

ADEBOT1: EDUCATIONAL ROBOTICS WITH ARDUINO AND ALGEBRAIC SIMULATION

Puguh Wahyu Prasetyo^{1*}, Adhy Kurnia Triatmaja², Fariz Setyawan³, Muhammad Ardiyansyah⁴, Angger Hafid Prabawa⁵, Shafa Violeta Chaila Putri⁶, Ayoudya Titan Widyasmara⁷, Hafizh Naufal Azmi⁸, Samsul Arifin⁹

^{1,3,5,6,7,8} Mathematics Education Study Program, Faculty of Teacher Training and Education, Universitas Ahmad Dahlan, 7th Floor, Main Building, Kampus 4 UAD, Jl. Ringroad Selatan, Kragilan, Tamanan, Banguntapan, Bantul, Yogyakarta 55191, Indonesia

² Electronics and Informatics Education Study Program, Faculty of Teacher Training and Education, Universitas Ahmad Dahlan, Laboratory Building, Kampus 4 UAD, Jl. Ringroad Selatan, Kragilan, Tamanan, Banguntapan, Bantul, Yogyakarta 55191, Indonesia

⁴ Department of Biological and Environmental Science, Faculty of Mathematics and Science, University of Jyväskylä, YA C425.1, Seminaarinkatu 15, PL 35 Jyväskylä, Surfontie 9, Finland

⁹ Data Science, Faculty of Engineering and Design, Institut Teknologi Sains Bandung, Deltamas Lot-A1 CBD, Jl. Ganesha Boulevard No.1 Blok A, Pasirranji, Bekasi, West Java 17530, Indonesia

*Corresponding author: puguh.prasetyo@uad.ac.id

Proceedings of the Kalijaga International Conference on Education 2026

Abstract. This study is focused on creating ADEBOT1, an educational robot powered by Arduino, to enhance STEM learning. The drive behind developing ADEBOT1 is the pressing need for educational robots that are innovative, adaptable, and efficient in terms of kinematic performance. Before bringing ADEBOT1 to life physically, its programming was tested through simulations using Google Colab to evaluate the effectiveness of its motion control algorithms. The simulation looked at two primary methods: Product of Exponentials (PoE) and Sequential Transformation (ST) for calculating the robot's forward kinematics. The findings revealed that the ST method is 2.47 times quicker than PoE, making it ideal for real-time applications in straightforward robotic systems for ADEBOT1. Nevertheless, PoE continues to be important for more intricate systems with multiple revolute joints. This simulation serves as a foundation for refining the programming before ADEBOT1 is physically deployed, ensuring it delivers reliable and responsive performance in educational settings.

Keywords: *Algebraic structure, Grobner basis, Lie algebra, Product of exponential, Sequential transformation.*

INTRODUCTION

The development of robotics technology has brought significant changes in various fields, ranging from industry, medical, education, to space exploration. According to data released by the International Federation of Robotics in 2024, the use of robots in the industrial world has experienced very significant growth, as shown in Figure 1 below.

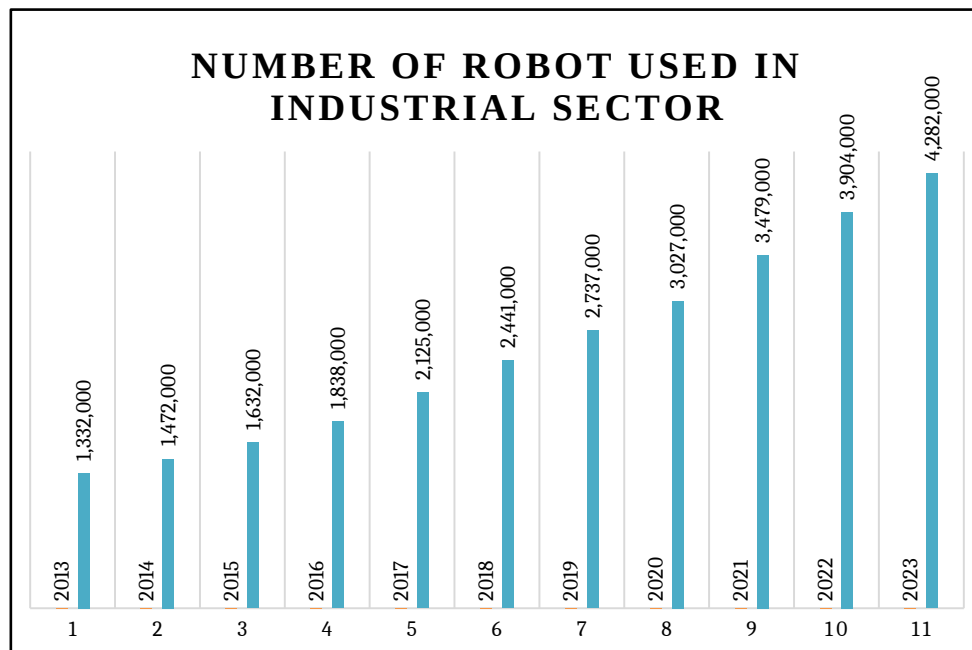


Figure 1. Data on the Use of Industrial Robots Worldwide (Source:[1])

The use of robots in Indonesia's industrial sector is also increasing, as indicated by statistical data reports found in [2]. One of the key aspects in robot development and study are the ability to control the movement of robot with high precision [3], [4], [5], [6], [7], [8]. One of the mathematical concepts used in optimizing robot motion control is Lie algebra[9], [10], [11], [12], [13]. More specifically, those frequently used are special Euclidean and special orthogonal groups as studied in[14], [15], [16], [17], [18].

Lie algebra is used to control nonlinear systems, particularly in mobile robot control. For instance, path following control of a mobile robot can be achieved using an input-output feedback linearization controller, which is based on Lie algebra [19]. This method helps in effectively managing the nonlinear dynamics of the robot. In mobile robot trajectory tracking, Lie algebra helps in considering the manifold constraints of the robot configuration. By utilizing the relationship between Lie groups and Lie algebra, the tracking control problem can be convexified, leading to more efficient and smoother trajectory tracking [20]. Lie algebra is also applied in formulating optimal control problems for mobile robots. For example, in the control of a vertical rolling disc (a simplification of a differential drive wheeled mobile robot), the controllable Lie algebras are calculated to determine energetically optimal trajectories between states [21]. Lie algebra is extensively used to describe the dynamics and control of robotic systems, including those with active and passive joints. The inverse dynamics of such systems are often described using Lie groups and Lie algebras, which are then used to implement computed torque control laws for better performance compared to traditional PID control [22]. For systems like the T-S fuzzy model of a flexible joint robot arm, Lie algebra is used to design controllers that satisfy stability criteria. This approach ensures the stability of the robotic system [23]. Additionally, Lie algebra methods are used to linearize nonlinear systems, such as in the control of an inverted pendulum, which can be applied to robotic systems for improved performance [24].

Lie algebra aids in the kinematic and dynamic modeling of robots, especially those with complex mechanisms. By representing the pose of components using the Euclidean group and its subgroups, the velocity, acceleration, and Jacobian matrix can be derived, simplifying the modeling process and reducing computational complexity [25]. The use of Lie algebra in robot dynamics and control allows for efficient simulation and real-time control. For example, the computed torque control method based on Lie algebra shows better performance in simulations, indicating its potential for real-time applications [11]. Lie algebra is part of geometric methods in robotics, which provide a unified and intuitive framework for modeling and control. These methods are computationally efficient and can be applied to various robotic tasks, including manipulation and trajectory tracking [20], [26]. Lie algebra plays a crucial role in the control and modeling of robotic systems, including Arduino robot cars. It provides tools for handling nonlinear dynamics, optimizing control strategies, and simplifying the kinematic and dynamic modeling of robots.

