

Journal of Digital Education and Learning Engineering

ดำเนินการวารสารโดย สมาคมการศึกษาดิจิทัลและวิศวกรรมการเรียนรู้

Simulating Robotic Kinematics for Learning: A Game Engine Approach

Kantinan Saivichit¹ & Narong Aphiratsakun^{2*}

King Mongkut's Institute of Technology Ladkrabang, Bangkok, Thailand¹, Research and Development Department STEAM
Kiddee Lab Skills Enhancement School, Bangkok, Thailand^{2*}

Received: March 4, 2026 Revised: March 11, 2026 Accepted: June 14, 2026

Abstract

Teaching robotic kinematics presents significant challenges due to the abstract mathematical concepts involved and students' difficulty in visualizing three-dimensional movements. Traditional teaching methods often fail to provide adequate connections between theoretical principles and practical applications. This study demonstrates the development and evaluation of a Unity-based simulation tool designed to enhance robotic kinematics education through interactive visual learning. The simulation includes four modules: coordinate systems, basic kinematics, the modified Denavit-Hartenberg matrix, and PID control. Performance analysis confirmed the tool's efficiency on standard hardware, while educational effectiveness was assessed using the Instructional Materials Motivation Survey (IMMS) with control ($n=31$) and experimental ($n=33$) groups. Results revealed statistically significant improvements in Attention ($p < .001$, Cohen's $d=0.659$) and Satisfaction ($p < .001$, Cohen's $d=0.749$) dimensions for students using the simulation. Qualitative feedback highlighted the tool's effectiveness in visualizing complex concepts and bridging the gap between abstract equations and real-world applications. Users suggested improvements in the interface design and the addition of guided tutorials. This research demonstrates that game engine-based simulations can meaningfully enhance motivation and engagement in robotics education, particularly through improved conceptual visualization and interactive exploration.

Keywords: Kinematic, robotic, feedback control, simulation, game engine

This paper is a revised and expanded version of a paper entitled '*Robotic Kinematics Learning Guidance Based on Game Engine Simulation*' presented at the 9th International STEM Education Conference, Thailand, July 31 - August 2, 2024.

*Corresponding author

Email address: narongphr@au.edu

■ Introduction

Robotic kinematics focuses on the motion of robots without considering the forces that cause this motion. It involves understanding how various joints and links of a robot move in relation to one another. In robotics education, kinematics is particularly significant as it forms the basis for programming robots to perform specific tasks (Ragusa and Leung, 2023). However, teaching kinematics presents several challenges. Many students struggle with the mathematical concepts involved, often lacking the visual intuition between the abstract equations and the actual movements, which are integral to comprehending kinematic principles (Alimisis and Boulougaris, 2014). Additionally, (Dreger, Chuang and Rinkenauer, 2024) suggests that the complexity of visualizing three-dimensional movements and the relationships between different joints can also hinder learning.

■ Research Questions

This study addresses the following research questions:

- 1) Can game engine technology be effectively utilized to develop an interactive simulation tool that enhances student engagement in learning robotic kinematics?
- 2) Does the use of a Unity-based simulation tool significantly improve student motivation across the four dimensions of the ARCS model (Attention, Relevance, Confidence, and Satisfaction) compared to traditional teaching methods?
- 3) To what extent does interactive visualization through game engine-based simulation enhance students' understanding of complex kinematic concepts, including coordinate systems, Denavit-Hartenberg parameters, and PID control?

■ Significance and Purposes

- 1) To demonstrate the creation of a high-performance real-time interactive simulation program utilizing game engine technologies for robotic kinematics education, with emphasis on enhancing interactivity, optimizing performance, and reducing development time.
- 2) To effectively visualize robotic kinematic concepts, including coordinate systems, basic kinematics, the modified DH matrix, and PID control, through the developed simulation program.
- 3) To evaluate the developed simulations based on their performance metrics and educational effectiveness in terms of student motivation and self-efficacy, using the IMMS to measure the four subscales of the ARCS model, complemented by qualitative feedback.

■ Literature Reviews

Traditional teaching methods may fail to engage students fully, leaving them with a superficial understanding of the concept. Traditional robotic kinematics teaching methods primarily consist of lectures, laboratory-based activities, and simulations, each with its advantages and limitations. Lectures serve as an essential foundation, introducing students to theoretical concepts and mathematical formulations (Wieman and Perkins, 2005, Prince, 2004, Bligh, 2000). However, without interactive visual components, lectures may fail to provide the necessary intuition for understanding three-dimensional movement and joint relationships (Lanos et al., 2024, Yang and Wang, 2023). Laboratory-based activities, such as working with physical robotic arms, offer direct experience with kinematic principles, reinforcing theoretical knowledge through practical application. This approach, however, presents challenges such as high costs associated with robotic hardware, safety concerns, and potential difficulty in explicitly linking abstract mathematical concepts to real-world motion (González-García et al., 2020; Ferreira and Freitas, 2018; Khalil and Creusot, 1997; Raz, 1989; Lloyd and Hayward, 1988). While effective for reinforcing concepts, laboratory-based activities are best suited for students who have already developed a foundational understanding through other means (Wang et al., 2023; Kirschner et al., 2006). Simulations provide an intermediate learning tool, bridging the gap between theory and physical application. However, not all simulation tools are equally effective (see Table 1). Some are overly technical, requiring significant prior knowledge, while others lack the necessary functionality to provide meaningful insights into kinematic relationships (González-García et al., 2020).

Table 1.

Comparison of the main properties of several robot simulation programs.

Program	Programming language	Program input	Interactive	GUI	Available robots	Educational
Proposed simulator	Unity3D	Mouse	Yes	Yes	Arbitrary serial-link manipulators	Yes
ROBOLAB	MATLAB	Mouse	Yes	Yes	Arbitrary serial-link manipulators	Yes
RCAAD	MATLAB	Mouse	Yes	Yes	Arbitrary serial-link manipulators	No
Robotics toolbox	MATLAB	Command line	No	No	Puma and Stanford	No
Planar manipulator toolbox	MATLAB	Command line	No	No	Planar manipulators	No

Program	Programming language	Program input	Interactive	GUI	Available robots	Educational
ROBOTICA	MATHEMATICA	Mouse	No	Yes	Arbitrary serial-link manipulators	Yes
ROBOOP	C++	Command line	No	No	Arbitrary serial-link manipulators	Yes
ROBOSIM	Common LISP	Mouse	No	Yes	Arbitrary serial-link manipulators	No

A well-designed simulation tool should be intuitive, interactive, and capable of representing key kinematic principles accurately, ensuring that students can visualize and manipulate robotic motion effectively. By combining these approaches, using lectures as an introduction, simulations to develop intuition, and hands-on activities to solidify practical skills, robotic kinematics education can be significantly improved (Wang et al., 2023). Effective educational strategies should seek to balance these methods, optimizing engagement and comprehension while addressing the limitations of each approach. To address the limitations that current simulations possess, dedicated educational simulation tools are being explored (Fernández et al., 2020; González-García et al., 2020; Kortemeyer, 2019; Ferreira and Freitas, 2018; Caccavelli et al., 2011; Hugues and Bredeche, 2006; Khalil and Creusot, 1997; Raz, 1989; Lloyd and Hayward, 1988). One major difficulty lies in maintaining an intuitive and interactive user experience. Traditional robotic simulation tools are not designed to achieve real-time interactable performance (Choi et al., 2021). Additionally, developing a flexible yet accurate interface that allows users to manipulate robotic models without requiring extensive programming knowledge is a key concern (Haces-Garcia and Zhu, 2024). This is where game engines play a crucial role.

Game engines have emerged as powerful tools for creating simulation environments that enhance the learning of robotic kinematics. Utilizing game engines allows for rapid development of interactive simulation tools that provide a real-time visual and dynamic way to explore robotic concepts with ergonomic software interface (Arnay et al., 2017). By simulating real-world scenarios, students can manipulate robotic models, observing how changes in joint configurations affect overall movement and behaviour. Incorporating learning theories into the design of these simulation tools is fundamental to their effectiveness. Constructivist principles emphasize the importance of active learning, where students build knowledge through exploration and experimentation (Zajda, 2021). Game engine-based simulations align well with this approach, allowing learners to test hypotheses and see immediate feedback on their actions. For example, students can adjust a robot's joint angles and observe the resulting position of the end effector, solidifying their understanding of forward and inverse kinematics. It is also beneficial to introduce students to simpler, well-understood concepts before progressing to more complex topics (Raslan, 2024). Gradually increasing complexity allows learners to familiarize themselves with the simulation environment while reinforcing foundational knowledge.

The simulation environment presented in this paper follows this structured learning approach by incorporating four distinct simulations: coordinate systems, basic kinematics, the modified Denavit-Hartenberg (DH) matrix, and basic proportional-integral-derivative (PID) control applied to kinematic models. The first simulation introduces students to coordinate transformations, a fundamental concept in robotic motion representation. The second simulation builds on this by allowing learners to explore basic kinematics, reinforcing the relationship between position, velocity, and acceleration. The third simulation integrates the modified DH matrix with forward kinematic and inverse kinematic which uses the Jacobian transpose method and damped least-squares (DLS) method, providing a systematic way to model robotic linkages and view their transformations. The modified DH convention is chosen over the standard DH convention due to its improved intuitiveness with how it arranges the parameters α and a , which describe the orientation and length of each robotic link. (Singh, Kukshal and Yadav, 2021) shows the comparison between classical and modified DH conventions in Table 2.

Table 2.

Comparison between modified and classical D-H models.

