Intelligent kinematic control of a quadrupedal mobile robot

Alex Csillag, László Dávid, Lőrinc Márton
Department of Electrical Engineering
Sapientia Hungarian University of Transylvania
Corunca, Romania
alex@csalex.org, (ldavid, martonl)@ms.sapientia.ro

Endre Györfi, Zsolt Köllő
Codespring, Software development
Târgu Mureș, Romania
alex@csalex.org
(gyorfi.endre, kollo.zsolt)@codespring.ro

***Abstract*—This work presents a control method for quadrupedal locomotion utilizing periodic gaits, alongside a neural network-based solution for the inverse kinematics problem inherent to these robots. The proposed method evaluates the stability and execution speed of both periodic-continuous and periodic-discontinuous gaits. Based on this analysis, the optimal gait for the robot locomotion can be chosen. For the inverse kinematics problem of the 12-DOF quadrupedal robot, a feed-forward neural network-based solver was proposed. Different training methods were analyzed. The performed simulation experiments demonstrate the effectiveness of the proposed computation and control approach.**

***Index Terms*—Robotics, Quadrupedal locomotion, Neural networks, Intelligent control, Inverse kinematics**

I. Introduction

Nowadays, mobile robots are gaining more ground and are being used in an increasing number of applications, such as construction and manufacturing industries, healthcare, space exploration, and military applications [1].

Mobile robots are classified into several categories based on their motion principles, including *wheeled robots*, *tracked robots*, and *legged robots*. The choice of robot class for a specific task depends on environmental conditions and the task requirements. The properly controlled legged robots are useful in outdoor environments even in the case of uneven terrains. The legged robots can be divided into three categories based on the actuation type: *hydraulic actuator*, *pneumatic actuator* and *electrical actuator* [2]. Based on the realized degrees of freedom of a leg, it can be further broken down into three categories: *prismatic legs*, *articulated legs* and *redundant articulated legs* [3]. The type of robots being discussed in this paper are legged robots with an electrical actuation.

Operating a quadrupedal mobile robot safely and efficiently involves addressing several challenges in both structural and software design. One such challenge is achieving stable mobility of the robot where both structural design and algorithm efficiency are big factors. The design of the gait algorithms is discussed in this paper, alongside the aforementioned study of stability and velocities for periodic gaits.

The mobile robots can be controlled by personnel through a controller or remote control software. Movement is also possible autonomously, that is, without the assistance of human operators or personnel [4]. Autonomous operation can be achieved with the use of *artificial intelligence (AI)*. The incorporation of artificial intelligence is widely used in quadrupedal robots due to the mobility and stability of locomotion [1].

Researchers have developed a range of control solutions to achieve stable locomotion in quadrupedal robots. One study suggested the use of *Proportional Integral Derivative (PID)* controllers and *Fuzzy Inference Systems (FIS)* to stabilize robot motion, with parameters tuned by an *ant-colony optimization* algorithm [5]. FIS was also utilized for high-velocity galloping gaits [6]. Managing a wide range of speeds is challenging due to quadruped systems being more responsive at high speeds than low speeds, and proportional derivative controllers being effective only within a narrow range around the calibrated speed and turning rate [7]. To address this issue, a *hybrid controller* using fuzzy inference systems and *lateral side-stepping* has been proposed. Additionally, a *novel intelligent controller* incorporating a new mechanical structure and optimized foot trajectory was proposed to enable precise trajectory tracking and a steady gait. It utilized *optimized cascade PID* and *improved fuzzy adaptive PID control systems* with parameters optimized by *the sparrow search* algorithm [8]. Furthermore, an effective control algorithm based on *reinforcement learning (RL)* and *artificial neural networks (ANN)* has been proposed to replace traditional analysis-based control theories like inverse kinematics and differential equations of motion [9].

ANFIS was explored for hexapod control, but membership functions were reduced due to computational limits [10]. Additionally, a hybrid FMM-RS algorithm was developed for bipedal locomotion. Results demonstrated that FMM-RS surpassed the FMM in both static and dynamic conditions [11].

This paper considers each step of the kinematic control of a quadruped robot. First, the gait method is chosen and its proprieties are discussed. Second, the prescribed path of each leg is designed. Third, the inverse kinematics problems of the legs are solved.