In this study, the definition of Lie algebra and its basic concepts can be accessed in [19]. To assist readers, we provide the formal definition as follows. A non-empty set g completed with binary operation $[\cdot, \cdot]: g \times g \rightarrow g$ formed a Lie algebra when satisfied the following properties: Linearity; for every x in g , $[x, x]=0$; and $x, y, z \in g$ implies $[x, [y, z]] + [y, [z, x]] + [z, [x, y]] = 0$. More

advanced and detailed algebraic concepts and some implementations in robotics programming are studied in [27], [28]. The trend of robot implementation in various fields is very positive, such as in the medical field where special robots, especially soft robots, are needed to handle certain cases in surgery and operations [29], [30]. This certainly makes robotics research increasingly challenging in the future.

To prepare a generation capable of facing these challenges, robot development is not only specifically designed to perform special tasks but there are also robots developed as educational robots intended for learning from elementary school to university levels, each with its own specifications. Some well-known companies have also produced educational robots such as Lego with its Lego Mindstorms, Makeblock with mBot and mBot2, and many others. The practical implementation of these robots in education has shown positive results as studied in [31], [32], [33], [34], [35].

In addition, the optimization of mBot2 using the concept of Lie algebra has also been studied in [12]. In that study, Lie algebra was able to optimize by minimizing orientation errors and the travel time duration of the mBot2 robot when following a route. Furthermore, the implementation of these two robots in the learning field is also very challenging. This includes learning activities with a STEM approach involving robot assembly, studying robot functions, and then programming them. From this series of activities, there are certainly weaknesses. In terms of creativity, these robots can only be assembled in fixed forms with variations recommended by Lego or Makeblock. This is certainly one of the weaknesses, especially for enhancing creativity in assembling educational robots. Therefore, in this study, we examine the construction of a robot model with a robot body that can be created using a 3D printer and using Arduino as its microcontroller. The reason for using Arduino is because it is easily obtainable, relatively affordable, and programming algorithms for it are already available in [27], for example. As the state of the art in this research, the educational robot is constructed with a body shaped using a 3D printer and can operate automatically with the help of an RGB colour sensor programmed with the concept of algebra and has potential applications in the field of learning. In addition, this study will uniquely discuss the Grobner basis of the developed robot, and its computational exploration and experiments will be presented.

METHODS

Based on the issues and problem formulation presented in the introduction section, this section will outline the research methods used in this study. This research was conducted through several stages as follows.

First Stage

The first stage is the development of the robot. This research is a form of Research and Development (RnD). In this stage, the researcher adopts the Analysis-Design-Development-Implementation-Evaluation (ADDIE) development model with the following phases:

Analysis

Analyzing literature studies on concepts of Lie algebra that can be applied to mobile robots with arms. This is followed by an analysis of existing educational robot products to gather relevant information. This stage strengthens the motivation to develop an android-based mobile robot with an arm.

Design

In this stage, the mobile robot with an arm is designed. The design process is divided into three parts: designing the arm or gripper, designing the robot's body, and designing related components such as circuits and servo positions. Illustrative designs of these components, registered with DJKI for industrial design creation rights, are provided in the next section.

Development

In this stage, the robot's arm and body are built with the aid of a standard-specification 3D printer. The assembly process involves the following components. The assembled robot is presented in the subsequent section. Additionally, in this stage, program installation incorporating Lie algebra concepts is conducted. These concepts are based on previous research results, including those presented in articles such as [12], [27], [28].

Implementation

After the robot is successfully assembled, it is implemented using specific routes. In terms of features, the robot has automatic movement capabilities aided by RGB colour sensors and a secondary feature with an android-based wireless remote. After this stage, evaluations are also applied, including addressing technical issues. To minimise the impact of technical issues, a simulation process of ADEBOT1 is carried

out on the Google Colab platform during the implementation stage to observe the effectiveness of the programming to be applied.

Evaluation

The evaluation stage is conducted at every development phase. One of the major evaluation outcomes, for example, is the replacement of burnt servos during implementation.

Second Stage

The second stage of the research involves developing a mathematical model for the robot, including formulating the Grobner basis of the developed robot arm system. The analysis of maneuverability and movements, both kinematics and dynamics of ADEBOT1, is also conducted. Computation is applied at this stage to assess the performance of ADEBOT1's movements. The Grobner basis concepts in this phase refer to references such as [12].

The third stage

In this stage, the simulation for ADEBOT1 is analyzed. The simulation of ADEBOT1 is conducted using an online Python-based programming environment, specifically Google Colab. This simulation aims to compare two distinct approaches: the exponential map (Product of Exponential, PoE) and sequential transformation. The rationale behind selecting these two approaches stems from a comprehensive analysis of their computational costs, as reviewed in prior research. For the Product of Exponential approach, several methods have been employed to analyze computational costs within the framework of Lie algebra. Notably, the Runge-Kutta-Munthe-Kaas (RKMK) methods utilize Lie-group solvers to minimize computational costs by applying linear transformations and examining graded free Lie algebras, effectively reducing the number of commutators [36]. Despite these optimizations, the computational complexity remains significant, particularly when addressing nonlinear systems, as evidenced by applications in unscented Kalman filters (UKF) for robot localization[37]. Another prominent method is Newton's method, which, when applied to Lie groups, demonstrates semi-local convergence with computational costs comparable to traditional techniques. However, it offers distinct advantages, such as weaker convergence conditions and tighter error bounds [38], [39]. Additionally, motion transformation techniques employ Lie algebra elements to represent group transformations, ensuring the shortest geodesic paths within the Lie group, although this can be computationally intensive [40].

Conversely, sequential transformation approaches have also been explored to assess computational efficiency. The Euler angle perturbation model introduces novel solutions that circumvent singularity issues associated with Euler angles. These methods focus on accelerating iterative processes and minimizing computational complexity while achieving equivalent global minima compared to traditional Lie algebra-based solutions [41]. Furthermore, recursive fast transforms are engineered to achieve the lowest computational complexity. This is accomplished through the application of recursive equations and Kronecker products, simplifying fast transforms and reducing computational overhead [42], [43].

From the comparative analysis, it is evident that sequential transformation methods, particularly those leveraging recursive fast transforms and Euler angle perturbation models, exhibit superior cost-effectiveness and efficiency in specific scenarios [41], [42], [43]. In contrast, while Lie algebra-based methods offer powerful and precise solutions, they generally incur higher computational complexity due to the necessity of managing nonlinearities and maintaining geometric constraints [36], [37].

In application-specific contexts, such as robot localization and motion transformation, Lie algebra methods provide robust and reliable outcomes, albeit at higher computational costs [37], [40]. Sequential transformation methods, notably those employing Euler angle perturbation, strike a favorable balance between accuracy and computational efficiency, rendering them suitable for real-time applications [41]. Thus, while Lie algebra-based methods deliver precise and robust solutions, their computational demands are substantial. Sequential transformation methods, particularly those utilizing recursive fast transforms and Euler angle perturbation models, present a more computationally efficient alternative, especially for real-time processing applications with stringent computational constraints [36], [37], [41], [42], [43]. This study solves a research gap concerning the impact of both approaches on the ADEBOT1 simulation, warranting further exploration. Based on this study, a research gap emerges that needs to be explored further, particularly regarding the impact of both approaches in the ADEBOT1 simulation.

In the context of computational complexity, a comparative study was conducted between the implementation of PoE (Proof of Existence) and Sequential Transformation. This comparison focuses on both theoretical and practical aspects to highlight the differences in their computational demands.