D-H parameter	Classical Convention	Modified Convention
Joint axis	Z_{i-1} is for $\dot{i}$ axis	Z_i is for $\dot{i}$ axis
Link length (a_i)	The distance from O_i to the intersection of the Z_{i-1} and X_i axes along the X_i axis	The distance from Z_i to Z_{i+1} measured along X_i
Twist angle (α_i)	The angle from the Z_{i-1} axis to the Z_i axis about the X_i axis	The angle between Z_i to Z_{i+1} measured about X_i
Joint offset (d_i)	The distance from O_{i-1} to the intersection of Z_{i-1} with the X_i axis along the Z_{i-1} axis	The distance from X_{i-1} to X_i measured along Z_i
Joint angle (θ_i)	The angle between X_{i-1} and X_i along Z_{i-1}	The angle between X_{i-1} and X_i along Z_i

DH parameters are defined such that the parameters α and a are associated with the next joint in the sequence, when defining a transformation, these values describe how the previous joint (i-1) transitions to the next one, while the parameters θ and d are associated with the current joint (i). In the standard DH convention, the parameters α_i and a_i are aligned with θ_i and d_i . This can make it harder to think about the direct impact of the parameters α_i and a_i on each joint's movement because the focus is on what happens after the joint rather than its immediate effect. In contrast, the modified DH convention aligns α_{i-1} and a_{i-1} with θ_i and d_i , meaning that these values directly describe how the current joint is positioned relative to the previous one. This shift makes it easier to visualize the transformation process because each joint's parameters define its own placement rather than being dependent on the next one. Finally, the fourth simulation

introduces basic PID control applied to kinematics. Each simulation acts as a steppingstone, guiding students toward a deeper understanding of robotic kinematics and bridging the gap between theory and application.

To evaluate the educational effectiveness of the simulation, this study employs the Instructional Materials Motivation Survey (IMMS), which is based on Keller's ARCS model of motivational design. The IMMS has been widely validated as a reliable instrument for assessing learners' motivation across four dimensions: Attention, Relevance, Confidence, and Satisfaction. In Keller's original validation study (2010), high internal consistency was reported with Cronbach's alpha coefficients of $\alpha = 0.89$ for Attention, $\alpha = 0.81$ for Relevance, $\alpha = 0.90$ for Confidence, and $\alpha = 0.92$ for Satisfaction. Subsequent research supports these findings; for instance, Looibach et al. (2015) reported $\alpha = 0.73$ (Attention), $\alpha = 0.68$ (Relevance), $\alpha = 0.73$ (Confidence), and $\alpha = 0.82$ (Satisfaction). A Brazilian Portuguese adaptation (IMMS-BRV) further confirmed the scale's reliability, yielding values of $\alpha = 0.76$, 0.93 , 0.87 , and 0.78 , respectively (Cardoso-Júnior and Faria, 2021). Together, these studies affirm the IMMS as a dependable measure for evaluating motivational impact in instructional settings.

■ Methods

This section describes the development of the Unity-based simulation tool and the methods used to evaluate its technical performance and educational impact. The application includes four modules: Object Coordinates, Kinematics, DH Matrix, and PID Control, each aligned with the key concepts in robotic kinematics.

Implementation

The application was developed using the game engine, Unity3D, chosen for its versatility and accessibility. Unity's user-friendly interface, comprehensive documentation, and C# scripting capabilities enable efficient development (Moiseienko et al., 2023). Additionally, its real-time rendering during development facilitates immediate feedback and iterative improvements. Another advantage of Unity is its built-in multi-platform export functionality, which allows seamless deployment across Windows, macOS, Linux, and other operating systems, thereby reducing distribution challenges (Moiseienko et al., 2023; Mosler et al., 2023). The application is structured into 4 primary simulations: Object Coordinates Simulation, Kinematic Simulation, DH Matrix Simulation, and PID Control Simulation. The implementation process for application control and each simulation will be elaborated in 5 subsections.

1) Application Control

The simulation environment allows users to freely navigate a 3D space using the mouse and keyboard, except in the Kinematic Simulation, which operates in a 2D environment. In 2D mode, users can pan and zoom. Several quality-of-life enhancements, such as adjustable movement speed and hotkeys, are incorporated for ease of navigation. Rotation and panning using the mouse are activated when the right

mouse button is held, while cursor interaction with UI elements remains independent of this function. The development of navigation controls was streamlined due to the extensive availability of online documentation and built-in Unity features. Various resources, including free tutorials and guides, facilitated the implementation process. Unity's built-in vector and matrix manipulation functions, along with UI design tools, enabled efficient integration of navigation and interaction mechanics.

2) Object Coordinates Simulation

This simulation enables users to manipulate a target object, represented as a cube, within a virtual space. The cube's coordinates are displayed in Cartesian, cylindrical, and spherical coordinate systems. Users can switch to a 2D mode that constrains movement and viewing angles accordingly. The key feature is that modifying a coordinate value in one system updates the corresponding values in the other systems, allowing dynamic transformations of the cube's position. Coordinate conversions are achieved using mathematical equations:

$$x = r \cdot \cos(\theta) \quad (1)$$

$$y = r \cdot \sin(\theta) \quad (2)$$

$$z = z \quad (3)$$

$$x = \rho \cdot \sin(\varphi) \cdot \cos(\theta) \quad (4)$$

$$y = \rho \cdot \sin(\varphi) \cdot \sin(\theta) \quad (5)$$

$$z = \rho \cdot \cos(\varphi) \quad (6)$$

where: x, y, z are the Cartesian coordinates; r is the radial distance on the x-y plane from the origin to the point; ρ is the distance from the origin to the point; θ is the rotation around the z-axis; and φ is the elevation from the positive z-axis. The Equation 1, 2 and 3 are used to convert the cylindrical coordinate to cartesian coordinate. The Equation 4, 5 and 6 are used to convert the spherical coordinate to cartesian coordinate. The inverses of the equations are also included in the code to convert from cartesian coordinate to cylindrical and spherical coordinate. Conversion between cylindrical and spherical coordinates uses cartesian coordinate as the medium. The implementation of this feature utilized existing Unity tools and trigonometric functions, ensuring a straightforward development process. Figure 1 shows a visual representation of each coordinate system, clearly labelling the variables in their geometric context. Figure 2 illustrates the object coordinates simulation interface.

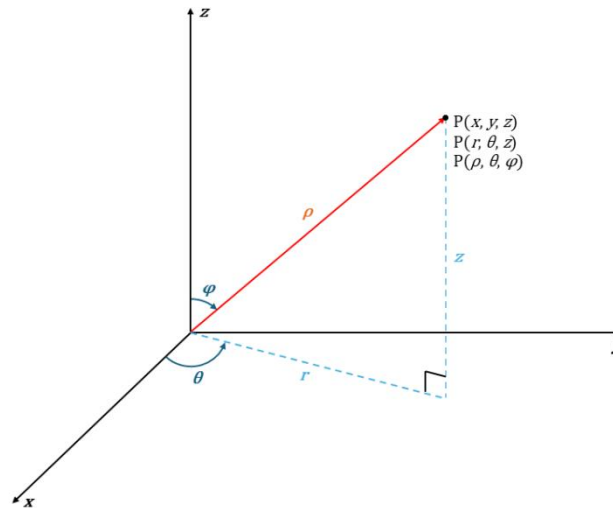


Figure 1. A visual representation of the coordinate systems

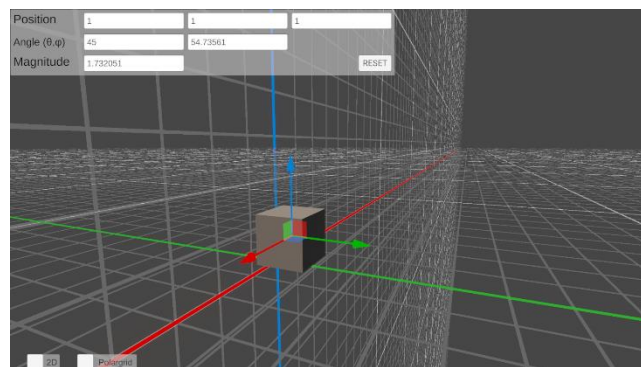


Figure 2. The object coordinates simulation interface

3) Kinematic Simulation

The kinematic motion is done using the midpoint Euler integration method, which calculates variations in position, velocity, and acceleration over discrete time steps at each frame. The Euler midpoint method follows these equations:

$$\Delta X = \Delta t \cdot f(X, t) \quad (7)$$

$$f_{mid} = f\left(\frac{X + \Delta X}{2}, \frac{t + \Delta t}{2}\right) \quad (8)$$

$$X(t + \Delta t) = X(t) + \Delta t \cdot f_{mid} \quad (9)$$

where: $\mathbf{X}$ is the kinematic vector values, Δt is the delta time between each frame, and $f(\mathbf{X}, t)$ is the rate of change function. The kinematic values change on each frame in accord to the Euler midpoint method, using the Equation 7, 8 and 9. The Euler midpoint method is a second-order Runge-Kutta method, having a truncation error proportional to the square of the time step. This results in better accuracy compared to the basic Euler method which is first-order (Burden and Faires, 2011; Butcher, 2008; Süli and Mayers, 2003). Unity's built-in frame time calculations and update functions facilitated implementation.

These kinematic parameters are displayed as x, y vector component values, magnitude values, and graphically represented within the coordinate space as an object and arrows. Users can directly modify these values or apply conditional instruction sets, which operate sequentially or independently based on the specified conditions. They can also be specified to trigger once or indefinitely. Instruction sets can be programmed to execute changes in kinematic parameters or timescales. For instance, setting the timescale to zero halts the simulation time. The execution can be between setting a value, adding a value, or multiplying a value. Additionally, extra utilities such as moving the coordinate grid relative to the object and tracking the object are available. These utilities are particularly useful for visualizing velocity and acceleration vectors. The kinematic simulation interface is illustrated in Figure 3.