II. Direct kinematics

In this paper, the mathematical and simulational models are based on an actual robot named *Dogzilla S1* which was developed by *Yahboom* [12]. Most four-legged robots use a *serial mechanism* per leg, each leg is mounted to one common base, in this case, the chassis of the robot, thus making the

architecture *parallel* [13]. Each leg consists of 3 electrically actuated joints *(DC motors)*, making one leg have 3 degrees of freedom (3-DOF). For each joint, we can match a coordinate system which follows the *Denavit-Hartenberg (D-H)* standard [14].

The link lengths (L_i) and joint angles (θ_i) for each leg are known and measurable. TABLE I presents the D-H parameters based on [12].

TABLE I
D-H TABLE FOR THE FRONT LEFT LEG.

K_i	θ_i	d_i	a_i	α_i
1	θ_1	0	0	0
2	θ_2	L_1	$\pi/2$	0
3	θ_3	0	L_2	0
4	θ_4	0	L_3	0

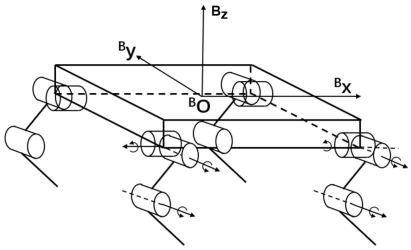

Fig. 1. Coordinate system for a four-legged parallel mobile robot [15].

Using the D-H parameters in TABLE I, the following transformation between the robot's base frame and foot end frame is derived:

$$R = \begin{pmatrix} s_{23} & s_{23} & 0 \\ -s_1 c_{23} & s_1 s_{23} & c_1 \\ c_1 c_{23} & -c_1 s_{23} & s_1 \end{pmatrix} \tag{1}$$

$$P = \begin{pmatrix} P_x \\ P_y \\ P_z \end{pmatrix} = \begin{pmatrix} L/2 + L_2 s_2 - L_3 s_{23} \\ W/2 + L_1 c_1 + L_2 s_1 c_2 - L_3 s_1 c_{23} \\ L_1 s_1 + L_2 c_1 c_2 - L_3 c_1 c_{23} \end{pmatrix} \tag{2}$$

where R is the rotation matrix, P is the translation vector, L is the length and W is the width of the robot. s and c abbreviate the sine and cosine trigonometric functions. Once we have the direct kinematics for one leg, we can apply the same transformation symmetrically for each leg. With known joint angle domains ($\theta_i \in [\theta_{i_{min}}, \theta_{i_{max}}]$) the workspace of the legs can be computed based on the direct kinematic relations, see eq. (2).

III. GAIT GENERATION

A gait is defined as the timing and coordination of placing and lifting each leg, synchronized with the body's motion across its six degrees of freedom, to propel the robot in the desired direction [15].

A. Events and event sequence

An *event* is defined as the placement or lifting of one leg [15]. A gait is possible with the implementation and timing of these events. Multiple events in a sequence are called an *event sequence*. For an n-legged robot, it can be defined with an index in the sequence. The placement of the foot i is defined as event i, while the lifting of the foot i is defined as event $i + n$. In this way, a gait can be expressed as a sequence of events where the number of unique indices in the sequence is $2n$, e.g. $2-4-5-7-3-1-8-6$ [15]. The sequence can also be expressed in graphs as seen in Fig. 2.

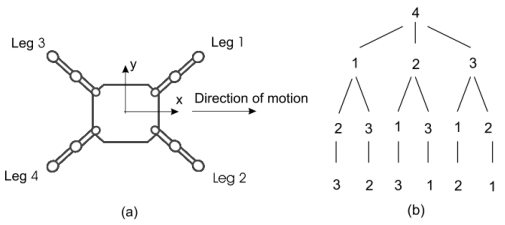

Fig. 2. **(a)** Top view of the robot; **(b)** Graph of event sequences [15].

Wave gaits provide a constant motion of the COG, while discontinuous gaits do not.

