Fast and Reliable Stand-Up Motions for Humanoid Robots Using Spline Interpolation and Parameter Optimization

Conference Paper · December 2021

DOI: 10.1109/ICAR53236.2021.9659325

CITATIONS
6

READS
484

4 authors:

Sebastian Stelter
Cranfield University
3 PUBLICATIONS **8** CITATIONS
SEE PROFILE

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

N. Hendrich
University of Hamburg
105 PUBLICATIONS **1,076** CITATIONS
SEE PROFILE

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

Fast and Reliable Stand-Up Motions for Humanoid Robots Using Spline Interpolation and Parameter Optimization

Sebastian Stelter[1], Marc Bestmann[1], Norman Hendrich[1] and Jianwei Zhang[1]

***Abstract*— Reliability and robustness to external influences are important characteristics for using humanoid robots outside of laboratory conditions. This paper proposes a closed-loop system to let the robots recover from falling over using stand-up motion trajectories based on quintic spline interpolation. By applying IMU-based PD controllers, faster and more reliable motions can be executed. The paper also explores the usage of parameter optimization methods, which can find suitable parameter sets quickly and are able to find solutions that are unlikely to be discovered via manual search.**

I. INTRODUCTION

In the context of autonomous humanoid robots outside of laboratories, a certain resistance to accidents and unforeseen situations is important. Due to their high center of mass, and small support polygon, humanoid robots are especially prone to falling over and thus being unable to resolve their designated task without reliable means to upright themselves. A further challenge to the stability of bipedal robots is added in the RoboCup [1] context, where robots are required to carry out soccer matches on artificial turf. Compared to a flat and solid underground, the turf is significantly more difficult to walk on and to get up from. In RoboCup humanoid soccer, a robot that cannot stand up is deemed incapable after 20 seconds. Thus fast and reliable methods for getting up are necessary. While in this situation, incapable robots are allowed to be removed from the game and reset after a penalty time, in other cases it cannot be helped by outside means, for example when carrying out tasks in dangerous environments. Furthermore, a robot that is lying on the ground is severely restricted in its movement and becomes vulnerable. It, therefore, would be useful to get out of this situation as fast as possible.

Despite this, there has not been a lot of research on the topic of reliable stand-up motions. In RoboCup soccer, the most common approach is still based on keyframe animations in joint space. Most teams instead focus on reducing falls by utilizing various tools to stabilize the robot. Yet, collisions between robots still occur regularly, often knocking at least one of the contenders over. While keyframe animations work well under perfect conditions, they are open-loop and thus cannot adapt to environmental changes. Influences such as changing actuator performance due to low batteries, hardware deformations, uneven turf, or being pushed while standing up cannot be accounted for, causing the motion to fail. Furthermore, keyframe animations require a lot of fine-tuning and good knowledge of the robot platform to work reliably. Additionally, they are difficult to transfer to other platforms, since they work in joint space. Even small changes to the same platform can require a lot of additional tuning.

To combat these problems, this paper proposes an approach based on parametrized Cartesian quintic splines utilizing an IMU to calculate the error term of a PD controller. This makes the creation of a motion more intuitive and adds stability to the motion, making it easily adaptable to different environments and ground materials. Parameter optimization is then applied to simplify finding good parameters for the proposed system.

The presented approach has been developed for and tested on the Wolfgang-OP robot platform developed by the Hamburg Bit-Bots (80 cm high, 7.5 kg, see [2] for more information). The platform has a 20-degree-of-freedom (DoF) layout, with six DoF in each leg, three in each arm, and an additional two in the head. This motor layout is fairly common in humanoid robotics, which makes it easy to transfer the proposed system to different robot models (compare Section V-A). Since the arms of the Wolfgang robot are not long enough to constantly touch the ground while standing up, the motion has to transition through some dynamically unstable states, which further stresses the need of a closed-loop solution.

The core contributions of this paper are:

- Development of a fast stand-up motion that works on multiple robot platforms
- Investigation of parameter optimization and its Sim2Real transfer

After the related work in the field is presented in Section II, our approach is explored in Section III. Section IV discusses the use of parameter optimization. In Section V the results of the experiments are presented, which are evaluated in Section VI. The paper concludes with an outlook on future work in Section VII.

II. RELATED WORK

In the past, there have mainly been three different approaches to generating stand-up motions. Besides the aforementioned keyframe-based approaches, other papers have proposed systems using either reinforcement learning or motion tracking. Apart from stand-up motions, parameter optimization has been used to generate various motion types.

This research was funded partially by the German Research Foundation (DFG) and the National Science Foundation of China (NSFC) in the project Crossmodal Learning, DFG TRR-169/NSFC 61621136008.

[1] All authors are with the Department of Informatics, Universität Hamburg, Hamburg, Germany 6stelter, bestmann, hendrich, zhang@informatik.uni-hamburg.de

A. Keyframe-based Approaches

Currently, keyframe animations seem to be the most common approaches in the industry, as they are easy to implement and work sufficiently well for a lot of problems. Keyframe animations record the exact motor positions and speeds at certain points of the motion to replay them later in an open-loop system. Suitable frameworks have been created by various researchers, such as Kuroki et al., whose system also supports Euclidean space manipulation and has been used by the Sony Corporation and Boston Dynamics [3].

Stückler et al. describe the shape of keyframe-based stand-up motions in detail and already stress the importance of keeping the center of pressure inside the support polygon [4]. This idea has further been contemplated by Hirukawa et al., who described a system that works on ten different support phases and utilizes feedback control to keep the more volatile transitions stable [5].

