Abstract: This paper presents a hybrid force/position control. We developed it for a hexapod walking robot that combines multiple bipedal robots to increase its load. The control method integrated Extenics theory with neutrosophic logic to obtain a two-stage decision-making algorithm. The first stage was an offline qualitative decision-applying Extenics theory, and the second was a real-time decision process using neutrosophic logic and DSmT theory. The two-stage algorithm separated the control phases into a kinematic control method that used a PID regulator and a dynamic control method developed with the help of sliding mode control (SMC). By integrating both control methods separated by a dynamic switching algorithm, we obtained a hybrid force/position control that took advantage of both kinematic and dynamic control properties to drive a mobile walking robot. The experimental and predicted results were in good agreement. They indicated that the proposed hybrid control is efficient in using the two-stage decision algorithm to drive the hexapod robot motors using kinematic and dynamic control methods. The experiment presents the robot’s foot positioning error while walking. The results show how the switching method alters the system precision during the pendulum phase compared to the weight support phase, which can better compensate for the robot’s dynamic parameters. The proposed switching algorithm directly influences the overall control precision, while we aimed to obtain a fast switch with a lower impact on the control parameters. The results show the error on all axes and break it down into walking stages to better understand the control behavior and precision.

Keywords: hybrid position/force control; sliding mode control; decision method; neutrosophic logic; extension set

1. Introduction

Worldwide, practical robot applications are diversifying more and more in the new world of robotics, automation, and artificial intelligence [1]. Researchers and engineers are working on developing solutions and solving problems for all kinds of robots. This research will enhance human motion and workspace investigation through different sensors and automate tasks for unattended robots [2]. Into this category falls every type of robot control method that can improve robot control and behavior, with or without autonomous capabilities [3]. For a robot to be capable of accomplishing a designated task, it must reject most of the uncertainties and disturbances within the work environment. The robot must also handle information from several sensors and fuse the data to reach a close decision to the truth value.

Most mobile robots combine kinematic and dynamic control methods to solve such a problem, each designated for certain joints in the robot structure. However, for a highly versatile robot structure, a hybrid position/force method is used. Although the technique is not new, having begun with Raibert and Craig [4], it has the attention of robotic control research, and it continues to bring adaptability to the robots using it. In recent years, different approaches have been researched. Zhang et al. [5] created a hybrid control method...
that could adjust the joint dynamic parameters online. The process allowed rough modeling of the robot parameters and left the fine-tuning to the control algorithm. A similar approach used a neural network to autocalibrate the control parameters [6]. Still, not all hybrid methods can use mechanical parameters because they increase the system’s complexity. For a robot with a simple structure for which the kinematic or dynamic equations are easier to define, a classic approach to hybrid control can be easier to implement. Some examples include the hybrid method with impedance control [7] or even the backstepping method with a Hamilton controller [8]. Applications of the hybrid control method can be found in all types of robots, from a robot used for mechanical tests [9] to an upper limb rehabilitation robot [10]. Because of its versatility, the hybrid position/force control was chosen as the primary control method for the walking robot.

As we know, hybrid control combines a kinematic control method for joints that do not require compensation of weight and inertia and a dynamic control method that can handle these parameters and reject environmental disturbances. The kinematic approach is used for positioning control of the robot and the dynamic approach for the force and torque control. Consequently, a classic proportional-integrative-derivative (PID) regulator was chosen for the position control and a sliding mode control (SMC) method [11] for the force and torque control. The main reason for using a SMC method was its robustness in the presence of external disturbances and uncertainties. Many scientists have used the control method to improve industrial robot trajectory [12], mobile robot trajectory in dynamic environments [13], n-link serial manipulator control [14], balance control of a two-wheel robot [15], and even airplane fuselage inspection [16]. SMC is not a perfect control method, and it has drawbacks, one of which is the chattering effect that it can introduce. New research is published every year [17] on eliminating the chattering impact in a general manner or for a specific robot structure or purpose.

Using multiple regulators or control methods on the same robot structure can separate the robot joints statically into two categories, starting with the design of the control law. However, this is not desirable if one needs to build a versatile robot. Hence, a real-time decision method must determine the degree of freedom (joints) controlled by each method. A combination of techniques and control methods was thus selected. The first one, Extenics [18] or extension logic [19], entails defining the control parameters and robot properties or abilities and is used by scientists to configure problem-solving algorithms [18] and even design toys for children with special needs [20].

For the proposed robot structure, Extenics helped ease the process of organizing the parameters of each control method and provided the offline means of solving potential conflicts, uncertainties, or mismatching of sensor data and regulators.

Neural networks [21,22] were considered for the decision method, but the process of training the network was too extensive for the proposed robot. Another possibility was using swarm optimization [23] to predict what the robot needs in terms of control methods. This also overcomplicated the control system, and it should be handled in future work. For the presented robot, neutrosophic logic [24] and the Dezert–Smarandache Theory (DSmT) [25] were chosen. DSmT combined with Extenics is used to manage decision making [20] and provides excellent results by combining the mapping process of Extension logic with the sensor fusion of DSmT in uncertain and contradictory conditions. As an extension of fuzzy logic, neutrosophic logic and DSmT have been used by researchers to develop applications [26] for aviation parking [27], multi-UAV surveillance [28], obstacle avoidance in unknown environments [29], and environmental detection and estimation [30].

A different approach in designing the decision algorithm of a hybrid system is using time triggers or an event-driven mechanism [31] with an event generation mechanism [32] to ensure control of the system at the precise times of important defined events. This approach is safe for robots in a known environment, but it can fail or make inconsistent decisions for robots moving inside unknown and unstructured environments.

To develop a mobile walking robot, one can reference many highly advanced robots, some designed by renowned institutes [33], that use dynamic control methods to provide
stability and error rejection for the control architecture. While efficient mathematical solutions [34] are desired for a control law to give it low computational requirements, these can be difficult to obtain when the robot model is complex. Moreover, the dynamic control of any robot must overcome external disturbances [35] and reject any influence from other sources, including within the sensor information.

Here, we propose and describe a mobile walking robot hybrid position/force control that can be used within a group of linked robots. The research aim to obtain a control algorithm and method using both kinematic and dynamic control methods and an intelligent switching method between them. As a result, several experiments were conducted to improve the performance of the developed hybrid control, taking advantage of the extension set and neutrosophic logic. The extension set and neutrosophic logic were used to enhance the decision making required by the hybrid control, the first as an offline set of characteristics to extend the system’s definition, and the second as an online switching mechanism that works with uncertain and contradictory information. The resulting hybrid control using a two-stage decision algorithm was a robot control method that took advantage of the best properties of the kinematic and dynamic control laws while the robot was fulfilling its tasks in uncertain environments. The data fusion provided by the neutrosophic theory in contradictory or uncertain conditions improved the decision switching mechanism, while the overall reference tracking of the robot did not decrease. The computational requirements of the proposed hybrid control were reduced because of the kinematic control method when the robot did not require dynamic compensation.

The paper is divided into six main sections. Section 2 provides a visual description of the robot used in the experiments. Section 3 presents the offline decision using the extension set, while Section 4 presents the decision method based on neutrosophic logic that takes advantage of DSmT [25]. The hybrid control is presented in Section 5 with an in-depth analysis of the kinematic (Section 5.1) and SMC dynamic (Section 5.2) control methods. Sections 6 and 7 contain the conducted experiments and simulations with the obtained results. In the end, Section 8 presents this paper’s conclusions.

2. System Description

Figure 1 presents the robot structure. The robot was a hexapod [36], and its design was selected to avoid stability problems. Future research will consider the stability of a single bipedal mobile walking robot. As can be seen, the robot platform was divided into three modules, resulting in a modular robot that could be further extended or reconfigured. Figure 1b presents the kinematic structure of the hexapod robot leg. Each leg had three degrees of freedom, ensuring the 3D positioning of the foot.

![Figure 1. The robot structure: (a) the hexapod walking robot; (b) kinematic structure of the robot leg.](image-url)
In the robot structure, aluminum was considered for rectangular bars with height, width, and length dimensions $h_i \times w_i \times l_i$ and weight $m_i$. $h_i$ and $w_i$ with the dimensions of $h_i = 3 \text{ cm}$ and $w_i = 3 \text{ cm}$. Table 1 presents the robot dimensions used in the testing and simulations presented throughout the paper.