There are six parameters that we can tune to optimize the event sequence to our target:

1) Duty factor/cycle (β_i): the fraction of the cycle for which one leg is on the ground. If β_i is equal for every leg we can say that the gait is *regular* [15];
2) Phase (Φ_i): The normalized activity offset in time from the reference leg (normally *leg 1* is considered the reference leg);
3) Stroke (R): The distance which a leg travels relative to the body during the support phase;
4) Stroke pitch ($P = (P_x, P_y, P_z)$): The distance between stroke centres of adjacent legs;
5) Stride length (λ): The distance travelled by the centre of gravity (*COG*) of the body along a cycle. If the gait is periodic, then
$$\lambda = \frac{R}{\beta} \tag{3}$$

Using these parameters, we can define a $+X$ type wave (continuous) gait, assuming that the leg workspaces have no overlap, i.e. $R \leq P$, the leg offsets (Φ_i) are the following:

$$\Phi_1 = 0 \quad \Phi_2 = \frac{1}{2} \quad \Phi_3 = \beta \quad \Phi_4 = F\left(\beta - \frac{1}{2}\right) \tag{4}$$

where $F(X)$ returns the fractional part of $|X|$. In Fig. 3 an example of such an event sequence is presented.

B. Stability analysis of gaits

During gait planning, we have to take into account the stability of the robot. This is done with the help of *stability margins* on the *support polygon*. The support polygon is defined as the convex shape formed by the footprints of the legs in the support phase [15].

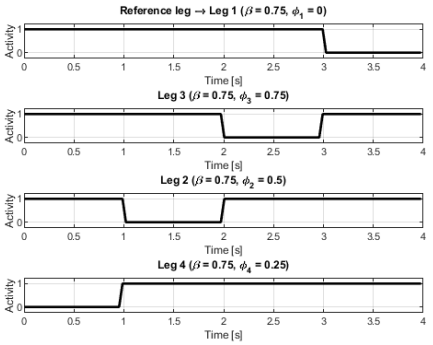

Fig. 3. Event timings with reference leg 1.

A stability margin takes into account the centre of gravity of the robot. If the support polygon changes and the COG falls outside the support polygon, an undesired torque created around the axis between the two footprints closest to the COG can make the robot unstable. We have to make sure that the COG always stays within the support polygon, or in cases when the COG falls outside the support polygon, the robot can catch and push itself back into a stable configuration [16].

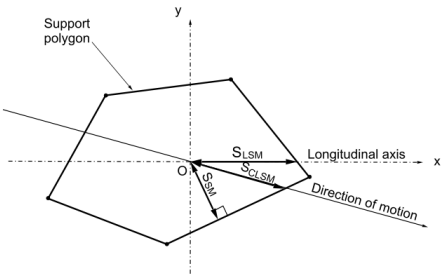

Fig. 4. Different stability margins defined geometrically [15].

There are different ways to measure stability. One such stability margin is called *static stability margin* (S_{SM}) which is defined as the smallest of distances from the projection of the centre of gravity to the edges of the support polygon. However, the equation for calculating S_{SM} can be quite complex, so a different kind of stability margin was proposed called *longitudinal stability margin* (S_{LSM}), which is defined as the smallest of distances from the centre of gravity's projection to the front and rear edges of the support polygon along the robot's longitudinal axis ($+X$). The support polygon can change between steps, stability margins take this fact into account.

The longitudinal stability margin is calculated by the diagonal defined by two contralateral non-adjacent feet, shown in Fig. 5. Using this information, we can calculate the longitudinal stability margin for discontinuous gaits. This diagonal goes from a foot in the middle of its workspace to a foot placed at its kinematic limit closest to the centre of gravity of the robot.

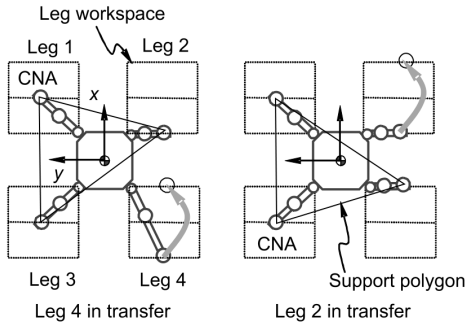

Fig. 5. Support polygon changing between steps [15].