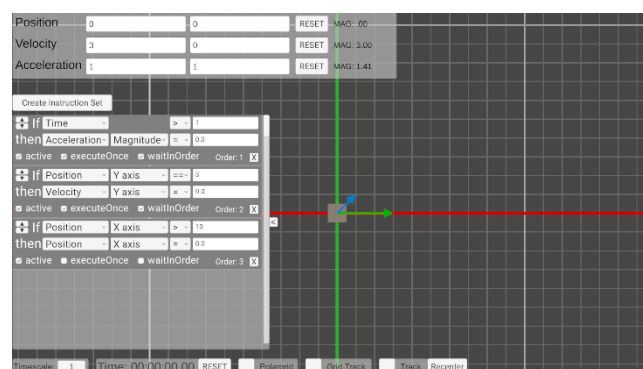


Figure 3. The kinematic simulation interface

4) DH Matrix Simulation

The DH matrix simulation utilizes Unity's object transform hierarchy system, which enables an object to be assigned as a child of another, linking the movement of the parent to its child. This approach serves as a substitute for performing forward kinematics calculations based on the modified DH table. The modified version is preferred over the original due to its intuitive nature, as changes to its parameters have a more direct and comprehensible effect on each link (Singh et al., 2021; Reddy, 2014). Both the original and modified DH representations are widely used and interchangeable (Granja et al., 2016; Wang et al., 2014; Corke, 2007). Through real-time visualization, students can directly observe how changes in Denavit–Hartenberg (DH) parameters affect the physical orientation and position of the robot manipulator. The simulation explicitly displays the origin and axis orientation of each joint coordinate frame, helping learners visualize rotational axes and translational directions that are otherwise difficult to interpret from mathematical equations alone. By interactively modifying parameters such as joint angle, link offset, link length, and twist angle, students can immediately observe corresponding changes in robot posture, end-effector position, and coordinate frame transformations in three-dimensional space. This real-time interaction helps bridge the gap between abstract kinematic formulations and observable robotic motion. Equation 10 and 11 define the modified DH homogeneous transformation equation and matrix, while Equation 12 and 13 represent the standard DH transformation equation and matrix. The equations are given as follows:

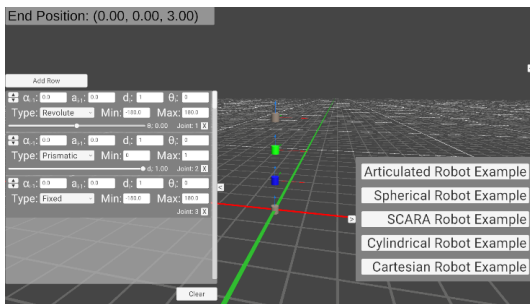
$${}^{i-1}T_i = R(x_{i-1}, \alpha_{i-1})T(x_{i-1}, a_{i-1})R(z_i, \theta_i)T(z_i, d_i) \quad (10)$$

$${}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) & 0 & a_{i-1} \\ \sin(\theta_i)\cos(\alpha_{i-1}) & \cos(\theta_i)\cos(\alpha_{i-1}) & -\sin(\alpha_{i-1}) & -d_i\sin(\alpha_{i-1}) \\ \sin(\theta_i)\sin(\alpha_{i-1}) & \cos(\theta_i)\sin(\alpha_{i-1}) & \cos(\alpha_{i-1}) & d_i\cos(\alpha_{i-1}) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (11)$$

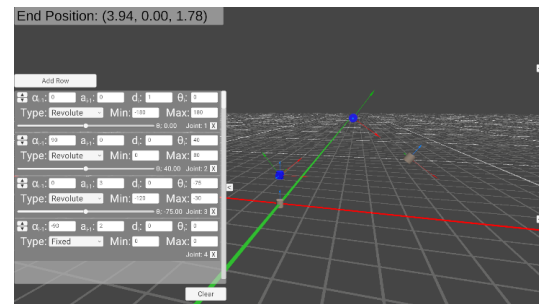
$${}^{i-1}T_i = R(z_{i-1}, \theta_i)T(z_{i-1}, d_i)T(x_i, a_i)R(x_i, \alpha_i) \quad (12)$$

$${}^{i-1}T_i = \begin{bmatrix} \cos(\theta_i) & -\cos(\alpha_i)\sin(\theta_i) & \sin(\alpha_i)\sin(\theta_i) & a_i\cos(\theta_i) \\ \sin(\theta_i) & \cos(\alpha_i)\cos(\theta_i) & -\sin(\alpha_i)\cos(\theta_i) & a_i\sin(\theta_i) \\ 0 & \sin(\alpha_i) & \cos(\alpha_i)\sin(\alpha_{i-1}) & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (13)$$

where: $R(v, n)$ is the function of rotation along v-axis of n value; $T(v, n)$ is the function of translation along v-axis of n value; and $\theta_i, d_i, a_i, \alpha_i$ are the DH parameters of i link. Each row in the simulation interface contains modified DH parameters, joint type, threshold values, and an adjustable slider. The parameters d and θ can be modified directly like others or additionally via the slider, depending on the joint type. Additionally, predefined configurations are available for common robotic arms. The final position of the robot's end effector is computed and updated in real-time by the game engine. By leveraging the transform hierarchy system, the simulation process is streamlined, as it eliminates the need for direct calculations using the modified DH transformation matrix (see Equation 12). This system allows for parent-child relationships between objects, applying rotations and translations in both local and global coordinate spaces. The local coordinate system is relative to the parent object's rotation within the hierarchy. This enables the modified DH transformation equation (Equation 10) to be replicated by applying corresponding local transformations. Figure 4 presents the Denavit–Hartenberg (DH) table simulation interface (a) and preset robot configurations. The interface allows users to modify DH parameters in real time and observe the resulting robot posture and coordinate frame transformations. Preset configurations include (b) an articulated robot with three revolute joints (RRR), (c) a spherical robot with two revolute joints and one prismatic joint (RRP), (d) a SCARA robot with two revolute joints and one prismatic joint arranged for planar motion, (e) a cylindrical robot consisting of one revolute joint and two prismatic joints (RPP), and (f) a cartesian robot composed of three orthogonal prismatic joints (PPP).



(a)



(b)

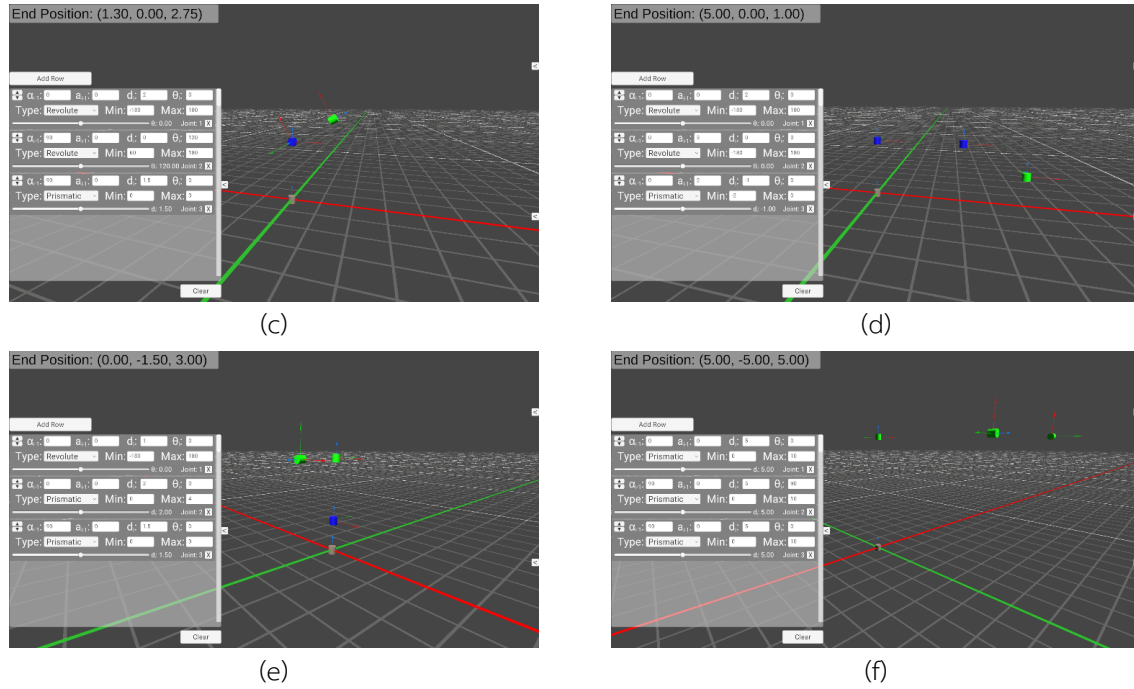


Figure 4. (a) The Denavit-Hartenberg table simulation interface; and the preset configurations: (b) articulated; (c) spherical; (d) SCARA; (e) cylindrical; (f) cartesian

For the inverse kinematics (IK), The system supports two IK solving methods: Jacobian Transpose and Damped Least Squares (DLS). These methods operate iteratively and are implemented using Unity's update loop which recalculates the joint angle adjustments at each frame based on the positional error between the current and desired end position. The Jacobian matrix J is calculated numerically at runtime using the current joint configuration (Haviland and Corke, 2020). For each joint i , the Jacobian column is constructed based on the joint type. For revolute joints, the linear and angular velocity components are defined as:

$$v_i = z_{i-1} \times (p_n - p_{i-1}) \quad (14)$$

$$\omega_i = z_{i-1} \quad (15)$$

where: z_{i-1} is the rotation axis of joint i in global coordinates, p_n is the position of the end effector, and p_{i-1} is the origin of the i^{th} joint's frame. For prismatic joints, the expressions are simplified as follows:

$$v_i = z_{i-1} \quad (16)$$

$$\omega_i = \vec{0} \quad (17)$$

The complete Jacobian matrix J is assembled column-wise from these vectors:

$$J = \begin{bmatrix} v_1 & v_2 & \cdots & v_n \\ \omega_1 & \omega_2 & \cdots & \omega_n \end{bmatrix} \quad (18)$$

The Jacobian Transpose method offers a simple and computationally inexpensive approach by approximating the solution through multiplication of the Jacobian transpose and the error vector (Fiala and Walvering, 1990). The update rule follows the equation:

$$\Delta\theta = \alpha J^T e \quad (19)$$

where: $\Delta\theta$ represents the incremental change in joint angles, α is a small scalar gain controlling convergence speed, J^T is the transpose of the Jacobian matrix, and e is the error vector between the current and target end effector positions. This method is suitable for small adjustments and performs well under most configurations but may exhibit instability or slow convergence near singularities or poorly conditioned Jacobians. Trading simplicity for precision, the DLS method introduces matrix inverse and damping term that stabilizes the solution, particularly in configurations near singularities (Deo and Walker, 1995; Nakamura and Hanafusa, 1986; Wampler, 1986). The DLS method is commonly represented by the following equation:

$$\Delta\theta = (J^T J + \lambda^2 I)^{-1} J^T e \quad (20)$$

where: λ is the damping factor, and I is the identity matrix. Unlike the standard formulation that uses $J J^T$, the simulation employs $J^T J$ due to the low degrees of freedom in the teaching examples and the increased computational efficiency of this form in such contexts (Buss, 2004). Additionally, the use of $J^T J$ allows for more lenient damping, resulting in smoother convergence behaviour (Buss, 2004). Since Unity does not provide built-in matrix inverse functions for arbitrary matrices, a custom inverse function was implemented using LU decomposition (Fiala and Walvering, 1990). The simulation allows real-time switching between the two IK solving modes via the interface. Adjustable parameters such as learning rate, damping factor, and step size are provided for educational experimentation. When an end effector target is set, the solver iteratively calculates $\Delta\theta$ and applies it to each joint's transform using Unity's local rotation and translation hierarchy system (Deo and Walker, 1995; Whitney, 1969). This process continues until the end effector reaches the desired target within a predefined threshold. Figure 5 shows the interface of the IK simulation.