Table 1. Robot values.

| Dimension | Value     |
|-----------|-----------|
| $r_1$     | 0.05 [m]  |
| $l_1$     | 0.1 [m]   |
| $m_1$     | 0.25 [kg] |
| $r_2$     | 0.25 [m]  |
| $l_2$     | 0.5 [m]   |
| $m_2$     | 1.25 [kg] |
| $r_3$     | 0.35 [m]  |
| $l_3$     | 0.7 [m]   |
| $m_3$     | 1.76 [kg] |

$l_i$ and $r_i$ represent, respectively, leg segment dimensions and the distance from the joint axis to the center of mass for each leg segment, and $m_i$ is the segment’s mass.

3. Extenics Theory and Extension Set Applied to Robots

Extenics is a scientific field that uses modeling and formal methods to extend elements or physical objects. The models and methods are then used to solve contradictory problems that cannot be solved in their defined form and conditions [37,38].

Because contradictory problems are omnipresent in any field, Extenics aims to define a set of methods that allows solving contradictory issues using virtual simulation with the help of computers.

The central parts of extension theory are the base element theory, extension set theory, and extension logic [39].

The fundamental element used to describe objects in extension theory is defined as:

$$M = (O_m, c_m, v_m)$$

where $M$ is the object element for which $O_m$ is the object, $c_m$ is the characteristic, and $v_m$ is the measure. If one characteristic exists, the other matter element can be of only one dimension. If the object has more characteristics, however, the multidimension matter element is be defined as:

$$M = \begin{bmatrix} O_m, & c_{m1}, & v_{m1} \\ c_{m2}, & v_{m2} \\ \vdots & \vdots & \vdots \\ c_{mn}, & v_{mn} \end{bmatrix} = (O_m, C_m, V_m).$$

If a hybrid control is used, then a multidimension matter element is used, described by many characteristics specific to the chosen control methods.

For a hybrid position/force control, the following matter element can be defined:

$$R_0 = \begin{bmatrix} \text{Robot Control} & \text{Control Type} & \text{Hybrid Control} \\ \text{Overall Computation Speed} & \text{Good} & \text{Very Good} \\ \text{Reference Tracking Speed} & \text{Very Good} & \text{Good} \\ \text{Reference Tracking Error} & \text{Good} & \text{Good} \\ \text{Inertia Compensation} & \text{Good} & \text{Good} \\ \text{Disturbance Rejection} & \text{Good} & \text{Good} \end{bmatrix}.$$
By using Extenics principle 2.2 [40], which Ren et al. [18] used to design low carbon products, the object from Equation (3) can be extended and decomposed into two matter elements for which \( O_1 \) and \( O_2 \) are defined as kinematic control and dynamic control, respectively. The two objects have the same characteristics as the primary object (called “hybrid control”) but with different values:

\[
R_1 = \begin{bmatrix}
\text{Robot Control,} & \text{Control Type} & \{ \text{Kinematic Control} \} \\
\text{Overall Computation Speed,} & \text{Very Good} \\
\text{Reference Tracking Speed,} & \text{Very Good} \\
\text{Reference Tracking Error,} & \text{Good} \\
\text{Inertia Compensation,} & \text{Very Poor} \\
\text{Perturbation Rejection} & \text{Poor} \\
\end{bmatrix}, \quad (4)
\]

\[
R_2 = \begin{bmatrix}
\text{Robot Control,} & \text{Control Type} & \{ \text{Dynamic Control} \} \\
\text{Overall Computation Speed,} & \text{Average} \\
\text{Reference Tracking Speed,} & \text{Good} \\
\text{Reference Tracking Error,} & \text{Very Good} \\
\text{Inertia Compensation,} & \text{Very Good} \\
\text{Perturbation Rejection} & \text{Very Good} \\
\end{bmatrix}. \quad (5)
\]

As can be seen, the two matter elements from Equations (4) and (5) describe the control types, the kinematic and the dynamic, briefly. The matter elements are customized for control types, but numerous other features can be added for which the matter characteristics are different according to the desired control type.

The contradiction between the two control types is found using the matter element characterization. The kinematic type has a better computational speed for a real-time controller. Still, when a robot is subject to inertial forces, it has worse positioning error and tracking speed. On the other hand, the dynamic control method takes into consideration the inertial forces that act on a robot and has a better tracking error. However, although the tracking error is better, the tracking speed is worse. The overall computational speed is greatly diminished, owing to the many calculations inside the control loop.

As a simple reference trajectory, an ideal trajectory of the foot (Figure 2) was used. When a robot foot has the role of support (the unbroken line in Figure 2), precise control is needed that considers the weight and inertia of the robot, so the robot’s position does not oscillate on the vertical axis during the support phase. Additionally, the joints of a robot leg must complete or partially support the overall robot weight, including the other legs in the advancing stage. The dotted line in Figure 2 represents the leg balance trajectory when a robot takes a step for which the positioning is not required to be precise but must be fast and smooth. During the second motion phase, the leg joints support only their leg weight.

![Figure 2](image)

**Figure 2.** A simple trajectory for the robot foot.
Knowing the gross characterization of the two types of trajectories that a robot foot takes, the problem of choosing the control type is solved by checking the properties of the two matter elements defined by Equations (4) and (5).

Thereby, to control the robot during the uniform pull and weight support phase, the matter element \( R_2 \) was chosen. Its properties provided better precision in tracking the position reference and considered the robot’s inertia to compensate for the robot’s weight and inertial motion forces.

When the robot foot must follow a curve in space during the advancing phase, the robot’s weight was not supported by it, so we used matter element \( R_1 \). The \( R_1 \) properties better corresponded to the robot motion criteria. The kinematic controller was used when the robot needed to make a forward or reverse motion that was not in direct contact with the support plane. High positioning precision was not required for the advancing movement, only a faster speed to position the foot as quickly as possible on the next support point.

One of the properties of the two matter elements is “overall computation speed,” which indicates how many mathematical operations are needed to compute the actual reference for each separate joint. Therefore, a kinematic controller is much more efficient in computational requirements, performing fewer mathematical operations than a dynamic one. Results have indicated that a kinematic controller supplies not only a better tracking speed but also minimal resource consumption.

When defining and separating the control type that is best for the job, a real-time switching method was required. Thus, with the help of neutrosophic logic, the robot leg phase was determined, based not on reference values but on sensor information. According to the data calculated by Extenics theory on which type of control law to use, a hybrid control could be obtained to resolve the transition problem between kinematic and dynamic control laws. As an offline result, it could be reiterated for future datasets to enhance control properties or add a second layer of details and properties.

4. Neutrosophic Logic in Robot Control

As defined in [41], neutrosophic logic is the foundation of neutrosophic mathematics. Neutrosophic logic works with neutrosophic sets that generalize fuzzy sets and describe neutrosophic elements. The elements are based on \(<A>, <anti A>, \text{ and } <\text{neutral } A>\), where \(<A>\) is an attribute, \(<\text{anti } A>\) is the opposite of the attribute, and \(<\text{neutral } A>\) is the neutral area between \(<A>\) and \(<\text{anti } A>\).

In neutrosophic logic, every affirmation \( Af \) is T\% true, I\% undetermined (uncertain), or F\% false. Therefore, we can say \( Af (T, I, F) \), where \( T, I, \) and \( F \) are standard or non-standard subsets of the interval \([-1, 1]\) [41].

If \( U \) is the work universe and \( M \) is a set included in \( U \), then one element \( x \) from \( U \) is written as \( x (T, I, F) \) according to set \( M \) and belongs to the same set in the following way: element \( x \) is \( t\% \) true in set \( M \); element \( x \) is \( i\% \) undetermined in set \( M \) (either true or false); and element \( x \) is \( f\% \) false in set \( M \). The value of \( t \) varies in \( T \), \( i \) varies in \( I \), and \( f \) varies in \( F \) [42,43].

As described in the current paper, the robot control diagram presented used both kinematic and dynamic elements. At a specific time, the robot used only one of the control methods to maximize and optimize computing and motion speed or the positioning error. A precision element was needed to switch between the two control types. Using Extenics and extension theory, the contradictory elements were defined to separate the two control types between which the decision algorithm switched using neutrosophic theory.

