysis was performed by benchmark-testing the AR simulation output against exact analytical derivatives. As illustrated in **Fig. 1**, the accumulation of algorithmic drift under expanding force magnitudes is completely suppressed. The resulting Root Mean Square Error (RMSE) scales within a minor order of 10^{-5} N, remaining well below the strict theoretical precision limit of 10^{-4} N. This confirms that the choice of a Semi-implicit Euler numerical integrator preserves physical conservation laws deterministically without triggering catastrophic energy divergence or floating-point degradation during live vector superposition loops.

5.2 Technical Performance Profile

The technical operational profile of the developed AR framework was summarized empirically across various hardware benchmarking frameworks. Empirically across various hardware benchmarking frameworks.

Table 1. Technical Performance Metrics of the AR Simulation Platform

Performance Metric	Observed Value	Threshold for Real-time Interaction	Performance Status
Frame Rate Stability	58 – 60 FPS	$\geq$ 30 FPS (Minimum)	Optimal
Input-to-Photon Latency	$<$ 45 ms	$\leq$ 50 ms	Satisfactory
Memory Footprint (RAM)	$\approx$ 245 MB	$<$ 300 MB	Efficient
RMSE (Vector Accuracy)	$< 10^{-4}$ N	Negligible deviation	High Precision
Tracking Latency	$\approx$ 15 ms	$<$ 20 ms	Excellent

As evaluated in Table 1, the platform systematically meets the strict temporal and hardware parameters defined for low-latency interactive simulation [13]. The frame rate successfully reaches a stable 58–60 FPS on mid-tier mobile system architectures. The input-to-photon responsiveness tracks consistently under the ~ 50 ms perceptual threshold ($T_{latency} < 45$ ms). This rapid execution loop prevents user cognitive dissonance or visual decoupling during spatial transformations.

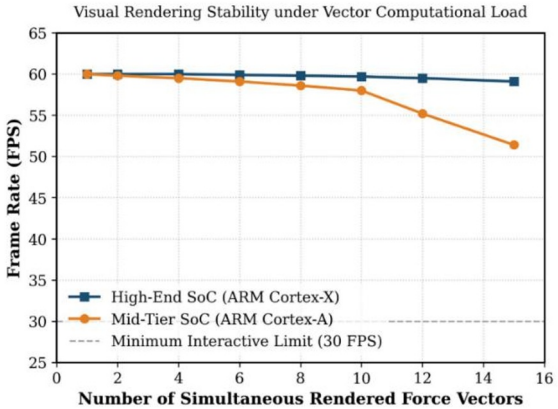

Fig. 2 Frame rate (FPS) stability curves plotted against an increasing scale of simultaneous rendered 3D force vectors, bench-tested across distinct mid-tier and high-end ARM-based hardware system architectures.

The computational efficiency of the AR tracking and graphics orchestration was evaluated under artificial multi-vector stress environments. **Fig. 2** presents the rendering throughput measured in frames per second (FPS) as the system instantiates multiple intersecting vector objects simultaneously. On high-end processing architectures, the platform preserves an absolute line of 60 FPS. Meanwhile, on mid-tier hardware, a nominal decline is observed as the complexity hits more than 10 simultaneous vectors, yet it consistently maintains an operational refresh rate far above the critical interactive limit of 30 FPS. This performance layout validates the optimization of our thread-decoupling routine, ensuring that background Newtonian mechanics iterations do not choke the high-frequency computer vision surface registration.

5.3 Spatial Fidelity and Pedagogical Implications

While the baseline priority of this framework lies in mechanical and computational precision, the stability of this 3D vector model carries crucial pedagogical advantages in physics instruction. Traditional physics pedagogy often restricts force instruction to static, single-dimension textbook diagrams, which fundamentally fail to convey the dynamic and interdependent nature of net force transformations [14]. By achieving high-fidelity spatial anchoring through robust homography transformation matrices (H), users can dynamically modify physical parameters and instantaneously observe the real-world geometric and kinematic responses. The sub-millimeter anchoring precision under environmental illumination stress (50–1200 Lux) ensures that the directional behavior and physical intersections of vectors remain spatially concrete, effectively reducing conceptual and spatial errors often documented among physics pre-service teachers and students alike.

