CPG-Based Gait Generator for a Quadruped Robot with Sidewalk and Turning Operations

Vladimir Danilov and Sekou Diane

May 22, 2022

CPG-based Gait Generator for a Quadruped Robot with sidewalk and turning operations

Danilov Vlaimir[1][0000-0002-6038-3889] and Diane Sekou[1][0000-0002-8690-6422]

[1] Institute of Control Problems of Russian Academy of Sciences, Moscow, Russia
vldanilov90@gmail.com, sekoudiane1990@gmail.com

Abstract. This article describes the quadruped robot gait generator algorithm based on the central pattern generator (CPG). The proposed architecture uses CPG as a phase signal generator for each leg, and a mapping function that builds the desired trajectory for the robot feet. The algorithm is able to change smoothly and online the gait type, movement direction, frequency, height and length of the stride. In order to test the performance of the algorithm, experiments were carried out both in the simulation and on the real robot. The results show the efficiency of the algorithm and its ease of implementation.

Keywords: quadruped robot, Central Pattern Generator (CPG), gait generator.

1 Introduction

Walking robots are increasingly being used in various fields of human activity, as they have a greater cross-country ability, unlike wheeled vehicles. Over the past forty years there has been tremendous progress in this area. But still, the complexity of algorithms for controlling stable locomotion on various surfaces is still a serious problem.

To implement a stable static gait, the method of maintaining the center of mass point inside the reference polygon is used [1]. An extended version of this algorithm is the search for zero moment point (ZMP) [2, 3]. Its essence is to keep ZMP of the robot in the center of the support polygon. This algorithm is well suited for six-legged robots [4] or slow four-legged ones [5]. This is due to the fact that the supported polygon is not presented during movement with gaits such as trot or gallop.

Another algorithm is CPG (Central Pattern Generator) inspired by research on mammalian locomotion. Neuroscientists have found that for the formation of rhythmic movements in the process of purposeful locomotion of animals, central pattern generators located in the spinal cord are used [6-7]. A large number of implementations of this approach for mobile robots have been produced [8-11]. When applied to walking robots, usually each CPG neuron generates the desired joint angle [12-14]. The disadvantage of this method is that, in reality, joint trajectories are much more complicated than generated by CPG, and tedious manual adjustment of the parameters of each neuron is required for stable movement.

There is a large group of control algorithms based on dynamics of a robot mechanics: SLIP (Spring Loaded Inverted Pendulum), VMC (Virtual Model Control), MPC (Model Predictive Control). The SLIP method was first proposed in [16] and is based on the “virtual leg” concept [15]. Many works have been done [17–20], including

Boston Dynamics' BigDog [19]. VMC has been successfully applied to HyQ [21] and StarlETH robots [22]. MPC is used in a series of robots Cheetah [23-27], ANYmal [28] and HyQReal [29]. As it can be seen, these methods are widely used for solving locomotion control problem. However, they require significant prior knowledge of the target robotic platform and the task [30], a very accurate dynamic robot model that is often difficult to obtain [31], and the need to manually adjust the parameters of the algorithms [32]. Moreover, separate controllers for each movement mode are used, which have a different architecture, and are designed and configured separately [35].

Recently, model-free reinforcement learning (RL) algorithms have been widely advanced that can avoid the disadvantages described above. Although these algorithms have a large number of parameters, nevertheless, their configuration can be automated. However, there are two serious problems: a long training time and the difficulty of migration from simulation to hardware called the reality gap.

Solving the first problem, one can use parallel training of several thousand agents, which reduces the training time to several minutes [36]. Predefined gait generators are also used, and the task of the agent is to optimize their trajectories [37]. The problem of the first implementation is that the robot could not change its speed of movement, and in the second case, the robot moved only along the lateral plane and could not do gait transition online.

The solution to the second problem is combining accurate mathematical model of servo drives, the randomization of mechanical parameters and proprioceptive sensor data during training, as well as the imitation of external force disturbances acting on the robot [35, 38]. This algorithm provided stable and dynamic locomotion on such robots as Minitaur [30, 31, 33, 34, 37, 38], Unitree A1 [32] and ANYmal [35,36].