B. Reinforcement-Learning-based Approaches

Reinforcement learning also presents huge opportunities for generating these kinds of motions, as these approaches can create usable motions without a deeper understanding of the underlying kinematics. One of the first approaches to this topic has been published by Morimoto and Doya [6]. Their system uses a hierarchical structure with the upper level selecting the current sub-goal while the lower level determines the required joint angles to reach the goal pose of each sub-goal.

Another system has been developed by Meng et al. [7]. Their neural network also accounts for ground force and friction. By reducing the robot model to a mathematical dynamic model, the problem is greatly simplified and can be trained faster. In this approach, more reward is given to the agent when the center of pressure is closer to the center of its support polygon. This, therefore, encourages more stable motions.

Jeong et al. created a Q-learning-based approach that utilizes the bilateral symmetries of humanoid robots to make their motions faster and more robust [8].

Most recently, Chatzinikolaidis et al. developed an extension to the differential dynamic programming (DDP) algorithm, using an implicit formulation and focusing on contact dynamics [9]. Their approach was able to create complex multi-contact motions, such as dynamical standing up for a robotic leg, but has only been tested in simulation.

C. Motion-Tracking-based Approaches

Even though reinforcement-learning-based approaches generate usable results in many cases, they almost always have to be trained in simulation due to the high number of trials. As the conditions of a simulation cannot perfectly mirror the real world, these approaches tend to introduce unwanted artifacts into the motion or exploit features of the simulated environment that cannot transition to the real world. To mitigate this problem, a common approach is utilizing reference motions.

This concept has been used by Mistry et al., particularly for the sit-to-stand transition [10]. They recorded humans standing up from a chair and then applied the recording to the robot model. Peng et al. combined the intuition of reinforcement learning approaches with the reliability of motion tracking with their DeepMimic system [11]. By using the reference motions as an additional goal for the reward function instead of a hard constraint, artifacts are prevented while still keeping a versatile and dynamic approach. The system can be applied to a variety of motions, such as standing up, kickflips, or walking. Recently, Lee et. al. also proposed a model-free deep reinforcement learning model for stand-up motions in quadruped robots [12].

D. Optimization

Using parameter optimization on motions for bipedal robots has been done in various contexts. Rodriguez et al. used Bayesian optimization to improve bipedal walking algorithms [13]. Using the rational quadratic kernel, they optimized a gait pattern generator with PID stabilizing.

Rai et al. also used Bayesian optimization, but instead created a custom kernel function based on certain characteristics of gait transformations used by physiotherapists [14]. This system was tested on two lower-dimensional controllers on an ATRIAS robot, as well as on a virtual neuromuscular controller that emulates the functionalities of human muscles. Liu et al. used policy gradient descent methods to optimize a pattern generator based on the linear inverted pendulum abstraction [15]. This approach is conducted online, which allows for real-time correction of errors or disturbances. Tassa et. al. developed an online trajectory optimization algorithm, that generates optimal trajectories using iterative linear quadratic Gaussians, and can solve many problems in real-time [16]. Finally, Silva et al. used temporal difference learning to optimize the default pattern generator of a Darwin-OP robot [17].

III. APPROACH

In comparison to known approaches, the proposed system differs mainly in the use of quintic splines and the modeling of motions in Cartesian coordinates. By using quintic splines, the resulting trajectories are continuous in the first two derivatives and thus guarantee a smooth transition of both position and velocity in the Cartesian space. The modular software of our system is based on the ROS-architecture (Robot Operating System) [18]. The program flow is shown in Figure 1. In a first step, splines are generated for the poses of the four end-effectors (hands and feet), so that their interpolated values can be calculated quickly for any requested point in time. This information is forwarded to the stabilizer component, which applies PD controllers to the robot pitch and roll offset, correcting for errors in the motion. The orientation readings of the torso IMU are used as the input term. Finally, the inverse kinematics are calculated and returned as motor goals, which are then sent out by the stand-up motion generator node. In the following sections, the approach is described in detail.

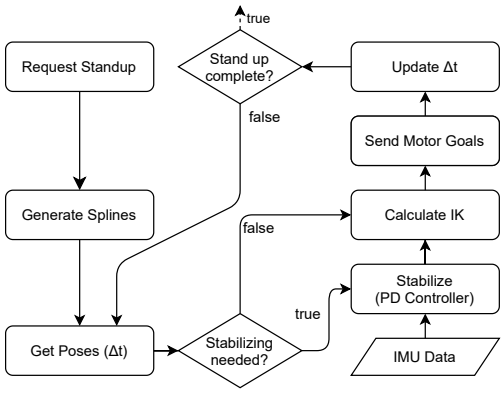


Fig. 1: Program flow of a stand-up request. Δt is the time passed since the request.

