Simulation Tool for Stationary Hexapod Robots: Visual Design through Kinematic Analysis

Aashish Singh Alag
Mechanical Engineering
Worcester Polytechnic Institute
Worcester, United States of America
aalag@wpi.edu

***Abstract*—This paper develops simulation software designed for stationary hexapod robots. This paper focuses to address the gap in existing tools for simulation that focus primarily on dynamic analysis. This simulation puts emphasis on the calculation and visualization of Forward Kinematics (FK) and Inverse Kinematics (IK). This would allow for accurate predictions of joint angles and end-effector poses based on the varied inputs for design.**

***Index Terms*—Hexapod, Simulation, Inverse Kinematics, Stationary**

I. Introduction

The current field of robotics has seen several advancements in simulation, especially for efficient designing processes and increasing the robot's performance in different environmental conditions. Prior research done by Michel (2004) shows the mobile robot simulation software which supported modeling, programming and simulation of various types of mobile robots [1]. The foundational methods developed by such a platform provide a certain framework for simulation tools. In addition, we can also see the work of Ma Jin et al. (2023) expanding on the work by showing dynamic modeling and simulation with the focus on trajectory prediction [2]. This paper shows the different simulation models. Even after this prior research, there remains a gap in the field of simulation tools for stationary hexapod robots. Existing tools have always dealt with the dynamic simulation, focusing on the movement analysis, leaving a need for a dedicated software that calculates as well as visualizes the outcomes of the Forward Kinematics (FK) and Inverse Kinematics (IK) for stationary applications. By extending the capabilities of such platforms a tool can be built to help people in designing and visualizing the robot. This will also assist in optimizing the joint angles, kinematic calculations, and refining end effector poses based on the different design configurations. This project is inspired by a Bare minimum Hexapod Robot Simulator [5], which simplifies complex motion into somewhat manageable components that can assist with understanding the design and configuration.

II. MODELING

A. Geometric Modeling

Mathematical modeling is one of the most important steps while starting the visualization and simulation of the robot. Here, we start with geometric modeling of the system which defines the physical structure in a mathematical framework which is helpful while allocating poses. The body is modeled as a hexagonal prism as seen in Figure. 1, to which all the legs are attached. This is the ideal starting shape, since it provides symmetry and balance for even distribution across the legs. The coordinate system here is centered at the Center of Gravity (COG) for further simplification of calculation. The body of the robot is defined by three key parameters: f, side length s, and mid-length m. These parameters f show the distance from the center to the front vertices of the hexapod along the x-axis.s sets show the distance from the center to the side vertices along the y-axis. m shows the midpoint of the front and back edges along the x-axis.

Using these parameters, the vertices are calculated as follows:

$$P_0 = (m, 0, 0), \qquad P_3 = (-m, 0, 0),$$
$$P_1 = (f, s, 0), \qquad P_4 = (-f, -s, 0),$$
$$P_2 = (-f, s, 0), \qquad P_5 = (f, -s, 0).$$

This hexapod shape provides symmetry which simplifies the simulation code and makes it more open to change. This also provides a modular design as each side is designed as its own modulus with legs, allowing for easy changes. This shape also provides a certain level of inherent stability which not only supports the stability but also helps with making the simulation environment.

B. Leg Kinematics

We then design the leg of the hexapod simulator which is important to establish the movements. The legs are usually modeled to copy kinematics of creatures like spiders, ants, etc. This consists of three main segments: the Coxa, Femur, and Tibia. Coxa, which is attached directly to the body, acts as the steering mechanism of the robot as it allows for movement in the horizontal direction. Then we have the Femur which is usually the longest segment and is required for the elevation of the robot. Then we have the Tibia, which makes contact with the ground and is crucial for adjusting the stance and grip on the surface. These three segments provide enough degrees of freedom to navigate different terrains.