Based on the information above, the most interesting and promising approach is RL. The proposed scheme of the algorithm is shown in fig. 1. It is based on the architecture obtained in [37] but it uses more efficient CPG as a gait generator. The CPG-based gait generator allows to smoothly change the gait pattern, stride frequency, length and height, as well as additionally perform sidewalk and turning online. The task of the trained agent is to set the input parameters for the CPG block and optimize its output trajectories of robot feet. GG State, Control Input and Robot Feedback signals form the agent observation vector. Thanks to a predefined gait generator, the training time of the agent should be greatly reduced, and its CPG implementation with sidewalk and turning operations will also allow the agent to perform these actions.

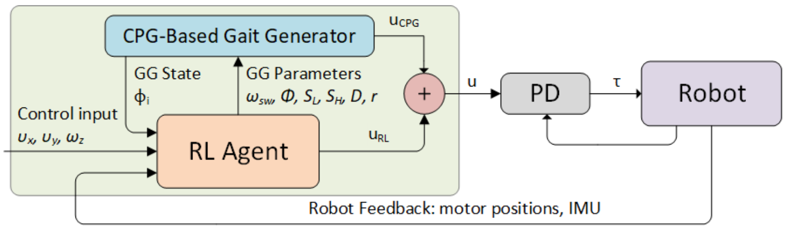

CPG-Based Gait Generator
u_{CPG}
GG State
ϕ_i
GG Parameters
$\omega_{sw}, \Phi, S_L, S_H, D, r$
Control input
$\upsilon_x, \upsilon_y, \omega_z$
RL Agent
u_{RL}
+
u
PD
τ
Robot
Robot Feedback: motor positions, IMU

Fig. 1. Proposed robot locomotion control algorithm architecture

In this paper, we focus on the CPG-based Gait Generator block. It must have the following properties:

- The network of CPG neurons must provide periodic clock signals for movement of a robot and smooth gaits transition by adjusting only one parameter. The frequency of the signals should also vary smoothly based on a single parameter;
- Foot trajectories are calculated based on the rhythmic signals obtained by the CPG network;
- Trajectories must take into account both changes in stride height and stride length, as well as changes in direction of movement.

The algorithm will be tested on a quadruped robot developed by Voltbro and MSU Institute of Mechanics.

2 Gait Generator

The formation of the foot trajectories during locomotion is usually carried out using generators based on inverse kinematics [39]. The disadvantage of this method is that it does not allow both to do gait transition and to change stride frequency smoothly. The most attractive way is to use CPG, which is capable to generate trajectories of the foot positions [40, 41]. These publications describe development of such generators, but only to move forward. In this paper, we create a generator capable of form locomotion trajectories in all directions.

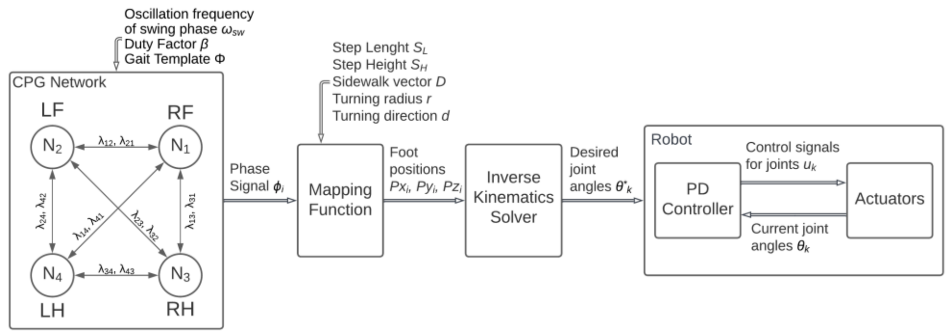

Oscillation frequency of swing phase ω_{sw}
Duty Factor β
Gait Template Φ
CPG Network
LF
RF
N_2
N_1
$\lambda_{12}, \lambda_{21}$
$\lambda_{24}, \lambda_{42}$
$\lambda_{14}, \lambda_{41}$
$\lambda_{23}, \lambda_{32}$
$\lambda_{13}, \lambda_{31}$
N_4
N_3
$\lambda_{34}, \lambda_{43}$
LH
RH
Phase Signal ϕ_i
Step Lenght S_L
Step Height S_H
Sidewalk vector D
Turning radius r
Turning direction d
Mapping Function
Foot positions Px_i, Py_i, Pz_i
Inverse Kinematics Solver
Desired joint angles θ^*_k
Robot
Control signals for joints u_k
PD Controller
Actuators
Current joint angles θ_k