A. Spline generation

In the first step of creating the stand-up motion, the movements of the four end-effectors are defined by six splines each. The splines are generated relative to the robot's base frame which is located in the torso between the first hip joints and describe the movement in x, y, z, ϕ, θ and ψ. Each polynomial of a quintic spline can be defined as

$$\begin{aligned} o_i(t) =& c_0 + c_1(t_{i+1} - t_i) + c_2(t_{i+1} - t_i)^2 + \\ & c_3(t_{i+1} - t_i)^3 + c_4(t_{i+1} - t_i)^4 + c_5(t_{i+1} - t_i)^5 \end{aligned} \tag{1}$$

with

$$\begin{cases} c_0 = p_i \\ c_1 = v_i \\ c_2 = \frac{1}{2} a_i \\ c_3 = \frac{1}{2T_i^3}[20d_i - (8v_{i+1} + 12v_i)T_i - (3a_i - a_{i+1})T_i^2] \\ c_4 = \frac{1}{2T_i^4}[-30d_i + (14v_{i+1} + 16v_i)T_i + (3a_i - 2a_{i+1})T_i^2] \\ c_5 = \frac{1}{2T_i^5}[12d_i - 6(v_{i+1} + v_i)T_i + (a_{i+1} - a_i)T_i^2] \end{cases} \tag{2}$$

where $d_i = p_{i+1} - p_i$ describes the displacement and $T_i = t_{i+1} - t_i$ describes the duration. p_i and p_{i+1} describe the positions, v_i and v_{i+1} the velocities and a_i and a_{i+1} the accelerations at t_i and t_{i+1} respectively [20].

The initial position vector of each spline p_0 is the current pose of the corresponding end-effector, which makes sure that the robot moves to the starting position of the motion in a controlled manner. The further positions $p_1, ..., p_n$ are either given by the kinematic structure of the robot (e.g. the length of the arms) or influenced by free parameters that need to be optimized (see Section IV). All durations $T_0, ..., T_n$ are free parameters. The velocities $v_0, ..., v_n$ and accelerations $a_0, ..., a_n$ at each spline point are set to 0. The general shape of the splines is modeled after the keyframe animations our team developed for the open-loop stand up approach used prior to this system.

The resulting motions can be seen in Figure 2 and 3. If the robot has fallen to the front, it first moves its arms from their current position, usually the pre-generated falling pose (Figure 2.a) to the front. Simultaneously, the legs are pulled towards the body (Figure 2.b - 2.d). Next, the robot pushes up on its arms and tilts its torso back to push itself onto its feet (Figure 2.e - 2.h). Finally, the robot slowly rises into a stable upright position.

The backward motion works similarly. The robot starts lying on its back and pushes its arms back into the ground to lift itself up, while simultaneously pulling its legs under its body (Figure 3.b). From this position, it can then push itself entirely onto its feet (Figure 3.c - 3.e). The rest of the motion is identical to the front stand-up procedure.

The unlikely case of the robot falling and landing on its side is resolved by rotating the leg the robot is lying on so that it rolls over to its back and can start the procedure as described above. Since the final part of the motion is identical in both directions, it is implemented separately. This also allows us to use it as a sit-down and get-up motion to ensure the safe shutdown of the robot.

B. Stabilizer

Standing up creates high dynamic forces on our bipedal robot. Especially when the robot pushes onto its feet, it transitions through several poses where the center of pressure leaves the support polygon and the robot would not be able to stand statically. This means that the robot's push has to be strong enough to reach the next stable position without falling backward, but not too strong, as this would cause the robot to fall in the other direction. To increase this stable area, and to account for uneven turf or other robots pushing, we use a stabilizer. Since all four end-effectors touch the ground during the first part of the motion, the robot is very stable and virtually cannot fall over. Due to this, the stabilizing is disabled during that part of the motion.

The stabilizing works by using two PD controllers that both use the Fused angles [21] representation of Cartesian space. As the current position of the right leg is known from the joint angle encoders, the position of the torso relative to the right foot can be calculated easily using forward kinematics. The target orientation of the torso can now be compared to the IMU-measured orientation, and the difference is forwarded to two separate PD controllers, one controlling the trunk's fused pitch, the other the trunk's fused roll. Fused angles represent a robot's orientation towards the three Cartesian planes. They are useful in this application, as they do not encounter singularities when the robot is rotated by 90 degrees (in contrast to Euler angles) but still clearly decompose the rotation into different axes (in contrast to quaternions). The corrected orientation is then transformed back to determine where the foot needs to be placed to reach the desired trunk orientation. During the periods where stabilizing is not required, the poses are passed through unchanged.

C. Inverse Kinematics

Finally, the corrected poses are sent to the IK solver, which computes the necessary joint commands. To do so, the MoveIt [22] IK interface is used, which, depending on