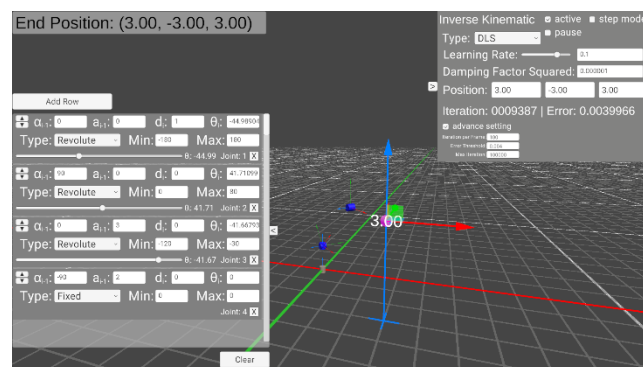


Figure 5. The inverse kinematic simulation

5) Proportional-Integral-Derivative Control Simulation

The PID control simulation demonstrates the application of PID control algorithms over a configurable target using the control system from section 2.1.3, Kinematic Simulation. The PID controller continuously calculates an error value as the difference between the current position of the target and the current position of the PID-controlled object, adjusting the control input accordingly (Åström and Murray, 2008). The PID control equation is given by:

$$u(t) = K_p e(t) + K_i \int_0^t e(t) dt + K_d \frac{de(t)}{dt} \quad (21)$$

where: $u(t)$ is the control output, $e(t)$ is the error at time t , K_p is the proportional gain, K_i is the integral gain, and K_d is the derivative gain. The general independent PID equation is chosen because the three values are independent, allowing for better observation of each value's effects as shown in Table 3 (Ang, Chong and Li, 2005). The table demonstrates the relationship between PID gain parameters and system response characteristics. Increasing the proportional gain (K_p) generally improves responsiveness by reducing rise time, but excessive values may reduce system stability. Increasing the integral gain (K_i) helps reduce steady-state error but may increase overshoot and settling time. Meanwhile, increasing the derivative gain (K_d) improves damping behaviour and system stability by reducing overshoot and oscillation. These relationships allow students to experimentally observe the trade-offs involved in PID controller tuning.

Table 3.

Effects of independent P, I, and D tuning.

Closed-Loop Response	Rise Time	Overshoot	Settling Time	Steady-State Error	Stability
Increasing K_p	Decrease	Small Increase	Small Increase	Decrease	Degrade
Increasing K_i	Small Decrease	Increase	Increase	Large Decrease	Degrade
Increasing K_d	Small Decrease	Decrease	Decrease	Minor Change	Improve

By guiding the students to tune the PID through trial and error as well as through analytical methods, they develop an understanding of the effects of each of the three values. Figure 6 shows the interface of the PID simulation.

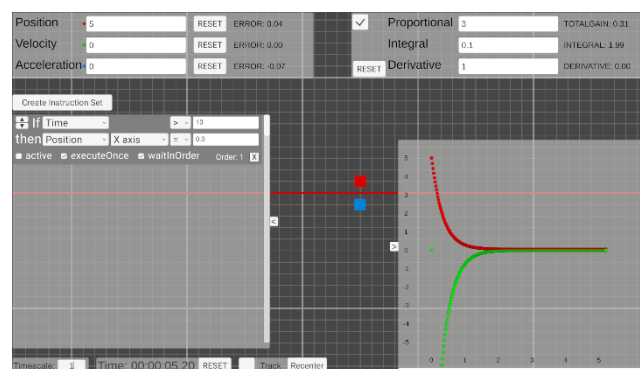


Figure 6. The PID simulation interface

Participants

The participants are students of a robotic kinematic class and are divided into two groups: a control group and an experimental group. The control group engages with traditional learning materials only, including a lecture book, presentation slides, and MATLAB exercises. In contrast, the experimental group receives the same instructional content, with the addition of the simulation program. The traditional learning content of this class covers DH conventions, forward kinematics, and inverse kinematics with analytic methods and Jacobian methods.

The study involved two distinct participant groups: a control group and an experimental group. As shown in Table 4, the control group totals 31 members (22 males and 9 females) with a mean age of 20 years ($SD = 1.31$, $Min = 18$, $Max = 22$). While the experimental group totals 33 members (25 males and 8 females) with a mean age of 20 years ($SD = 0.83$, $Min = 18$, $Max = 21$).

Table 4.

Demographic characteristics of the 2 groups of participants.

	Control Group (n=31)	Experimental Group (n=33)
Age (yrs) Mean $\pm$ SD	20 $\pm$ 1.31	20 $\pm$ 0.83
Gender (male/female)	22/9	25/8

No statistically significant difference in age was found between the two groups (Wilcoxon's $W = 466.0$, $p = 0.55$) as a negligible effect size was measured by the rank biserial correlation ($r = 0.09$, 95% CI ≈ -0.13 to 0.17). For gender distribution, the analysis yielded no significant difference in gender ratios between the groups either ($\chi^2 = 0.02$, $df = 1$, $p = 0.88$), with a very small effect size (Cramer's $V = 0.019$, 95% CI ≈ -0.13 to 0.17).

Instruments and Data Collection

The evaluation of the implemented simulations focuses on the performance analysis and user engagement. The effectiveness of the simulations is assessed using both quantitative and qualitative methods, including IMMS score and user feedback.

1) Performance

The performance of the simulations is analysed using Unity's built-in profiling tools, which provide insights into frame rate stability, computational efficiency, and resource utilization. Key performance indicators such as frame rate, CPU usage, memory allocation, and storage size are monitored to ensure the simulations run smoothly across various hardware configurations and loads.

2) Instructional Materials Motivation Survey

To evaluate the educational effectiveness and user engagement of the simulations, the IMMS is conducted among participants. The IMMS survey is a validated tool, derived from the ARCS construct, that

measures motivation and engagement levels with instructional materials (Keller, 2010). The survey consists of 36 items assessing four key dimensions:

- i) Attention (12 items): Measures how well the simulations capture and maintain user interest.
- ii) Relevance (9 items): Assesses whether the simulations are perceived as useful and applicable to the users' learning goals.
- iii) Confidence (9 items): Evaluates the extent to which users feel they can successfully understand and apply the concepts demonstrated in the simulations.
- iv) Satisfaction (6 items): Gauges user enjoyment and overall satisfaction with the learning experience.

Participants provide feedback using a 5-point Likert scale, ranging from strongly disagree to strongly agree. As the instructional material in this context is the simulation program, the wording of the IMMS survey was adjusted accordingly while retaining the meaning.

3) User Feedback

User feedback was collected from the experimental group after completing the simulation-based learning activity. Participants were asked to provide open-ended responses regarding their experience with the simulation software, via the question "In what ways did this software enhance or hinder your learning experience? Please provide specific examples of features that were helpful or challenging and any suggestions for improvement." Feedback was analyzed qualitatively to identify common themes and areas for improvement.

Analysis of Data

Psychometric reliability was assessed using Cronbach's alpha for each ARCS subscale in both groups. The Shapiro-Wilk test assessed normality of score distributions to verify assumptions for parametric analysis. Independent samples t-tests compared mean scores between control and experimental groups, with statistical significance set at $\alpha = 0.05$. Effect sizes were calculated using Cohen's d with 95% confidence intervals, interpreted as small ($d \approx 0.2$), medium ($d \approx 0.5$), or large ($d \approx 0.8$). Qualitative data from open-ended feedback underwent thematic analysis to identify patterns in visualization effectiveness, interface usability, and learning impact.

■ Results and Discussion

Results

This section presents the outcomes of the simulation's performance evaluation and its educational effectiveness based on the demographic analysis, internal consistency of the survey tool, and motivational responses measured by the IMMS. Results are organized into five subsections: software performance,

participant demographics, psychometric reliability, descriptive statistical results, and user feedback summary.

1) Software Performance

The software's performance was assessed using Unity's built-in profiling tool, focusing on frame rate, CPU usage, memory allocation, and storage size. Across all simulation modules, the application maintained smooth real-time interaction, averaging 200 frames per second in general use and 60 frames per second during inverse kinematics. CPU usage remained efficient, with frame times rarely exceeding 16 milliseconds, even under computational load on the Intel Core i7-9700. Memory usage was stable at 1.35 GB, and the final build was lightweight, requiring only 86.5 MB of disk space. The high frame rate and stable frame timing indicate that the simulation is capable of maintaining smooth real-time interaction even during computationally intensive inverse kinematics calculations. This responsiveness is important in educational environments because delayed feedback may negatively affect learner engagement and conceptual understanding. The relatively low memory consumption and compact storage size also suggest that the software can be deployed on standard educational computers without requiring specialized hardware. Figure 7 shows the result from the built-in profiling tool.

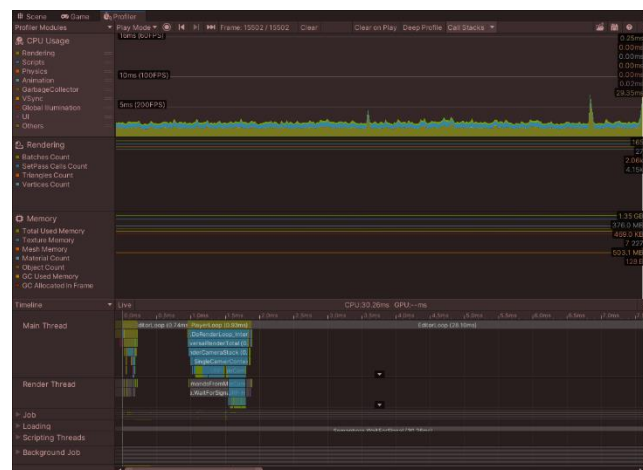


Figure 7. The built-in profiling tool result

2) Reliability

The internal consistency was assessed separately for the control and experimental groups. The results are summarized in Tables 5 and 6. In the control group, Cronbach's alpha was highest for Attention ($\alpha = 0.754$) and Relevance ($\alpha = 0.675$), indicating acceptable reliability. Confidence ($\alpha = 0.606$) and Satisfaction ($\alpha = 0.555$) showed moderate to low reliability. In the experimental group, alpha values were generally higher: Attention ($\alpha = 0.764$), Relevance ($\alpha = 0.711$), Confidence ($\alpha = 0.675$), and Satisfaction ($\alpha = 0.682$), suggesting improved internal consistency.

