Bipedal Walking on Humanoid Robots through Parameter Optimization

Conference Paper · July 2022

CITATIONS
5

READS
408

2 authors:

Marc Bestmann
German Aerospace Center (DLR)
28 PUBLICATIONS **216** CITATIONS
SEE PROFILE

Jianwei Zhang
University of Hamburg
433 PUBLICATIONS **6,283** CITATIONS
SEE PROFILE

Bipedal Walking on Humanoid Robots through Parameter Optimization

Marc Bestmann[0000−0002−7857−793X] and Jianwei Zhang[0000−0002−7856−5760]

Department of Informatics, University of Hamburg, 22527 Hamburg, Germany
{marc.bestmann, jianwei.zhang}@uni-hamburg.de

Abstract. This paper presents a open-source omnidirectional walk controller that provides bipedal walking for non-parallel robots through parameter optimization. The approach relies on pattern generation with quintic splines in Cartesian space. Additionally, baselines of achieved walk velocities in simulation for all robots of the Humanoid Virtual Season, as well as some commercial robot models, are provided.

Keywords: Bipedal Walking · RoboCup · ROS2

1 Introduction

Bipedal walking is one of the biggest challenges in humanoid robotics. Some impressive results have been achieved, e.g. on the Cassie robot [18]. Still, there is, to the best of our knowledge, no simple-to-use open-source controller available that works on a majority of bipedal robots. Commercial robots, e.g. the Darwin-OP [16] or the NAO [15] may come with a walk controller provided by the manufacturer. Some of these widespread platforms may also have open-source solutions available, e.g. the different walk approaches from the RoboCup Standard Platform League (SPL) for the NAO robot. But if a team constructs its own robot, it needs to program its own walk controller or at least modify an existing solution due to differences in the kinematics and dynamics. This is one of the key issues that teams are still facing in the RoboCup Humanoid League [7] and it leads to two problems. First, it creates a high entry barrier for new teams since these need to build working hardware and a complete software stack. For some trivial parts of this software stack, e.g. the connector to the game controller, and even for some more complex parts, e.g. the computer vision, software of other teams can easily be used. Since the robot designs differ, this is not true for the walking, which is arguably one of the most complex parts. Second, even established teams may run into problems with their walking controller if they want to modify their hardware, as this can require changes to the controller or at least new parameter tuning. Therefore, teams may hesitate to introduce hardware changes which is problematic, as this kind of development is one of the goals of the league.

We present an open-source omnidirectional walk controller that is easy to use and works on all non-parallel robots in the Humanoid League Virtual Season (HLVS). Additionally, we provide baselines for the achieved walk velocities

that are reached on different robots. These are measured in the standardized simulation environment of HLVS and are therefore easily comparable. Our goal is to allow new teams an easier entry into the league as well as allowing comparison of self-created walk controllers to a baseline for any specific robot. Furthermore, our controller allows to easily test new robot platforms or modifications to existing ones, thus enabling a faster hardware development cycle.

The walking consists of two parts: an open loop pattern generator based on parameterized quintic splines and a closed loop stabilization module. First, the pattern generator describes the desired trajectory of the feet in Cartesian space. Then, PID controllers modify the torso orientation based on IMU data in Cartesian space and optionally step phase modulations can be applied based on either joint torque or foot pressure data. Due to the usage of the standard MoveIt [11] interface, different inverse kinematics (IKs) can be applied, including a generally applicable memetic approach. This facilitates the general usability of the walking as we can abstract from the concrete kinematic structure of the robot. Additionally, the walking does not require a model of the robot's dynamics but a set of parameters for the spline definitions and the PID controllers. We show how these can be optimized automatically by using the Multi-Objective Tree-structured Parzen Estimator (MOTPE). Integrating the walking into an existing code base is convenient, since we provide a ROS 1 and a ROS 2 version as well as direct interfaces in C++ and Python.

For our experiments, it was necessary to create URDF models as well as MoveIt configurations for all evaluated robots. We provide these too, as they may facilitate others in running software on different robots.