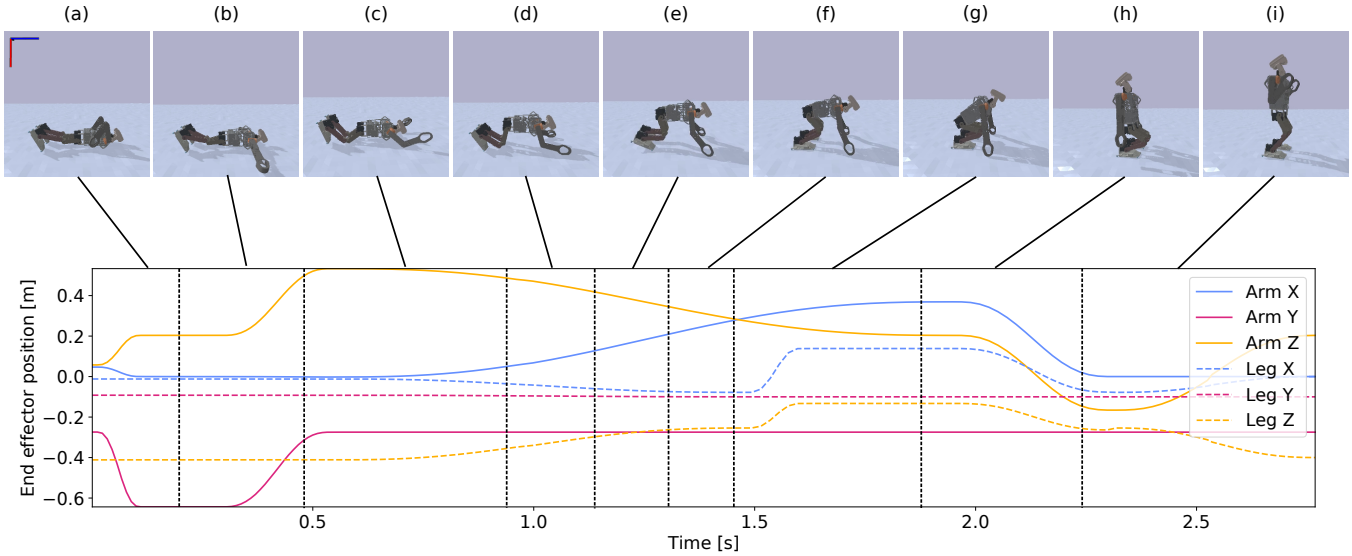


Fig. 2: "Front" stand-up motion using the proposed dynamic stand-up approach with optimized parameters, mapped to the end-effector poses as described by the generated splines, prior to stabilizing. The coordinates are plotted for the right arm and foot, measured relative to the robot's torso. The initial orientation of the robot coordinate system is indicated in 2.a. Corresponding curves for the left arm and foot follow from symmetry considerations. Recorded in the PyBullet simulator [19].

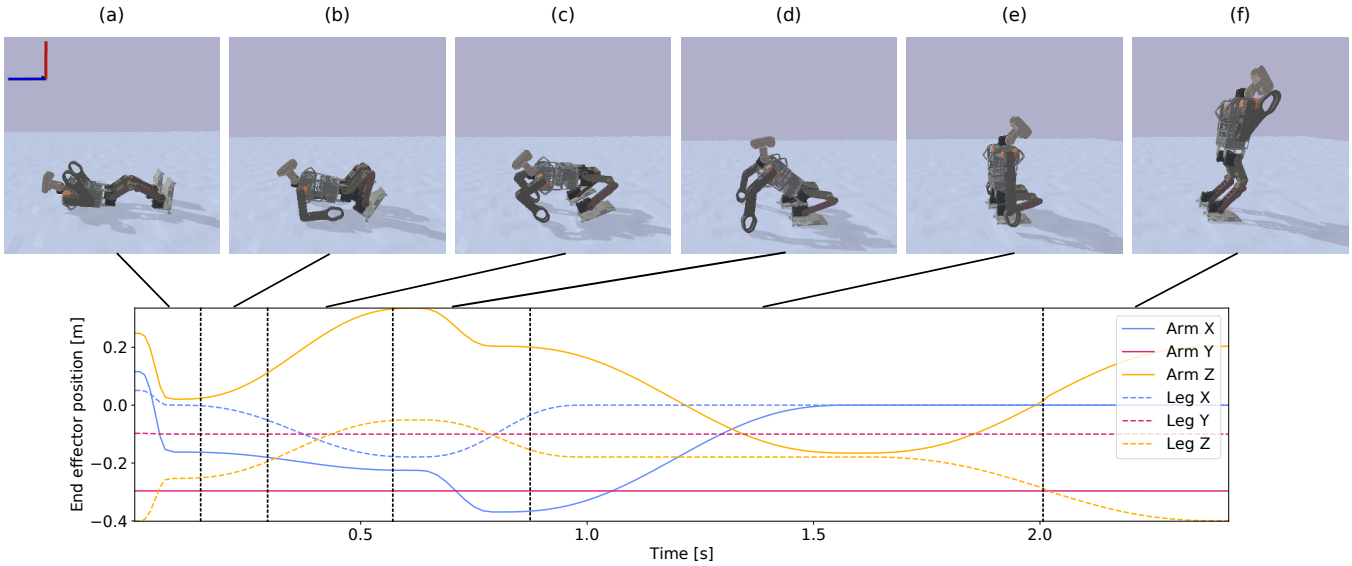


Fig. 3: "Back" stand-up motion using the proposed dynamic stand-up approach, same as Figure 2.

the robot model, provides different kinematic solvers. A problem with the 20 DoF robot model as described above is the low DoF of the arms. Compared to a human, this robot model is missing a shoulder yaw and thus cannot reach all possible poses in its range. Therefore, we use the BioIK [23] solver, which combines genetic algorithms and particle swarm optimization, and therefore allows for the robot's arms to approximately reach the desired pose, even when it would not be possible to reach it exactly. This approach might produce erroneous solutions, if the target positions are too far off of the possible movements. However, we did not encounter any problems related to this, as the divergences are small enough.

IV. PARAMETER OPTIMIZATION

As described in Section (III), the definition of the splines leaves multiple free parameters. Therefore, it is necessary to find a suitable values for these to ensure getting up stably and fast. Standing up from the front has 9 free parameters describing the length of the different phases (see also Figure 2) and 6 free parameters describing Cartesian poses of the end-effectors. These parameters define the lateral and forward offset of the arm when pushing up, the distance between foot and torso at the shortest point, as well as the

angle of the legs and the offset and overshoot of the torso. For standing up from the back, there are 6 free parameters for the phases (see also Figure 3) and 9 for the end-effector poses. The back motion has equivalents for all pose parameters, except for the trunk offset. Instead, it also defines the angle of the arms, the trunk height when pushing up, as well as the shift of the center of mass during two distinct phases of the motion. Additionally, four parameters describe the goal pose of the robot after the stand-up motion, which can be chosen freely by the user and do not need to be optimized.