Table 5.

Cronbach's alpha values of the control group's IMMS results.

Coefficient α	Estimate	Std. Error	95% CI	
			Lower	Upper
Attention	0.754	0.065	0.627	0.881
Relevance	0.675	0.102	0.475	0.875
Confidence	0.606	0.094	0.423	0.79
Satisfaction	0.555	0.158	0.245	0.865

While these results did not fully reach the high reliability levels reported in prior studies, they remain within acceptable bounds for exploratory educational research. The comparison of Cronbach's alpha values between the control and experimental groups reveals a noticeable improvement in the internal consistency of the IMMS subscales in the experimental group. All four motivational dimensions: Attention, Relevance, Confidence, and Satisfaction, showed increased reliability by 1.33%, 5.33%, 11.39%, and 22.88% respectively.

Table 6.

Cronbach's alpha values of the experimental group's IMMS results.

Coefficient α	Estimate	Std. Error	95% CI	
			Lower	Upper
Attention	0.764	0.07	0.626	0.902
Relevance	0.711	0.1	0.514	0.907
Confidence	0.675	0.08	0.518	0.832
Satisfaction	0.682	0.149	0.39	0.974

3) Descriptive Results

Descriptive statistics for each ARCS motivation subscale: Attention, Relevance, Confidence, and Satisfaction, were calculated separately for the control and experimental groups to evaluate the impact of the simulation-based learning tool on learner motivation (see Table 7 and 8).

Table 7.

ARCS scores of the control group's IMMS results.

	Mean	Std. Deviation	Shapiro-Wilk	P-value of Shapiro-Wilk	Minimum	Maximum
Attention	3.336	0.521	0.975	0.68	2.17	4.33
Relevance	3.846	0.493	0.978	0.764	2.67	4.89
Confidence	3.44	0.475	0.962	0.337	2.67	4.44

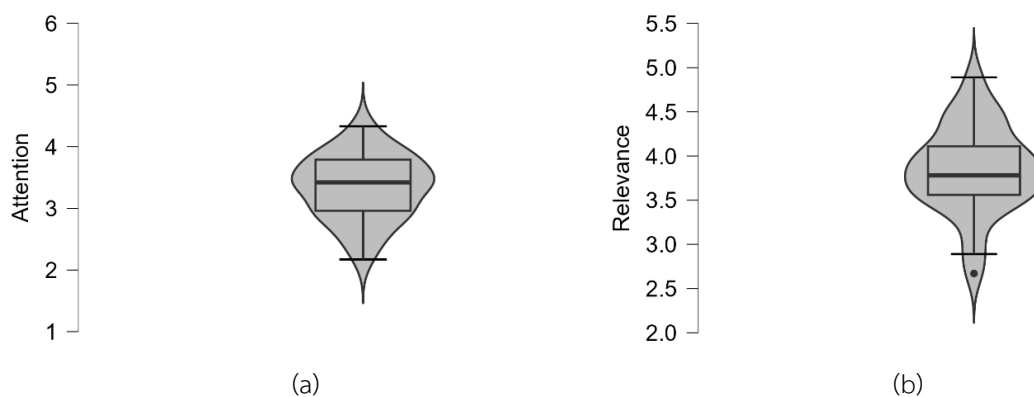
	Mean	Std. Deviation	Shapiro-Wilk	P-value of Shapiro-Wilk	Minimum	Maximum
Satisfaction	3.554	0.514	0.926	0.035	1.83	4.5

Table 8.

ARCS scores of the experimental group's IMMS results.

	Mean	Std. Deviation	Shapiro-Wilk	P-value of Shapiro-Wilk	Minimum	Maximum
Attention	3.485	0.535	0.983	0.883	2.42	4.83
Relevance	3.802	0.496	0.969	0.46	2.67	4.89
Confidence	3.448	0.506	0.946	0.103	2.67	4.56
Satisfaction	3.813	0.559	0.912	0.011	1.83	4.83

Box-violin plots of each ARCS motivation subscale distribution for the control and experimental groups are also shown in Figure 8 and 9 respectively. In the control group, Relevance recorded the highest average score ($M = 3.846$, $SD = 0.493$), followed by Satisfaction ($M = 3.554$, $SD = 0.514$), Attention ($M = 3.336$, $SD = 0.521$), and Confidence ($M = 3.440$, $SD = 0.475$). In contrast, the experimental group showed an overall increase in mean scores across most dimensions, with Satisfaction reaching the highest level ($M = 3.813$, $SD = 0.559$), followed by Relevance ($M = 3.802$, $SD = 0.496$), Attention ($M = 3.485$, $SD = 0.535$), and Confidence ($M = 3.448$, $SD = 0.506$). All four motivational dimensions: Attention, Relevance, Confidence, and Satisfaction, showed changes in mean scores of 4.46% increase, 1.15% decrease, 0.23% increase, and 7.29% increase respectively from the control to the experimental group. Although the difference in Relevance and Confidence scores between the groups was minimal, the increase in Satisfaction and Attention in the experimental group was more pronounced.



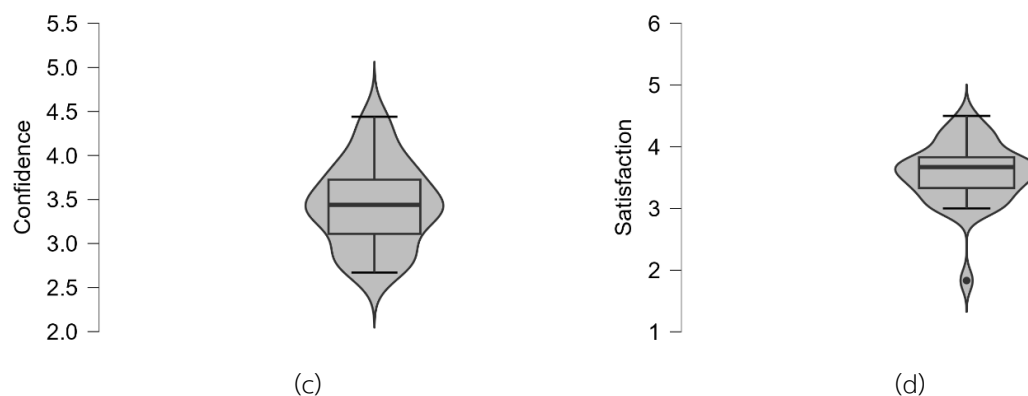


Figure 8. Box-violin plots of each ARCS motivation subscale distribution for the control group: (a) Attention; (b) Relevance; (c) Confidence; (d) Satisfaction

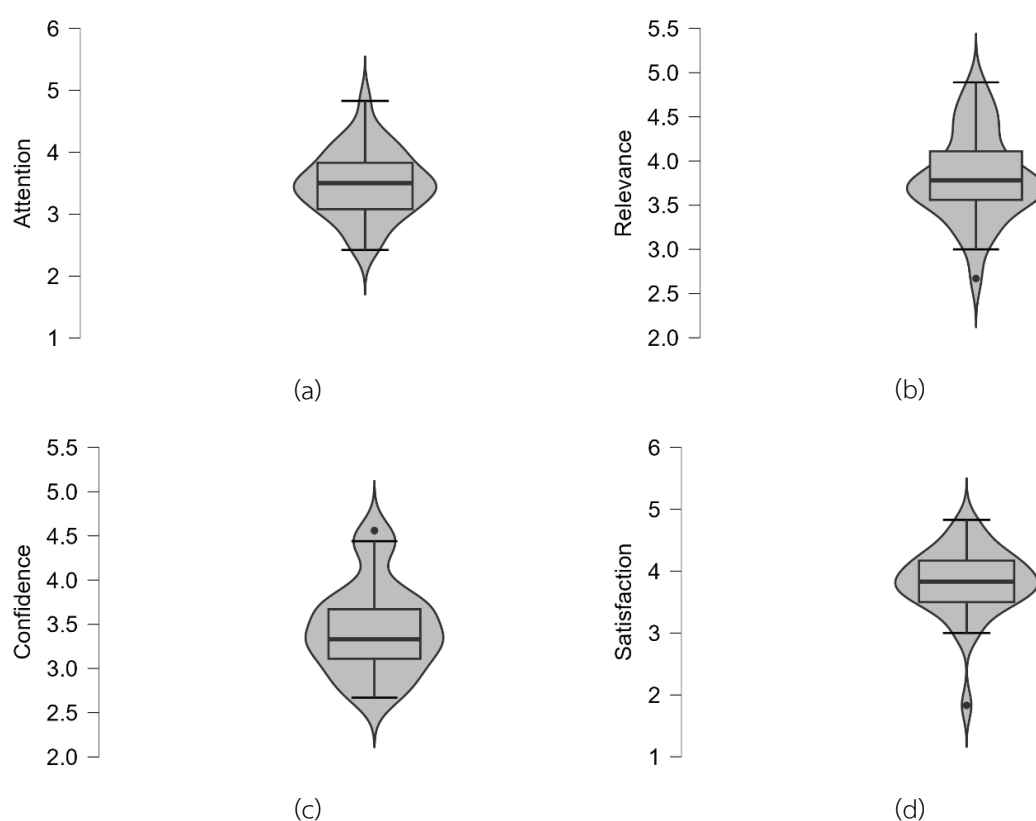


Figure 9. Box-violin plots of each ARCS motivation subscale distribution for the experimental group: (a) Attention; (b) Relevance; (c) Confidence; (d) Satisfaction