When leg 4 is in transfer, we can define S_{LSM_d} as follows:

$$S_{LSM_d} = \left| -y_2 \left(\frac{x_3 - x_2}{y_3 - y_2} \right) + x_2 \right| \tag{5}$$

where (x_2, y_2) and (x_3, y_3) are the end points of the diagonal. We can observe in Fig. 5 that the point coordinates are the following: $(-P_x/2, P_y/2)$ and $(P_x/2 - R_x/2, -P_y/2)$. Substituting these into eq. (5), yields:

$$S_{LSM_d} = \frac{R_x}{4}. \tag{6}$$

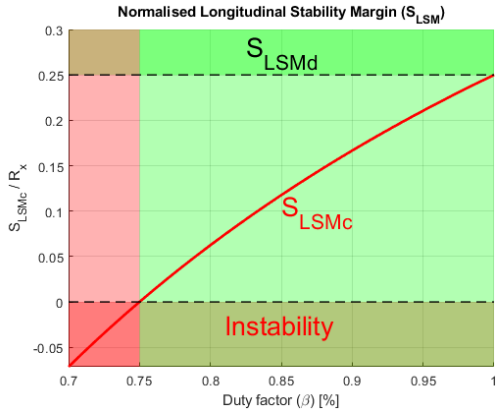

Fig. 6. Stability of wave and discontinuous gaits.

In 1968 McGhe and Frank [15] proved that the S_{LSM} of a continuous gait is given by:

$$S_{LSM_c} = \left(\beta - \frac{3}{4} \right) \lambda;\ 1 > \beta \geq \frac{3}{4}. \tag{7}$$

Substituting eq. (3) into eq. (7) yields the formula for continuous longitudinal stability margin:

$$S_{LSM_c} = \left(\beta - \frac{3}{4} \right) \frac{R_x}{\beta};\ \frac{3}{4} \leq \beta \leq 1. \tag{8}$$

As you can see in Fig. 6, as the duty factor (β) goes above 0.75 the gait stays stable. However, if it reaches values below 0.75 the gait becomes unstable. In other words, if the foot stays in contact with the support surface for longer periods of time, the stability of the gait increases.

IV. Gait period

A. Period of a discontinuous gait

While calculating the gait period we have to make the assumption that the path is rectangular, this simplification makes the calculation much easier. Consider the path in space on Fig. 7. We segmented this path into three sections [15]:

1) $[A, B]$ - leg lifting section (h height);
2) $[B, C]$ - leg motion forward (R_x distance);
3) $[C, D]$ - leg placement (h height).

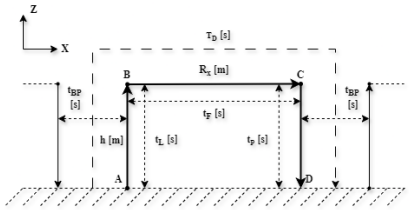

Fig. 7. Foot path.

Since we know the distances for step height and stroke (h and $P = R_x$ respectively), we can calculate the gait period as the function of two velocities (V_x velocity on the X-axis, V_z velocity on the Z-axis). The period of a two-phase gait consists of the time it takes to lift each leg, to propel the leg forward and to place the foot down. With no overlapping events in the event sequence (see Fig. 3), i.e. there is only one leg in motion at any given time, we have to multiply this number by the number of legs, four in this case. In addition, we have to factor in the body propulsion time between each sub-phase of the gait, which makes it possible to move the body of the robot in the desired direction. This gives us the following equation, see Fig. 7.

$$T_D = 4(t_L + t_F + t_P) + 2t_{BP} \tag{9}$$

Substituting the height and leg strokes into eq. (9), we get:

$$T_D = 4\left(\frac{h}{V_z} + \frac{R_x}{V_x} + \frac{h}{V_z}\right) + 2\frac{R_x}{2V_x} = 8\frac{h}{V_z} + 5\frac{R_x}{V_x} \tag{10}$$

We have to note that the duty cycle does not affect the period of discontinuous gaits.

B. Period of a continuous gait

Calculating the period for a continuous gait is very similar to the discontinuous case. The path followed by the foot is described in section IV-A. In a continuous gait, a leg is in support for $t_s = \beta T_C$ seconds and in transfer for $t_t = (1 - \beta)T_C$ seconds, where T_C is the period of one cycle. Assuming that the feet follow the same trajectory that discontinuous gaits do, we can calculate the transfer leg time.

$$t_t = 2\frac{h}{V_z} + \frac{R_x}{V_x}. \tag{11}$$

therefore

$$T_C = \frac{1}{1-\beta}t_t = \frac{1}{1-\beta}\left(2\frac{h}{V_z} + \frac{R_x}{V_x}\right). \tag{12}$$

Notice that in contrast to discontinuous gaits, the duty factor of the gait is proportional to the gait period, i.e. if β increases, T_C also increases and vice versa.