Manually finding good parameters is possible and eased by the fact that they have a clear semantic relation to the robot's movement. Still, this is a laborious task that requires some insight into the approach by the user. Therefore, we apply an automatic parameter optimization. In the past, different algorithms were proposed to tackle this issue. Mathematically, this problem can be expressed as finding the parameter set x^* which minimizes an objective function $f(x)$. The parameters x need to be part of the set of possible parameters X.

$$x^* = \underset{x \in X}{\operatorname{argmin}} \; f(x) \tag{3}$$

Commonly used examples for this are the covariance matrix adaptation evolution strategy (CMAES) [24] and the Tree-structured Parzen Estimator (TPE) [25].

In our case, we have two different optimization objectives: stability and speed. It is possible to formulate both as a single objective problem by using scalarization, e.g. taking the weighted sum of both values, but this can lead to issues if the objectives are scaled in different units. Another approach is to use a multi-objective optimization that can be defined by using multiple objective functions.

$$x^* = \underset{x \in X}{\operatorname{argmin}}(f_1(x), f_2(x), ..., f_n(x)) \tag{4}$$

Solving such a multi-objective problem can be done by Multi-Objective Tree-structured Parzen Estimator (MOTPE) [26]. We used the implementations provided by the Optuna library [27].

V. EXPERIMENTS

To evaluate the performance of our approach, different experiments were conducted in simulation, as well as on the real robot.

A. Simulation

To reduce the number of trials on the actual robot, optimization was conducted in a simulation using PyBullet [19]. The used URDF model of the robot was created by using the CAD files for the links and estimations from the servo datasheets for the joints. No specific model identification was used. To make the simulation more similar to the artificial grass, the ground is a randomly generated height-map of up to 1cm. Early termination was done if the IK does not find a valid solution or if the robot falls back to the ground. This speeds up the optimization process, as no time was wasted on further simulating unsuccessful parameters.

Each parameter configuration was simulated three times to account for uncertainties due to the uneven ground and small errors in the IK solution. The mean objective value of these trials was then taken as the final objective value. Different objective functions were investigated which are based on the success rate s, the total time t, the minimal reached fused pitch p and the maximal reached head height h. In the case of using single-objective algorithms, these were summed up to form one single function. Since all of them have different units, the values were scaled between [0,100].

$$s = \begin{cases} 0 & \textit{if successful} \\ 100 & \textit{if not successful} \end{cases} \tag{5}$$

$$t = 10 * \textit{time\_until\_standing} \; [s] \tag{6}$$

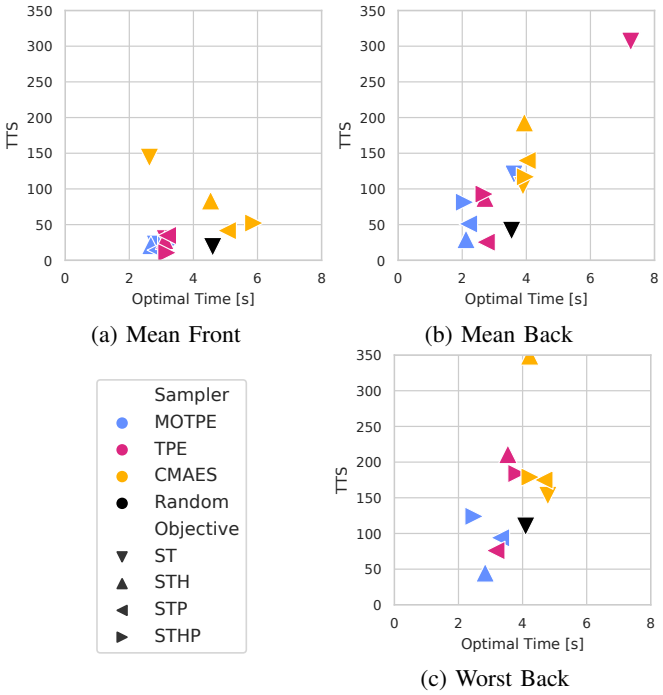
$$p = \min(\textit{fused\_pitch} \; [deg]) \tag{7}$$

$$h = 100 - \max(\textit{head\_height} \; [cm]) \tag{8}$$

Time t includes all the time the robot shook after getting up (clipped after 10s). This was necessary to prevent learning parameters that quickly reach a standing pose but remain unstable for a long time afterward. The values p and h are not directly related to our objectives of time and stability but were added in an attempt to guide the algorithm in the right direction as s is not a smooth function. In the following, we used combinations of these letters to describe different objective functions, e.g. ST means using the success and time objective and in cases of single-objective approaches, the sum of them.

The parameters for the stabilizing PD controller were not optimized with the other parameters to prevent compensating bad trajectories with stabilization and to reduce the problem dimensionality. They can be optimized in a second optimization step where the trajectory parameters are fixed to the previously found ones, however, for the following experiments the PD gains have been calculated manually using the Ziegler-Nichols method [28].