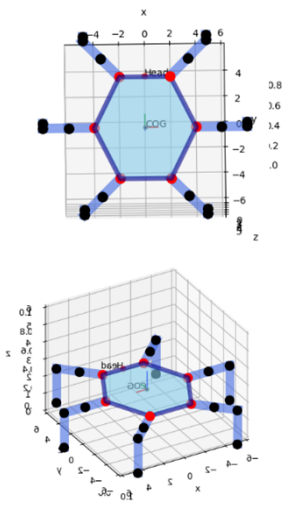

Fig. 1. Initial Geometry of the Hexapod Robot

The kinematic chain for each leg is described by a series of transformations from the robot's body (base frame) to the end of the Tibia (foot). The origin of the Coxa is at the point where it attaches to the body. It rotates around by an angle θ_{coxa}. The Femur extends from the end of the Coxa and rotates around its joint by an angle θ_{femur}. The Tibia, extending from the Femur, adjusts the final position of the leg and rotates by θ_{tibia}.

The position of the leg's end (foot) can be mathematically expressed in terms of the joint angles and the lengths of the segments. Assuming the plane of movement is the X-Y plane, the transformation matrices can be represented as:

$$T_{\text{coxa}} = \begin{bmatrix} \cos(\theta_{\text{coxa}}) & -\sin(\theta_{\text{coxa}}) & 0 \\ \sin(\theta_{\text{coxa}}) & \cos(\theta_{\text{coxa}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & L_{\text{coxa}} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_{\text{femur}} = \begin{bmatrix} \cos(\theta_{\text{femur}}) & -\sin(\theta_{\text{femur}}) & 0 \\ \sin(\theta_{\text{femur}}) & \cos(\theta_{\text{femur}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & L_{\text{femur}} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

$$T_{\text{tibia}} = \begin{bmatrix} \cos(\theta_{\text{tibia}}) & -\sin(\theta_{\text{tibia}}) & 0 \\ \sin(\theta_{\text{tibia}}) & \cos(\theta_{\text{tibia}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & L_{\text{tibia}} \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The following figure illustrates the segmentation and basic joint configuration of a typical hexapod leg. Each segment is labeled to demonstrate the connection and articulation points.

C. Inverse Kinematics

Inverse kinematics (IK) is an important computational step used in robotic systems to determine the necessary joint angles that achieve a desired position of the end effector (in this case, the foot of the hexapod's leg). For the hexapod, this involves calculating the angles at which each leg's joints must be oriented to allow the robot's foot to reach a specific point in space. Unlike forward kinematics, where the positions are computed from given joint angles, inverse kinematics might not have a unique solution, or in some cases, any solution at all, because of the physical constraints of the joints.

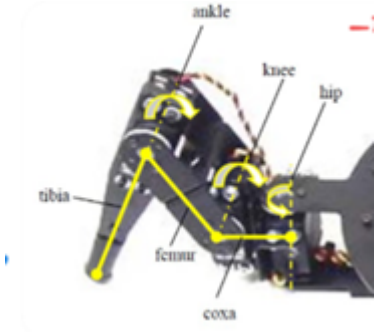

Fig. 2. Leg Components of the Hexapod Robot

Given the leg structure consisting of the Coxa, Femur, and Tibia we can describe the position of the foot in terms of these joint angles. The positions (x, y) of the end-effector can be derived by backward chaining from the foot to the body, considering the lengths of the segments $L_{\text{coxa}}, L_{\text{femur}}, L_{\text{tibia}}$ and the angles $\theta_{\text{coxa}}, \theta_{\text{femur}}, \theta_{\text{tibia}}$.