V. Gait velocity

We have already calculated the gait period for wave and discontinuous gaits, we know the leg stroke (step length). This information is enough to calculate the gait velocity with a few assumptions:

1) The velocities along the X and Z axis are equal;
2) The step height is proportional to the workspace size (R_x) and inversely proportional to parameter K which weighs the step height ($h = R_x/K$).

We have to calculate the velocity for wave and discontinuous gaits separately. In continuous gaits, we can describe the velocity as

$$v_c = \frac{\lambda}{T_C} = \frac{\lambda(1-\beta)V_xV_z}{2hV_x + R_xV_z} \tag{13}$$

and in discontinuous gaits as

$$v_d = \frac{R_x}{T_D} = \frac{R_xV_xV_z}{8hV_x + 5R_xV_z}. \tag{14}$$

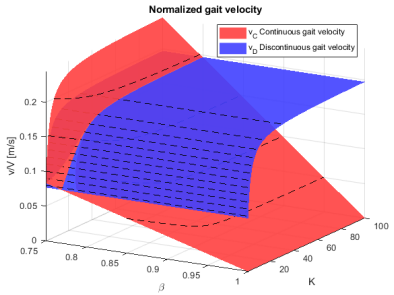

Fig. 8. Wave and discontinuous gait velocities.

In Fig. 8 we can observe that the duty factor does not influence the discontinuous gait velocity. However, in the case of wave gaits the gait velocity is inversely proportional to the duty factor. We can also notice that as the parameter K approaches infinity, the surface given by eq. (14) flattens around $v_{d_N} = 0.2m/s$. On the line of intersection between the two surfaces, we cannot differentiate between wave gait velocities and discontinuous gait velocities.

If we want to achieve a normalized velocity around $0.2m/s$, where the surface for discontinuous gait velocity flattens, we have to choose the β and K parameters in the blue region, otherwise, we have to choose parameters in the red region and implement a wave gait.

VI. Path planning for one leg

It is important to note the difference between one leg's path and the path of the robot that it takes while walking. The path must be inside the leg's workspace. We can ensure that the path stays inside the leg's workspace by implementing a rectangular path which can be described with 6 parameters. These parameters are limited by the workspace.

- Start position: $S = (X_s, Y_s, Z_n)$
- Height of step: $h = Z_m$
- Goal position: $F = (X_g, Y_g, Z_n)$

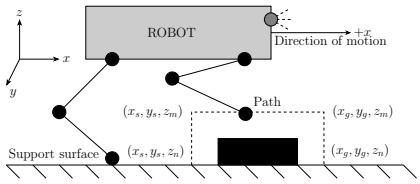

Fig. 9. Path that the leg has to follow.

In Fig. 9 we can see the aforementioned rectangular path that the leg has to follow. The motion of the robot is in the $+X$ direction (forward). We can clearly distinguish four different points in space and we can interpolate between them to calculate the intermediate points, this improves the path's resolution.

This presents a problem: we know the points the leg must follow in space but not the joint angles needed to actuate the motors. A solution to this problem is called *inverse kinematics*. Inverse kinematics computes the joint angles for each leg to achieve a target position in the body-frame coordinate system. The calculation of the inverse kinematics can be a challenging task. The classic solutions include geometric, iterative and algebraic methods, but these can be computationally expensive if the joint space and configuration of the robot itself are complex [17]. An alternate solution to calculating the inverse kinematics is proposed in this work using feed-forward artificial neural networks, as it will be presented in section VII.

VII. Inverse Kinematics Solver Using Neural Networks

As mentioned previously, the calculation of the inverse kinematics is a computationally expensive problem if the joint space and configuration of the robot are complex. In this case, one leg consists of three revolute joints in series, so we have to figure out three joint angles based on a target position, see eq. (2). The solution for inverse kinematics can have multiple results among which solutions might not be optimal.

An alternate method for computing the inverse kinematics of a serial robot is the use of feed-forward artificial neural networks. Remember that one leg of the robot can be thought of as a serial robot with 3 degrees of freedom. The two main network types used in robot model estimation are Radial Basis Functions (RBF) and Multilayer Perceptrons (MLP). Approximating a robot model using RBF neural networks gives us greater accuracy than MLP neural networks. However, the computation of the base functions is computationally expensive. For this reason, the use of offline-trained MLP neural networks is more convenient and straightforward.