Figure 4 shows a comparison between different objective functions and optimization algorithms for standing up from the front and back in simulation. In this study, only the open-loop parameters were searched, without PD parameters. All combinations were run 5 times. Each optimization process was run for 500 trials, a number that is based on the observation that there was no significant improvement after this. Each trial consisted of three repetitions of standing up since it was not deterministic due to noise in the IK solution and the structure of the floor. The score is based on the mean value of these repetitions. For each combination of scoring and algorithm, two measurements of quality are given. Firstly, the number of trials until at least one standing-up procedure of the three repetitions was successful (**TTS**) shows how quickly this combination can find a roughly working stand-up motion. Secondly, the minimal time that was achieved while being able to stand up in all three repetitions shows how optimal the found solution is. See also Figure 5 for an exemplary optimization history.



(a) Mean Front (b) Mean Back

(c) Worst Back

Fig. 4: Results from optimizing the spline parameters five times with different samplers and objectives each. Subfigure **a** shows the mean value of the optimization results with respect to the number of trials to success (TTS, at least one successful stand-up motion in three attempts) and the optimal time (all three attempts successful). Subfigure **b** shows the same data for standing up from the back. Subfigure **c** shows the worst result while optimizing the backward motion. Here the marker for the ST objective is missing for TPE and MOTPE as they did not find a solution.

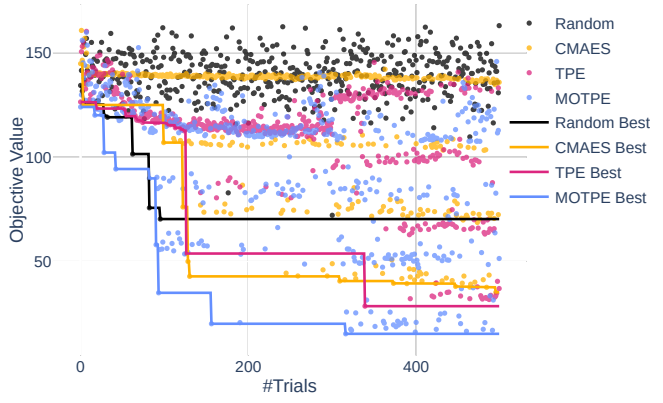


Fig. 5: Exemplary optimization history for the four used algorithms for standing up from the back. Each point in the scatter plot shows one trial result, while the lines show the best value. As an objective value ST was plotted, while STH was used during optimization. This shows how the actual objectives were improved when giving an additional guidance objective (compare Figure 5.b).

TABLE I: Results on Different Platforms using MOTPE and STH Objective. Note that the Darwin robot could not execute our stand-up-from-the-back motion due to kinematic limitation.

	front		back	
	Time [s]	TTS	Time [s]	TTS
Wolfgang	2.7	21.0	2.1	29.4
Darwin	2.9	35.8	n/a	n/a
Sigmaban	2.2	34.2	2.2	62.4

As an additional experiment, the optimization was performed using MOTPE and the STH objective on two other similar robot models, to investigate how well the defined splines generalize (see Table I). The Darwin-OP (45.5 cm high, 2.8 kg [29]) and the Sigmaban [30] (64 cm high, 5.5 kg [31]) robots were chosen, since they have a similar layout of the DoF, yet very different sizes and masses.

B. Real Robot

Three different types of stand-up motions were evaluated on the real Wolfgang robot. The previously used open-loop keyframe approach was taken as a baseline. The quintic-spline-based approach was then evaluated with manually tuned parameters as well as parameters optimized in simulation. For each type, both front and backward direction were evaluated 30 times. All experiments were performed on the same artificial turf with a blade length of approximately 3 cm. Similarly to the time measurement during optimization, the time was measured until the robot's angular velocity, as measured by the IMU, was below 0.15 rad/s. The results are displayed in Figure 6.

Since the optimization does not always produce parameters that are usable on the real robot, the optimization was run multiple times for both directions. Using MOTPE, an optimization was done 3 times for each objective function. The best set of each run was chosen by considering first success rate and then speed. These 12 sets were each tested once on the robot, and the best one was manually chosen. Out of these 12 sets for the front motion, 2 could be transferred to the robot directly, 5 needed minor tweaking, and 5 could not be transferred to the real robot. The failures mainly stem from a flawed assumption about the used motors, which lead to the simulated actuators being able to move faster than the actual robot could. Thus, the minor tweaking involved either slowing down individual timing parameters or shifting the center of mass. For the back motion, it was 0, 6, and 6 respectively. Since no directly working parameter set was found for the back motion, a set that only required the increase of one time parameter was used for the comparison.

The significance of the PD controllers can be seen in Figure 7 when comparing the targets of the dynamic approach with the keyframe-based approach. As the motion approached the dynamically unstable part and error was introduced at approximately 2 seconds, the dynamic approach sent modified goals to correct the orientation of the robot. The keyframe-based approach could not react to these

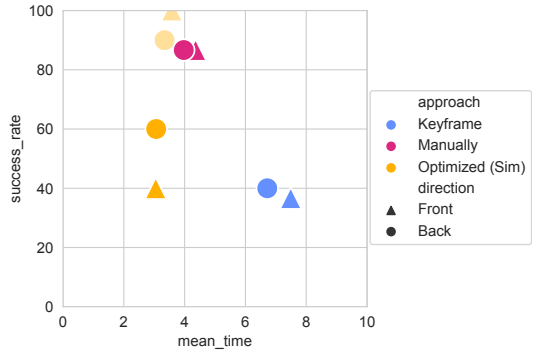


Fig. 6: Mean time and success rate for standing up from both directions with the keyframe animations, as well as manually and automatically tuned parameters. Additionally, the values for the optimized parameterset in simulation are plotted transparently to show the sim-to-real gap. While the optimized parameters perform the quickest, the manual ones are more stable. Still, both are superior to the previously used keyframe animation.