The position equations for a 2D projection on the X-Y plane are:

$$\begin{aligned} x &= L_{\text{coxa}} \cos(\theta_{\text{coxa}}) + L_{\text{femur}} \cos(\theta_{\text{coxa}} + \theta_{\text{femur}}) \\ &\quad + L_{\text{tibia}} \cos(\theta_{\text{coxa}} + \theta_{\text{femur}} + \theta_{\text{tibia}}), \\ y &= L_{\text{coxa}} \sin(\theta_{\text{coxa}}) + L_{\text{femur}} \sin(\theta_{\text{coxa}} + \theta_{\text{femur}}) \\ &\quad + L_{\text{tibia}} \sin(\theta_{\text{coxa}} + \theta_{\text{femur}} + \theta_{\text{tibia}}). \end{aligned}$$

Due to the non-linear nature of the equations, solving these equations analytically is complex and often not feasible. Therefore, numerical methods such as the Newton-Raphson method or optimization techniques are employed. This approach involves initializing the joint angles with a guess, often based on the last known configuration.Then iteratively adjusting the angles to minimize the difference between the current position of the end-effector and the desired position. Finally using the Jacobian matrix of partial derivatives of the position functions with respect to the joint angles to guide the adjustments.

For a leg with three joints, the Jacobian J is given by:

$$J = \begin{bmatrix} \frac{\partial x}{\partial \theta_{\text{coxa}}} & \frac{\partial x}{\partial \theta_{\text{femur}}} & \frac{\partial x}{\partial \theta_{\text{tibia}}} \\ \frac{\partial y}{\partial \theta_{\text{coxa}}} & \frac{\partial y}{\partial \theta_{\text{femur}}} & \frac{\partial y}{\partial \theta_{\text{tibia}}} \end{bmatrix}$$

This matrix is used to compute the change in angles needed to achieve a small movement toward the target position, typically through the equation:

$$\Delta \boldsymbol{\theta} = J^{-1} \Delta \mathbf{x},$$

where $\Delta \mathbf{x}$ is the vector of differences between the current and desired end-effector positions, and $\Delta \boldsymbol{\theta}$ is the vector of changes in joint angles.

D. Forward Kinematics

Forward kinematics (FK) in robotic systems is the calculation of the position and orientation of the robot's end-effector (e.g., Platform or in this case the foot-tip of a hexapod) based on the specified joint angles without considering forces or physics involved. This process involves the use of kinematic chains.

Each leg of the hexapod robot can be considered as a separate kinematic chain. The legs are articulated with three segments. To calculate the position of each leg's end-effector , we define the transformation matrices for each segment. The total transformation from the base of the leg to the foot is obtained by multiplying the transformation matrices of each segment in sequence. Each joint then contributes a rotational transformation based on its angle, followed by a translational transformation based on the segment's length. The transformation matrix for each segment can be expressed as a product of rotation and translation matrices:

$$\begin{aligned} T_{\text{coxa}} &= \text{Rot}_Z(\theta_{\text{coxa}}) \cdot \text{Trans}(L_{\text{coxa}}, 0, 0), \\ T_{\text{femur}} &= \text{Rot}_Z(\theta_{\text{femur}}) \cdot \text{Trans}(L_{\text{femur}}, 0, 0), \\ T_{\text{tibia}} &= \text{Rot}_Z(\theta_{\text{tibia}}) \cdot \text{Trans}(L_{\text{tibia}}, 0, 0). \end{aligned}$$

Where $\text{Rot}_Z(\theta)$ represents a rotation matrix around the Z-axis by θ degrees, and $\text{Trans}(x, y, z)$ represents a translation matrix that moves the point by x units along the X-axis, y units along the Y-axis, and z units along the Z-axis.

The position of the end-effector in the robot's base frame is then calculated by sequentially applying these transformations from the base of the leg to the foot:

$$T_{\text{foot}} = T_{\text{coxa}} \cdot T_{\text{femur}} \cdot T_{\text{tibia}}$$

This final transformation matrix T_{foot} provides the coordinates of the foot relative to the base of the leg, considering all the joint angles and segment lengths.

In a simulation environment, forward kinematics is used to update the position of each leg's foot in real-time as the joint angles change. This allows for visualizing the robot's walking pattern and testing different gait strategies under various simulated conditions. Accurate forward kinematics calculations are essential for ensuring that the simulated movements correspond to what would be physically possible in a real robot.