Our key contributions are:

- Open source walk controller that is easily usable on any non-parallel humanoid robot
- Baselines of stable walk velocities on different robot platforms
- Collection of ROS 2 URDF description and MoveIt configurations for various humanoid robots
- Comparison of different parameter optimization approaches

2 Related Work

There are many existing approaches to bipedal walking, but we are focusing in this section on approaches that are either applied in the RoboCup domain or similar to our presented approach. Model-free approaches that consist of an open-loop trajectory generator and a stabilizing mechanism are often used in RoboCup [8,23]. These do not require exact dynamic models, which are difficult to obtain for the typical low-cost robots due to sensor noise and delay, joint backlash, and imperfect actuation. The trajectories can be generated in joint space or in Cartesian space, but specifically designed leg representations can also be used [8]. Generating the trajectory in joint space does not require an IK and therefore less computation. On the downside, it can only be applied on robots with a certain joint configuration and is, therefore, less transferable between

different robots. While some stabilization methods only work in Cartesian space, there are different methods such as the hip and the ankle strategy that work directly in joint space [3]. As an improvement to Euler angles, fused angles [5] were proposed for improved modeling of balance and applied as the basis for stabilization mechanisms [6].

Typically, a set of parameters needs to be optimized for a walk controller since their performance highly depends on the used parameter set. Therefore, different approaches have been investigated. Shafii et al. used Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [17] to search for optimal walking parameters in simulation [27]. CMA-ES was also used by Seekircher et al. to optimized their model parameters to improve its predictions [26]. Rodriguez et al. used Bayesian optimization for walk parameters with a combination of simulated and real experiments [22]. Silva et al. used reinforcement learning on two of their walking parameters [28]. In the domain of quadrupeds, Saptura et al. used the Nondominated Sorting Genetic Algorithm II (NSGA-II) [13] to optimize the walk parameters [25].

We have used quintic splines with parameter optimization for controlled stand-up motions [29]. While the general idea of our previous work is similar, a different optimization procedure and different objectives are required for bipedal walking. Additionally, our previous work was not evaluated on so many robots and with no standardized simulation parameters, as these have only become available recently.

3 Walk Controller

The presented walk controller expects a goal walk velocity as input, which is typically provided by the path planning or human teleoperation. Based on this, a finite state machine (FSM) decides on the kind of step that needs to be performed and keeps track of the current step phase. Depending on the step type, movements for the foot and torso are defined through Cartesian quintic splines. Different stabilization approaches can be used based on sensor feedback. An overview of the approach can be seen in Figure 1 and the following sections explain the different parts in more detail.

3.1 Finite State Machine

The walking controller needs to be able to stably start and stop the walking. Therefore, it needs to perform different types of steps, that handle the movement of the torso, and thereby the movement of the CoM, differently. Before lifting a foot for the first step, it starts moving the torso to the side, initializing the lateral swinging motion of the robot. Then the first step is performed with a phase offset between the torso and foot movement. A similar procedure is performed when stopping to walk so that the torso ends up in a centered position. This is modeled with a FSM which leads to a clearer code structure (see Figure 2). While the robot is walking, it can also perform small dribbling kicks on request

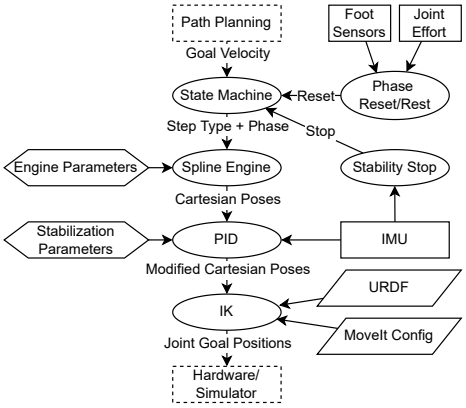

Fig. 1: Overview of the approach with used parameters, sensors and interfacing software parts.