In terms of computational complexity, Lie Algebra generally exhibits polynomial complexity, with algorithms specifically designed to efficiently manage symbolic computations[44]. The complexity tends to

increase with the rank of the algebra, posing significant computational challenges for higher ranks unless specialized software is employed to mitigate these issues [45], [46], [47]. Conversely, Sequential Transformations can exhibit high levels of complexity, particularly when optimized through parallelization. This complexity arises from the requirements for synchronization and the preservation of dependencies, with variability depending on the specific transformation and its application, such as in change-point detection or dynamical modeling [48], [49], [50].

From a practical applications standpoint, Lie Algebra is widely utilized in physics, particularly in the analysis of systems characterized by Hamiltonians and other operators expressed via Lie algebra elements. The computational complexities are often managed using specialized software packages that support symbolic computations [47], [51], [52]. Sequential Transformations, on the other hand, find applications across diverse fields such as computer engineering, dynamical modeling, and graph transformation systems. The inherent complexities in these applications are typically addressed through parallelization and optimization techniques, which are crucial for handling large datasets and real-time data processing requirements [50].

The ADEBOT1 simulation was conducted using Google Colab to compare the Product of Exponential (PoE) approach with sequential transformation. The analysis focused on the computational cost efficiency of these two methods. The PoE approach proved to be computationally complex, particularly in nonlinear systems such as the unscented Kalman filter applied to robot localization [37]. In contrast, the Euler angle perturbation model offers an efficient solution by addressing singularity issues [41]. Recursive fast transformation also reduces computational complexity using recursive equations [50]. Lie Algebra demonstrates polynomial complexity that increases with the rank of the algebra, while sequential transformations can be optimized through parallelization [50]. In real-world applications such as robot localization and motion manipulation, Lie Algebra-based approaches yield robust results but are computationally expensive [37]. Sequential transformation methods, particularly those utilizing Euler angle perturbation models and recursive fast transformations, are more efficient for real-time applications [41], [50].

RESULTS AND DISCUSSION

In the Results and Discussion section, the development process of ADEBOT1, an Arduino-based robot equipped with an articulated arm, is thoroughly explained. The **analysis stage** highlights that the research background underscores the necessity of designing an Arduino-based robot with a body and accessories, such as an arm, which can be fabricated using filament materials and assembled with the aid of a 3D printer.

In the design stage, the ADEBOT1 is designed with the following details: ADEBOT1's features include autonomous movements facilitated by Red Green Blue (RGB) color sensors and wireless control capabilities through a remote, operated via an application compatible with Android devices. These features are illustrated in Figure 2 below.

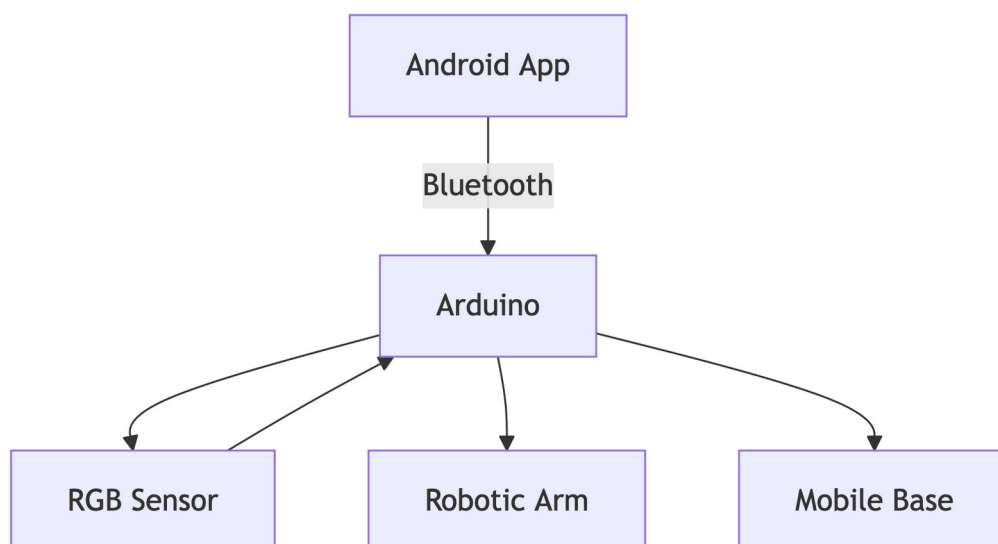


Figure 2. Robot Feature Concept

After drafting the concept, still in the design phase, the robot's arm and body are designed. The appearance of these robot parts is illustrated in Figure 3 below.

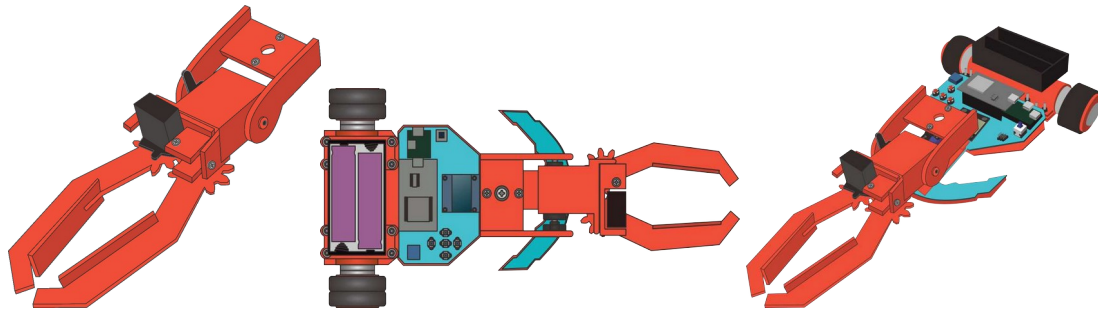


Figure 3. ADEBOT1 Concept Design

The ADEBOT1 Construction

In this stage, the robot is physically constructed using a standard-specification 3D printer. The development process includes assembling the robot's arm and body, installing software that integrates Lie Algebra concepts based on previous research, and integrating electronic components such as sensors and actuators. After the assembly, the robot undergoes implementation and testing on specific routes. Key features include automatic movement capability, supported by RGB colour sensors for path detection, and wireless remote control operated via an Android-based remote. During implementation, technical issues such as burnt servos are identified and addressed. This development process has adopted the ADDIE research and development model as described in the previous section.

Evaluation is conducted continuously at every development phase with the aim to identify technical deficiencies and improve the robot's design and functionality based on testing results. A significant evaluation result includes the replacement of burnt servos encountered during implementation trials for example. The development of the mobile robot with an arm demonstrates a systematic approach from analysis to evaluation. The assembled appearance of ADEBOT1 with all its components is shown in Figure 4 below.

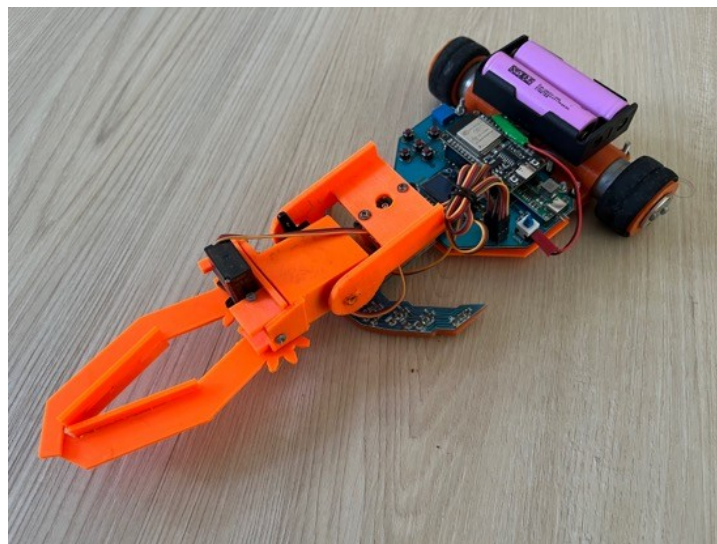


Figure 4. Assembled ADEBOT1

Meanwhile, the detailed specifications are presented in Table 1 below.