The classic neutrosophic theory [25] chooses between the two control methods. The general equation is presented in Relation (6) and defines the generalized basic belief assignment:

\[
    m(C) = \sum_{A, B \in D^\phi} m_1(A) \cdot m_2(B), \forall C \in D^\phi
\]

\[
    A \cap B = C
\]
where \( D^\Theta \) is a hyperpower set from the frame \( \Theta = \{\theta_1, \theta_2, \ldots, \theta_n\} \) of \( n \) exhaustive elements and \( A, B \in 2^\Theta \). The basic belief assignment is \((\cdot): 2^\Theta \rightarrow [0,1]\), where \( 2^\Theta = \{\emptyset, \theta_1, \theta_2, \theta_3, \theta_1 \cup \theta_2, \theta_1 \cup \theta_3, \theta_2 \cup \theta_3, \theta_1 \cup \theta_2 \cup \theta_3\} \) when \( \Theta = \{\theta_1, \theta_2, \theta_3\} \).

In the case of the presented robot, two belief assignments were assigned to the two observers. The two observers were the force and proximity sensors that must determine which type of control was required at one time.

The experimental data for the two observers are presented in Table 2. These values divided the sensors’ measurement interval in a decision percentage, where the force sensor was more likely to decide on a dynamic control (75%) than the proximity sensor (65%). The rate was reversed for the kinematic control and was the same for the uncertain interval.

### Table 2. Observer experimental data.

| \( \theta \)       | Force Sensor \( m_1(\theta) \) | Proximity Sensor \( m_2(\theta) \) |
|--------------------|-------------------------------|-----------------------------------|
| \( \theta_D^* \)   | 0.75                          | 0.65                              |
| \( \theta_C^{**} \) | 0.2                           | 0.3                               |
| \( \theta_D \cap \theta_C^{***} \) | 0.05                        | 0.05                             |

* \( \theta_D \) = dynamic control; ** \( \theta_C \) = kinematic control; *** \( \theta_D \cap \theta_C \) is the indeterminate area.

The values meant that the decision could be computed with a certain approximation by using Equation (6) if the robot was in contact with the support surface, according to the data received from the sensors, and a decision was made whether it would switch from one control type to another. The kinematic control type was used in the foot balancing phase and the dynamic control type in the support phase. The decision was made between the two contradictory objects, defined with the help of Extenics and extension theory.

Table 3 presents the cases in which the neutrosophic values \( m_1(\theta_D), m_1(\theta_C), m_2(\theta_D), m_2(\theta_C), m_1(\theta_D \cup \theta_C), \) or \( m_2(\theta_D \cup \theta_C) \) can be found in any combination for \( A \) and \( B \) to correspond to Equation (6), meaning that \( A \cap B = C \). The results were obtained using Equation (6) and represent the neutrosophic probabilistic values of truth (certainty of a valid value), falsity (assurance of a false value), uncertainty (the unknown state between two possible outcomes), and contradiction (two observers provide contradictory information with high certainty for both).

### Table 3. The experimental data after using Equation (6).

| \( C = A \cap B \)                  | \( m(C) \)                  |
|------------------------------------|-----------------------------|
| \( \emptyset \)                    | 0                           |
| \( \theta_D \)                     | 0.5575                      |
|                                   | Truth value for \( \theta_D \) and falsity value for \( \theta_C \) |
| \( \theta_C \)                     | 0.085                       |
|                                   | Truth value for \( \theta_C \) and falsity value for \( \theta_D \) |
| \( \theta_D \cap \theta_C \)      | 0.0025                      |
|                                   | Uncertainty between \( \theta_C \) and \( \theta_D \) |
| \( \theta_D \cup \theta_C \)      | 0.355                       |
|                                   | The contradiction between \( \theta_C \) and \( \theta_D \) |

The values from Table 3 were computed in Equation (7).

\[
m(\emptyset) = 0
\]

\[
m(\theta_D) = m_1(\theta_D) \times m_2(\theta_D \cup \theta_C) + m_1(\theta_D \cup \theta_C) \times m_2(\theta_D) + m_1(\theta_D) \times m_2(\theta_D) = 0.5575
\]

\[
m(\theta_C) = m_1(\theta_C) \times m_2(\theta_D \cup \theta_C) + m_1(\theta_D \cup \theta_C) \times m_2(\theta_C) + m_1(\theta_D) \times m_2(\theta_C) = 0.085
\]

\[
m(\theta_D \cup \theta_C) = m_1(\theta_D \cup \theta_C) \times m_2(\theta_D \cup \theta_C) = 0.0025
\]

\[
m(\theta_D \cap \theta_C) = m_1(\theta_D) \times m_2(\theta_C) + m_1(\theta_C) \times m_2(\theta_D) = 0.355
\]

where \( m(\theta_D) \) and \( m(\theta_C) \) are the probabilistic values of certainty to choose a certain control law; \( m(\theta_D \cup \theta_C) \) is the probabilistic uncertainty value of the two sensors; and \( m(\theta_D \cap \theta_C) \) is...
the probabilistic contradiction values between the two sensors. As a test, when all five values are added, their sum must be equal to 1 (100%): 

\[ m(\varphi) + m(\theta_D) + m(\theta_C) + m(\theta_D \cup \theta_C) + m(\theta_D \cap \theta_C) = 1. \]

Using the computed values, each observer’s decision (force and proximity sensor) had a certain probability that each of the two control systems required to control the robot.

Table 4 presents all cases presented in Figure 3a,b for the force sensor and proximity sensor where X and \( \alpha \) were defined according to the sensor type.

### Table 4. Control probability.

| \( C = A \cap B \) | \( A \) (Force Sensor) | \( B \) (Proximity Sensor) | \( m (C) \) | Control Type |
|---------------------|------------------------|-----------------------------|-------------|--------------|
| \( \varphi \)        | \( \varphi \)          | \( \varphi \)                | 0           | Robot stopped |
| \( \theta_D \)       | \( \theta_D \)         | \( \theta_D \cup \theta_C \) | 0.5575      | Dynamic Control |
|                      | \( \theta_D \cup \theta_C \) | \( \theta_D \)            |             |               |
| \( \theta_C \)       | \( \theta_C \)         | \( \theta_D \cup \theta_C \) | 0.085       | Cinematic Control |
|                      | \( \theta_D \cup \theta_C \) | \( \theta_C \)            |             |               |
| \( \theta_D \cap \theta_C \) | \( \theta_D \cap \theta_C \) | \( \theta_D \cap \theta_C \) | 0.0025 | Uncertainty |
| \( \theta_D \cap \theta_C \) | \( \theta_D \cap \theta_C \) | \( \theta_D \cap \theta_C \) | 0.355    | Contradiction |

**Figure 3.** Control-deciding graphs: (a) force sensor graph; (b) proximity sensor graph.

For the last two cases in Table 4, the \( C = \theta_D \cup \theta_C \) or \( C = \theta_D \cap \theta_C \) uncertainty in decision making was due to the sensor values, leading to a contradiction. If, in the case of uncertainty, the control type running at that time could be kept, in the case of contradiction between sensor data, a decision must be made on which control should be used. Because the contradiction could appear only under specific conditions, a decision was made to use the same type of control as the robot in the case of uncertainty.

One exceptional or typical case is the robot stepping on very uneven ground. The force sensor indicates that the foot is on the floor, but the proximity sensor does not provide the same conclusion since it reads a value greater than the reference threshold. Therefore, the decision should be to switch to a dynamic controller. On the other hand, if a kinematic control is used and the foot is subject to external factors, the force sensor records high peak values in short periods, leading to the chattering effect. The algorithm switched from kinematic to dynamic control for any case of uncertainty. To prevent additional chattering effects, the algorithm switched the control method when the force sensor retained its
contradictory value for a minimum $\Delta t$ time interval. The time threshold provided a precision control law in uneven terrain and contradictory cases between input sensors and observers.

A supplementary condition was required in addition to the selected requirements for the control type. The condition was bound to the way the robot moves. Because the dynamic control was slower to compensate for high errors and its stationary points were unnecessary, we chose the control law based on robot kinematics to save computing time.

One could argue that the switching control law is unnecessary and uses simple triggers that act as switching mechanisms. However, a simple control switch cannot decide between options when the information received is inaccurate, which is one of the main reasons the neutrosophic switching mechanism was chosen and used.

5. The Walking Robot Leg Control Architecture

To control a walking robot, one has to design a control law for each leg, and the control has many walking phases that depend directly on the reference signal of the foot. Therefore, the design of a general control law is needed to control foot position and the motor’s torque according to the computed reference and to use the sensor signal (force and proximity) for environmental interaction and detection.