E. Stability Analysis

Stability analysis is essential for ensuring that the hexapod robot remains balanced and functional across various operational scenarios and configurations. This analysis focuses on both the static and dynamic aspects of stability, with a primary emphasis on the relationship between the robot's COG and its support polygon (SP), formed by the feet currently in contact with the ground. For stability, the COG must stay within the SP during all times. If the COG falls outside the SP, the robot risks tipping over. For this the instantaneous position of the COG is calculated based on the robot's configuration. To enhance the precision and reliability of stability checks, Barycentric coordinates are utilized. These coordinates help determine if the COG, projected onto the ground plane, remains within the triangle formed by any three of the robot's feet. For this we calculate the normal vector to the plane formed by any three contact points.then project the COG onto this plane to determine its position relative to the SP.

F. Gait Walking Sequence Generation

To simulate the walking motion of the hexapod robot, a gait walking sequence generation function is implemented within the modeling framework. This function allows the specification of various parameters to define the gait pattern, including the type of gait, hip swing angle , lift swing angle, the number of steps, the direction of movement, and rotation angle.

The implemented function makes a walking sequence based on the specified parameters. Currently, in this paper we focus on the Tripod gait. In the Tripod gait, the robot moves by alternating between two sets of three legs, forming a stable tripod configuration at all times. The function generates coordinated movements for each leg, including variations in the angles of the coxa, femur, and tibia joints over the specified number of steps. These sequences are stored in the dictionary, indexed by the leg number.

The coxa angles determine the lateral movement of the leg, while the femur and tibia angles control the lifting and lowering of the leg, respectively. By adjusting these angles over time, the hexapod can achieve forward movement while maintaining stability.

Future work on the simulator will include the implementation of additional gait patterns, such as the Symmetric, Symmetric Forward Wave, and Crab gait, and the integration of more sophisticated gaits based on the real animal motion.

G. Kinematic Phase Diagram

The kinematic phase diagram is another crucial tool for understanding the coordinated movement of the hexapod robot's legs during locomotion. It shows how the phases of leg movement evolve over time, helping in the design of gait patterns for efficient and stable motion.

To generate the kinematic phase diagram, a Python function is implemented. This function takes parameters such as the duty factor, number of legs, and scale in the x-direction to compute the phase diagram array and its dimensions.

The duty factor represents the proportion of time a leg spends in the supporting phase versus the swing phase during a gait cycle. Once the phase diagram array is generated, a visualization function is called to create a heatmap plot using the Plotly library. This plot displays the phase diagram in a graphical format, with the x-axis representing the kinematic phase and the y-axis indicating the individual legs of the hexapod.

The resulting kinematic phase diagram provides valuable insights into the coordination and synchronization of leg movements during various gait patterns. By analyzing and

optimizing these phase diagrams, researchers and engineers can enhance the hexapod robot's locomotion capabilities, improving its performance across different terrains and environments.

III. Visualization and User Interface

For the hexapod robot simulator, an interactive graphical user interface (GUI) is developed using Matplotlib, a versatile plotting library and Dash, which is a framework for building data visualization interfaces that supports complex 3D graphics.

The body of the hexapod, including its head, is visualized using scatter plots for focal points and a 3D polygon collection to depict the body's geometry. Each leg is represented as a series of line segments connected by points that denote the joints and extremities of the leg segments, providing a clear depiction of the hexapod's articulation.

Movement animation for the hexapod is facilitated by an animation function in python, which enables animation by repeatedly calling an update function that recalculates the positions of the hexapod's body and legs based on pose that is calculated . This ensures that the visualization updates reflect the changes in the robot's posture and position, offering real-time feedback on its dynamic behavior.