Table 1. Specification of ADEBOT1

Component	Detail
Microcontroller	ESP32
Actuator	Motor servo and motor DC 12v
Battery Capacity	Li on 12v 5000 mAh
Sensor	Photodiode Sensor

Component	Detail
LED Screen	OLED
Robot Casing	3D printing plastic

Algebraic Property of ADEBOT1

The Grobner basis is an effective method for solving inverse kinematics problems in robotic arms, which is crucial for determining joint angles to achieve a specific end-effector position. This method is highly beneficial for polynomial equation systems derived from kinematic constraints [53], [54], [55]. The Degree of Freedom (DOF) in an arm robot's system refers to the count of independent variables needed to fully determine its configuration or state. For 3DOF robot manipulators, the Gröbner basis can be computed using computer algebra systems such as SymPy and Risa/Asir integrated through Python, enabling symbolic and numerical solutions that enhance the efficiency and accuracy of kinematic calculations [53].

In addition to inverse kinematics, the Grobner basis can also be applied to forward kinematics analysis, where the end-effector position is determined based on known joint angles. This process involves mapping solutions from high-dimensional kinematic space to three-dimensional Euclidean space [56]. The resultant method and Grobner basis in a seven-dimensional kinematic space provide robust solutions for planar parallel robots with identical arm structures. For trajectory planning and path feasibility, the Comprehensive Gröbner Systems (CGS) method ensures the existence of solutions, avoids redundant computations, and verifies the feasibility of motion paths [54].

The application of the Grobner basis in robotic systems demonstrates accuracy and computational efficiency. This condition delivers highly accurate results in solving kinematic equations for both industrial robotic arms and multi-legged robots [55]. Its computational performance is comparable to conventional kinematic models with minimal reference errors, making it a reliable choice for real-time control and simulation [55]. Moreover, the Gröbner basis is versatile as it can be applied to various types of robotic systems, including Cartesian robots, SCARA, and non-redundant open-chain manipulators [55]. Integration with computer algebra systems and systematic procedures further enhances its utility in real-time control and path planning [53], [54], [55], [56]. This section will present the algebraic properties of ADEBOT1, including the Gröbner basis of the arm system of ADEBOT1. The use of algebraic techniques in robot kinematics enables symbolic analysis and simplification of nonlinear systems. ADEBOT1 is an educational robot designed with two degrees of freedom (horizontal and vertical motion) of the arm and one degree of freedom of a gripper for manipulation tasks. While it lacks inverse vertical motion, its simple kinematic structure provides an ideal framework for exploring symbolic transformations. This paper formulates the kinematic model of ADEBOT1 and applies trigonometric rationalization via Weierstrass substitution to express its forward kinematics as a polynomial system. We define the robot's configuration by the parameters described in the following theorem:

Theorem 1. The position of the end-effector of ADEBOT1 (x, y, z) is given by where θ is the angle formed by the horizontal rotation (yaw), ranging from -90^0 to $+90^0$; ϕ is the angle formed by the elevation (tilt), ranging from 0^0 to $+90^0$; l is the length of the arm of the ADEBOT1.

Proof.

It is obvious by the basic concept of trigonometry.

To help the reader to understand what are the θ and ϕ , please consider the Figure 5 below.

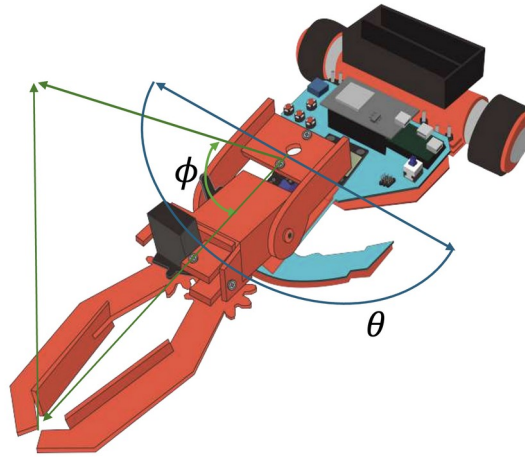


Figure 5. Yaw and Tilt Illustrations of ADEBOT1

It follows from Theorem 1 that the following consequence is asserted.

Corollary 1. The Grobner basis of the arm system of ADEBOT1 is given by

$$x = \frac{l(1-s^2)(1-t^2)}{(1+s^2)(1+t^2)}$$

$$y = \frac{2l(1-s^2)t}{(1+s^2)(1+t^2)}$$

$$z = \frac{2ls}{1+s^2}$$

where $s = \tan \frac{\phi}{2}$ and $t = \tan \frac{\theta}{2}$.

Proof. It follows from Theorem 1 that the end-effector of ADEBOT1 (x, y, z) is given by

$$x = l \cos \phi \cdot \cos \theta$$

$$y = l \cos \phi \cdot \sin \theta$$

$$z = l \sin \phi$$

To enable symbolic computation with algebraic methods, we apply the Weierstrass substitution: $s = \tan \frac{\phi}{2}$ and $t = \tan \frac{\theta}{2}$ to get the representations

$$\cos \theta = \frac{1-t^2}{1+t^2}, \sin \theta = \frac{2t}{1+t^2}, \cos \phi = \frac{1-s^2}{1+s^2}, \sin \phi = \frac{2s}{1+s^2}$$

Substituting these into the previous expression, we therefore have:

$$x = \frac{l(1-s^2)(1-t^2)}{(1+s^2)(1+t^2)}$$

$$y = \frac{2l(1-s^2)t}{(1+s^2)(1+t^2)}$$

$$z = \frac{2ls}{1+s^2}$$

Which completes the proof.

ADEBOT1 Movement Simulation

For Fundamental aspect of this section, we refer to [57], [58], [59]. In robotics, the **configuration** of a rigid body in 3D space (i.e., its position and orientation) is represented as a **transformation matrix** in the **Special Euclidean group**, denoted as $SE(3)$. This matrix combines a rotation $R \in SO(3)$ and a translation vector $p \in R^3$. Each transformation matrix $T \in SE(3)$ can be written as:

$$T = \begin{bmatrix} R & p \\ 0 & 1 \end{bmatrix}$$

The **Lie algebra** associated with $SE(3)$ is called $\mathfrak{se}(3)$ [59]. Elements of $\mathfrak{se}(3)$ are called **twists**[60]. The twists represent **infinitesimal motions** (rotations and translations).

A twist $\xi \in R^6$ is often written as:

$$\xi = \begin{bmatrix} v \\ \omega \end{bmatrix}$$

where $\omega \in R^3$ is the angular velocity (rotation) and $v \in R^3$ is the linear velocity vector (translation, dependent on the rotation center). The twist is turned into a 4x4 matrix (called a hat operator $\hat{\xi}$) for use in matrix exponentiation:

$$\hat{\xi} = \begin{bmatrix} [\omega]_{\times} & v \\ 0 & 0 \end{bmatrix}$$

where $[\omega]_{\times}$ is the skew-symmetric matrix of ω . The Product of Exponentials (PoE) formula provides a powerful method to model a robot's forward kinematics [59], [61], [62]. It describes the position and orientation of the robot's end-effector in space as a sequence of motions, each expressed as the exponential of a twist from the Lie algebra $\mathfrak{se}(3)$. A twist is a 6-dimensional representation of a joint's motion combining both linear and angular velocity and it is crucial in the modeling of robot joints as part of a continuous motion group. The PoE formula is written as:

$$T(\theta_1, \theta_2, \dots, \theta_n) = e^{\hat{\xi}_1 \theta_1} \cdot e^{\hat{\xi}_2 \theta_2} \dots e^{\hat{\xi}_n \theta_n} M$$

where $\hat{\xi}_i$ represents the twist corresponding to the screw axis of the i -th joint, expressed in the space frame, θ_i is the joint variable, such as the angle for a revolute joint or the displacement for a prismatic joint and M is the home configuration of the end-effector, which describes the pose of the end-effector when all joint variables are zero.

