20th RoboCup International Symposium, Leipzig, Germany, July 2016.

A Closed-Loop Gait for Humanoid Robots Combining LIPM with Parameter Optimization

Andreas Seekircher and Ubbo Visser

University of Miami, Department of Computer Science,
1365 Memorial Drive, Coral Gables, FL, 33146, USA
E-Mail: {aseek|visser}@cs.miami.edu

Abstract. Even with the recent advances in the area of dynamic walking on humanoid robots there is still a significant amount of manual calibration required in practice due to the variances in the hardware. That is in order to achieve the performance needed in environments such as RoboCup. We present a LIPM-based closed-loop walk, that adapts to differences in the physical behavior of the robot by optimizing parameters of the model directly on the NAO while walking and executing other tasks. A significant amount of errors in the model predictions can be reduced without using a more complex model simply by adjusting the LIPM to fit the observed behavior. Our experiments show that the optimized model yields a more controlled, faster and even more energy-efficient walk on different NAO robots and on various surfaces without additional manual parameter tuning.

1 Introduction

A stable walk is one of the most important skills of a humanoid robot for most tasks. In an environment such as RoboCup, being able to walk to the ball and position for a kick is an essential skill that decides whether a team is successful. The walk has to be fast, but also stable and accurate. On physical robots, this often requires manual calibration and parameter tuning.

In this paper, we present a new walking engine for creating a balancing, dynamic walk for the NAO with the focus on optimizing parameters of the used model from observations gathered while the robot is walking. The goal is a walk that does not require manual fine-tuning, exact calibration or a long training to be stable in different simulators or on different NAOs or surfaces (Fig. 1).

The paper is organized as follows: we discuss relevant work in the next section and describe the generation of the walk motion in Section 3. Section 4 describes the optimization of the walk model. Our experiments and results are explained in Section 5, followed by the conclusion in Section 6.

2 Related Work

Many approaches for dynamic walking are based on simplified physical models, such as the inverted pendulum [11] or extensions of it, e.g. [14]. These models can also be used to define constraints on CoM/ZMP trajectories for a stable walk [10] or for preview control or model predictive control for walking [26, 23].

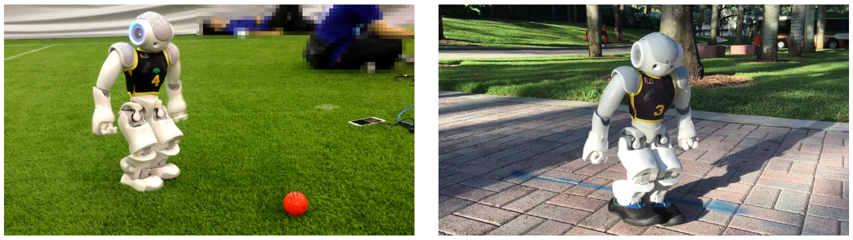


Fig. 1: The NAO walking on artificial grass and outside with shoes without extra calibration.

In [9] a ZMP-based approach is use for walking on slopes or stairs. Additionally, there are many approaches for balance controllers, e.g. for lateral disturbance rejection [16], balancing through foot placement adjustment [25], ankle-,hip- and step-strategies [1, 28] or balancing based on contact forces [20]. These approaches can produce very stable, dynamic walking motions and balancing when there are external disturbances. However, the models are never perfect and variances in physical robots could add a bias that reduces the stability of the walk. Therefore, in practice a lot of time might be needed to calibrate individual robots.

Other approaches for walking use methods such as evolutionary strategies, genetic algorithms or parameter optimization to generate or improve walking motions. These approaches can fine-tune a motion to a specific robot and do not depend on an accurate calibration. A common approach is to create a parameterized walk motion (e.g. feet trajectories) and use parameter optimization or evolutionary strategies to find good parameters [3, 27, 21]. Approaches for walking based on optimization are often evaluated using only simulations. For example, [12] optimize parameters for walking in Webots using Gaussian PSO, policy gradient RL and evolutionary hill climbing and compare the performance of the different methods. [15] use CMA-ES [7] to optimize walk parameters for the RoboCup 3D Simulation League. Nevertheless, there are promising result for similar approaches on physical robots [2, 8]. In [4], the forward walking speed of a physical humanoid robot is improved significantly by optimizing parameters of an open-loop walk (e.g. step frequency, swing amplitude) and parameters for a feedback controller using Policy Gradient Reinforcement Learning (PGRL) and Particle Swarm Optimization (PSO). [19] optimize a gait on a physical humanoid robot using PSO. Other approaches optimize the parameters of central pattern generators that generate the walk motion [29, 13, 5, 24, 22]. Many approaches for gait generation using optimization show promising results, but they usually require a long training and can be tedious to execute on physical robots. Also, a human operator might be needed to prevent damages from evaluating parameters that create an unstable motions. A method that needs 1000 fitness functions evaluations [4] or a 3 hour training [19] is too time-consuming if it has to be done for 6 robots within a limited time.

Ideally, the robot should not require an explicit walk training, but improve the walk while it is performing other tasks and it should not evaluate parameters directly that might cause it to fall. Therefore, we use a model-based approach (using LIPM, a linear inverted pendulum) to generate a walk that we combine with an offline optimization that can constantly run in the background and provide improved parameters based on current measurements.

3 LIPM-based Gait Generation

This section describes the implementation of a walking engine for a dynamically generated walk on the NAO. It uses a linear inverted pendulum as a model for the motion of the center of mass of the robot. A center of mass reference trajectory is generated for a requested walk speed and the steps are created according to the model. We estimate the robots state using sensor measurements to react to errors and disturbances.