The neural network was trained and implemented in MATLAB using the Deep Learning Toolbox. The structure of the feed-forward artificial neural network consists of one input vector with $n = 3$ elements, $q = 10$ hidden layers and $m = 3$ outputs.

The input of the neural network is the position of the foot in the body frame reference while the outputs are the joint angles required to achieve the target position.

A. Generating training data

The first step to training a feed-forward neural network is collecting the training data, this was done with three different methods.

1) Method 1: We have generated three different sawtooth signals for each joint with an evenly distributed randomized phase offset in the interval $[-\pi, \pi]$, with angular velocities 2π, 10π and 20π respectively, we added high-frequency sinusoids and a randomized vector to avoid duplicate data points in the set, see Fig. 10. Every generated angle was also mapped to the interval $[\theta_{i_{\min}}, \theta_{i_{\max}}]$ which is the limit of the given joint.

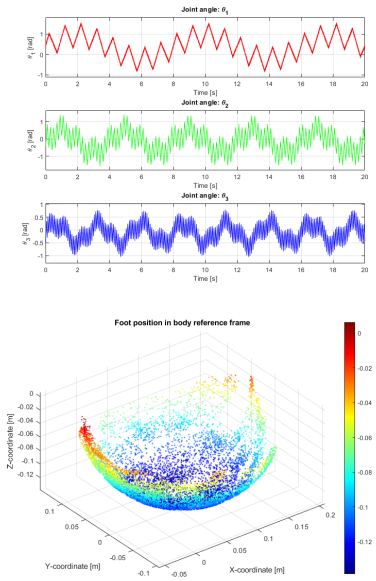

Fig. 10. Generated points based on sawtooth signals.

2) Method 2: The joint angles were randomly generated with an even distribution in the interval $[\theta_{i_{\min}}, \theta_{i_{\max}}]$. This gives us a more evenly distributed result.

3) Method 3: The foot positions were generated by calculating all possible permutations of the joint angles with a specified resolution.

B. Training the neural network

After generating the training data, we trained the neural network with a training set the size of $100,000$ data points, using the scaled conjugate gradient (SCG) method, with a maximum epoch count of 2000 and 50 validation checks. During training the data was randomly shuffled. The training, validation and test ratio was 0.7, 0.15 and 0.15 respectively. The training was conducted using an AMD Ryzen 9 5900HX processor with 16GB of RAM. The mean performance metrics for the three training sets were $120.97s$, $14.514s$, and $132.327s$, respectively. The performance analysis shows that

method 1, see VII-A1, amongst the two other methods, see VII-A2, VII-A3, performs the best, see Fig. 11.

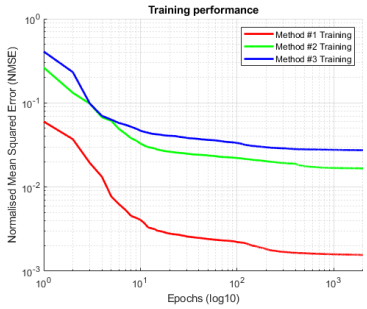

Fig. 11. Performance analysis.

C. Evaluating the neural network

The testing set was generated with the method mentioned in section VII-A2. We can also cross-validate the results found in VII-B. The testing set contained 2000 data points in total.

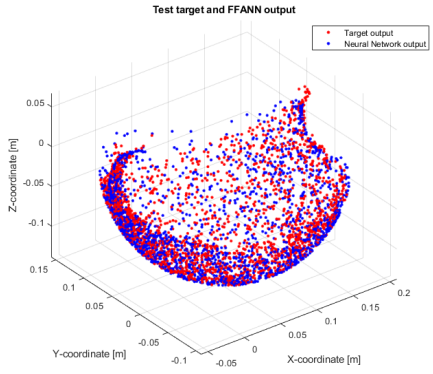

Fig. 12. Test target and neural-network output.

We can observe that the neural network output is close to the target.

VIII. Conclusions

This paper introduces an intelligent control approach for quadrupedal robots by combining an efficient periodic gait method with a cost-effective inverse kinematics solver that utilizes feed-forward artificial neural networks. Our approach aims to reduce computational expenses while maintaining high performance during robot control. The simulations and calculations conducted have successfully met our expectations, demonstrating results that are comparable to those achieved by existing solutions.