To simplify the simulation process, ADEBOT1 can be considered as a 2-DOF robot car with an arm by ignoring the gripper first. ADEBOT1 is a mobile robot equipped with:

- **Differential-Drive Base:** Allows movement in the $x-y$ plane and rotation around the vertical z -axis, providing 3 degrees of freedom (DOF) for the base.
- **2-DOF Arm:** Typically consists of two revolute joints that enable vertical planar motion, similar to shoulder and elbow movements.

The complete configuration of ADEBOT1 includes:

- **Base Motion:** q_1 (Translation along the x -axis), q_2 (Translation along the y -axis), q_3 (Rotation about the z -axis)
- **Arm Joints:** θ, ϕ : Rotations of joint 1 and joint 2, respectively where $-90^\circ \leq \theta \leq 90^\circ$ and $0 \leq \phi \leq 90^\circ$

Furthermore, each motion is represented by a twist in $\mathfrak{se}(3)$:

- **Base Motion Twists:** Modeled using twists $\hat{\xi}_{base1}, \hat{\xi}_{base2}, \hat{\xi}_{base3}$.
- **The arm motion is modeled** twists $\hat{\xi}_1, \hat{\xi}_2$ for each joint.

It follows from the concept of PoE in [63] that we therefore have.

Theorem 2. The PoE kinematics for ADEBOT1 is

$$T(\theta, \phi) = e^{\hat{\xi}_{base1} q_1} \cdot e^{\hat{\xi}_{base2} q_2} \cdot e^{\hat{\xi}_{base3} q_3} e^{\hat{\xi}_1 \theta} \cdot e^{\hat{\xi}_2 \phi} M$$

where q_1, q_2, q_3 are the position and orientation of the mobile base; θ, ϕ are the arm joint and M is the home pose of the end-effector relative to the robot base when the joints are in their zero configuration.

Proof. It follows from the previous explanation.

Based on Theorem 2 above, for the operation of the product of exponential as the impact model of ADEBOT1, it will mathematically result in a large computational cost. In the subsequent section, we will conduct a simulation of ADEBOT1's movement utilizing the Google Colab online programming platform. We use this algorithm

Forward Kinematics Computation Algorithm with PoE

The computation steps can be summarized as follows:

1. Initialize the robotic design parameters: anchor length (fixed part) and arm length (moving part).
2. Define the spatial twist vectors ξ_1 and ξ_2 for each joint.
 1. ξ_1 is the rotation about the Z-axis at the base point (0,0,0).
 2. ξ_2 is the rotation about the Y-axis at the point (0,0,ANCHOR_LENGTH).
3. Convert the twist vectors into a 4x4 se(3) matrix format using the `vec_to_se3` function.
4. Multiply the se(3) matrix by the joint angle in radians, then take the matrix exponential (using `expm` from `scipy.linalg`) to obtain the SE(3) transformation.
5. Sequentially multiply the SE(3) transformations of each joint, then multiply with the home configuration matrix M to obtain the end-effector pose relative to the base.
6. Add the transformation from the mobile robot base to the world frame to obtain the end-effector position in the world frame.

From this algorithm, the simulation display obtained is shown in Figure 6 below.

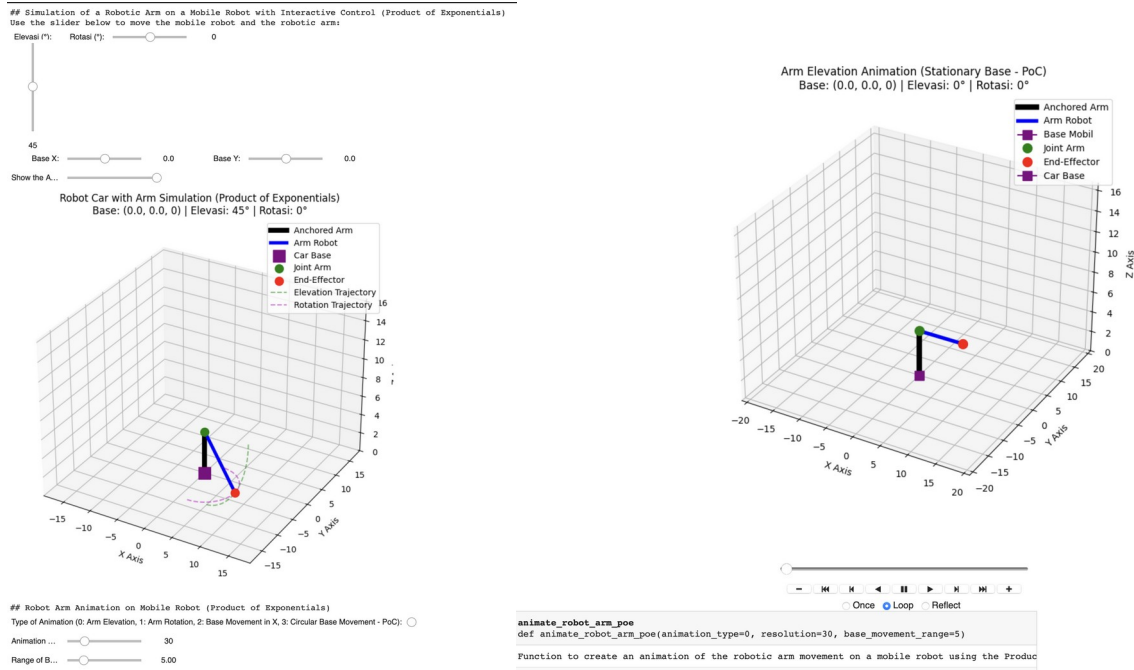


Figure 6. ADEBOT1 Simulation PoE

This approach starts by defining an initial configuration with all joints in their zero positions, establishing reference frames on both the base and tool links, and deriving joint twists based on the joint axes' geometry. The robot's kinematics are then elegantly formulated as a product of exponentials of these joint twists, followed by the transformation matrix between the initial frames. The POE formula offers benefits such as straightforward modeling [64]. In this study, we compare the PoE approach with sequential transformation. The sequential transformation uses homogeneous transformation matrices and Denavit-

Hatemberg parameters for systematic modeling [65]. In previous study, [64] compared the PoE with specifically Denavit-Hartenberg (D-H) approach. According to [64] the D-H method remains the most commonly utilized approach for kinematics modelling.

Remark 1. It follows from the concept of sequential transformation, the end-effector pose of ADEBOT1 can be denoted by T_{world}^{end} and it is determined through chained transformations:

$$T_{world}^{end} = T_{world}^{base} \cdot T_{base}^1 \cdot T_1^2(\theta) \cdot T_2^3(\phi) \cdot T_3^{end}$$

T_{world}^{base} is the Base translation and $T_1^2(\theta), T_2^3(\phi)$ are rotations about the (Z) and (Y) axes respectively.

Based on the Remark 1. We deploy the following algorithm and compare with the PoE approach computationally.

Algorithm for Numerical Analysis of Robot Forward Kinematics (FK): ST vs PoE

1. *Preparation:*

1. *Import libraries: numpy, scipy.linalg.expm, time.*
2. *Set robot parameters like ANCHOR_LENGTH and ARM_LENGTH.*

2. *ST Method (Sequential Transformations):*

1. *Create a function for rotation (rotation_matrix_4x4_st).*
2. *Create a function for translation (translation_matrix_st).*
3. *Combine rotation and translation in forward_kinematics_sequential to calculate FK.*

3. *PoE Method (Product of Exponentials): Convert angular vectors to matrices (vec_to_skew_poe).*

Convert twists to transformation matrices (vec_to_se3_poe).