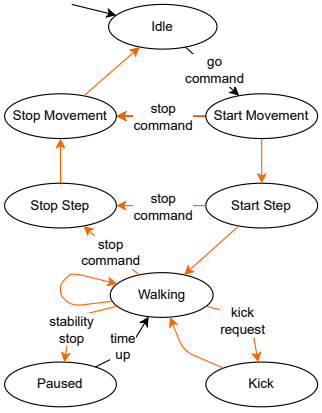

Fig. 2: Visualization of the finite state machine. Transitions are activated if the corresponding condition is met. Orange transitions can only be activated when a step is finished.

by modifying the spline to move the foot quickly to the front before setting it down. Generally, the controller may prevent further steps during double support to let oscillations of the robot settle (*Stability Stop*, see Section 3.3). A step is normally finished after a fixed time period (defined by one parameter) but the step duration can be modified by the *Phase Reset* and *Phase Rest* (see Section 3.3).

3.2 Spline Engine

The core part of the walk controller is the spline engine that generates a fixed cycle gait which is partially based on the IKWalk from team Rhoban [23]. Dependent on the current step type, two sets of six quintic splines are generated to describe the Cartesian pose (x, y, z, roll, pitch, yaw) of the torso and the moving foot in relation to the support foot over the duration of the step. The usage of quintic splines ensures that the trajectories are continuous in the first and second derivatives. This leads to smooth motions which are crucial for a stable walk.

The quintic spline is defined by a list of knots that each specifies the position, velocity and acceleration at a given time. Between these, polynomials are used for interpolation. The values of these knots can be seen as the parameters of the spline engine. Some of these are naturally defined, e.g. the foot is on the ground at the end of the step, and some can be derived from the commanded walking speed, e.g. how far the foot is put forward at the end of the step. The remaining parameters need to be optimized as they depend on kinematic and dynamic properties. A visualization of these parameters is shown in Figure 3.

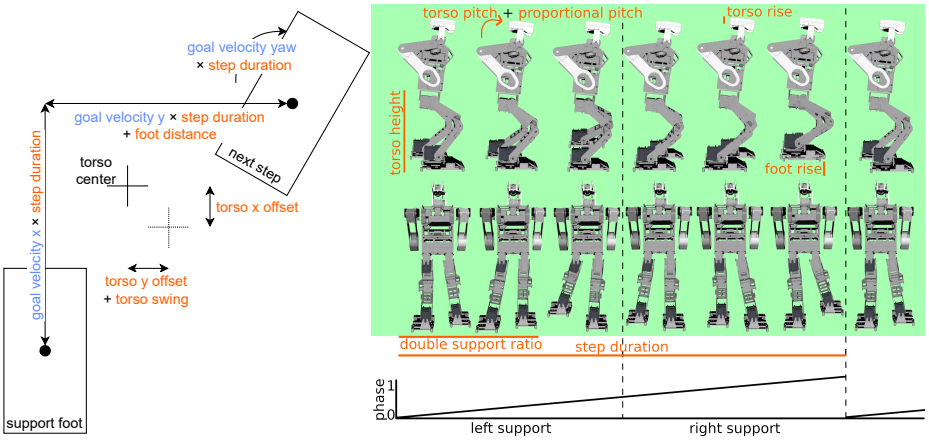

Fig. 3: Illustration on how the parameters (**orange**) and the goal walk velocity (**blue**) influence the motion. On the **left**, the positioning of the next step and the torso is shown. On the **right**, the walk cycle is shown (with exaggerated double support) and the phase variable is illustrated on the **bottom**. Additionally, a not shown parameter defines phase shift of the torso movement towards the foot movement. These are all parameters that are optimized.