Fig. 2. Structure of the Gait Generator

2.1 Design Architecture

Fig. 2 shows the general block diagram of the gait generator. The CPG Network block consists of four oscillators, each of which is responsible for formation of a phase trajectory of each leg. Stride frequency ω_{sw} and the vector Φ describing the gait type are taken as input. The output contains four phase signals ϕ_i, which are fed to the Mapping Function. The task of this block is to form stride cycles based on ϕ_i. The input parameters are: S_L - stride length, S_H - stride height, D – linear direction, r – turning radius and *d* - for turning direction (-1 - clockwise, 0 - straight, 1 - counterclockwise). The output parameters are desired foot position vectors P_{xi}, P_{yi}, P_{zi}. The Inverse Kinematics Solver block solves the inverse kinematics problem and, based on the desired positions, gives the desired joint angles, which are fed to the low-level part of the robot that controls the servos. This block is implemented by widely used matrix method [43-45].

2.2 CPG Network

CPG network consists of four symmetrically connected neurons, each of them is an oscillator that modulates the phase signal for a particular leg. There are several types of oscillators [42]. We choose the Hopf oscillator because it is the least computationally demanding, moreover it is able to provide fast switching between different gaits. The oscillator equations look like this:

$$\dot{x}_i = \alpha(\mu - r_i^2)x_i - \omega_i y_i + \eta \sum_{j=1}^{4}\left(x_j \cos\lambda_{ji} - y_j \sin\lambda_{ji}\right) \tag{1}$$

$$\dot{y}_i = \alpha(\mu - r_i^2)y_i + \omega_i x_i + \eta \sum_{j=1}^{4}\left(x_j \sin\lambda_{ji} - y_j \cos\lambda_{ji}\right) \tag{2}$$

$$r_i = \sqrt{x_i^2 + y_i^2} \tag{3}$$

$$\omega_i = \left(\frac{\beta}{(e^{-ay_i+1})} + \frac{1}{e^{ay_i+1}}\right)\omega_{sw} \tag{4}$$

$$\phi_i = atan2(y_i, x_i), \tag{5}$$

where α – convergence rate, $\sqrt{\mu}$ – amplitude, i – oscillator number, ω_i – stride frequency, β – duty factor, ω_{sw} – swing phase frequency, a – alternation rate coefficient between swing and stance phases, $\lambda_{ji} = \varphi_i - \varphi_j$ – phase difference between legs *i* and *j*, $\varphi_i \in [0; 2\pi]$, x_i, y_i – internal oscillator state, η – coupling coefficient, ϕ_i – current phase of the oscillator, that is considered as a synchronization signal for the formation of the foot trajectories, $\phi_i \in [-\pi; 0)$ – stance phase, $\phi_i \in [0; \pi]$ – swing phase.

The phase differences φ_i for four different gaits are shown in Table 1. By setting these parameters for λ_{ji}, one can obtain different locomotion rhythms. The output values of the oscillators ϕ_i for gait patterns Φ from Table. 1 are shown in fig. 3. On the above graphs, one can observe a slight curvature of the lines. This is due to the value of β: the larger it is, the smaller the slope ϕ_i in the stance phase and the greater in the swing phase. By changing the parameters Φ, β and ω_{sw}, it is possible to perform a smooth change of ϕ_i. An example of a transition from walk to a more dynamic trot gait is shown in Fig. 4. Note, that gait transition occurs a little more than just one second.

Table 1. Phase relationship for different gait patterns.

Gait	RF φ_1	LF φ_2	RH φ_3	LH φ_4
Walk	π	0	0.5 π	1.5 π
Trot	π	0	0	π
Pace	π	0	π	0
Gallop	0	0	π	π