Define initial end-effector position and twists.

Use forward_kinematics_poe with the PoE formula to calculate FK.

4. *Testing Setup:*

Set how many tests to run (e.g., 10,000).

Randomly pick values for angles (theta, phi) and positions (base_x, base_y).

5. *Measure Execution Time:*

Create time_fk_method to measure how long each FK method takes.

6. *Run and Compare:*

Execute both ST and PoE methods.

Measure and compare the total time and average time per calculation.

Display the results.

7. *End of Algorithm*

It follows from the numerical method explained by the above algorithm that the comparison between PoE and the Sequential Transformation is illustrated in Table 2 below.

Table 2. Comparison in Simulation of The Forward Kinematics of ADEBOT1 with PoE vs Sequential Transformation

Aspect	Sequential Transformation	PoE
Computational Efficiency in Avg. Time per Call (μ s)	56.30	138.82
Speed up	2.47×	1.00×
Total time	0.363803 seconds	0.938879 seconds
Average time per call	36.380 microseconds	93.888 microseconds

The Sequential Transformation (ST) method has demonstrated superior computational efficiency, with an average execution time 2.5 times faster than the Product of Exponentials (PoE) method. In 10,000 test iterations, ST completed computations in only 0.364 seconds, whereas PoE required 0.939 seconds. This significant performance difference indicates that ST is more suitable for real-time applications in simple robotic systems where high-speed computations are essential. It is advisable to employ the Sequential Transformation method for robotic applications that demand fast and efficient computations. Conversely, the PoE method should be considered when theoretical consistency with Lie algebra formulations is required or for complex robotic systems featuring numerous revolute joints.

As a final part of this section, we explore the Lie algebraic structure generated by the motion components of ADEBOT1, consisting of a differential-drive mobile base and a 2-DOF articulated arm. Each joint or planar motion corresponds to a twist $\xi_i \in se(3)$, where the system generates a Lie subalgebra $\mathfrak{g} \subset se(3)$ spanned by:

ξ_1 : base rotation and translation,

ξ_2 : shoulder elevation,

ξ_3 : elbow yaw.

We aim to analyze the structure of $\mathfrak{g}$ using a radical $\oplus$ compact decomposition framework, inspired by the stability criterion for switched differential-algebraic equations (DAEs) introduced in [66] and it is also mentioned in [12].

The motion behavior of ADEBOT1 switches between autonomous tracking mode and manual android based remote-control mode, forming a simple switching system. In [66], Raj and Pal proved that if the Lie algebra generated by system matrices admits a decomposition into its radical (a solvable ideal) and a compact subalgebra, then global exponential stability is guaranteed under switching with nonzero dwell time. Although ADEBOT1 does not fall under classical DAEs, we adopt this decomposition framework as a qualitative tool to analyze the behavior of its switching control system. Symbolic computation was performed using the initial basis of $\mathfrak{g} = span\{\xi_1, \xi_2, \xi_3\}$. The Lie brackets among these twists were calculated:

$$[\xi_1, \xi_2] \neq 0, [\xi_2, \xi_3] \neq 0, [\xi_3, \xi_1] \neq 0,$$

and subsequent brackets of these results with each other (i.e., the second derived algebra) were also found to be nonzero. This indicates that the algebra is not nilpotent in two steps, and its solvability is not immediately evident. Nevertheless, the structure reveals restricted closure patterns within a subspace of $se(3)$, suggesting that a radical decomposition is plausible if one extends the basis or explores further derived steps. Hence, the radical-compact decomposition is used in this paper as an analytical lens, rather than as a proven algebraic classification.

Under the assumption that ADEBOT1's generated algebra admits a radical decomposition,

$$\mathfrak{g} = \mathfrak{r} \oplus \mathfrak{s}$$

where:

$\mathfrak{r}$ represents solvable motion generators (e.g., base and elbow movement),

$\mathfrak{s}$ represents compact subalgebra components (e.g., cyclic joint symmetries),

then stability under switching between control modes can be analyzed following the criterion in [66], particularly if the dwell time between mode switches is bounded below. While this decomposition is not yet formally derived, its use here aligns with the geometric intuition and motion structure of ADEBOT1. A full classification of the Lie algebra remains a valuable topic for future symbolic and geometric investigation especially for further ADEBOT1 exploration.

CONCLUSION

From the analysis we've done, it's clear that the Sequential Transformation (ST) method is more efficient than the Product of Exponentials (PoE) method when it comes to speed. The ST method is about 2.5 times faster than PoE, making it ideal for real-time applications, especially in simple robotic systems where quick processing is essential. So, if you need fast and efficient calculations, the ST method is the way to go. This is particularly relevant in the case of ADEBOT1, which has a 2-DOF (Degrees of Freedom) arm joint, making it relatively simple. For such a straightforward robotic system, the ST method's speed advantage becomes even more significant, enabling faster computations and more responsive control. Thus, for ADEBOT1, the ST method is highly suitable and efficient. On the other hand,

the PoE method still holds its ground in situations where aligning with Lie algebra formulations is crucial or in complex robotic systems with many revolute joints. Even though it takes longer to compute, PoE has its own set of benefits that make it the better choice in these scenarios. Therefore, deciding between the ST and PoE methods should depend on the specific computational and theoretical requirements of the robotic application.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to the Institute for Research and Community Service of Universitas Ahmad Dahlan (LPPM UAD) and the Center for Intellectual Property (Sentra HKI UAD) for their invaluable support throughout this research. We are also deeply thankful to BRIn UAD and all parties who have contributed, directly or indirectly, to the success of this study. This research was funded under the research contract Number: PT-117/SP3/LPPM-UAD/XI/2024. We would also like to express our sincere gratitude to Kunti Robiatul Mahmudah, Ph.D., for her invaluable assistance and guidance in ensuring that the author adheres to the research schedule for monitoring this paper. We also express our gratitude to the Centre of Intellectual Property of Universitas Ahmad Dahlan, supporting ADEBOT1 to be submitted for industrial design with submission number (A00202407822 (end-effector); A00202503230 (ADEBOT1 Body); A00202503236 (ADEBOT1 with end-effector)) and simple patent with submission number (S00202505479).