3.3 Stabilization

Different methods for stabilization can be applied to the open-loop pattern that is created by the quintic splines. Two PID controllers modify the torso's orientation in relation to the ground, based on the IMU orientation. For this, we use fused angles [5] which are an orientation representation that is specifically designed for balancing. The PID controllers deal with arising oscillations and external pushes. Another source of forces that lead to instabilities is making ground contact with the moving foot, either by pushing too far downwards into the ground or by starting a new step without having yet made ground contact. To prevent this, we apply phase modulation which either ends a step early or waits for the ground contact based on sensor data. This data can either come from foot pressure sensors, if the robot has those, or from the joint torque feedback, which the widespread Dynamixel servos provide. The latter approach is less reliable due to noise in the sensor data, but is applicable without additional sensors. If the robot gets too unstable, it is also possible to do a stability stop during a double support phase, which completely stops the walk motion, allowing larger oscillations to settle. While this is undesired, as it slows the robot, it is still better than falling. The arms are not used for stabilization, since movements can lead to fouls, i.e. ball holding. On the real hardware, the arms also might get damaged during falls dependent on their position. Therefore, most teams typically keep the arms in a fixed position in the RoboCup Humanoid League.

3.4 Inverse Kinematics

Since the movement is defined in Cartesian space, we need to apply an IK to compute the joint goal positions. To keep our walk controller generally usable, we do not rely on a specific analytic IK. This would need to be adapted to new robot models, and some of the used models, i.e. the Wolfgang-OP [10] and the NUgus [14], do not have an analytic solution because the hip joint axes do not intersect. We use the widespread MoveIt IK interface which provides multiple solvers. For this work, we chose the BioIK [24] solver as it works for all non-parallel robots. It combines genetic algorithms and particle swarm optimization to allow solving of different IK goals. Still, it is fast enough to run in real-time on computationally limited robot platforms.

3.5 Interfacing

We provide multiple interfaces for the walk controller, that allow simple integration into different code bases. The standard interface uses ROS 2 based message passing to provide the motor goal positions and the odometery. Additionally, multiple debug topics are provided that show the internal state of the walk controller and can be visualized with standard ROS tools, i.e. PlotJuggler. This allows simple integration for users of this middleware. We also provide a ROS 1 version. Since the code is written in C++, it is also possible to directly call the corresponding methods without ROS 2 message transfers. Still, either ROS 2 or ROS 1 packages are necessary as dependencies for building the code. Additionally, we provide a direct Python 3 interface to the C++ methods which is also used for the described parameter optimization (see Section 4).

4 Optimization

As described above, the definition of the splines does not only rely on the goal walk velocity but also on a set of parameters. Naturally, the performance of the walk control correlates to the quality of these parameters and the optimal values differ for each robot type. It is possible to tune these by hand, but it is time-consuming and might result in a local maximum. Therefore, we apply black-box parameter optimization to automatically tune them in simulation. The different stabilization approaches also require parameters, but these are either easy to find, i.e. the thresholds for the phase modulation, or have an existing method for tuning, i.e. the Ziegler-Nichols method [30] for the PID controllers. Therefore, we do not optimize these parameters automatically and will not cover them in the following section.

4.1 Problem Definition

Generally, such an optimization problem can be expressed as finding the parameter set x^* which maximizes a single objective function $f(x)$ (Equation 1) or

multiple objective functions $f_1(x), f_2(x), ..., f_n(x)$ (Equation 2). The parameters x need to be part of the set of possible parameters X.

$$x^* = \underset{x \in X}{\operatorname{argmax}} \ f(x) \quad (1) \qquad x^* = \underset{x \in X}{\operatorname{argmax}}(f_1(x), f_2(x), ..., f_n(x)) \quad (2)$$

For each of the spline parameters, a continuous range is defined to create the set of possible parameters X. This range can be identically for all robots, e.g. the $double\_support\_ratio$, and is defined either through natural boundaries, e.g. $double\_support\_ratio \geq 0$, or our previous experience on which values make sense, e.g. $double\_support\_ratio \leq 0.5$. Some ranges are dependent on the robot type since they are heavily influenced by its size, e.g. $foot\_distance$. These need to be specified for each robot but can be found easily by trying out maximal reachable poses of the feet. It is important to note, that these ranges do not need to be exact and can be larger than necessary, but it can prolongate the optimization process if they are chosen extremely large. The user might also narrow these ranges to achieve a walking with certain desired properties, e.g. with slow steps or a certain torso height.