Figure 4 presents the general control diagram for one leg of the walking robot. The graph contains a reference generation block to generate the foot trajectory using detailed data chosen to test the control law.

![Figure 4. The general control diagram.](image)

The reference generation was made in the operational space, and the data were converted to the joint space from the operational space by using inverse kinematics. An inverse kinematics algorithm based on the Jacobian transpose was used and is presented in

\[
\Delta e = e_{\text{goal}} - e_{\text{real}}
\]

\[
J_{\text{d}} = J \times J^T \times \Delta e
\]

\[
\alpha = \Delta e^T \times J_{\text{d}}
\]

\[
\Delta \theta = (\alpha \times J^T \times \Delta e)^T
\]
Equation (8). Compared to other algorithms, it provides a reference speed for the leg joint motors and not the angular position reference.

\[
\begin{align*}
\Delta e &= e_{\text{goal}} - e_{\text{real}} \\
\alpha &= \frac{\Delta e^{T} \times JTde}{JTde^{T} \times JTde} \\
\Delta \theta &= \left( \alpha \times J^{T} \times \Delta e \right)^{T}
\end{align*}
\]

The speed reference value cannot be used to control the robot joints by the dynamic controller because the dynamic controller needs the angular reference for all the degrees of freedom it controls, and this is the reason why the angular values for each joint were computed using the foot position as the origin. The equations are:

\[
\begin{align*}
q_{1} &= \arctan\left( \frac{My}{Mz} \right) \\
q_{2} &= 2 \times \arctan\left( \frac{\sin q_{3}}{\sin^{2} q_{3} + \cos^{2} q_{2}} \right) \\
q_{3} &= \arctan\left( \frac{-\sin q_{3}}{\cos q_{3}} \right)
\end{align*}
\]

where the sine and cosine values are given by

\[
\begin{align*}
\cos q_{2} &= \frac{(Mz-l_{1}) \times (l_{2}+l_{3} \cos q_{4}) + Mx \times l_{3} \sin q_{3}}{(Mz-l_{1})^{2} + Mx^{2} + My^{2}}, \\
\sin q_{2} &= \sqrt{1 - \cos^{2} q_{2}}, \\
\cos q_{3} &= \frac{(Mz-l_{1})^{2} + Mx^{2} + My^{2} - l_{2}^{2} - l_{3}^{2}}{2l_{3}}, \\
\sin q_{3} &= \sqrt{1 - \cos^{2} q_{3}}.
\end{align*}
\]

For Equation (9) to be valid and to condition the leg posture, additional conditions were added:

\[
\begin{align*}
1: & \quad \text{if} \quad My = 0 \quad \text{then} \quad q_{1} = 0 \\
2: & \quad q_{1} \in \left( -\frac{\pi}{2}, \frac{\pi}{2} \right) \\
3: & \quad q_{3} \leq 0
\end{align*}
\]

The two sensors’ data (proximity and force) were used as input signals for the neutrosophic block to decide. Because generated information was used, the two sensors were simulated. Therefore, the proximity sensor had a function based on the calculated distance from the foot to the support surface considered a plane, but to which a sinusoidal signal was added to generate the measurement error of the sensor. Regarding the force sensor, the foot–ground interaction was simulated using the system from Figure 5. The simulation was achieved with the help of a damper and a spring.

Figure 5. Interaction between the robot and ground surface.
The equation used for contact modeling and determining the reaction force of ground interaction was the classical one:

\[ F_{\text{tot}} = -(k \cdot x + c \cdot \dot{x}) \]  

(12)

where \( k \) and \( c \) are the constants of the spring and damper, respectively.

Having the two parameters, reaction force and proximity distance, as inputs for the decision method, the two control methods are defined in the following sections.

5.1. The Kinematic Control Method

This method used the data provided by the computing algorithm of inverse kinematics (Figure 6) and fed the output to the PI (proportional-integrative) regulator that drove the robot joint motors.

![Figure 6. The kinematic control diagram.](image)

As previously described in the Extenics method, the control method has fewer calculations. Still, the positioning error is not the best because of the inverse kinematics method. It does not consider the inertial force that the robot experiences during the actual motion.

The main component of the controller is the Jacobian matrix:

\[
J = \begin{bmatrix}
-s_1(l_2s_2 + l_3s_{23}) & c_1(l_2c_2 + l_3c_{23}) & l_3c_1c_{23} \\
-c_1(l_2s_2 + l_3s_{23}) & s_1(l_2c_2 + l_3c_{23}) & l_3s_1c_{23} \\
0 & -l_2s_2 - l_3s_{23} & -l_3c_{23}
\end{bmatrix}
\]  

(13)

where \( s_i = \sin(\theta_i) \), \( c_i = \cos(\theta_i) \), \( s_{ij} = \sin(\theta_i + \theta_j) \), and \( c_{ij} = \cos(\theta_i + \theta_j) \).

The matrix was computed from the direct kinematics equations and was used to find the foot position in the operational space:

\[
M(x, y, z) = \begin{bmatrix}
c_1(l_2s_2 + l_3s_{23}) \\
 s_1(l_2c_2 + l_3c_{23}) \\
l_1 + l_2c_2 + l_3c_{23}
\end{bmatrix}
\]  

(14)

The entire kinematic control loop was based on the Jacobian matrix. First, the matrix containing the actual angular joint position was calculated. After it followed its transpose matrix and the operational space reference position, the positioning error \( \Delta \theta \) was obtained.

The positioning error was sent to two PI (proportional-integrative) feedback control loops for controlling the angular speed and motor torque. Thereby, the torque control for each joint was obtained, and the switch from one controller to another (from the kinematic control to the dynamic one, and vice versa) was more accessible since they both used torque to control the robot joints.
The transpose Jacobian method is not new and is based on using the transpose matrix of the Jacobian instead of the inverse matrix. Therefore, $\Delta \theta$ was computed using Equation (15):

$$\Delta \theta = a J^T e$$

for specific values of constant $\alpha$.

The transpose-Jacobian-matrix-based algorithm presented in Equation (8) eliminated stability problems. The algorithm was also chosen because it had a higher computation speed than the control values of other algorithms, even if the computed values were not as precise as the inverse-Jacobian-matrix-based method [44].

Because the method of solving the inverse kinematics problem uses the Jacobian matrix, the final results are always formed by angular speeds that the robot joints must follow. Therefore, the control is suitable for PI and PID regulators and for controlling angular velocities. The downside is that the method cannot be used to compute a dynamic control reference since it needs a precise joint angular value. In contrast, if the values given by the Jacobian-based inverse kinematic problem are integrated, the result is not as accurate as is required.

5.2. The Dynamic Control Method

The dynamic control method used the same reference data as the kinematic one. Nevertheless, it computed the torque reference of the motors considering kinematic parameters, the inertial ones provided by the inertia matrix, and the Coriolis and gravity force effects supplied by the Coriolis and gravity matrices.

Figure 7 presents the dynamic control diagram. The most critical control blocks are shown, including those that compute the inertial parameters and values used by the slide control block. The three control blocks that formed the dynamic controller from Figure 7 were the PID error controller, the fuzzy controller, and the slide control. The first block passed the positioning error through a PID controller so that the control method could consider the error variations. Using the PID error controller data, the fuzzy amplification was obtained through the membership functions presented in Figure 8a,b. The command torque for each motor joint could be calculated after computing the fuzzy gain, using the inertial data and the reference values [45].

![Figure 7. The dynamic control scheme.](image)

All the control values were computed from the presented robot structure, characteristics, structural weights, and measurements.

For Figure 8a,b the abbreviations are N = Negative, P = Positive, ZE = zero, S = Small, M = Medium, B = Big, and V = Very. For the membership functions, Table 5 presents the values for the membership parameters so the gain value $K_{fuzzy}$ could be chosen. The membership functions that provided the gain were selected according to the values of the two parameters $s$ and $\hat{s}$, where $s$ represents the error through the PID error controller and $\hat{s}$...
is its derivate. A constant gain was not desired for each case, leading to a standard-step fuzzy controller, but a function-based one was selected.