The GUI allows for interactive control, enabling users to adjust parameters such as gait, swing, rotation and more to observe different behaviors. This interactive capability is coupled with immediate visual feedback on changes to the robot's configuration, making the GUI a great tool for debugging and enhancing the robot's design.

The user interface integrates a control panel that allows users to input or adjust parameters affecting the robot's movement, such as limb rotation and size. Additionally, data outputs are provided to show real-time calculated values, including joint angles, the position of the COG, and stability metrics, increasing the understanding and control over the simulation.

IV. Testing and Results

In this section, we present the testing results for various components of the hexapod robot simulator, including leg pattern generation, inverse kinematics, forward kinematics, kinematic phase diagram, and walking gait simulation. Each subsection details the inputs, outputs, and limitations of the tested functions.

A. Leg Pattern Generation

The Leg Pattern tab in Figure. 3 provides users with the capability to manipulate the angles of the coxa, femur, and tibia joints of the hexapod robot's legs. By adjusting these angles, users can observe how the leg configuration changes in real-time within the graphical user interface. During testing, we found that the Leg Pattern tab effectively allows for intuitive exploration and visualization of different leg configurations. However, it is important to note that this function lacks real-world constraints such as joint limits or collision detection, which may limit its applicability in certain scenarios.

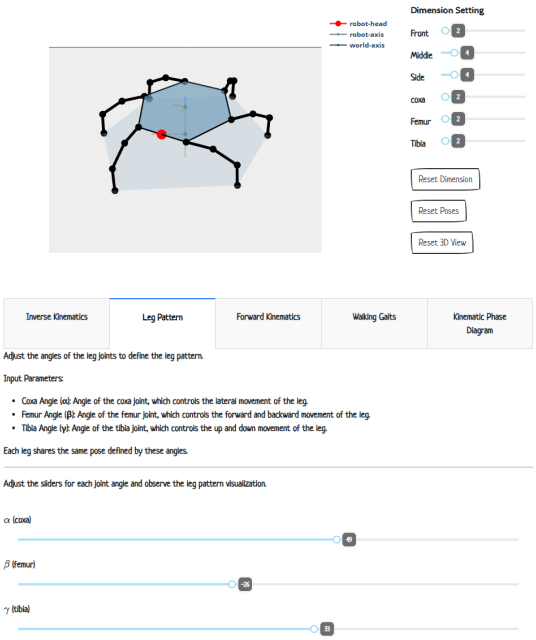

Fig. 3. Visualization of leg pattern generated for tripod gait.

B. Inverse Kinematics

The Inverse Kinematics tab in Figure. 4 enables users to specify desired positions for the end effectors of the hexapod robot's legs. Using mathematical algorithms, the application calculates the corresponding joint angles required to achieve these desired positions. Through testing, we observed that the Inverse Kinematics function successfully computes accurate joint angle solutions for a wide range of end effector positions. This was also compared to other inverse kinematic models to verify the solutions. However, it is important to note that like the previous function, this function also assumes ideal conditions and may not account for practical limitations such as mechanical constraints or joint limits.

C. Forward Kinematics

The Forward Kinematics tab in Figure. 5 allows users to visualize the forward kinematics of the hexapod robot, displaying the resulting end effector positions based on given joint angles. By inputting specific joint angle configurations, users can observe how these angles translate into corresponding end effector positions. During testing, we found that the Forward Kinematics function provides insightful visualizations of the robot's motion.

D. Kinematic Phase Diagram

The tab in Figure. 6 presents a graphical representation of the hexapod robot's kinematic phase diagram, illustrating the phase relationships between different legs during supporting and transfer phase. By adjusting parameters such as duty factor and number of legs, users can observe how these factors influence the robot's gait patterns. In testing, we found that the Kinematic Phase Diagram function offers valuable insights

Fig. 4. Visualization of leg configuration obtained from inverse kinematics.