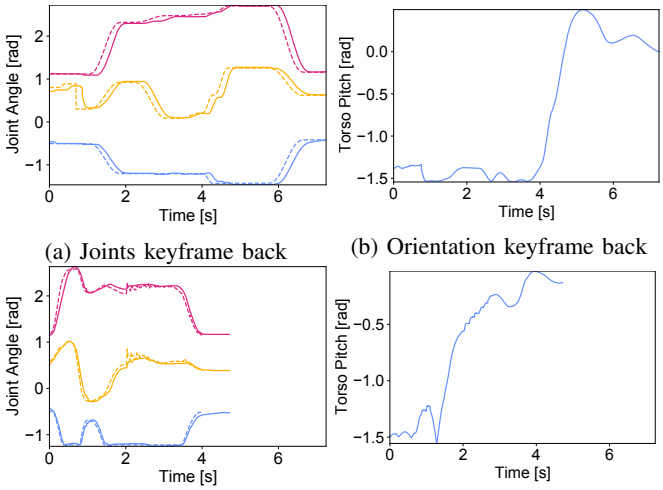
changes and thus had to conduct the motion significantly slower to have a chance to succeed.

VI. DISCUSSION

Our results show that the use of PD controllers allows for significantly faster motions with higher stability (compare Figure 6). While the manual search based on intuition and trial-and-error eventually leads to near perfect results, it is also a very tedious process. We demonstrated that working parameter sets can be generated quickly with parameter optimization approaches, and can be transferred from simulation to the real world. In comparison to motions generated with reinforcement learning, our approach is based on parameters with intuitive meaning and results can be modified after training by simply tweaking parameters manually. In doing so, the transfer from simulation to reality is significantly eased.

Even though the optimized parameters outperform the keyframe animation in regards to speed, the success rate only improved slightly. Furthermore, it was necessary to try multiple optimization results to get a working parameter set. While this is not optimal, it is still vastly less laborious than tuning the parameter completely manually. This could be improved by bridging the reality gap further. One of the main problems encountered in our parameter sets was the robot using self collisions to exploit the conditions of the simulation in a way that does not transition to the real world. Filtering out any attempts that generate self-collisions would combat this issue.

Another difference to the real world was the ground friction. Several of the tested parameter sets did not succeed in the real world, as the robots arms stuck to the turf and thus moved too slow. Changing the properties of the turf in the simulation or setting a minimum value for each timing parameter could stop the optimizer from generating unrealistically fast motions. Doing the latter could also help



(a) Joints keyframe back (b) Orientation keyframe back

(c) Joints manual back (d) Orientation manual back

Fig. 7: Leg joint angles for back stand-up motion using keyframes (**a**) and dynamic standup (**c**) together with the corresponding torso pitch (**b**, **d**). The expected goals are displayed dotted, current positions with plain lines. Blue represents the ankle pitch, red the knee, and yellow the hip pitch joints. The error correction of the PD controller can be seen at around two seconds in Subfigure **c**.

to avert another problem we encountered which stems from the different torques and speeds of the motors between robot model and real robot. This sometimes allowed the optimizer to generate faster motions than the robot could carry out.

Regarding the optimization approaches, CMAES performed inferiorly to TPE and MOTPE (compare Figure 4 and 5). While MOTPE and TPE were similarly fast in finding a successful solution, MOTPE managed to achieve better times. The difference between the used objective functions becomes clearer when optimizing the back motion. As it seems to be harder to find working parameters for it than for the front motion, neither TPE nor MOTPE did always find a solution when only using the success and time objective, probably since the success objective is binary. In combination with other objectives, this issue does not occur. The head height seems to be the best secondary objective to quickly achieve a working stand-up motion.

The versatility of the proposed system has been shown by applying it to three different robot platforms. Of the six tested motions, five succeeded (see Table I). Only the back stand-up motion for the Darwin robot could not be generated. This is most likely because its hip movement is severely restricted, which makes it impossible to move the legs into the position required by our system. The generated front motion, however, reached speeds comparable to the ones mentioned in the Darwin-OP user manual[1].

VII. CONCLUSION

This paper demonstrates that stabilized Cartesian spline movements can be superior in stability and speed compared

[1]https://emanual.robotis.com/docs/en/platform/op/getting_started/

to commonly used open-loop joint space keyframe animations. By optimizing parameters in simulation, the manual effort can be reduced and faster solutions can be found. The proposed system can cope with the difficult environment of artificial turf, which is significantly more difficult to walk on compared to flat surfaces. As demonstrated by our simulations with the Darwin and Sigmaban platforms, our approach can easily be applied to other robot models, due to the abstraction in Cartesian space. Being amongst the first researchers that conducted a large-scale stand-up experiment on a real-world humanoid robot on artificial grass, in this paper we also provide a baseline for stand-up motions that new approaches can be measured against.

Future work will consider optimizing the PD parameters, as well as running longer optimization trials to generate even more reliable results. Furthermore, optimizing parameters directly on the robot could bridge the gap between simulation and the real world and therefore improve the results. Pruning unrealistic trials and preventing impossible motions could also improve transferability. Another point of interest is optimizing with lower torque in the simulation, to find solutions that stress the joints less.

The project source code is available at

github.com/bit-bots/bitbots_motion/tree/master/bitbots_dynup

REFERENCES

[1] Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, and Eiichi Osawa. Robocup: The robot world cup initiative. In *Proceedings of the first international conference on Autonomous agents*, 1997.