![Membership functions](image)

**Figure 8.** Membership functions: (a) member function for input $s$; (b) member function for input $\dot{s}$.

**Table 5.** The output fuzzy gain computation.

| $s$  | $\dot{s}$ | NB   | NM  | NS  | Z   | PS  | PM  | PB  |
|------|----------|------|-----|-----|-----|-----|-----|-----|
| $\dot{s}$ | $s < -10$ | S    | VS  | S   | M   | B   | VB  | VB  |
| $N$   | $-10 \leq \dot{s} < 0$ | M    | VS  | S   | M   | B   | VB  | VB  |
| $Z$   | $\dot{s} = 0$ | B    | M   | S   | VS  | S   | M   | B   |
| $PS$  | $0 \leq \dot{s} \leq 10$ | VB   | B   | M   | S   | VS  | S   | M   |
| $PB$  | $10 < \dot{s}$ | VB   | VB  | B   | M   | S   | VS  | S   |

Using Table 5 data, the parabola in Figure 9 was considered for computing the $K_{fuzzy}$ gain, according to the two inputs $s$ and $\dot{s}$. The parabola equation was computed from Equation (16):

$$y(x) = 2x^2 + 50,$$

and we modified it to introduce the fuzzy parameters:

$$K_{fuzzy}(\dot{s}) = 2(\dot{s} - 10 \cdot s)^2 + 50.$$  

![Parameter $K_{fuzzy}$](image)

**Figure 9.** Parameter $K_{fuzzy}$ for $s = 0$ and $\dot{s}$ between −20 and 20.
Equation (15) now provides the $K_{\text{fuzzy}}$ parameter in the dynamic control.

The sliding control was made with the help of the slide control block (Figure 7). The control type was inspired by Shafiei [12] and modified to match the robot kinematic structure used, a design with three degrees of freedom instead of the two used by Shafiei [12]. Following that, the dynamic equations that allowed the dynamic controller’s development are presented.

The basic dynamic control equation was:

$$H(q) \ddot{q} + C(q, \dot{q}) \dot{q} + G(q) + \tau_d = \tau.$$  \hspace{1cm} (18)

From Equation (18), the signal for motor torque control was calculated. All the parameters from Equation (18) are required to be known. The unknown values are the torque $\tau$, the matrices $H$ (inertial parameters), $C$ (Coriolis and centrifugal forces), and $G$ (gravity effect), which are given by the following equations, in which the angles $\theta$ from the joint space are equal to the ones in the operational space:

$$H = M = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix}$$ \hspace{1cm} (19)

where $M$ is the inertial parameters matrix.

The inertial matrix parameters can be computed using the following:

$$T(\theta, \dot{\theta}) = \frac{1}{2} \dot{\theta}^T \cdot \dot{\theta} = \frac{1}{2} \sum_{i,j} M_{ij} (\dot{q}_i \dot{q}_j) \geq 0,$$

$$T(\theta, \dot{\theta}) = \frac{1}{2} m_1 (\ddot{x}_1 + \ddot{y}_1 + \ddot{z}_1) + \frac{1}{2} m_2 (\ddot{x}_2 + \ddot{y}_2 + \ddot{z}_2) + \frac{1}{2} m_3 (\ddot{x}_3 + \ddot{y}_3 + \ddot{z}_3) + \frac{1}{2} I_{11} \ddot{\theta}_1 + \frac{1}{2} I_{22} \ddot{\theta}_2 + \frac{1}{2} I_{33} \ddot{\theta}_3$$

$$+ \frac{1}{2} l_1 \ddot{\theta}_1 \ddot{\theta}_2, \quad \ddot{x}_1 = 0, \quad \ddot{x}_2 = -r_2 \sin \theta_1 - r_2 \cos \theta_1 \cdot \cos \theta_2 \cdot \ddot{\theta}_2,$$

$$\ddot{y}_2 = r_2 \cos \theta_1 \cdot \sin \theta_1 + r_2 \cos \theta_1 \cdot \cos \theta_2 \cdot \ddot{\theta}_2,$$

$$\ddot{z}_2 = l_1 + r_2 \cos \theta_2, \quad \ddot{x}_2 = -r_2 \sin \theta_1 - r_2 \cos \theta_1 \cdot \cos \theta_2 \cdot \ddot{\theta}_2,$$

$$\ddot{y}_2 = r_2 \cos \theta_1 \cdot \sin \theta_1 + r_2 \cos \theta_1 \cdot \cos \theta_2 \cdot \ddot{\theta}_2,$$

$$\ddot{z}_2 = l_1 + r_2 \cos \theta_2, \quad x_3 = \cos \theta_1 (l_2 \sin \theta_2 + r_3 \sin (q_2 + q_3)), \quad y_3 = \sin \theta_1 (l_2 \sin \theta_2 + r_3 \sin (q_2 + q_3)),$$

$$\ddot{z}_3 = x_3 = -\sin \theta_1 (l_2 \sin \theta_2 + r_3 \sin (q_2 + q_3)) \dot{q}_1 + \cos \theta_1 (l_2 \cos \theta_2 + r_3 \cos (q_2 + q_3)) \dot{q}_2 + r_3 \cos \theta_1 \cos (q_2 + q_3) \dot{q}_3,$$

$$\ddot{z}_3 = -l_2 \sin \theta_2 + r_3 \sin (q_2 + q_3) \dot{q}_2 + r_3 \sin (q_2 + q_3) \dot{q}_3,$$

$$l_{x1} = m_l (w_1^2 + h_1^2), \quad l_{y1} = m_l (l_1^2 + h_1^2), \quad l_{z1} = m_l (l_1^2 + w_1^2),$$

where $x_i, y_i, z_i, x_i, y_i, z_i$ are the coordinates of the center of mass for each element of a leg, and, respectively, their first derivative; and $I_{x1}, I_{y1},$ and $I_{z1}$ represent the inertia tensors of each leg element.

By using inertia matrix requirement parameters, the inertia matrix elements were computed by the following equations:
\[ M_{11} = r_2^2 m_2 + r_3^2 m_3 + \frac{1}{12} r_1^2 m_1 + \frac{1}{12} \dot{r}_2^2 m_1 + \frac{1}{12} \dot{r}_2 m_2 + \frac{1}{12} \dot{r}_3^2 m_2 + \frac{1}{12} \dot{r}_3 m_3 + \frac{1}{12} \dot{w}^2 m_3 + \]
\[ 2l_2 r_5 m_5 \sin(q_2 + q_3) - \cos^2(q_2) (l_2^2 m_3 + r_2^2 m_2) - \] \[ r_3^2 m_3 \cos^2(q_2 + q_3), \]
\[ M_{12} = \frac{1}{6} \dot{r}_2^2 m_2 + \frac{1}{6} \dot{w}^2 m_2, \]
\[ M_{13} = \frac{1}{6} \dot{w}^2 m_3 + \frac{1}{6} \dot{r}_3^2 m_3, \]
\[ M_{22} = r_2^2 m_2 + r_3^2 m_3 + \frac{1}{12} \dot{r}_2^2 m_2 + \frac{1}{12} \dot{r}_3^2 m_2 + \frac{1}{12} \dot{w}^2 m_2 + 2l_2 r_3 m_5 [\sin(q_2 + q_3) + \cos q_2 \cos(q_2 + q_3)], \]
\[ M_{23} = 2l_2 r_3 m_5 [\sin q_2 \cdot \sin(q_2 + q_3) + \cos q_2 \cdot \cos(q_2 + q_3)], \]
\[ M_{31} = \frac{1}{6} r_2^2 m_3, \]
\[ M_{32} = \frac{1}{6} \dot{w}^2 m_3 + \frac{1}{6} \dot{r}_3^2 m_3 + 2r_3^2 m_3, \]
\[ M_{33} = r_2^2 m_3 + \frac{1}{12} \dot{r}_3^2 m_3 + \frac{1}{12} \dot{w}^2 m_3. \]