REFERENCES

- [1] "Record of 4 Million Robots in Factories Worldwide." [Online]. Available: <https://ifr.org/ifr-press-releases/news/record-of-4-million-robots-working-in-factories-worldwide>
- [2] "Robotics - Indonesia." [Online]. Available: <https://www.statista.com/outlook/tmo/robotics/indonesia>
- [3] H. Nijmeijer, "Control-Oriented Models for Hyperelastic Soft Robots Through Differential Geometry of Curves," *Soft Robot*, vol. 10, no. 1, pp. 129–148, Feb. 2023, doi: 10.1089/soro.2021.0035.
- [4] J. Adamy, *Nonlinear Systems and Controls*. Springer Berlin Heidelberg, 2024. doi: 10.1007/978-3-662-68690-4.
- [5] S. Kuriakose, M. M. Harshitha, D. G. Keerthana, S. Adarsh, and K. Harshitha, "Wireless Voice Controlled Robot," presented at the 2023 9th International Conference on Advanced Computing and Communication Systems (ICACCS), IEEE, 2023, pp. 189–194. doi: 10.1109/ICACCS57279.2023.10113022.
- [6] M. Khairudin, R. Refalda, S. Yatmono, H. S. Pramono, A. K. Triatmaja, and A. Shah, "The mobile robot control in obstacle avoidance using fuzzy logic controller," *Indones. J. Sci. Technol.*, vol. 5, no. 3, pp. 334–351, 2020, doi: 10.17509/ijost.v5i3.24889.
- [7] J.-S. Li, W. Zhang, and L. Tie, "On separating points for ensemble controllability," *SIAM J. Control Optim.*, vol. 58, no. 5, pp. 2740–2764, 2020, doi: 10.1137/19M1278648.
- [8] N. O. Agwunedu, O. Oshiga, L. O. Chizea, O. A. Oluwafemi, and S. Thomas, "Arduino Based Voice Controlled Delivery System (Robot)," presented at the 2021 1st International Conference on Multidisciplinary Engineering and Applied Science (ICMEAS), IEEE, 2021. doi: 10.1109/ICMEAS52683.2021.9692356.
- [9] P. J. McCarthy and C. Nielsen, "Global stability of a class of difference equations on solvable Lie algebras," *Math. Control Signals Syst.*, vol. 32, no. 2, pp. 177–208, Jun. 2020, doi: 10.1007/s00498-020-00259-7.
- [10] D. Hildenbrand, J. Hrdina, A. Návrát, and P. Vašík, "Local Controllability of Snake Robots Based on CRA, Theory and Practice," *Adv. Appl. Clifford Algebr.*, vol. 30, no. 1, 2020, doi: 10.1007/s00006-019-1022-8.
- [11] Z. Mei, L. Chen, and J. Ding, "Modeling and adaptive torque computed control of industrial robot based on lie algebra," in *Journal of Physics: Conference Series*, 2021. doi: 10.1088/1742-6596/1780/1/012029.
- [12] P. W. Prasetyo, "Optimizing Robotic Movement: Applying Lie Algebra to Enhance the Performance of mBot2," *Int. J. Adv. Sci. Eng. Inf. Technol.*, vol. 15, 2025.
- [13] P. Raj and D. Pal, "Lie Algebraic Criteria for Stability of Switched Systems of Differential Algebraic Equations (DAEs)," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1333–1338, Oct. 2021, doi: 10.1109/LCSYS.2020.3036577.
- [14] F. Aslam and M. F. Haydar, "Nonlinear H ∞ Filtering on the Special Orthogonal Group SO(3) Using Vector Directions," *IEEE Control Syst. Lett.*, vol. 6, pp. 2599–2604, 2022, doi: 10.1109/LCSYS.2022.3172648.
- [15] D. Shi, L. Li, Y. Shao, W. Zhang, and X. Ding, "Multimode Control Strategy for Robotic Rehabilitation on Special Orthogonal Group SO(3)," *IEEE Trans. Ind. Electron.*, vol. 71, no. 2, pp. 1749–1757, 2024, doi: 10.1109/TIE.2023.3257378.
- [16] M. Sinner, X. Chen, and L. Pao, "Controllability of Formation Systems on Special Orthogonal Groups Over Directed Graphs," *IEEE Trans. Control Netw. Syst.*, vol. 8, no. 2, pp. 872–883, Jun. 2021, doi: 10.1109/TCNS.2021.3050125.
- [17] Y. T. Tsai and J. T. Yu, "Quadrotor UAV with Independent Yaw Control Using Sliding-Mode under Special Orthogonal Group SO(3) Attitude Representation," presented at the 2023 International Automatic Control Conference (CACS), IEEE, 2023. doi: 10.1109/CACS60074.2023.10326160.
- [18] H. Zhang, D. Ye, Y. Xiao, and Z. Sun, "Adaptive Control on SE(3) for Spacecraft Pose Tracking with Harmonic Disturbance and Input Saturation," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 58, no. 5, pp. 4578–4594, Oct. 2022, doi: 10.1109/TAES.2022.3165524.
- [19] P. Coelho and U. Nunes, "Lie algebra application to mobile robot control: A tutorial," *Robotica*, vol. 21, no. 5, pp. 483–493, 2003, doi: 10.1017/S0263574703005149.

- [20] J. Tang *et al.*, “GMPC: Geometric Model Predictive Control for Wheeled Mobile Robot Trajectory Tracking,” *IEEE Robot. Autom. Lett.*, vol. 9, no. 5, pp. 4822–4829, 2024, doi: 10.1109/LRA.2024.3381088.
- [21] M. Stodola, M. Rajchl, M. Brabc, S. Frolík, and V. Křivánek, “Maxwell points of dynamical control systems based on vertical rolling disc—numerical solutions,” *Robotics*, vol. 10, no. 3, 2021, doi: 10.3390/robotics10030088.
- [22] B. Shao, H. Wu, S. Cheng, and S. Yu, “Dynamics and control of robot with active and passive joints using lie groups and lie algebras,” *Zhongguo Jixie GongchengChina Mech. Eng.*, vol. 21, no. 3, pp. 253–257, 2010.
- [23] E. Gürkan, S. P. Banks, and I. Erkmén, “Stable Controller Design for the T-S Fuzzy Model of a Flexible-Joint Robot Arm based on Lie Algebra,” in *Proceedings of the IEEE Conference on Decision and Control*, 2003, pp. 4717–4722.
- [24] P. Van Lam and Y. Fujimoto, “Completed hardware design and controller of the robotic cane using the inverted pendulum for walking assistance,” in *IEEE International Symposium on Industrial Electronics*, 2017, pp. 1935–1940. doi: 10.1109/ISIE.2017.8001547.
- [25] L. H. Ma, Y. B. Zhong, G. D. Wang, and N. Li, “The Kinematic and Dynamic Modeling and Numerical Calculation of Robots with Complex Mechanisms Based on Lie Group Theory,” *Math. Probl. Eng.*, vol. 2021, 2021, doi: 10.1155/2021/6014256.
- [26] T. Low and S. Calinon, “Geometric Algebra for Optimal Control With Applications in Manipulation Tasks,” *IEEE Trans. Robot.*, vol. 39, no. 5, pp. 3586–3600, 2023, doi: 10.1109/TRO.2023.3277282.
- [27] J. Deray and J. Solà, “Manif: A micro Lie theory library for state estimation in robotics applications,” *J. Open Source Softw.*, vol. 5, no. 46, p. 1371, Feb. 2020, doi: 10.21105/joss.01371.
- [28] C. C. Cossette, M. Cohen, V. Korotkine, A. Del Castillo Bernal, M. A. Shalaby, and J. R. Forbes, “navlie: A Python Package for State Estimation on Lie Groups,” in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Detroit, MI, USA: IEEE, Oct. 2023, pp. 5282–5287. doi: 10.1109/IROS55552.2023.10342362.
- [29] C. Armanini, I. Hussain, M. Z. Iqbal, D. Gan, D. Prattichizzo, and F. Renda, “Discrete Cosserat Approach for Closed-Chain Soft Robots: Application to the Fin-Ray Finger,” *IEEE Trans. Robot.*, vol. 37, no. 6, pp. 2083–2098, Dec. 2021, doi: 10.1109/TRO.2021.3075643.
- [30] Kinjoll Dey, “Global Medical Robotics Market Overview,” *Market Research Future*, 2022.
- [31] M. I. Ahmad, M. Khordi-Moodi, and K. S. Lohan, “Social robot for STEM education,” in *ACM/IEEE International Conference on Human-Robot Interaction*, IEEE Computer Society, Mar. 2020, pp. 90–92. doi: 10.1145/3371382.3378291.
- [32] M. Alemdar, J. A. Lingle, and J. Rosen, “Assessing STEM knowledge gain in integrated robotics-based learning environments,” *J. STEM Educ.*, vol. 25, no. 2, pp. 34–42, 2024.
- [33] R. Antunes, M. L. Aguiar, and P. D. Gaspar, “A Dynamic STEM-Driven Approach through Mobile Robotics to Enhance Critical Thinking and Interdisciplinary Skills for Empowering Industry 4.0 Competencies,” *Technologies*, vol. 11, no. 6, p. 170, Dec. 2023, doi: 10.3390/technologies11060170.
- [34] R. Jimenez Gaona and M. Maldonado Gonzalez, “STEAM Skills Developed in Basic General Education Through the Creation of the Digital Robot Through the mBlock Platform,” in *Proceedings - 3rd International Conference on Information Systems and Software Technologies, ICI2ST 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 152–157. doi: 10.1109/ICI2ST57350.2022.00029.
- [35] M. Jones *et al.*, “Learning contexts and visions for STEM in schools,” *Int. J. Sci. Educ.*, vol. 47, no. 3, pp. 337–357, 2025, doi: 10.1080/09500693.2024.2323032.
- [36] F. Casas and B. Owren, “Cost efficient lie group integrators in the RKMK class,” *BIT Numer. Math.*, 2003.
- [37] Y. Wang, C. Xie, Y. Liu, and J. Qin, “A Multi-Sensor Fusion Underwater Localization Method Based on Unscented Kalman Filter on Manifolds,” *Sensors*, 2024, [Online]. Available: <https://www.scopus.com/record/display.uri?eid=2-s2.0 85206334503&origin=scopusAI>
- [38] I. K. Argyros and S. Hilout, “Extending the applicability of Newton’s method on Lie groups,” *Appl. Math. Comput.*, 2013.
- [39] I. K. Argyros and S. George, “Extended semi-local convergence of Newton’s method on lie groups using restricted regions,” *Commun. Appl. Nonlinear Anal.*, 2019.
- [40] E. Bayro-Corrochano and J. Ortégón-Aguilar, “Lie algebra approach for tracking and 3D motion estimation using monocular vision,” *Image Vis. Comput.*, 2007.
- [41] B. Nie, Z. Cai, J. Zhao, and Y. Wang, “Motion transformation solutions based on Euler angle perturbation model,” *Meas. J. Int. Meas. Confed.*, 2025.
- [42] C. Fu and B. J. Falkowski, “Ternary recursive fast transforms: Properties, mutual relations, and circuit realization,” *J. Circuits Syst. Comput.*, 2007.
- [43] C. Fu and B. J. Falkowski, “Two classes of recursive fast transforms over GF(3): Properties, relations and computational costs,” presented at the IEEE Asia-Pacific Conference on Circuits and Systems, Proceedings, APCCAS, 2004.
- [44] A. V. Leont’ev, “Lower bounds for algebraic complexity of classical simple Lie algebras,” *Sb. Math.*, 2008.
- [45] J. C. Benjumea, J. Núñez, and Á. F. Tenorio, “Computing the law of a family of solvable Lie algebras,” *Int. J. Algebra Comput.*, 2009.
- [46] J. C. Benjumea, M. D. Morales, J. Núñez, and A. F. Tenorio, “Computational calculus of the law of a family of solvable Lie algebras,” *J. Comput. Methods Sci. Eng.*, 2012.
- [47] A. Nazarov, “Affine.m - Mathematica package for computations in representation theory of finite-dimensional and affine Lie algebras,” *Comput. Phys. Commun.*, 2012.