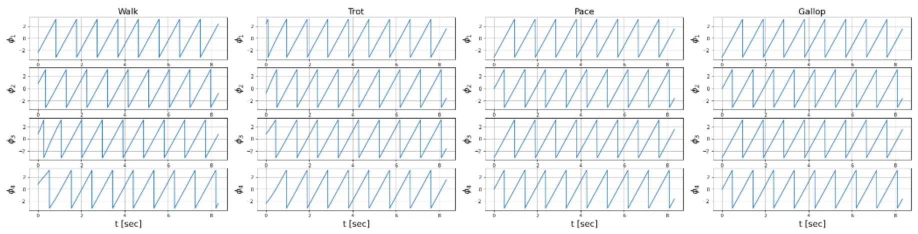


Fig. 3. CPG output curves ϕ_i for different gait patterns

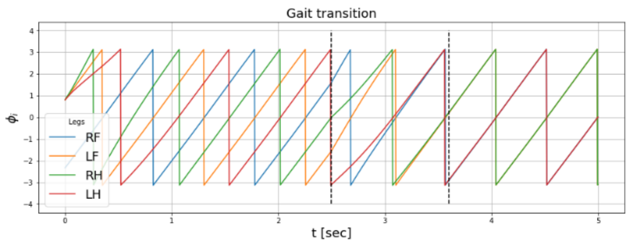


Fig. 4. Walk to trot transition. Dashed lines show the beginning and the end of the transition

2.3 Mapping Function

The task of the Mapping Function block is to form the desired trajectory of the robot feet based on the clock signal φ_i. Also, this block must take into account the desired stride height and stride length, and at the same time carry out turning and sidewalk. The algorithms used for the mapping function [40, 41] assume only forward movement. To implement other types of movement, the algorithm was refined; its block diagram is shown in Fig. 5. At the input, it takes the values of the clock signal ϕ_i, the stride length S_L, the stride height S_H, the two-dimensional velocity vector in the XY plane $D = \begin{bmatrix} v_x & v_y \end{bmatrix}^T$ and the turning radius r. Next, the desired foot positions are calculated. After that, the lateral deflection is sequentially calculated by the values sidewalk vector D, turning radius r and turning direction d. At the very end, the obtained trajectory is translated into the coordinate system of the robot foot relative to the center of mass of the body and the desired foot position P_x, P_y, P_z is calculated.

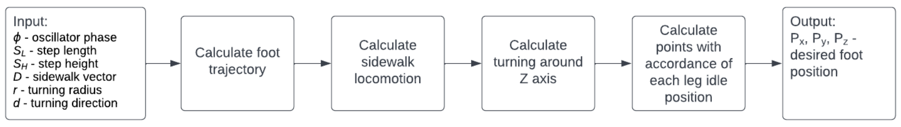

Input:
ϕ - oscillator phase
S_L - step length
S_H - step height
D - sidewalk vector
r - turning radius
d - turning direction
Calculate foot trajectory
Calculate sidewalk locomotion
Calculate turning around Z axis
Calculate points with accordance of each leg idle position
Output: P_x, P_y, P_z - desired foot position

Fig. 5. Block diagram of Mapping Function algorithm

The basic calculation of the foot trajectory is carried out using the following equations:

$If\ \phi_i \in [-\pi; 0)$:

$$P_{x0} = -\left(\frac{1}{\pi} S_L \phi + \frac{S_L}{2}\right) \tag{6}$$

$$P_{z0} = 0 \tag{7}$$

$$P_{y0} = 0 \tag{8}$$

$If\ \phi_i \in [0; \pi]$:

$$P_{x0} = -\frac{1}{2\pi} S_L \sin(4\phi) + \frac{1}{\pi} S_L \phi + \frac{S_L}{2} \tag{9}$$

$$P_{z0} = \begin{cases} -\frac{1}{2\pi} S_H \sin(4\phi) + \frac{2}{\pi} S_H \phi, & \phi \in \left[0; \frac{\pi}{2}\right] \\ \frac{1}{2\pi} S_H \sin(4\phi) - \frac{2}{\pi} S_H \phi + 2S_H, & \phi \in \left[\frac{\pi}{2}; \pi\right] \end{cases} \tag{10}$$

$$P_{y0} = 0, \tag{11}$$

where P_{x0}, P_{y0}, P_{z0} – foot position, S_H, S_L – stride height and stride length respectively.