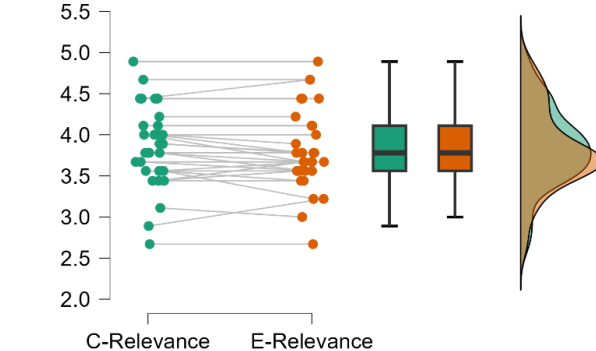
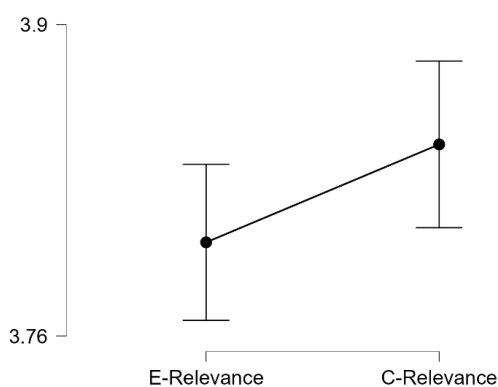
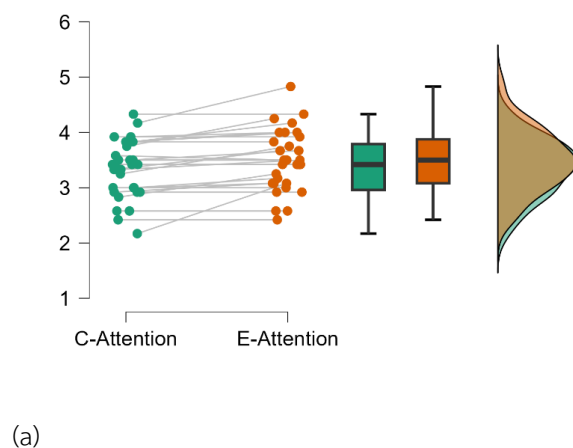
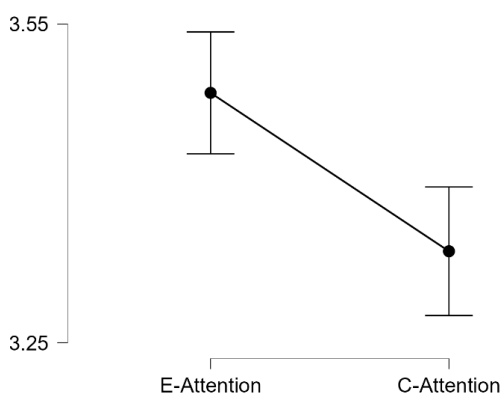
Normality of the score distributions was assessed using the Shapiro-Wilk test. All subscales in both groups satisfied the normality assumption ($p > .05$), except for Satisfaction in both groups ($p = .035$ for control; $p = .011$ for experimental), which showed mild deviation. This supported the use of parametric tests for group comparisons, with appropriate caution regarding the Satisfaction subscale. An independent samples t-test was conducted to compare the mean scores between the two groups (see Table 9).

Table 9.

Paired student's *t*-test between the control and experimental groups' IMMS results.

Experimental Group		Control Group	<i>t</i>	df	<i>p</i>	Cohen's <i>d</i>	SE Cohen's <i>d</i>	95% CI for Cohen's <i>d</i>	
								Lower	Upper
E-Attention	-	C-Attention	3.668	30	< .001	0.659	0.085	0.327	∞
E-Relevance	-	C-Relevance	-1.083	30	0.856	-0.194	0.053	-0.491	∞
E-Confidence	-	C-Confidence	0.697	30	0.245	0.125	0.051	-0.172	∞
E-Satisfaction	-	C-Satisfaction	4.168	30	< .001	0.749	0.127	0.408	∞

Descriptive and raincloud plots of each ARCS motivation subscale are also shown in Figure 10. Statistically significant differences were observed in both the Attention ($t(30) = 3.668, p < .001$, Cohen's $d = 0.659$) and Satisfaction ($t(30) = 4.168, p < .001$, Cohen's $d = 0.749$) subscales, indicating medium to large effect sizes. No statistically significant differences were found for Relevance ($t(30) = -1.083, p = 0.856$) or Confidence ($t(30) = 0.697, p = 0.245$), although Confidence was slightly higher, and Relevance was moderately lower in the experimental group.



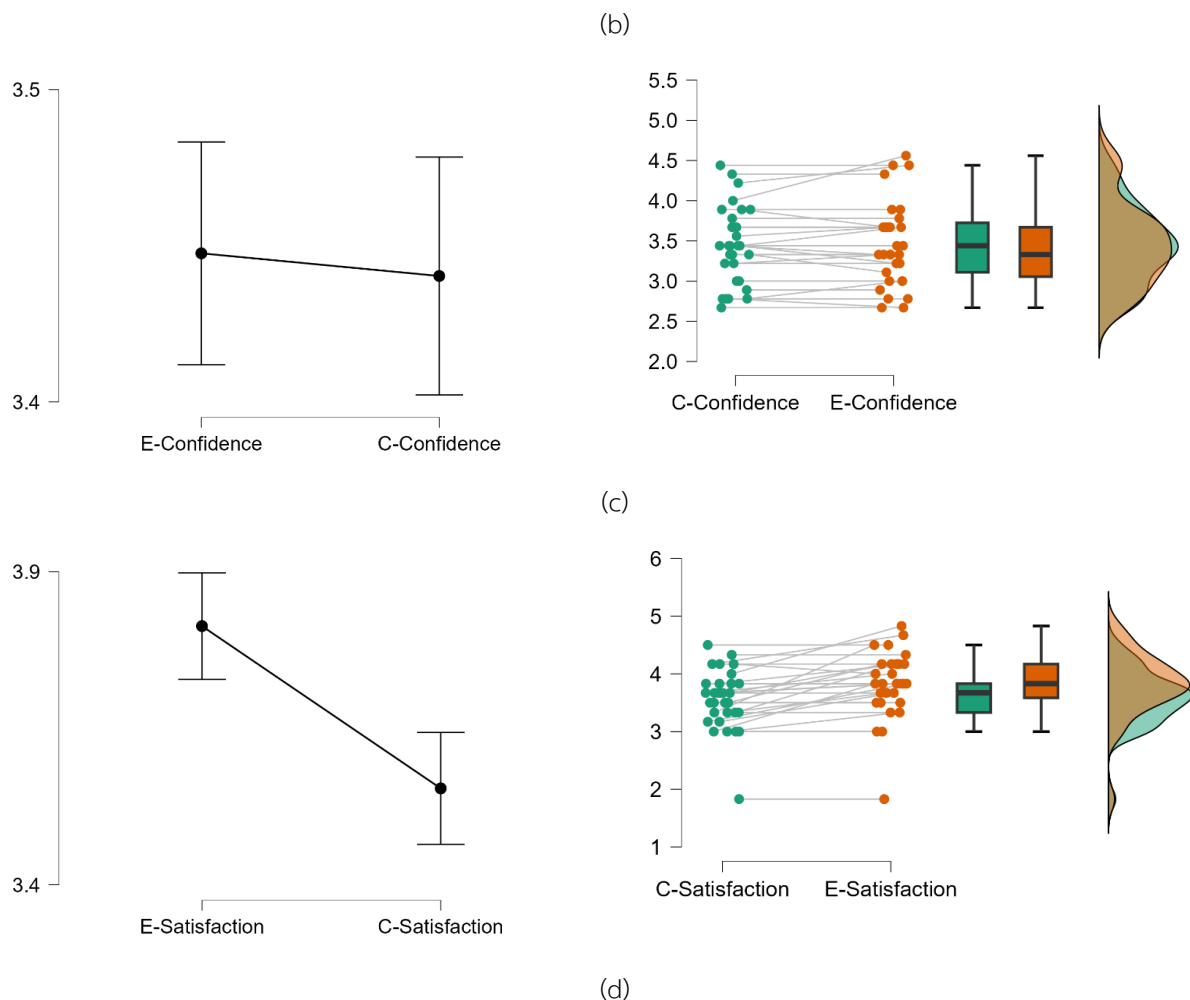


Figure 10. Descriptive and raincloud plots of each ARCS motivation subscale: (a) attention; (b) relevance; (c) confidence; (d) satisfaction

4) User Feedback Summary

Feedback from the experimental group, shown in Table 10, highlighted several key strengths and areas for improvement in the simulation tool. Most participants emphasized that the software significantly enhanced their understanding of robotic kinematics by offering clear, real-time visualizations of complex concepts, particularly in coordinate systems, Denavit–Hartenberg (DH) parameters, and PID control. The ability to input parameters and instantly observe changes helped bridge the gap between abstract equations and real-world application. Many noted that the simulation made it easier to visualize spatial transformations and joint movements, especially for learners who struggle with imaginative outlook. Participants also appreciated the tool’s convenience, allowing them to explore concepts without writing code, as well as its potential to replace expensive physical hardware for basic kinematic experimentation. Several comments expressed enthusiasm for using the tool in classroom instruction, highlighting its value in visual reinforcement of theoretical content. However, users also reported notable limitations. The most common concern was the user interface (UI): while generally functional and navigable, some found it overly large or visually cluttered. Issues such as having to manage multiple windows within modules (especially in

the DH and PID sections), the lack of quick reset or view recentring functions, and absence of intuitive exit shortcuts were frequently mentioned. Others noted the lack of guided tutorials, tooltips, or embedded explanations, which made it difficult to follow certain derivations without external help. Additionally, a few users suggested enhancing the visual design of robotic arms, currently represented as generic cylinders, to improve realism and engagement.

Table 10.

Student responses on the open feedback question.

	In what ways did this software enhance or hinder your learning experience? Please provide specific examples of features that were helpful or challenging and any suggestions for improvement.
Student 1	I liked the visualization of objects moving in 3D space, as I find it difficult to imagine such scenarios on my own.
Student 2	It really helped me understand mechanics without needing to visualize everything in my head.
Student 3	The software provided a valuable learning experience in understanding robot kinematics through interactive visualizations and step-by-step calculations. The simulation tools were particularly useful for demonstrating forward and inverse kinematics, making complex concepts easier to grasp. Additionally, the ability to input different link parameters and view real-time transformations helped reinforce theoretical concepts with practical application. However, there were some challenges. The user interface could be more intuitive, as navigating between modules felt cumbersome at times. Also, the lack of detailed explanations for certain derivations made it difficult to follow without external resources. Including more guided tutorials or tooltips explaining key equations would be beneficial. Overall, the software was useful for visualizing and experimenting with robot kinematics, but improvements in UI design and educational support would enhance the learning experience further.
Student 4	For me, not yet—but in my opinion, I would like the design to be more attractive. It would also be great if it could import code from MATLAB.
Student 5	The software was interesting. You don't need to write code—you can just input numbers, which is very convenient. You can also see the robot's movement in 3D. I think you should improve the visual design.
Student 6	It helped me to understand and visualize the objects better.