The Coriolis matrix was computed using the following equation:

\[ C_{ij}(q, \dot{q}) = \sum_{k=1}^{3} \Gamma_{ijk} \dot{q}_k = \frac{1}{2} \sum_{k=1}^{3} \left( \frac{\partial M_{ij}}{\partial q_k} + \frac{\partial M_{ik}}{\partial q_j} - \frac{\partial M_{kj}}{\partial q_i} \right) \dot{q}_k \]

for which the \( \Gamma_{ijk} \) parameters are:

\[ \Gamma_{111} = \Gamma_{122} = \Gamma_{123} = \Gamma_{132} = \Gamma_{133} = \Gamma_{212} = \Gamma_{213} = \Gamma_{221} = \Gamma_{222} = \Gamma_{231} = \Gamma_{312} = \Gamma_{313} = \Gamma_{321} = \Gamma_{323} = \Gamma_{331} = 0, \]
\[ \Gamma_{112} = \left( \frac{2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) + \cos q_2 \sin(q_2 + q_3)) + 2l_2^2 m_2 \cos q_2 \sin q_2 + + 2m_2 r_3^2 \cos q_2 \sin q_2 + 2m_3 r_3^2 \cos(q_2 + q_3) \sin(q_2 + q_3))}{3} \right) \dot{q}_2, \]
\[ \Gamma_{113} = (2l_2 m_3 \sin \theta q_2 \cos(q_2 + q_3) + 2m_3 r_3^2 \cos(q_2 + q_3) \sin(q_2 + q_3)) \dot{q}_3, \]
\[ \Gamma_{121} = \left( \frac{2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) + \cos q_2 \sin(q_2 + q_3)) + 2l_2^2 m_2 \cos q_2 \sin q_2 + + 2m_2 r_3^2 \cos q_2 \sin q_2 + 2m_3 r_3^2 \cos(q_2 + q_3) \sin(q_2 + q_3))}{3} \right) \dot{q}_1, \]
\[ \Gamma_{131} = (2l_2 m_3 \sin \theta q_2 \cos(q_2 + q_3) + 2m_3 r_3^2 \cos(q_2 + q_3) \sin(q_2 + q_3)) \dot{q}_1, \]
\[ \Gamma_{211} = \left( \frac{2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) + \cos q_2 \sin(q_2 + q_3)) + 2l_2^2 m_2 \cos q_2 \sin q_2 + + 2m_2 r_3^2 \cos q_2 \sin q_2 + 2m_3 r_3^2 \cos(q_2 + q_3) \sin(q_2 + q_3))}{3} \right) \dot{q}_2, \]
\[ \Gamma_{223} = 2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) - \cos q_2 \sin(q_2 + q_3)) \dot{q}_3, \]
\[ \Gamma_{232} = 2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) - \cos q_2 \sin(q_2 + q_3)) \dot{q}_2, \]
\[ \Gamma_{233} = 4l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) - \cos q_2 \sin(q_2 + q_3)) \dot{q}_3, \]
\[ \Gamma_{311} = -2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) + 2m_3 r_3^2 \cos(q_2 + q_3) \sin(q_2 + q_3)) \dot{q}_2, \]
\[ \Gamma_{322} = -2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) - \cos q_2 \sin(q_2 + q_3)) \dot{q}_2, \]
\[ \Gamma_{332} = -2l_2 m_3 r_3 (\sin q_2 \cos(q_2 + q_3) - \cos q_2 \sin(q_2 + q_3)) \dot{q}_2. \]

The last part of the dynamic equation is given by Equation (24), which computed the gravity effect matrix on the robot leg:

\[ G(q) = N(q, \dot{q}) = \frac{\partial U}{\partial q} = \left[ \begin{array}{c} \frac{\partial U}{\partial q_1} \\ \frac{\partial U}{\partial q_2} \\ \frac{\partial U}{\partial q_3} \end{array} \right] \]

where

\[ U(q) = m_1 x_1 g + m_1 y_1 g + m_2 z_1 g + m_2 g(x_2 + y_2 + z_2) + m_3 g(x_3 + y_3 + z_3) \]
\[ = m_1 r_1 g + m_2 g(l_1 + r_2 \cos q_2 + r_2 \sin q_2 \sin q_1 + \cos q_1) + + m_3 g(l_1 + l_2 \cos q_2 + r_3 \cos(q_2 + q_3) + (\sin q_1 + \cos q_1)(l_2 \sin q_2 + r_3 \sin(q_2 + q_3))) \]

and its derivative components are:
\[
\frac{\partial U}{\partial q_1} = m_2 r_2 g \sin q_2 (-\sin q_1 + \cos q_1) + m_3 g (-\sin q_1 + \cos q_1) (l_2 \sin q_2 + r_3 \sin (q_2 + q_3)) \\
\frac{\partial U}{\partial q_2} = m_2 g (-r_2 \sin q_2 + r_2 \cos q_2 (\cos q_1 + \sin q_1)) + m_3 g (-l_2 \sin q_2 - r_3 \sin (q_2 + q_3) + (\sin q_1 + \cos q_1) (l_2 \cos q_2 + r_3 \cos (q_2 + q_3)) \\
\frac{\partial U}{\partial q_3} = m_3 g (-r_3 \sin (q_2 + q_3) + r_3 \cos (q_2 + q_3) (\sin q_1 + \cos q_1)).
\]

(26)

Using the dynamic control equations, the PID sliding control can be developed, which computes the torque \( \tau \) used in the joint motor torque control so that the position vector \( q \) can track the desired trajectory \( q_d \). The tracking error vector is defined as:

\[
e = q_d - q.
\]

(27)

Sliding motion control requires a sliding surface, given by Equation (28), and contains both the derivative term and the integral one:

\[
s = \dot{e} + \lambda_1 e + \lambda_2 \int_0^t \dot{e} dt
\]

(28)

where \( \lambda_i \) is a positive diagonal matrix; it turns out that, for \( s = 0 \), a stable sliding surface is obtained (as shown by Shafiei in [12]). The dynamic robot equations can be written by using the sliding surface equation:

\[
H \dot{s} = -Cs + \tau_f - \tau
\]

(29)

where

\[
f = H (\dot{q}_d + \lambda_1 \dot{e} + \lambda_2 \dot{e}) + C (\dot{q}_d + \lambda_1 e + \lambda_2 \int_0^t \dot{e} dt) + G.
\]

(30)

The control module input becomes:

\[
\tau = \tilde{f} + K_v s + K \text{sgn}(s)
\]

(31)

where

\[
\tilde{f} = \tilde{H} (\tilde{q}_d + \lambda_1 \dot{e} + \lambda_2 \dot{e}) + \tilde{C} (\dot{q}_d + \lambda_1 e + \lambda_2 \int_0^t e dt) + \tilde{G}.
\]

(32)

Equation (32) represents a force estimation \( f \), and \( K_v s = K_v \dot{e} + K_v \lambda_1 e + K_v \lambda_2 \int_0^t \dot{e} dt \) is the outer PID loop; \( K_\nu \) and \( K \) are positive diagonal matrices built so that the stability conditions are fulfilled and guaranteed. The \( \text{sgn}(s) \) function is the sign function. The function can also be written as:

\[
|\tilde{f}| = \|H (\tilde{q}_d + \lambda_1 \dot{e} + \lambda_2 \dot{e}) + C (\dot{q}_d + \lambda_1 e + \lambda_2 \int_0^t e dt) + G\| \leq F
\]

(33)

where \( \tilde{f} = f - \tilde{f}, \tilde{H} = H - \tilde{H}, \) and \( \tilde{G} = G - \tilde{G} \). The vector \( F \) is:

\[
F = \|\tilde{H} (\tilde{q}_d + \lambda_1 \dot{e} + \lambda_2 \dot{e}) + \tilde{C} (\dot{q}_d + \lambda_1 e + \lambda_2 \int_0^t e dt)\| + \|\tilde{G}\|.
\]

(34)

To control the system states \((e, \dot{e})\) and to reach the sliding surface \( s = 0 \) in a limited time by staying on the surface, the control law should be formulated so that Condition (35) is fulfilled:

\[
\frac{1}{2} \frac{d}{dt} |s^T H s| < \eta \left( s^T s \right)^{\frac{1}{2}}, \quad \eta > 0.
\]

(35)
Using the sign function in the control law, high oscillations in the control torque are found as the undesired phenomenon called chattering. To overcome this drawback, a saturation function was used (36) for the discontinuous part of the control law:

\[
\text{sat}\left(\frac{s}{\varphi}\right) = \begin{cases} 
1 & s \geq \varphi \\
\frac{s}{\varphi} & -\varphi < s < \varphi \\
-1 & s \leq -\varphi 
\end{cases}.
\] (36)

As a result, a layer \(\varphi\) around the sliding surface was obtained so that, when the robot foot trajectory was inside the layer, it remained there. The values of \(\lambda_1, \lambda_2, K,\) and \(K_v\) were adjusted to better position the mobile walking robot foot.

6. Hybrid Control Simulation

The simulation was built with the help of MATLAB Simulink software to test the proposed methods and control laws. Figure 10 presents the diagram of the main components of the hybrid controller. The reference generation block for the OXYZ axis in the Cartesian space is shown in Figure 11, and the constant generation block defining the walking robot is presented in Figure 12. All the values were sent to a reference system on the robot structure, illustrated in Figure 1b. The foot’s vertical position was at a distance of 1.1 m from the origin set on the robot platform, not the foot.