Sidewalk direction is calculated by determining the angle ψ (Fig. 6):

$$\psi = atan2(D_y, D_x) \tag{12}$$

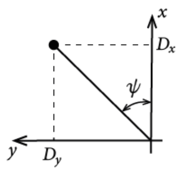


Fig. 6. Robot movement direction

And then the points of the generated trajectory are multiplied by the rotation matrix R_z:

$$P_1 = R_z(\psi) \cdot P_0 = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{x0} \\ P_{y0} \\ P_{z0} \end{bmatrix} = \begin{bmatrix} P_{x0}\cos\psi - P_{y0}\sin\psi \\ P_{x0}\sin\psi + P_{y0}\cos\psi \\ P_{z0} \end{bmatrix} \tag{13}$$

Turning of the robot is performed with respect to a certain radius r. To implement the turn, it is necessary to imagine the position of the legs, as shown in Figure 7. The modification of the trajectories for the turn is carried out using the equations [46]:

$$\gamma = d * atan2(h, 2r + d * b),\ \xi = d * \text{atan2}\ (h, 2r - d * b) \tag{14}$$

$$P_2 = \begin{bmatrix} \cos\sigma_{\gamma\xi} & -\sin\sigma_{\gamma\xi} & 0 \\ \sin\sigma_{\gamma\xi} & \cos\sigma_{\gamma\xi} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_{x1} \\ P_{y1} \\ P_{z1} \end{bmatrix}, \tag{15}$$

where d – turning direction, $\sigma_{\gamma\xi}$ - γ or ξ angles. $\sigma_{\gamma\xi} = \gamma$, if the leg is right, otherwise $\sigma_{\gamma\xi} = \xi$.

The final operation is shifting each vector according to the base position of each foot:

$$P = P_2 + P_{idle}, \tag{16}$$

where P_{idle} – the vector of the idle position of the foot relative to the center of mass of the robot body.

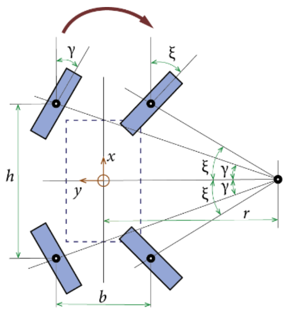


Fig. 7. Turning implementation

3 Results

This section presents the results of experiments on robot locomotion using the control algorithm described above.

3.1 Simulation Experiments

PyBullet is used as a simulation environment [46]. The robot model is described using URDF. Fig. 8 shows a visual image of the robot in the PyBullet environment.

In all tests, the robot moves on a flat surface. The model processing frequency is 300 Hz. The parameters of the algorithm are presented in Table 2. The values of a, μ, α did not lead to any major changes in the operation of the algorithm and were set as in [40].

Table 2. Parameters used in experiments for the CPG model and mapping function.

Parameter	Walk, trot, pace	Gallop
ω_{sw}	4π	6π
β	2	2
α	100	100
μ	1	1
a	100	100
S_H	0.06	0.08
S_L	0.06	0.08

The first experiment was the movement along the longitudinal plane of the robot, using all the gaits from Table. 1. The simulation was performing for 5 seconds. Each gait showed the possibility of locomotion. Trot turned out to be the fastest gait, and in 5 seconds the robot covered ≈1.95 m. The slowest gait was walk. Using it, the robot covered only 1 meter in the same time. Trot, pace and walk have minor deviations along the transverse Y axis, while the deviation of gallop was 0.2 m. Also, given the increased frequency of movement and stride length, the gallop was almost the slowest and least stable. Setting values, as in other gait types, the robot refused to make a steady movement. All this confirms the need to stabilize robot locomotion by the method presented in Chapter 1.

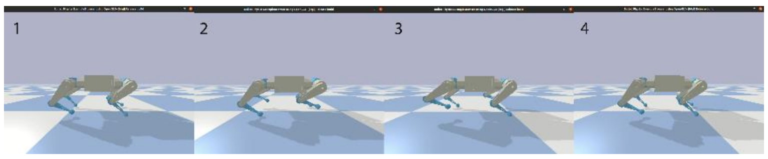
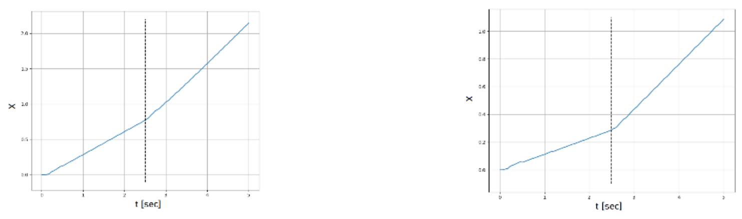


Fig. 8. Robot visualization in the PyBullet simulation environment while executing trotting gait



a) b)

Fig. 9. Dependences of the CoG X position of the robot base on time with an increase in (a) stride frequency, (b) stride length

The second experiment is the effect of changing stride frequency and stride length. The robot walked using the trot gait, each simulation episode lasted 5 seconds. At the time t=2.5 sec, the following was performed: a) increase in step frequency from 4π to 6π with a step length of 0.03 m and b) increase in step length from 0.03 m to 0.08 m with a frequency of 4π. As can be seen in fig. 9, the algorithm successfully changes these parameters, and the robot increases locomotion speed.