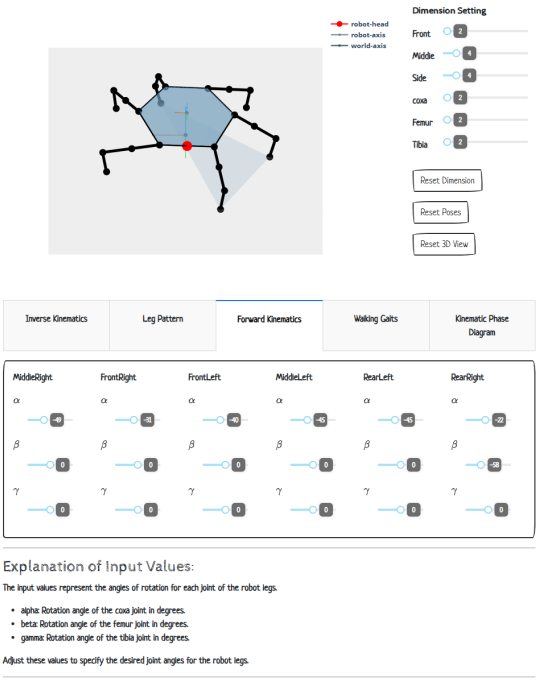

Fig. 5. Visualization of end effector obtained from forward kinematics.

into the robot's locomotion behavior. However, it is crucial to note that this function currently supports only forward symmetric wave gaits, limiting its applicability to other gait types.

Fig. 6. Visualization of kinematic phase diagram for coordinating leg movements.

E. Walking Gait

The Walking Gait tab adapted from the Bare minimum Hexapod simulator project [5] allows users to select and visualize walking gaits such as tripod gaits. By adjusting parameters such as step number and gait speed, users can customize the robot's walking behavior. During testing, we observed several areas for improvement. This function currently supports only tripod wave gaits, limiting its applicability to other gait types.

V. Conclusion

In conclusion, the development of the hexapod robot simulator provides step forward in the field of robotics simulation.

By providing engineers, researchers, and hobbyists with a powerful tool for designing, visualizing, and testing hexapod robot configurations, this simulator can be used in various application domains.

The simulator offers a lot of flexibility in configuring various parameters such as joint angles, leg lengths, and body dimensions, allowing a wide range of design options. This real-time visualization capability is particularly valuable for iteratively refining robot designs and optimizing performance. However, despite these advantages, the simulator may still can be improved in various fields implementing different gaits, including real life constraints and environmental simulation. It can integrate different of gait patterns into the simulator would allow users to simulate various locomotion strategies such as symmetric, symmetric forward wave, and and many more gaits. Additionally, the development of kinematic phase diagrams could provide insights into the relationships between joint angles and gait parameters during different phases of locomotion.

.

REFERENCES

[1] Michel, O. (2004). Cyberbotics Ltd. Webots™: Professional Mobile Robot Simulation. International Journal of Advanced Robotic Systems, 1(1), 39-42. Available at: https://www.cyberbotics.com/doc/guide/index. [2] Jin, M., Ding, L., Gao, H., Su, Y., Zhang, P. (2023). Dynamics Modeling and Simulation of a Hexapod Robot with a Focus on Trajectory Prediction. Journal of Intelligent Robotic Systems, 108(8). https://doi.org/10.1007/s10846-023-01839-w

[3] Wikipedia. (n.d.). Barycentric coordinate system. Retrieved from https://en.wikipedia.org/wiki/Barycentric$_{c}oordinate_{s}ystem$

[4] Wolfram Research, Inc., "Barycentric Coordinates – from Wolfram MathWorld." https://mathworld.wolfram.com/BarycentricCoordinates.html

[5] Mithi, "Mithi/hexapod: Blazing fast hexapod robot simulator for the web.," GitHub, https://github.com/mithi/hexapod (accessed May 1, 2024).

[6] "Vector projection," Wikipedia, https://en.wikipedia.org/wiki/Vector$_{p}rojection(accessedMay1, 2024)$.