Figure 10. MATLAB Simulink diagram for the hybrid control.

The three lines in Figure 11 represent the reference system as follows: the top signal (green line) is the reference for the robot foot on the OZ axis, the trapezoidal signal (blue line) is the reference for the robot foot on the OX axis, and for the OY axis, a zero-value signal was used (purple line). These three datasets represent the Cartesian position of the robot foot for a complete cycle of a leg’s walking step. The reference on the OY axis is the heading direction of the robot and has a trapezoidal shape because the foot is moving relative to the robot platform.
Hybrid Control Simulation

The simulation was built with the help of MATLAB Simulink software to test the proposed methods and control laws. Figure 10 presents the diagram of the main components of the hybrid controller. The reference generation block for the OXYZ axis in the Cartesian space is shown in Figure 11, and the constant generation block defining the walking robot is presented in Figure 12. All the values were sent to a reference system on the robot structure, illustrated in Figure 1b. The foot’s vertical position was at a distance of 1.1 m from the origin set on the robot platform, not the foot.

Figure 10. MATLAB Simulink diagram for the hybrid control.

Figure 11. Foot reference signals. The three lines in Figure 11 represent the reference system as follows: the top signal (green line) is the reference for the robot foot on the OZ axis, the trapezoidal signal (blue line) is the reference for the robot foot on the OX axis, and for the OY axis, a zero-value signal was used (purple line). These three datasets represent the Cartesian position of the robot foot for a complete cycle of a leg’s walking step. The reference on the OY axis is the heading direction of the robot and has a trapezoidal shape because the foot is moving relative to the robot platform.

Figure 12. Constant values.

Figure 13 shows the diagram corresponding to the sliding control method made in MATLAB Simulink in which all the described elements are found. With their help, the command signal for the three joint motors was calculated. Algorithm 1 controlled the kinematic control block. It computed the angular speeds using the Jacobian matrix and the formula from Equation (8), which provided the angular reference speed.

Figures 14 and 15 present the simulation diagrams for the two sensors used in determining which control law should be used at a particular moment in time according to the switching algorithm based on neutrosophic logic. Using what was presented in Section 4 regarding the neutrosophic decision, the neutrosophic control switching block was implemented. It is illustrated in Figure 16 with its inputs and outputs. The two inputs already described are shown, bringing proximity and force information into the switching mechanism. In addition to these two, there was a third input called stable-state, and it provided the block with additional information. When the robot was homing or reached specific points, it was controlled only by the kinematic control law. The solution was chosen to save computing power and provide a higher speed for arriving at the initial position (homing phase).
Figure 13. The sliding control diagram made in MATLAB Simulink.

Algorithm 1 The kinematic control block

```matlab
function [Mx, My, Mz, qd] = func(Ref_X, Ref_Y, Ref_Z, A1, L2, L3, q1, q2, q3)
    f1 = q1; f2 = q2; f3 = q3;
    J11 = -sin(f1) * (L2 * sin(f2) + L3 * sin(f2 + f3));
    J12 = L2 * cos(f1) * cos(f2) + L3 * cos(f1) * cos(f2 + f3);
    J13 = L3 * cos(f1) * cos(f2 + f3);
    J21 = cos(f1) * (L2 * sin(f2) + L3 * sin(f2 + f3));
    J22 = L2 * sin(f1) * cos(f2) + L3 * sin(f1) * cos(f2 + f3);
    J23 = L3 * sin(f1) * cos(f2 + f3);
    J31 = 0;
    J32 = -L2 * sin(f2) - L3 * sin(f2 + f3);
    J33 = -L3 * sin(f2 + f3);
    Jb = [J11 J12 J13; J21 J22 J23; J31 J32 J33];
    Mx = cos(f1) * (L2 * sin(f2) + L3 * sin(f2 + f3));
    My = sin(f1) * (L2 * sin(f2) + L3 * sin(f2 + f3));
    Mz = L2 * cos(f2) + L3 * cos(f2 + f3) + A1;
    M_err = [Ref_X - Mx; Ref_Y - My; Ref_Z - Mz];
    JTde = Jb * Jb' * M_err;
    qd = alpha * Jb' * M_err;
```
The actual neutrosophic switching block followed the detailed conditions already described.
7. Experimental Results

Following the simulation results, we observed several things. One of them was that, to successfully simulate the control law, which was bounded by the interaction between the support surface and the robot, different conditions were needed by the decision and control methods. This case was observed during the support phase, for which the robot leg must hold the entire robot weight and carry out the forward robot motion. The force and proximity sensors must have values that assumed the support surface contact in actual case conditions. In contrast, in simulation conditions, if the positioning control error placed the robot foot slightly above the support surface, then the sensors could affect the control laws and the entire system. Consequently, the switching mechanism was built with the condition that switched the control law when there was permanent contact with the support surface. An example is the homing motion of the foot, for which the robot was controlled only through the kinematic control method.

Figure 17 presents the reference and position tracking for the robot foot in the operational space in Cartesian coordinates on the OX axis. The positioning error on the OX axis is shown in Figure 18. The movement represents the forward direction of motion for the robot and its legs. Thus, three steps are presented, for which the trapezoidal shape of the signal represents the forward and retreat motion relative to the robot platform. The movement was computed according to the reference system relative to the robot platform. Because the reference was considered in the robot’s operational space, the first coordinate system was selected at the point where the first joint of the robot was placed.

![Figure 17. Foot reference and position tracking on the OX axis.](image1)

![Figure 18. Foot positioning error on the OX axis.](image2)
On average, the error on the OX axis was below 1 cm, but there were some spikes in the error signal. The high amplitude errors were due to the sudden change in the reference speed, which was used in controlling the angular velocity through the torque of the joint's motor. The high amplitude errors were found at the points where the reference changed its path slope and control type. The error had a more continuous shape when the kinematic control was in place, and in the case of the dynamic control, the error tended to oscillate.

Figure 19 presents the robot foot's reference, positioning, and error signals on the OY axis. The reference value was zero, and the positioning error was less than 1 mm. On the other two axes, spikes were found in the error signal at the moment when the control law changed.

Figure 20 presents the diagram for the reference and positioning signals of the robot foot on the OZ axis, which corresponds to the perpendicular axis on the support surface, meaning the vertical motion. The diagram presents the foot position during the leg's swing phase. The leg was positioned on the vertical axis so it would not hit an obstacle or the support surface. Also, the vertical trajectory of the foot was in the support phase, for which the reference was zero. The foot followed a continuous and uniform reference value during the swing phase. In the support and moving-forward phase, a positioning error was observed. The error may have been due to the platform weight compensating at the moment the robot foot crossed the point of intersection with the platform center of the vertical gravity axis. The positioning error became zero on the OZ axis.

Figures 17–21 present, in the same time frame, the motion of the robot leg stepping three times to move the robot forward. The diagrams show all motion stages for the robot foot to complete a step. They present reference and tracking signals. The homing occurred...
in the first second of the simulation, and a high error was observed. In the time interval of [2–5 s], the leg moved on the vertical axis and forward, controlled by the kinematic control law to reach a new position for the foot. In the next second of the virtual experiment at [5–6 s], the control method reached the vertical reference position to allow the robot leg to support the robot’s weight. Between 6 and 9 s, the leg moved backward in relation to the robot platform and was controlled by the dynamic SMC method. A different error pattern is observed in Figures 18 and 21, considering that the robot leg supported the weight at this stage. After this stage was completed, the next step continued in the same manner, excluding the homing sequence.

![Figure 21. Foot positioning error on the OZ axis.](image)

In Figure 21, a maximum error of 4 cm was found at the amplitude peaks and an average value of 1 cm. The high amplitude values appeared as described above when the control laws were switched, which is the subject of future work to stabilize the system at the switch. The error peaks were also due to the sudden shape-change of the reference signal. By changing from a curve signal to a straight line, the derivative part of the controller received an extremely high value, which in turn affected the control signal. The influence should be attenuated or removed entirely in future work.