The third experiment is the transition between gaits during locomotion. Similar to previous experiments, the simulation lasted five seconds. At the beginning of the experiment, the robot moved with a walk gait, and at the time t=2.5 sec, the gait changed to a trot. As can be seen on the plot in Fig. 10, the robot successfully performed a gait transition in approximately 1 second, which can be seen from the change in the nature of the oscillations along the vertical Z axis.

The fourth experiment is testing robot turning capability. The simulation lasted 15 seconds, during which the robot made four turns. The experiment ended successfully, showing capability of implementing this operation. The results are presented in fig. 11.

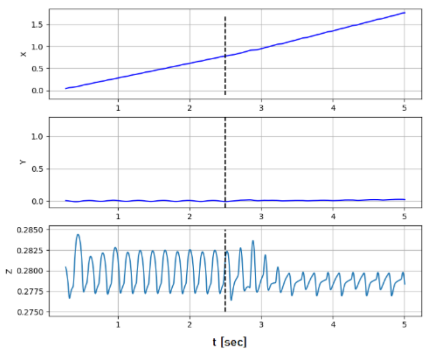
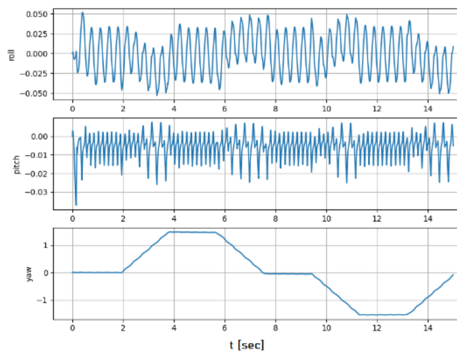
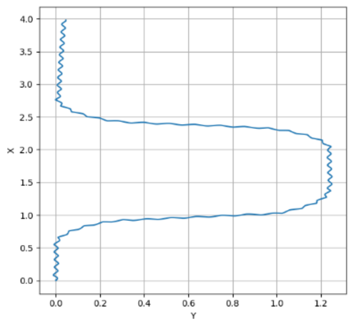


Fig. 10. Gait transition between walk and trot



a)



b)

Fig. 11. Implementation of changing the direction of the robot. (a) CoG orientation angles, (b) CoG movement in the horizontal plane.

3.2 Hardware experiments

While testing the performance of the algorithm on the hardware platform, the initial data from Table 2 was used. The robot moved on a flat surface.

As a result of the experiments, it was possible to move with all the proposed gait types with the ability to switch between them on the go and change other parameters (frequency, length and height of the step), even with the absence of feedback to stabilize the movement. The robot also coped with the task of changing the locomotion direction, executing both sidewalk and turning. Photographs of the robot movement during the experiments are shown in fig. 12 and 13. Also you can watch the experiments in the supplementary video1[1].

[1] https://youtu.be/bFAk3QXRYn4

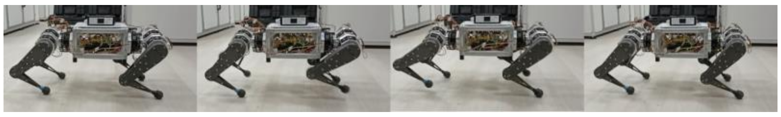


Fig. 12. Trot gait experiments

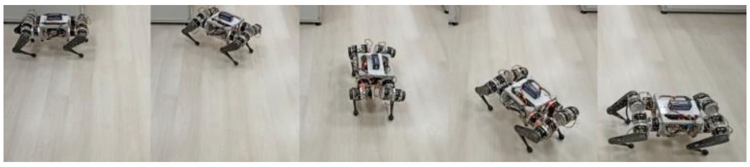


Fig. 13. Changing moving direction experiments

4 Conclusions

In this paper, the Gait Generator algorithm is featured that combine CPG and mapping function with sidewalk and turning operations. All the gaits such as trot, pace, walk and gallop were executed both in simulation and real world. Moreover, this technique is able to change gait type online, locomotion direction, stride frequency, stride height and stride length. The results show the efficiency and ease of this approach.