A natural objective function for walking is the maximum speed that the controller can achieve without the robot falling. Since our goal is an omnidirectional walk controller, there are three continuous goal velocities $(x, y, theta)$. Thus we have an infinite amount of goals and can not describe this as an objective function. Still, our tests have shown that it is enough to optimize the parameters for the four directions forward, backward, sideward, and turn. If the parameters work for these, combinations, e.g. walking to the front-left while turning right, are also working. For sideward and turn movements, only one direction has to be tested, as the parameters are symmetrical enough. This results in four objective functions $f_f(x)$, $f_b(x)$, $f_s(x)$, and $f_t(x)$.

This multi-objective optimization problem can directly be solved by Multi-objective Tree-structured Parzen Estimator (MOTPE) [20]. But it can also be scalarized a priori and then solved as a single-objective optimization problem, e.g. by using Tree-structured Parzen Estimator (TPE) [9]. When using MOTPE, an a posteriori scalarization is necessary for deciding on a parameter set, since a multi-objective optimization will provide a Pareto front, not a single solution. The following scalarization is used. Its factors are based on the typically achieved maximum walk speeds and ensure that each of the directions is contributing equally to the objective. Without these factors, the optimization might focus on the turn direction as the values are typically higher due to the different unit (rad/s instead of m/s).

$$f(x) = f_f(x) + f_b(x) + 2 * f_s(x) + 0.2 * f_t(x) \quad (3)$$

4.2 Optimization Process

The optimization process is based on the Optuna library [4] which provides implementations of different optimization algorithms (called *sampler*) and a framework to run the whole optimization process (called *study*). During the execution

of a study, several *trials* will be executed. For each, the sampler will propose a parameter set, based on the previously evaluated sets. The library user needs to implement a function that evaluates these parameters and returns their objective values to the study. Multiple trials of a study can also be executed in parallel by using an SQL database to share data between the processes, thus allowing good scalability on a computer cluster.

Basing our code on this library leads to a clear interface that is compatible with different samplers and removes the need to implement any sampler ourselves. The only part that is necessary to implement is the evaluation of a trial. To do this, we use the Webots simulator [19] since the robots models of the other teams are available for this simulator and we can use the exact same artificial grass simulation environment as defined by the HLVS. Still, our code also supports PyBullet [12] and has an interface to implement the usage of other simulators, but this is not further discussed in this paper.

For each of the directions, we do the following procedure to compute the objective value. First, the robot is initialized by performing steps in the air without gravity and then put back onto the ground. This is necessary, to ensure that the robot starts with the correct pose for the parameter set. Then, the robot walks for 10 s in the corresponding direction, including an acceleration and deceleration phase, as well as a complete stop at the end. This procedure is repeated with linearly incrementing speed until the robot either falls or the traveled distance does not increase. When all four directions are evaluated, the maximally reached velocities are returned either as an array, in case of multi-objective optimization, or scalarzied as a single value in case of single-objective optimization (see Equation 3).

5 Evaluation

To evaluate the presented walk controller, we first compare different samplers that can be used to optimize the parameters. Then we show how well it generalizes to different robot platforms and provide the achieved baseline values. Additionally, we discuss the previous impact of the approach.