Student 7	The first issue I encountered was that Windows warned me about the program being potentially unsafe. This could be improved, especially if the software is intended for educational use. That said, the program installed quickly and didn't require much memory, which is good. The main menu is simple and not difficult to navigate. The simulation works, but it's still hard to interpret what I'm seeing. The UI feels oversized and could benefit from scaling down. That's all the feedback I have.
Student 8	In my opinion, the DH simulation function is very helpful for learning kinematics, as it allows simulation of various robotic arms. It helps me learn visually by filling in all DH table variables—such as a , α , d , θ —and then viewing the position of every joint. This visual representation makes it easier to understand. For improvement, I suggest adding visible links between the robot arms in the simulation to better convey the robot's posture.
Student 9	Instead of relying solely on calculations, it was interesting to use this software because it's graphical and easy to use. Sometimes we make calculation errors, so this was very helpful. The DH simulation part was especially interesting. The software is good but could be further developed.
Student 10	This software helped me learn and understand simulation better, but the settings were a bit difficult to use at first.
Student 11	The program is very good. It helps visualize the robot and coordinate systems well. Entering parameters is easy. However, the UI could be improved—closing and reopening windows in the DH and PID sections requires navigating through 2–3 windows. Combining them into one would be better.
Student 12	The visualization is good, parameters are easy to enter, and the UI is simple to navigate. However, having to repeatedly close and reopen the 3 UI windows in DH simulation is annoying during long sessions.
Student 13	The simulation app really helped me understand how the robot arm works. Free body diagrams only show joints as blocks or cylinders, but the simulation displays an actual robot, which makes it easier to understand.
Student 14	It helped me visualize concepts from class, such as X and Y axes, which can be difficult to understand when moving a robot arm. The app could be improved with better graphic design and shortcuts to tools. Adding a realistic robot arm model would also be helpful.
Student 15	It helped me visualize what I learned in class, especially the DH part. It shows how each type of robot works with DH parameters. However, the robot model looks boring and

doesn't help much with understanding. A realistic robot model would be better than the cylindrical link.

Student 16 It would be more exciting if the simulation showed a robot arm instead of just a cylinder. The UI isn't very interesting, and sometimes when I select a position, the object flies away and I can't find it. A reset view button would help solve this problem. Still, the simulation helped me understand kinematics better.

Student 17 The software simulates concepts that are hard to imagine. For example, it helped me understand twist angles and object positioning. I suggest improving the PID section, as it's hard to adjust and calibrate the values.

Student 18 The interactive simulations effectively demonstrated kinematics concepts. However, the steep learning curve and limited tutorials were challenging. Real-time data monitoring helped me better understand robot movements. Hardware compatibility was a drawback. More tutorials and troubleshooting guides would improve the experience.

Student 19 I work on many projects that require kinematic knowledge. It's hard to do everything by calculation because of factors like friction and inertia. This software made my work easier and more efficient.

Student 20 Overall, it's good. The coordinate simulation helps understand positions, and the kinematics simulation offers many robot types. The PID tool was confusing, so that part should be improved.

Student 21 This program offers great visualization. I liked being able to adjust parameters and see the changes instantly—it's satisfying as a kinematics and dynamics student, especially with the MD simulation. I would like a dedicated exit button for simulation sections and the ability to adjust the camera vertically (maybe it exists, but I couldn't find it). Overall, I really approve of this program and can see Ajarn Narong using it for teaching future students.

Student 22 This software greatly enhanced my understanding by providing visual simulations of complex kinematics concepts. Coordinate and Kinematics Simulations helped me understand transformations. DH Simulation clarified parameter effects, though the UI was a bit complex. PID Simulation made tuning easier but required trial and error. Overall, this is a good software. I don't have any suggestions for improvement.

Student 23 It helped reinforce what I learned in class. I felt like I was using a real ABB robot without actually needing one. One downside is the absence of an ESC button to exit simulations, but overall, I liked the program.

Student 24	This program really helped me with visualization. Sometimes, I struggle to picture how a robot moves, especially with PID control. This made it easier to understand.
Student 25	The simulation and UI are easy to understand, and they helped me quickly learn about kinematics
Student 26	The UI is very flashy. This could be a pro or a con, depending on your perspective.
Student 27	It made my learning experience easier and kept me more engaged during lessons.
Student 28	I didn't encounter any errors using this software. It provided a smooth learning experience, and the simulations were clear and effective in enhancing my understanding.

*Note: All responses have been edited for grammar, clarity, and readability, while preserving the original intent of each student. Not all participants in the experimental group provided feedback.

Discussion

Early findings suggest a positive correlation between simulation use and student motivation (Teixeira, 2015; Caccavelli et al., 2011). Students reported an enhanced understanding of kinematic principles and increased interest in robotics as a result of hands-on engagement with the simulations. From a performance perspective, the Unity-based simulation platform demonstrated efficient use of system resources and consistently high frame rates, indicating that it is lightweight and suitable for use on most standard educational devices. This ensures that accessibility and hardware limitations are unlikely to interfere with learning outcomes.

In terms of educational impact, the use of the simulation tool has a meaningful effect on learners' motivational responses, particularly in the dimensions of Attention and Satisfaction. Statistically significant improvements were observed in both subscales ($p < .001$), with medium to large effect sizes (Cohen's $d = 0.659$ and 0.749 , respectively). The increase in Attention suggests that the visual and interactive aspects of the simulation successfully engaged learners more effectively than traditional methods. Given the often abstract and mathematical nature of robotic kinematics, the simulation's real-time feedback and manipulable interface likely helped maintain learner interest throughout the activity. Similarly, the increase in Satisfaction scores among the experimental group supports the notion that learners appreciated the inclusion of dynamic, game-like features in their learning experience. Despite the subscale's modest internal consistency ($\alpha = 0.555$ in control; $\alpha = 0.682$ in experimental), the increase is still meaningful. The lower alpha may be attributed to the limited number of items used to measure satisfaction, as well as possible variation in how students interpret subjective enjoyment in relation to technical content. Nonetheless, the statistically significant gain in Satisfaction (Cohen's $d = 0.749$) suggests that students found the simulation rewarding and enjoyable, which is a critical factor in sustained engagement with STEM content. The Relevance subscale showed a small, non-significant

decrease in the experimental group. This may be explained by the fact that while the simulation addressed coordinate frames, inverse kinematics, and feedback, these elements were not explicitly contextualized within the broader robotics curriculum. In other words, the simulation may have failed to clearly connect with students' perceptions of what is useful or applicable to real-world robotic systems. Furthermore, Relevance typically depends on learners' prior understanding of how content relates to their goals, something that may not be easily conveyed without integrated reflection or discussion during the lesson. Confidence scores showed a slight increase in the experimental group, but the difference was not statistically significant. This may be due to a combination of factors: while the simulation offered intuitive feedback and visual clarity, it did not provide formal problem-solving guidance or correctness verification mechanisms. Confidence is often built not only through practice, but through positive reinforcement and structured feedback, which were not primary features of the simulation. The subscale's moderate internal consistency ($\alpha = 0.606$ in control; $\alpha = 0.675$ in experimental) may also have limited sensitivity in detecting subtle differences.

Open-ended feedback revealed that students repeatedly cited improved conceptual clarity, especially for visualizing joint movement, coordinate transformations, and control responses, as one of the tool's major strengths. However, they also identified areas for future enhancement, including adding tooltips, integrating MATLAB compatibility, and refining visual design to better represent real robot arms instead of abstract geometries.

Altogether, these findings suggest that simulation-based learning, when implemented through an interactive and accessible platform, can meaningfully improve motivation and engagement in robotics education. The positive impact on Attention and Satisfaction highlights the simulation's effectiveness as a supplemental tool for concept visualization and learner-centred exploration. However, realizing its full potential in supporting Confidence and Relevance may require deeper curricular integration, more structured learning pathways, and UI improvements tailored to instructional clarity.

■ Conclusion

This study introduced and evaluated a Unity-based simulation tool designed to support the teaching of robotic kinematics through interactive, visual, and real-time experimentation. The tool integrates modules for coordinate systems, kinematics, Denavit–Hartenberg (DH) parameters, and PID control, with a focus on making abstract concepts more accessible to learners.

Performance profiling confirmed that the simulation runs efficiently on typical hardware, ensuring broad accessibility in classroom and self-directed learning environments. Educationally, the simulation produced meaningful motivational gains, particularly in the dimensions of Attention and Satisfaction, with statistically significant improvements and medium-to-large effect sizes (Cohen's $d = 0.659$ and 0.749 , respectively). These results suggest that the simulation successfully increased learner engagement and enjoyment, largely through its intuitive visualizations and interactive design. Confidence and Relevance,

while slightly improved or stable, did not show significant differences, likely due to the absence of feedback mechanisms, guided support, and direct curriculum alignment.

Student feedback reinforced these findings, praising the tool's ability to visualize transformations, demonstrate parameter effects, and simplify experimentation without requiring programming. At the same time, users highlighted opportunities for improvement, including streamlining the user interface, enhancing visual realism, and adding tutorials or tooltips for deeper understanding.

■ Limitations and Recommendations

Limitations

A limitation of this study was the moderate internal consistency in some IMMS subscales, possibly due to the limited number of items and the technical specificity of the content. Additionally, the lack of structured in-tool guidance may have constrained learning depth for certain students.

Recommendations

Future development should focus on improving instructional support through embedded tutorials, tooltips, automated feedback, and tighter integration with robotics curricula. Additional improvements in UI organization, navigation efficiency, visual realism, and interface scalability across different platforms and display resolutions may further enhance usability and learner engagement. Providing customizable UI layouts, adjustable interface scaling, and configurable hotkeys could also improve accessibility and user experience for different learning preferences and hardware environments. Features such as MATLAB code import and expanded robot models could further improve the educational value of the platform.