Since the robot has to manage with rough terrain and stabilize its body under external disturbances, an additional work with feedback control is needed. Proposed approach is feet trajectory optimization with RL approach which is the aim of the feature work.

References

1. Lee, T.T., Liao, C.M., Chen, T.K.: On the stability properties of hexapod tripod gait. IEEE J. Robotics Autom. 4, 427–434 (1988).
2. Vukobratovic, M., Borovac, B.: Zero-moment point - thirty-five years of its life. International Journal of Humanoid Robotics 01(01), 157–173 (2004).
3. Li, J., Wang, J., Yang, S.X., Zhou, K., Tang, H.: Gait planning and stability control of a quadruped robot. Computational Intelligence and Neuroscience 2016, 9853070 (2016).
4. Xin, G., Deng, H., Zhong, G., Wang, H.: Gait and trajectory rolling planning for hexapod robot in complex environment. In: Mechanism and Machine Science, pp. 23–33. Springer Singapore, Singapore (2017).
5. Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., Schaal, S.: Fast, robust quadruped locomotion over challenging terrain. In: 2010 IEEE International Conference on Robotics and Automation, pp. 2665–2670 (2010).
6. Delcomyn, F.: Neural basis of rhythmic behavior in animals. Science 210(4469), 492–498 (1980).
7. Grillner, S., Zangger, P.: On the central generation of locomotion in the low spinal cat. Experimental Brain Research 34(2), 241–261 (1979).
8. Fukuoka, Y., Kimura, H., Cohen, A.H.: Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. The International Journal of Robotics Research 22(3-4), 187–202 (2003).

9. Grillner, S., Wallen, P., Brodin, L., Lansner, A.: Neuronal network generating locomotor behavior in lamprey: Circuitry, transmitters, membrane properties, and simulation. Annual Review of Neuroscience 14(1), 169–199 (1991).
10. Ijspeert, A.J., Crespi, A., Ryczko, D., Cabelguen, J.M.: From swimming to walking with a salamander robot driven by a spinal cord model. Science 315(5817), 1416–1420 (2007).
11. Conradt, J., Varshavskaya, P.: Distributed central pattern generator control for a serpentine robot. In: ICANN 2003 (2003)
12. Liu, C., Chen, Y., Zhang, J., Chen, Q.: CPG driven locomotion control of quadruped robot. In: Proceedings of the 2009 IEEE International Conference on Systems, Man and Cybernetics, SMC'09, p. 2368-2373. IEEE Press (2009)
13. Righetti, L., Ijspeert, A.J.: Pattern generators with sensory feedback for the control of quadruped locomotion. In: 2008 IEEE International Conference on Robotics and Automation, pp. 819–824 (2008).
14. Santos, C.P., Matos, V.: Gait transition and modulation in a quadruped robot: A brainstem-like modulation approach. Robotics and Autonomous Systems 59(9), 620–634 (2011).
15. Sutherland, I.E., Ullner, M.: Footprints in the asphalt. The International Journal of Robotics Research 3, 29 – 36 (1984)
16. Cavagna, G.A., Heglund, N.C., Taylor, C.R.: Walking, running and galloping: mechanical similarities between different animals (1976)
17. Raibert, M.H., Chepponis, M., Brown, H.B.: Running on four legs as though they were one. IEEE J. Robotics Autom. 2, 70–82 (1986)
18. Poulakakis, I., Papadopoulos, E., Buehler, M.: On the stability of the passive dynamics of quadrupedal running with a bounding gait. The International Journal of Robotics Research 25, 669–687 (2006)
19. Raibert, M., Blankespoor, K., Nelson, G., Playter, R.: Bigdog, the rough-terrain quadruped robot. In: 17th IFAC World Congress41(2), 10822–10825 (2008).
20. De, A., Koditschek, D.E.: Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking, and trotting. The International Journal of Robotics Research 37(7), 743–778 (2018).
21. Havoutis Ioannis, S.C.C.D.: Virtual model control for quadrupedal trunk stabilization. Dynamic Walking (2013)
22. Hutter, M., Gehring, C., Hopflinger, M.A., Blosch, M., Siegwart, R.: Toward combining speed, efficiency, versatility, and robustness in an autonomous quadruped. IEEE Transactions on Robotics 30(6), 1427–1440 (2014).
23. Hyun, D.J., Seok, S., Lee, J., Kim, S.: High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the MIT Cheetah. The International Journal of Robotics Research 33(11), 1417–1445 (2014).
24. Park, H.W., Wensing, P.M., Kim, S.: High-speed bounding with the MIT Cheetah 2: Control design and experiments. The International Journal of Robotics Research 36(2), 167–192 (2017).
25. 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: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2245–2252 (2018).
26. Di Carlo, J., Wensing, P.M., Katz, B., Bledt, G., Kim, S.: Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1–9 (2018).
27. Katz, B., Carlo, J.D., Kim, S.: Mini Cheetah: A platform for pushing the limits of dynamic quadruped control. In: 2019 International Conference on Robotics and Automation (ICRA), pp. 6295–6301 (2019).