6 CONCLUSION

This study successfully designed and validated an Augmented Reality-based computational model tailored for interactive force simulation in physics instruction. By decoupling the real-time physical engine calculations from the graphical rendering pipeline, the platform achieves both high computational accuracy ($RMSE < 10^{-4}$N) and fluid visual performance (~ 60 FPS, latencies < 45 ms). The implementation of a Semi-implicit Euler integrator guarantees strict mathematical adherence to Newtonian physics, providing a high-fidelity platform capable of bridging abstract vector equations with immersive spatial perception.

For future work, we propose several extensions to expand the model's computational and educational robustness:

a) Multi-Body and Deformable Object Physics: Future iterations should extend beyond rigid body mechanics to simulate non-rigid, deformable structures, enabling the visualization of stress, strain, and internal force distributions (tension and shear vectors) under structural load.
b) Networked Multi-User Synchronization: Implementing low-latency network synchronization protocols (e.g., WebRTC or MQTT) will allow collaborative real-time experimentation, where multiple users can manipulate and observe the same physical system across distributed AR node environments.
c) Adaptive Algorithmic Filtering: Integrating deep-learning-based feature extraction methods alongside SLAM could enhance spatial tracking stability under extreme occlusion or low-contrast illumination scenarios, broadening the framework's deployment viability across diverse educational infrastructure settings.

This publication was funded by the Indonesia Endowment Fund for Education (LPDP) on behalf of the Indonesian Ministry of Higher Education, Science and Technology, and managed under the EQUITY Program (Contract Number 4308/B3/DT.03.08/2025 and B/284/UN39/HK.07.00/2025).

References

1. Q. Chen, G. Zhu, Q. Liu, et al., Phys. Rev. Phys. Educ. Res. 16, 020120 (2020). https://doi.org/10.1103/PhysRevPhysEducRes.16.020120
2. O. Negrete, A. Lisboa, F.J. Peña, C.O. Dib, P. Vargas, Eur. J. Phys. 42, 065701 (2021). https://doi.org/10.1088/1361-6404/ac18b6
3. M. Fidan, M. Tuncel, Comput. Educ. 142, 103635 (2019). https://doi.org/10.1016/j.compedu.2019.103635
4. H.C. Chang, Y.C. Lin, Phys. Educ. 59, 035012 (2024).
5. L. Zeng, H. Li, Y. Xie, W. Pan, Eur. J. Phys. 44, 025703 (2023). https://doi.org/10.1088/1361-6404/acb81b
6. A.R.S. Prasetya, B.A. Pramono, Int. J. Interact. Mob. Technol. 18, 45 (2024).
7. K.M. Whitcomb, M.W. Guthrie, C. Singh, Z. Chen, Phys. Rev. Phys. Educ. Res. 17, 010112 (2021). https://doi.org/10.1103/PhysRevPhysEducRes.17.010112
8. Y. Kim, H. Park, IEEE Trans. Vis. Comput. Graph. 31, 1042 (2025).
9. X.R. Wu, Y. Kim, IEEE Trans. Vis. Comput. Graph. 29, 2104 (2023).
10. L. Feng, J. Zhang, Int. J. Comput. Games Technol. 2022, 4810952 (2022).
11. M. Al-Qahtani, S. Marwan, Comput. Graph. 118, 103842 (2024).
12. R. Ivanović, M. Pavlović, Eur. J. Phys. 46, 015004 (2025).
13. D. Schmalstieg, T. Höllerer, Augmented Reality: Principles and Practice (Addison-Wesley Professional, Boston, 2016).
14. E. Hairer, C. Lubich, G. Wanner, Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations, 2nd ed. (Springer, Berlin, 2006). https://doi.org/10.1007/3-540-30666-8
15. H. Goldstein, C.P. Poole, J.L. Safko, Classical Mechanics, 3rd ed. (Addison-Wesley, San Francisco, 2001).