8. Conclusions

To summarize and conclude the results, Figure 22 presents the robot foot trace in a 3D space with the reference pattern. All three steps overlap in the same diagram. The coordinates are given in the Cartesian space. The first stage that was easily found was the homing curve, seen in the lower section of the image. The maximum error was found at the start or end of a step, where the reference system must be improved to avoid sharp changes of direction. For the simulation, a fixed Cartesian coordinates system was considered, with the origin placed in the first joint where the robot top platform joined with to the robot leg. The shape of the horizontal motion was not uniform. The trajectory had minor errors in the range of millimeters and hundreds of micrometers when the control method was not changed, and we considered the shape of the foot trajectory as close to the reference.

Overall, as presented, the dynamic controller was better at following a continuous reference than a simple positioning kinematic controller. Although the positioning was more precise when a dynamic-based controller was used, sudden changes were added in the reference value when changing from a linear trajectory to a half ellipse. Since these were the points where the decision algorithm should also switch the used control method, these points of interest became essential areas of disturbance in the system. We will dedicate our attention to mitigating the reference and switching effects in the reference-tracking algorithm in future work.
The dynamic controller tended to be slower than the kinematic controller in compensating for the disturbance, but it did not override the PI kinematic controller. Moreover, the PI kinematic controller oscillated around the reference value when the robot leg was subjected to exterior forces. The gravitational acceleration acted upon the entire leg when high gains were used inside the control loops to lower the reference tracking time. It resulted in a high tracking error on the vertical axis during the kinematic control. The reference tracking time was considered the time difference between the change to the kinematic control method and until the foot reached the target point, with an error small enough to consider that the position was reached. The target position was chosen near the support surface but far enough for the leg to lower its speed before hitting the surface.

A PI kinematic-based controller was used during the swing motion of the robot leg because high precision was not required to move the robot foot. Instead, a constant foot speed was needed to reach the point of contact with the support surface in a short time. In addition, along with the positioning precision, the controller did not need to compensate for the gravity and inertial forces during the leg and robot motion and could have severe and undesired consequences in the support phase. Therefore, we concluded that a dynamic-based controller was required to compensate for all the inertial forces and to better track the reference.

Finally, the proposed hybrid control efficiently used the two control methods for the mobile walking robot leg. The biggest problem was found during the transition between the two control techniques.

The main conclusion of the paper was on the decision algorithm side. By having a two-stage decision, the information that could be analyzed offline between simulations defined the outline of the critical decision in each case or phase of the robot. The final algorithm distinguished between robot motion phases and rejected contradictory conditions at the online stage.

The consequence of the presented hybrid force/position control with a two-stage decision algorithm was that it can successfully be used and further developed for other types of robots or tasks.

Our future work will focus on studies for removing the high peaks of positioning error. At the same time, new, improved simulations are required to visualize the leg motion cycle.

**Author Contributions:** Conceptualization, I.-A.G. and L.V.; simulation, I.-A.G.; validation, I.-A.G., L.V. and A.-C.C.; investigation, I.-A.G. and L.V.; resources, I.-A.G. and A.-C.C.; writing—original draft preparation, I.-A.G.; writing—review and editing, I.-A.G., L.V. and A.-C.C.; supervision, L.V. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.
Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: This work was supported by the Institute of Solid Mechanics of the Romanian Academy and by an interacademic project of IMSAR–Yanshan University: “Joint Laboratory of Intelligent Rehabilitation Robot” (KY201501009), a collaborative research agreement between Yanshan University, China, and the Romanian Academy by IMSAR, RO.

Conflicts of Interest: The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Glossary

\( \theta_i \)  robot leg join rotation
\( \theta_{\text{ref}} \)  robot joint angular position reference
\( \theta_{\text{real}} \)  robot joint angular position measured by angular sensors
\( \Delta \theta_i \)  leg joint positioning error
\( \omega_i \)  joint angular speed
\( \Delta \omega_i \)  joint angular speed error
\( r_i \)  distance from the previous joint axis to the \( i \)th leg segment center of mass
\( l_i \)  length of the \( i \)th leg segment
\( m_i \)  mass of the \( i \)th leg segment
\( P_{\text{ref}} \)  position reference of the robot foot
\( P_{\text{real}} \)  real position of the robot foot
\( J \)  Jacobian matrix
\( J_{ij} \)  Jacobian matrix element with coordinates \( i \) (row) and \( j \) (column)
\( F_{\text{sim}} \)  simulated ground reaction force
\( F_{\text{real}} \)  real force given by force sensors placed on robot foot
\( F_{\text{ground}} \)  ground reaction force
\( d \)  distance from robot foot to the ground given by proximity sensors
\( s_i \)  \( \sin(\theta_i) \)
\( c_i \)  \( \cos(\theta_i) \)
\( s_{ij} \)  \( \sin(\theta_i + \theta_j) \)
\( c_{ij} \)  \( \cos(\theta_i + \theta_j) \)
\( \tau_p \)  joint torque computed by position control
\( \tau_f \)  joint torque computed by force control
\( \tau \)  joint torque selected by the decision algorithm
\( q_i \)  leg joint used by the dynamic control method
\( s \)  sliding surface error in terms of SMC
\( \text{SMC} \)  sliding motion control
\( O_m \)  extension theory object
\( e_m \)  extension theory characteristic
\( v_m \)  extension theory measure
\( m(C) \)  the neutrosophic generalized basic belief assignment

\( D^\theta \)  a hyperpower set
\( \theta_C \)  neutrosophic kinematic state
\( \theta_D \)  neutrosophic dynamic state
\( \theta_{D\cup C} \)  neutrosophic uncertain state
\( \theta_{D\cap C} \)  neutrosophic contradiction state
28. Li, H.; Xie, X.; Du, P.; Xi, J. Cooperative Object Recognition Method of Multi-UAVs Based on Decision Fusion. In Proceedings of the 2021 33rd Chinese Control and Decision Conference (CCDC), Kunming, China, 22–24 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 5424–5429.

29. Chai, H.; Lv, S.; Fang, M. Obstacle Avoidance by DSmT for Mobile Robot in Unknown Environment. In Proceedings of the 2019 4th International Conference on Automation, Control and Robotics Engineering, Shenzhen, China, 19–21 July 2019; pp. 1–6.

30. Yuan, S.; Guo, P.; Han, X.; Luan, F.; Zhang, F.; Liu, T.; Mao, H. DSmT-Based Ultrasonic Detection Model for Estimating Indoor Environment Contour. IEEE Trans. Instrum. Meas. 2019, 69, 4002–4014. [CrossRef]

31. Liu, J.; Wei, L.; Cao, J.; Fei, S. Hybrid-Driven H∞ Filter Design for T–S Fuzzy Systems with Quantization. Nonlinear Anal. Hybrid Syst. 2019, 31, 135–152. [CrossRef]

32. Aslam, M.S.; Qaisar, I.; Saleem, M.A. Quantized Event-triggered feedback control under fuzzy system with time-varying delay and Actuator fault. Nonlinear Anal. Hybrid Syst. 2019, 35, 100823. [CrossRef]

33. Bledt, G.; Powell, M.J.; Katz, B.; Di Carlo, J.; Wensing, P.M.; Kim, S. MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2245–2252.

34. Zhang, J.; Jin, L.; Yang, C. Distributed Cooperative Kinematic Control of Multiple Robotic Manipulators with an Improved Communication Efficiency. IEEE/ASME Trans. Mechatron. 2021, 27, 149–158. [CrossRef]

35. Zhang, B.; Wu, J.; Wang, L.; Yu, Z. Accurate dynamic modeling and control parameters design of an industrial hybrid spray-painting robot. Robot. Comput. Manuf. 2019, 63, 101923. [CrossRef]

36. Ion, I.; Vladareanu, L.; Simionescu, I.; Vasile, A. The Gait Analysis for Modular Walking Robot MERo Walks on the Slope. Nature 2008, 6, 8.

37. Vladareanu, V.; Schiopu, P.; Deng, M.; Yu, H. Intelligent extended control of the walking robot motion. In Proceedings of the 2014 International Conference on Advanced Mechatronic Systems, Kumamoto, Japan, 10–12 August 2014; pp. 489–495. [CrossRef]

38. Wen, C.; Chun-Yan, Y.; Bin, H. New Development of the Basic Theory of Extenics. Eng. Sci. 2003, 5, 80–87.

39. Gal, A.; Vladareanu, L.; Munteanu, A.; Melinte, O. PID Sliding Motion Control by Using Fuzzy Adjustment. In Proceedings of the SISOM 2012 and Session of the Commission of Acoustics, Bucharest, Romania, 30–31 May 2012.