Furthermore, we offer an in-depth analysis of locomotion, addressing key aspects such as periodic gaits, stability, gait period, and velocity. This comprehensive study not only highlights the effectiveness of our proposed method but also contributes valuable insights into the dynamics and control of quadrupedal robots. Our findings underscore the potential of artificial neural networks in enhancing robotic control systems, paving the way for more efficient and accessible solutions in the field.

To expand upon the findings of this research, we intend to implement the proposed gait planning and control methods in a physical system for experimental testing. The testing will be performed using the aforementioned Dogzilla S1 [12] quadruped mobile robot.

References

[1] P. Biswal and P. K. Mohanty, "Development of quadruped walking robots: A review," *Ain Shams Engineering Journal*, vol. 12, no. 2, pp. 2017–2031, 2021.

[2] X. Meng, S. Wang, Z. Cao, and L. Zhang, "A review of quadruped robots and environment perception," in *2016 35th Chinese Control Conference (CCC)*. IEEE, 2016, pp. 6350–6356.

[3] Y. Zhong, R. Wang, H. Feng, and Y. Chen, "Analysis and research of quadruped robot's legs: A comprehensive review," *International Journal of Advanced Robotic Systems*, vol. 16, no. 3, 2019.

[4] F. Rubio, F. Valero, and C. Llopis-Albert, "A review of mobile robots: Concepts, methods, theoretical framework, and applications," *International Journal of Advanced Robotic Systems*, vol. 16, no. 2, 2019.

[5] A. A. Aldair, A. Al-Mayyahi, and W. Wang, "Design of a stable an intelligent controller for a quadruped robot," *Journal of Electrical Engineering & Technology*, vol. 15, no. 2, pp. 817–832, 2020.

[6] D. Marhefka, D. Orin, J. Schmiedeler, and K. Waldron, "Intelligent control of quadruped gallops," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 4, pp. 446–456, 2003.

[7] L. R. Palmer and D. E. Orin, "Intelligent control of high-speed turning in a quadruped," *Journal of Intelligent and Robotic Systems*, vol. 58, pp. 47–68, 2010.

[8] M. Li, Z. Liu, M. Wang, G. Pang, and H. Zhang, "Design of a parallel quadruped robot based on a novel intelligent control system," *Applied Sciences*, vol. 12, no. 9, 2022.

[9] C. Lee and D. An, "Reinforcement learning and neural network-based artificial intelligence control algorithm for self-balancing quadruped robot," *Journal of Mechanical Science and Technology*, vol. 35, no. 1, pp. 307–322, 2021.

[10] E. Burkus and P. Odry, "Autonomous hexapod walker robot" szabad (ka)"," in *2007 5th International Symposium on Intelligent Systems and Informatics*. IEEE, 2007, pp. 103–106.

[11] R. K. Mandava, K. Mrudul, and P. R. Vundavilli, "Dynamic motion planning algorithm for a biped robot using fast marching method hybridized with regression search," *Acta Polytech. Hung*, vol. 16, pp. 189–208, 2019.

[12] Yahboom, "Dogzilla s1 documentation," http://www.yahboom.net/study/DOGZILLA-S1.

[13] F. Gao, C. Qi, Q. Sun, X. Chen, and X. Tian, "A quadruped robot with parallel mechanism legs," in *2014 IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2014, pp. 2566–2566.

[14] A. A. Hayat, R. G. Chittawadigi, A. D. Udai, and S. K. Saha, "Identification of denavit-hartenberg parameters of an industrial robot," in *Proceedings of conference on advances in robotics*, 2013, pp. 1–6.

[15] P. G. De Santos, E. Garcia, and J. Estremera, *Quadrupedal locomotion: an introduction to the control of four-legged robots*. Springer, 2006, vol. 1.

[16] E. Garcia, J. Estremera, and P. G. De Santos, "A comparative study of stability margins for walking machines," *Robotica*, vol. 20, no. 6, pp. 595–606, 2002.

[17] R. Köker, C. Öz, T. Çakar, and H. Ekiz, "A study of neural network based inverse kinematics solution for a three-joint robot," *Robotics and autonomous systems*, vol. 49, no. 3-4, pp. 227–234, 2004.