■ Acknowledgement

This study under King Mongkut's Institute of Technology Ladkrabang was funded by the Research and Development Department STEAM Kiddee Lab Skills Enhancement School, Bangkok, Thailand.

■ References

Alimisis, D., & Boulougaris, G. (2014). Robotics in physics education: fostering graphing abilities in kinematics. *In Proceedings of the 4th International Workshop Teaching Robotics, Teaching with Robotics*, Padova, Italy.

- Ang, K.H., Chong, G., & Li, Y. (2005). PID control system analysis, design, and technology. *IEEE Transactions on Control Systems Technology*, 13, 559–576.
- Arnay, R., Hernández-Aceituno, J., González, E., & Acosta, L. (2017). Teaching kinematics with interactive schematics and 3D models. *Computer Applications in Engineering Education*, 25, 420–429.
- Åström, K.J., & Murray, R.M. (2008). *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press: Princeton, NJ.
- Bligh, D.A. (2000). *What's The Use of Lectures?* (1st ed.). Jossey-Bass: San Francisco, CA.
- Burden, R.L., & Faires, J.D. (2011). *Numerical Analysis* (9th ed.). Brooks/Cole, Cengage Learning: Boston, MA.
- Buss, S.R. (2004). Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods. *IEEE Journal of Robotics and Automation*, 17, 16.
- Butcher, J.C. (2008). *Numerical Methods for Ordinary Differential Equations* (1st ed.). Wiley: Hoboken, NJ.
- Caccavelli, J., Pedre, S., de Cristóforis, P., Katz, A., & Bendersky, D. (2011). A New Programming Interface for Educational Robotics. In D. Obdržálek & A. Gottscheber (Eds.) *Research and Education in Robotics - EUROBOT 2011* (pp. 68–77). Springer: Berlin/Heidelberg, Germany.
- Cardoso-Júnior, A., & Faria, R.M.D.D. (2021). Psychometric assessment of the Instructional Materials Motivation Survey (IMMS) instrument in a remote learning environment. *Revista Brasileira de Educação Médica*, 45, 197.
- Choi, H., Crump, C., Duriez, C., Elmquist, A., Hager, G., Han, D., Hearl, F., Hodgins, J., Howe, A., Jeong, D., et al. (2021). On the use of simulation in robotics: Opportunities, challenges, and suggestions for moving forward. *Proceedings of the National Academy of Sciences of the USA*, 118, e1907856118.
- Corke, P.I. (2007). A Simple and Systematic Approach to Assigning Denavit–Hartenberg Parameters. *IEEE Transactions on Robotics*, 23, 590–594.
- Deo, A.S., & Walker, I.D. (1995). Overview of damped least-squares methods for inverse kinematics of robot manipulators. *Journal of Intelligent and Robotic Systems*, 14, 43–68.
- Dreger, F.A., Chuang, L.L., & Rinkenauer, G. (2024). Analysis of learning the bimanual control of (tele)operating joint space controlled robotic arms with 4 degrees of freedom using the two-timescales power law of learning. *Ergonomics*, 67, 759–771.
- Fernández, B.G., Cañas, J.M., García-Pérez, L., & Rivas-Montero, F.M. (2020). Robotics vs. Game-Console-Based Platforms to Learn Computer Architecture. *IEEE Access*, 8, 95153–95169.
- Fiala, J., & Walvering, A. (1990). Implementation of a Jacobian-transpose algorithm. National Institute of Standards and Technology: Gaithersburg, MD. NIST IR 90-4286.
- Fonseca Ferreira, N.M., & Freitas, E.D.C. (2018). Computer applications for education on industrial robotic systems. *Computer Applications in Engineering Education*, 26, 1186–1194.
- González-García, S., Rodríguez-Arce, J., Loreto-Gómez, G., & Montaña-Serrano, V.M. (2020). Teaching forward kinematics in a robotics course using simulations: transfer to a real-world context using LEGO mindstorms™. *International Journal on Interactive Design and Manufacturing*, 14, 773–787.

- Granja, M., Chang, N., Granja, V., Duque, M., & Llulluna, F. (2016). Comparison between Standard and Modified Denavit-Hartenberg Methods in Robotics Modelling. *In Proceedings of the 2nd World Congress on Mechanical, Chemical, and Material Engineering*, Barcelona, Spain.
- Haces-Garcia, A., & Zhu, W. (2024). Building an Accessible and Flexible Multi-User Robotic Simulation Framework with Unity-MATLAB Bridge. *Computers*, 13, 282.
- Haviland, J., & Corke, P. (2020). A Systematic Approach to Computing the Manipulator Jacobian and Hessian using the Elementary Transform Sequence. *arXiv preprint*.
- Hugues, L., & Bredeche, N. (2006). Simbad: An Autonomous Robot Simulation Package for Education and Research. In S. Nolfi, G. Baldassarre, R. Calabretta, J.C.T. Hallam, D. Marocco, J.-A. Meyer, O. Miglino, & D. Parisi (Eds.) *From Animals to Animats 9* (pp. 831–842). Springer: Berlin/Heidelberg, Germany.
- Keller, J.M. (2010). *Motivational Design for Learning and Performance: The ARCS Model Approach*. Springer: Boston, MA.
- Khalil, W., & Creusot, D. (1997). SYMORO+: A system for the symbolic modelling of robots. *Robotica*, 15, 153–163.
- Kirschner, P.A., Sweller, J., & Clark, R.E. (2006). Why Minimal Guidance During Instruction Does Not Work: An Analysis of the Failure of Constructivist, Discovery, Problem-Based, Experiential, and Inquiry-Based Teaching. *Educational Psychologist*, 41, 75–86.
- Kortemeyer, G. (2019). Game Development for Teaching Physics. *Journal of Physics: Conference Series*, 1286, 012048.
- Lanos, M.E.C., Handyani, W., Manullang, J.G., Winartiningih, A., & Festiawan, R. (2024). Development of interactive manipulative motion learning media using adobe animate for elementary school students. *JK*, 12, 184–194.
- Lloyd, J., & Hayward, V. (1988). Kinematics of common industrial robots. *Robotics and Autonomous Systems*, 4, 169–191.
- Loorbach, N., Peters, O., Karreman, J., & Steehouder, M. (2015). Validation of the Instructional Materials Motivation Survey (IMMS) in a self-directed instructional setting aimed at working with technology. *British Journal of Educational Technology*, 46, 204–218.
- Moiseienko, N. V., Moiseienko, M. V., Kuznetsov, V. S., Rostalny, B. A., & Kiv, A. E. (2023). Teaching computer game development with Unity engine: a case study. *CEUR Workshop Proceedings*.
- Mosler, P., Schönbeck, P., Braach, M., Hudert, M., König, R., & Donath, D. (2023). Using the Game Engine Unity Efficiently in Teaching: Development of a fully-automated webserver-based build pipeline. *In Proceedings of the eCAADe 2023: Digital Design Reconsidered*, Graz, Austria.
- Nakamura, Y., & Hanafusa, H. (1986). Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. *Journal of Dynamic Systems, Measurement, and Control*, 108, 163–171.
- Prince, M. (2004). Does Active Learning Work? A Review of the Research. *Journal of Engineering Education*, 93, 223–231.

- Ragusa, G., & Leung, L. (2023). The Impact of Early Robotics Education on Students' Understanding of Coding, Robotics Design, and Interest in Computing Careers. *Sensors*, 23, 9335.
- Raslan, G. (2024). The Impact of the Zone of Proximal Development Concept (Scaffolding) on the Students Problem Solving Skills and Learning Outcomes. In K. Al Marri, F.A. Mir, S.A. David, & M. Al-Emran (Eds.) *BUID Doctoral Research Conference 2023* (pp. 59–66). Springer Nature Switzerland: Cham, Switzerland.
- Raz, T. (1989). Graphics robot simulator for teaching introductory robotics. *IEEE Transactions on Education*, 32, 153–159.
- Reddy, A.C. (2014). Difference between Denavit-Hartenberg (DH) classical and modified conventions for forward kinematics of robots with case study. In *Proceedings of the International Conference on Advanced Materials and Manufacturing Technologies (AMMT)*, JNTUH College of Engineering Hyderabad Chandigarh, India.
- Singh, R., Kukshal, V., & Yadav, V. (2021). A Review on Forward and Inverse Kinematics of Classical Serial Manipulators. In *Advances in Engineering Design* (pp. 417–428). Springer: Singapore.
- Süli, E., & Mayers, D.F. (2003). *An Introduction to Numerical Analysis* (1st ed.). Cambridge University Press: Cambridge, UK; New York, NY.
- Teixeira, J.V. (2015). Educational Robotic Simulators: A Systematic Literature Review. *Nuevas Ideas en Informática Educativa TISE*, 20, 340–350.
- Wampler, C. (1986). Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods. *IEEE Transactions on Systems, Man, and Cybernetics*, 16, 93–101.
- Wang, H., Liu, Y., Zhou, K., Aiyama, Y., & Chen, T. (2014). Research on the Relationship between Classic Denavit-Hartenberg and Modified Denavit-Hartenberg. In *Proceedings of the 2014 Seventh International Symposium on Computational Intelligence and Design*, Hangzhou, China.
- Wang, K., Sang, G.-Y., Huang, L.-Z., Li, S.-H., & Guo, J.-W. (2023). The Effectiveness of Educational Robots in Improving Learning Outcomes: A Meta-Analysis. *Sustainability*, 15, 4637.
- Whitney, D.E. (1969). Resolved Motion Rate Control of Manipulators and Human Prostheses. *IEEE Transactions on Man-Machine Systems*, 10, 47–53.
- Wieman, C., & Perkins, K. (2005). Transforming Physics Education. *Physics Today*, 58(11), 36–41.
- Yang, F.Y., & Wang, H.Y. (2023). Tracking visual attention during learning of complex science concepts with augmented 3D visualizations. *Computers & Education*, 193, 104659.
- Zajda, J. (2021). Constructivist Learning Theory and Creating Effective Learning Environments. In J. Zajda (Ed.) *Globalisation and Education Reforms: Creating Effective Learning Environments* (pp. 35–50). Springer International Publishing: Cham, Switzerland.