- [48] N. Mukhopadhyay, "A discussion on 'Detection of intrusions in information systems by sequential change-point methods' by Tartakovsky, Rozovskii, Blažek, and Kim," *Stat. Methodol.*, 2006.
- [49] S. Dutta, D. Sarkar, and A. Rawat, "Synchronization Validation for Cross-Thread Dependences in Parallel Programs," *Int. J. Parallel Program.*, 2017.
- [50] S. Sharma and A. Majumdar, "Sequential Transform Learning," *ACM Trans. Knowl. Discov. Data*, 2021.
- [51] M. Torres-Torriti and H. Michalska, "A software package for lie algebraic computations," *SIAM Rev.*, 2005.
- [52] F. Iachello, "Spectrum Generating Algebras and Dynamic Symmetries," in *Lecture Notes in Physics*, 2015.
- [53] N. Horigome, A. Terui, and M. Mikawa, "A Design and an Implementation of an Inverse Kinematics Computation in Robotics Using Gröbner Bases," in *Lecture Notes in Computer Science*, vol. 12097, Springer, 2020, pp. 3–17.
- [54] M. Yoshizawa, A. Terui, and M. Mikawa, "Inverse Kinematics and Path Planning of Manipulator Using Real Quantifier Elimination Based on Comprehensive Gröbner Systems," in *Lecture Notes in Computer Science*, vol. 13875, Springer, 2023, pp. 225–237.
- [55] J. Guzmán-Giménez, A. V. Fernández, V. M. Amela, and M. A. Díaz-Rodríguez, "Synthesis of the Inverse Kinematic Model of Non-Redundant Open-Chain Robotic Systems Using Groebner Basis Theory," *Appl. Sci.*, vol. 10, no. 6, p. 2122, 2020, doi: 10.3390/app10062122.
- [56] D. Naderi, M. Tale-Masouleh, and P. Varshovi-Jaghargh, "Gröbner Basis and Resultant Method for the Forward Displacement of 3-DoF Planar Parallel Manipulators in Seven-Dimensional Kinematic Space," *Robotica*, vol. 34, no. 7, pp. 1594–1611, 2016, doi: 10.1017/S0263574715000941.
- [57] P. Donelan and J. M. Selig, "Hyperbolic pseudoinverses for kinematics in the Euclidean Group," *SIAM J. Matrix Anal. Appl.*, vol. 38, no. 4, pp. 1541–1559, 2017, doi: 10.1137/16M1090971.
- [58] P. Donelan, "Invariants for Multi-twists, Screw Systems and Serial Manipulators," in *Springer Proceedings in Advanced Robotics*, vol. 15, 2021, pp. 133–141. doi: 10.1007/978-3-030-50975-0_17.
- [59] M. Carricato, "Classification of the three-dimensional persistent POE manifolds of SE(3)," *Mech. Mach. Theory*, vol. 206, 2025, doi: 10.1016/j.mechmachtheory.2025.105926.
- [60] M. Daher and P. Donelan, "Invariant properties of the Denavit–Hartenberg parameters," presented at the Mechanisms and Machine Science, 2015, pp. 43–51. doi: 10.1007/978-3-319-10723-3_5.
- [61] T. Du, W. Gao, and Y. Chu, "An Improved Minimal Error Model Based on the Adjoint Transformation Using the POE," presented at the 2022 6th International Conference on Robotics and Automation Sciences, ICRAS 2022, 2022, pp. 119–125. doi: 10.1109/ICRAS55217.2022.9842189.
- [62] R. Luo, W. Gao, Q. Huang, and Y. Zhang, "An improved minimal error model for the robotic kinematic calibration based on the POE formula," *Robotica*, vol. 40, no. 5, pp. 1607–1626, 2022, doi: 10.1017/S0263574721001284.
- [63] K. (Kevin M.) Lynch and F. C. Park, *Modern robotics : mechanics, planning, and control*.
- [64] L. Wu, R. Crawford, and J. Roberts, "Geometric interpretation of the general POE model for a serial-link robot via conversion into D-H parameterization," presented at the Proceedings - IEEE International Conference on Robotics and Automation, 2019, pp. 7360–7366. doi: 10.1109/ICRA.2019.8794384.
- [65] B. Wei, S. H. Shu, Y. Zhang, R. Yang, and B. Xing, "Kinematics Analysis and Trajectory Planning Simulation of Manipulator Based on AUBO-I5," presented at the Proceedings - 2020 4th Annual International Conference on Data Science and Business Analytics, ICDSBA 2020, 2020, pp. 72–75. doi: 10.1109/ICDSBA51020.2020.00025.
- [66] P. Raj and D. Pal, "On Stability of Switched Differential Algebraic Equations: A Decomposition-Based Approach," *IEEE Control Syst. Lett.*, vol. 8, pp. 358–363, 2024, doi: 10.1109/LCSYS.2024.3382612.