[2] Marc Bestmann, Jasper Güldenstein, Florian Vahl, Jianwei Zhang, Marc Bestmann, Timon Engelke, Niklas Fiedler, Jasper Güldenstein, Jan Gutsche, Jonas Hagge, et al. Wolfgang-op: A robust humanoid robot platform for research and competitions. *IEEE Humanoids 2021*, 2021.

[3] Yoshihiro Kuroki, Bill Blank, Tatsuo Mikami, Patrick Mayeux, Atsushi Miyamoto, Robert Playter, Ken'ichiro Nagasaka, Marc Raibert, Masakuni Nagano, and Jin'ichi Yamaguchi. Motion creating system for a small biped entertainment robot. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[4] Jörg Stückler, Johannes Schwenk, and Sven Behnke. Getting Back on Two Feet: Reliable Standing-up Routines for a Humanoid Robot. In *IAS*, 2006.

[5] Hirohisa Hirukawa, Shuuji Kajita, Fumio Kanehiro, Kenji Kaneko, and Takakatsu Isozumi. The human-size humanoid robot that can walk, lie down and get up. *The International Journal of Robotics Research*, 24(9), 2005.

[6] Jun Morimoto and Kenji Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36(1), 2001.

[7] Libo Meng, Marco Ceccarelli, Zhangguo Yu, Xuechao Chen, and Qiang Huang. Gait Transition Between Standing and Falling Down for a Humanoid Robot. In *IFToMM World Congress on Mechanism and Machine Science*. Springer, 2019.

[8] Heejin Jeong and Daniel D Lee. Efficient learning of stand-up motion for humanoid robots with bilateral symmetry. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016.

[9] Iordanis Chatzinikolaidis and Zhibin Li. Trajectory optimization for contact-rich motions using implicit differential dynamic programming. *arXiv preprint arXiv:2101.08246*, 2021.

[10] Michael Mistry, Akihiko Murai, Katsu Yamane, and Jessica Hodgins. Sit-to-stand task on a humanoid robot from human demonstration. In *10th IEEE-RAS International Conference on Humanoid Robots*, 2010.

[11] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4), 2018.

[12] Joonho Lee, Jemin Hwangbo, and Marco Hutter. Robust recovery controller for a quadrupedal robot using deep reinforcement learning. *arXiv preprint arXiv:1901.07517*, 2019.

[13] Diego Rodriguez, André Brandenburger, and Sven Behnke. Combining simulations and real-robot experiments for Bayesian optimization of bipedal gait stabilization. In *Robot World Cup*. Springer, 2018.

[14] Akshara Rai, Rika Antonova, Seungmoon Song, William Martin, Hartmut Geyer, and Christopher Atkeson. Bayesian optimization using domain knowledge on the ATRIAS biped. In *2018 IEEE International Conference on Robotics and Automation*, 2018.

[15] Chengju Liu, Jing Ning, and Qijun Chen. Dynamic walking control of humanoid robots combining linear inverted pendulum mode with parameter optimization. *International Journal of Advanced Robotic Systems*, 15(1), 2018.

[16] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[17] Isaac J Silva, Danilo H Perico, A Helena Reali Costa, and Reinaldo AC Bianchi. Using reinforcement learning to optimize gait generation parameters of a humanoid robot. *XIII Simpósio Brasileiro de Automaçao Inteligente*, 2017.

[18] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, 2009.

[19] Erwin Coumans and Yunfei Bai. PyBullet, a Python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[20] Luigi Biagiotti and Claudio Melchiorri. *Trajectory planning for automatic machines and robots*. Springer Science & Business Media, 2008.

[21] Philipp Allgeuer and Sven Behnke. Fused Angles: A representation of body orientation for balance. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2015.

[22] David Coleman, Ioan Sucan, Sachin Chitta, and Nikolaus Correll. Reducing the barrier to entry of complex robotic software: a Moveit! case study. *arXiv preprint arXiv:1404.3785*, 2014.

[23] Philipp Ruppel, Norman Hendrich, Sebastian Starke, and Jianwei Zhang. Cost functions to specify full-body motion and multi-goal manipulation tasks. In *2018 IEEE International Conference on Robotics and Automation*, 2018.

[24] Nikolaus Hansen and Andreas Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of IEEE international conference on evolutionary computation*, 1996.

[25] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in neural information processing systems*, 2011.

[26] Yoshihiko Ozaki, Yuki Tanigaki, Shuhei Watanabe, and Masaki Onishi. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, 2020.

[27] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, 2019.

[28] John G Ziegler, Nathaniel B Nichols, et al. Optimum settings for automatic controllers. *Trans. ASME*, 64(11), 1942.

[29] Inyong Ha, Yusuke Tamura, Hajime Asama, Jeakweon Han, and Dennis W Hong. Development of open humanoid platform DARwIn-OP. In *SICE Annual Conference 2011*. IEEE, 2011.

[30] Loic Gondry, Ludovic Hofer, Patxi Laborde-Zubieta, Olivier Ly, Lucie Mathé, Grégoire Passault, Antoine Pirrone, and Antun Skuric. Rhoban Football Club: RoboCup Humanoid KidSize 2019 Champion Team Paper. In *Robot World Cup*. Springer, 2019.

[31] L. Hofer P. Laborde-Zubieta O. Ly S. N'Guyen G. Passault A. Pirrone Q. Rouxel J. Allali, L. Gondry. *Rhoban Football Club – Robot Specification*, 2019.