5.1 Optimizer

There are multiple different optimization approaches available (see also Section 2). Optuna provides samplers for CMA-ES [17], NSGA-II [13], TPE [9], and MOTPE [20]. Additionally, for TPE and MOTPE a multivariate version (MTPE/MMOTPE) is provided. Optuna claims that it outperforms the independent TPE and MOTPE sampler. We compare the achieved objective value of these samplers with a budget of 1,000 trials. Additionally, we provide the baseline of a random search with 1,000 and 10,000 trials to highlight that these values can not be easily found randomly. The experiment was performed on three different robots, the Wolfgang-OP [10], the OP3 [2], and the Bez robot. These robots were chosen, as they have different sizes, foot shapes (with cleats

Table 1: Achieved objective values for different robot-sampler combinations (higher is better)

Name	Wolf.	OP3	Bez
Multi. MOTPE	1.33	1.81	0.24
MOTPE	1.39	1.89	0.72
Multi. TPE	1.25	1.83	0.31
TPE	1.35	1.81	0.55
CMA-ES	0.90	1.53	0.69
NSGA-II	1.39	1.78	0.63
Random 1000	0.64	1.36	0.17
Random 10000	1.09	1.48	0.36
MOTPE Comb.	1.52	1.95	0.74

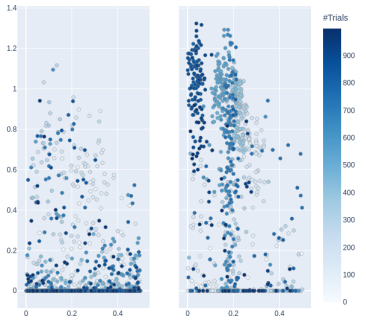

Fig. 4: Comparission between sampled *double_support_ration* parameters (x-axis) of MMOTPE (**left**) and MOTPE (**right**) with achieved objective values (y-axis).

and without), and different degrees of realism in the model. Due to the long time that is needed for the optimization process, each sampler-robot combination was only run a single time. Since the approaches are not deterministic (due to a random initialization), the results may be influenced by randomness.

Of all samplers, the independent MOTPE performs the best (see Table 1). These results are in accordance with our previous work on humanoid stand-up motions [29], where MOTPE also performed better than CMA-ES. While we only compare the samplers in the metric of trials, it is noteworthy that the necessary time for one study is also significantly influenced by the choice of the sampler. Since one parameter set is evaluated by trying to walk with increasing velocity, bad parameter sets that directly lead to a fall are evaluated quicker. For example, the multivariate MOTPE sampler needs only ca. 8 hours for 1,000 trials while the independent version needs ca. 24 hours on the same machine with an AMD Ryzen 9 5900x 12-core CPU. We observed that the independent (MO)TPE tries to improve more on the current local maximum that it has found, while the multivariate version explores the parameter space more (see Figure 4). As it tries out more bad parameter sets due to this, the multivariate version is also faster to compute in our scenario. But, it is not performing well in fine-tuning the best parameter set. Therefore, we propose to use a mixed approach of first optimizing 1,000 trials using the multivariate version and then optimizing further 500 trials with the independent version, as this leads to slightly better results with similar computation time (see *MOTPE Combination* in Table 1).

5.2 Generalization

We evaluate the presented walk controller on all robots of the Humanoid Virtual Season [1], as well as some popular humanoid robots that are included in Webots using the simulation parameters of the Humanoid Virtual Season. To do this, for each robot a URDF is needed to solve the IK. These were only available for the commercial robots. For the other robots, the URDF was created using the URDF export function of Webots based on the simulator model. For each

Table 2: Achieved walking velocities on different robot platforms

Robot Platform	Team/Company	Height [m]	Forward [m/s]	Backward [m/s]	Sideward [m/s]	Turn [rad/s]
Bez	UTRA	0.50	0.21	0.05	0.12	2.45
Chape	ITAndroids	0.53	0.30	0.36	0.10	2.09
Gankenkun	CITBrains	0.65	-	-	-	-
KAREN	MRL-HSL	0.73	0.54	0.48	0.18	3.10
NUgus	NUbots	0.90	0.38	0.42	0.26	1.97
RFC2016	01.RFC Berlin	0.65	0.37	0.40	0.36	1.99
Wolfgang-OP	Hamburg Bit-Bots	0.83	0.48	0.51	0.22	1.89
Darwin/OP2	Robotis	0.45	0.29	0.29	0.12	2.47
OP3	Robotis	0.51	0.45	0.54	0.13	2.01
NAO	Aldebaran	0.57	0.43	0.58	0.25	0.85

robot, we optimized the spline parameters using the above-described approach of doing 1,000 trials using the multivariate MOTPE and then 500 trials using the independent MOTPE. We did not use any of the stabilization methods for this comparison, since they are not necessary in a simulation without external forces.