28. Sleiman, J.P., Farshidian, F., Minniti, M.V., Hutter, M.: A unified mpc framework for whole-body dynamic locomotion and manipulation. IEEE Robotics and Automation Letters 6(3), 4688–4695 (2021).
29. Villarreal, O., Barasuol, V., Wensing, P.M., Caldwell, D.G., Semini, C.: Mpc-based controller with terrain insight for dynamic legged locomotion. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2436–2442 (2020).
30. Ha, S., Xu, P., Tan, Z., Levine, S., Tan, J.: Learning to walk in the real world with minimal human effort (2020).
31. Haarnoja, T., Ha, S., Zhou, A., Tan, J., Tucker, G., Levine, S.: Learning to walk via deep reinforcement learning (2018).
32. Kumar, A., Fu, Z., Pathak, D., Malik, J.: RMA: Rapid motor adaptation for legged robots (2021).
33. Yang, Y., Caluwaerts, K., Iscen, A., Zhang, T., Tan, J., Sindhwani, V.: Data efficient reinforcement learning for legged robots (2019).
34. Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., Levine, S.: Soft actor-critic algorithms and applications (2018).
35. Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., Hutter, M.: Learning agile and dynamic motor skills for legged robots. Science Robotics 4(26) (2019).
36. Rudin, N., Hoeller, D., Reist, P., Hutter, M.: Learning to walk in minutes using massively parallel deep reinforcement learning (2021).
37. Iscen, A., Caluwaerts, K., Tan, J., Zhang, T., Coumans, E., Sindhwani, V., Vanhoucke, V.: Policies modulating trajectory generators (2019).
38. Tan, J., Zhang, T., Coumans, E., Iscen, A., Bai, Y., Hafner, D., Bohez, S., Vanhoucke, V.: Sim-to-real: Learning agile locomotion for quadruped robots (2018).
39. Lee, J.H., Park, J.H.: Turning control for quadruped robots in trotting on irregular terrain. In: Proceedings of the 18th International Conference on Circuits Advances in Robotics, Mechatronics and Circuits, pp. 303–308 (2014)
40. Wang, M., Tang, Z., Chen, B., Zhang, J.: Locomotion control for quadruped robot based on central pattern generators. In: 2016 35th Chinese Control Conference (CCC), pp. 6335–6339 (2016).
41. Liu, C., Chen, Q., Wang, D.: CPG-inspired workspace trajectory generation and adaptive locomotion control for quadruped robots. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 41(3), 867–880 (2011).
42. Yu, J., Tan, M., Chen, J., Zhang, J.: A survey on CPG-inspired control models and system implementation. IEEE Transactions on Neural Networks and Learning Systems 25(3), 441–456 (2014). doi: 10.1109/TNNLS.2013.2280596
43. Craig, J.J.: Introduction to robotics: mechanics and control. Pearson Educacion (2005)
44. Danilov, V., Diane, S., Gonchareko, V., Artamonov, A.: Algorithms for intelligent control of multi-link walking robots with self-learning capabilities. In: 2020 22th International Conference on Digital Signal Processing and its Applications (DSPA), pp. 1–5 (2020).
45. Danilov, V., Goncharenko, V.: Development and implementation of a six-legged walking robot prototype. In: 2020 13th International Conference "Management of large-scale system development" (MLSD), pp. 1–5 (2020).
46. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning, http://pybullet.org (2016–2021)