The reached walk velocities of the best parameter set are shown in Table 2. There is only one robot in the HLVS, the Gankenkun, that does not work since it has parallel-kinematics in the legs. This is generally not supported by URDF and therefore it is not possible to solve the IK. We assume, that the robot would be able to walk if a custom IK is used. All robots without parallel kinematics were able to walk. Naturally, the larger robots, e.g. the Wolfgang-OP, reached higher velocities than the smaller robots, e.g. the Bez. Noteworthy is that the commercial robot models are less realistic, especially the OP3, which has a motor torque of 1,000 Nm in simulation. Therefore, they reach high walk velocities.

5.3 Previous Usages

The Hamburg Bit-Bots used earlier versions of the presented walk algorithm successfully on real hardware at the RoboCup championship in 2018 and 2019. Furthermore, the walk controller won the teen-size push recovery technical challenge in 2019. We made the experience that the optimized parameters are often not directly applicable to our real world robot. Mainly the torso pitch needs to be adjusted. Our assumption is that the model of the robot has an inaccurate weight distribution. Still, adapting the parameters from simulation to the real robot is simpler than optimizing them from scratch. Furthermore, in our experience, it leads to better walking, as humans tend to focus on the first local maximum when optimizing manually.

In simulation, it has been used by the Hamburg Bit-Bots to score third place in the RoboCup 2021, first place in RoboCup Brazil Open 2021, and second place in the HLVS 22. The Hamburg Bit-Bots also used it successfully on a simulated Darwin-OP robot in the running robot competition 2020 and 2021, including a

modified controller version for walking on stairs. The team NUbots is using the walk engine since 2019 [14]. The quintic spline engine has been used by Putra et al. as a basis for their walking approach [21].

6 Conclusion

We presented our open-source walk controller that is working on all non-parallel robots in the RoboCup Humanoid Virtual Season, as well as on widespread commercial robots. It reduces the entry barrier for new teams by providing a simple-to-use solution to the bipedal walk problem. The usefulness of this approach has been evaluated in the multiple real and virtual RoboCup tournaments.

In the future, we would like to test the algorithm on further robot models and are, therefore, asking more teams to make their models open-source. The optimization procedure could be improved further by automatically limiting the search space based on the kinematic restrictions of the robot.

The presented software, as well as accompanying videos and URDFs, can be found at: **https://bit-bots.github.io/quintic_walk/**

References

1. Robot Models HLVS. https://humanoid.robocup.org/hl-vs2022/teams/
2. Robotis OP3. http://emanual.robotis.com/docs/en/platform/op3/introduction/
3. Aftab, Z., Robert, T., Wieber, P.B.: Ankle, hip and stepping strategies for humanoid balance recovery with a single model predictive control scheme. In: 12th IEEE-RAS International Conference on Humanoid Robots. IEEE (2012)
4. Akiba, T., Sano, S., Yanase, T., Ohta, T., Koyama, M.: Optuna: A next-generation hyperparameter optimization framework. In: 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019)
5. Allgeuer, P., Behnke, S.: Fused angles: A representation of body orientation for balance. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE (2015)
6. Allgeuer, P., Behnke, S.: Omnidirectional bipedal walking with direct fused angle feedback mechanisms. In: IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). IEEE (2016)
7. Asada, M., von Stryk, O.: Scientific and technological challenges in robocup. Annual Review of Control, Robotics, and Autonomous Systems **3**, 441–471 (2020)
8. Behnke, S.: Online trajectory generation for omnidirectional biped walking. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE (2006)
9. Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B.: Algorithms for hyper-parameter optimization. In: Advances in neural information processing systems (2011)
10. Bestmann, M., Güldenstein, J., Vahl, F., Zhang, J.: Wolfgang-op: A robust humanoid robot platform for research and competitions. In: IEEE-RAS 20th International Conference on Humanoid Robots (Humanoids). IEEE (2021)
11. Chitta, S., Sucan, I., Cousins, S.: Moveit![ros topics]. IEEE Robotics & Automation Magazine **19**(1), 18–19 (2012)
12. Coumans, E., Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org (2016–2021)

13. Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In: International conference on parallel problem solving from nature. Springer (2000)
14. Dziura, N., et al.: The nubots team extended abstract 2022
15. Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B.: Mechatronic design of nao humanoid. In: IEEE international conference on robotics and automation. IEEE (2009)
16. Ha, I., Tamura, Y., Asama, H., Han, J., Hong, D.W.: Development of open humanoid platform darwin-op. In: SICE Annual Conference 2011. IEEE (2011)
17. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary computation **9**(2), 159–195 (2001)
18. Li, Z., Cheng, X., Peng, X.B., Abbeel, P., Levine, S., Berseth, G., Sreenath, K.: Reinforcement learning for robust parameterized locomotion control of bipedal robots. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE (2021)
19. Michel, O.: Cyberbotics ltd. webots^{TM}: professional mobile robot simulation. International Journal of Advanced Robotic Systems **1**(1), 5 (2004)
20. Ozaki, Y., Tanigaki, Y., Watanabe, S., Onishi, M.: Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In: Proceedings of the 2020 Genetic and Evolutionary Computation Conference (2020)
21. Putra, B.P., Mahardika, G.S., Faris, M., Cahyadi, A.I.: Humanoid robot pitch axis stabilization using linear quadratic regulator with fuzzy logic and capture point. arXiv preprint arXiv:2012.10867 (2020)
22. Rodriguez, D., Brandenburger, A., Behnke, S.: Combining simulations and real-robot experiments for bayesian optimization of bipedal gait stabilization. arXiv preprint arXiv:1809.05374 (2018)
23. Rouxel, Q., Passault, G., Hofer, L., N'Guyen, S., Ly, O.: Rhoban hardware and software open source contributions for robocup humanoids. In: Proceedings of 10th Workshop on Humanoid Soccer Robots, IEEE-RAS Int. Conference on Humanoid Robots, Seoul, Korea (2015)
24. Ruppel, P., Hendrich, N., Starke, S., Zhang, J.: Cost functions to specify full-body motion and multi-goal manipulation tasks. In: IEEE International Conference on Robotics and Automation (ICRA). IEEE (2018)
25. Saputra, A.A., Takeda, T., Botzheim, J., Kubota, N.: Multi-objective evolutionary algorithm for neural oscillator based robot locomotion. In: 41st Annual Conference of the IEEE Industrial Electronics Society (IECON). IEEE (2015)
26. Seekircher, A., Visser, U.: A closed-loop gait for humanoid robots combining lipm with parameter optimization. In: Robot World Cup. Springer (2016)
27. Shafii, N., Lau, N., Reis, L.P.: Learning to walk fast: Optimized hip height movement for simulated and real humanoid robots. Journal of Intelligent & Robotic Systems **80**(3-4), 555–571 (2015)
28. Silva, I.J., Perico, D.H., Costa, A.H.R., Bianchi, R.A.: Using reinforcement learning to optimize gait generation parameters of a humanoid robot. XIII Simpósio Brasileiro de Automaçao Inteligente (2017)
29. Stelter, S., Bestmann, M., Hendrich, N., Zhang, J.: Fast and reliable stand-up motions for humanoid robots using spline interpolation and parameter optimization. In: 20th International Conference on Advanced Robotics (ICAR). IEEE (2021)
30. Ziegler, J.G., Nichols, N.B., et al.: Optimum settings for automatic controllers. trans. ASME **64**(11) (1942)