3.1 The Inverted Pendulum Model and Walk State Representation

We use the linear inverted pendulum model as in [11]. It assumes that the center of mass stays at a constant height h. The position and velocity of the center of mass in the plane at height h can be calculated by

$$x(t) = x_0 * cosh(k * t) + \dot{x}_0 * \frac{1}{k} * sinh(k * t) \tag{1}$$

$$\dot{x}(t) = x_0 * k * sinh(k * t) + \dot{x}_0 * cosh(k * t) \tag{2}$$

with $k = \sqrt{\frac{g}{h}}$, where g is the gravitation and $x_0, \dot{x}_0 \in \mathbb{R}^2$ are the position and velocity of the mass at $t = 0$ (see [6]).

The state of the robot is reduced to the foot positions relative to the center of mass and the mass velocity. Given the position of the supporting foot and assuming a center of pressure in the center of the support polygon, we can use the model to predict the position and velocity of the mass. The swing leg is independently controlled and moved along an interpolated trajectory towards a step target position. However, the movement of the swing leg changes the mass position which has to be taken into account in the movement of the support foot. Using this representation, we can predict the outcome of full steps given the step duration and step targets (assuming the swing leg reaches the target position). We try to keep the ZMP always very close to the center of the support polygon. Thus, the mass trajectory for the current step can not be changed much. Most errors are compensated by adjusting the step duration and the target of the swing leg, which affects how the mass continues to move after the support exchange.

In contrast to some other implementations, we did not derive equations for step positioning or balancing directly from the equations of the pendulum model. Instead, the model is used as a black box, which is used to predict the successor state for a given walk state and selected a step values. We use the model in several components to approximate the correct values for walk actions numerically using the model predictions. This is not the most efficient implementation, but it allows us to experiment with the model without having to update other modules.

3.2 Step Planning

The behavior of the robot can request different walk directions and velocities. The walking engine has to make sure that this velocity is reached as fast as possible without falling. The previous walking engines used by the RoboCanes agent did not always generate stable motions. Sudden changes in the requested walk speed could make the robot fall. This could be avoided by adding additional limits on the acceleration, but choosing limits manually often reduced the responsiveness of the walk too much.

In this approach we let the walking engine control the acceleration. The maximum acceleration depends on the physical behavior of the robot, which is modeled by the pendulum model. The walk should be generated based on the physical behavior of the model and guarantee a feasible mass acceleration. Generating steps is done in two steps. First, we calculate the reference trajectory for the requested walk speed. The states defining this trajectory are then used as desired states for the step planning.

The center of mass reference trajectory describes the required mass movement for a given walk velocity and step frequency. It can be described completely by the two states (foot positions and mass velocity) during the support exchange. These states and the step values have to be chosen such that the velocity is maintained and the same steps can be repeated to walk with the requested speed. Since the duration of the steps is given by the step frequency and the average velocity is given, we can calculate the distance that has to be covered by the two steps. Thus, the step length and the target positions for the swing legs can directly be calculated. The only unknown value is the mass velocity at the support exchange that yields the same velocity after executing the two steps. If the velocity is too high, the velocity will increase over the two steps. If the velocity is too small, it will decrease further. Using these predictions and an approximated gradient we can approximate the velocity in a few iterations (error less than 1 mm/s after 5 iterations). Therefore, the optimal values at the support exchange states for a requested speed are known and can be used as desired values in the step planning.

The task of the step planning is to choose the current step, such that the robot can reach the reference trajectory. The duration and the swing target of the current step are chosen such that the predicted state after the next action is as close as possible to the corresponding support exchange state defined by the reference trajectory (similar to the capture step foot placement strategies used e.g. in [18, 17, 16]). We approximate the best step values similar to the calculation of the velocities in the reference trajectory iteratively using the pendulum model. Due to errors in the joint control or external disturbances, it can be necessary to adjust the step time and step target position for the current step. The step planning updates these values in every control cycle. First, the remaining step time is calculated such that the lateral velocity at the end of the step is close to the velocity defined in the reference trajectory. Using this time and the current position of the support foot, the mass position at the end of the step can be predicted using the model. The support position and mass movement during the next step depend on the step target for the current step. Therefore, the step target position is chosen, such that the state after the next step is close to the ideal support exchange state from the reference trajectory (again using model predictions). If the requested walking speed changes, the reference trajectory will immediately change to the new trajectory. However, the step planning will

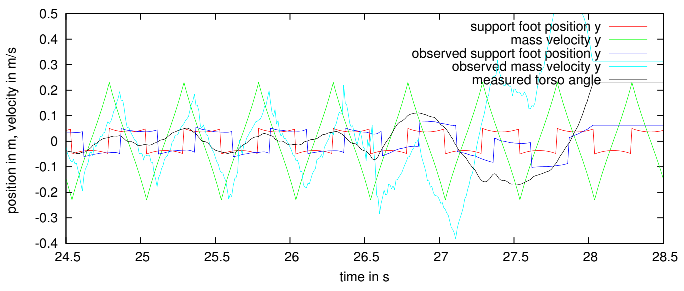


Fig. 2: Generated walk motion without using sensor feedback on a physical NAO (lateral direction). The robot is pushed a little and can not recover from it.

choose feasible steps that will not make the robot fall and reach the requested speed after a few steps.

If there were no errors in the model, the predictions would be accurate and the steps optimal. However, the variances and disturbances on a physical robot add too many errors that add up and make the robot fall quickly (see Figure 2). Sensor feedback has to be used to react to errors. Nevertheless, the better the model fits to the actual physical behavior of the robot, the smaller are the errors and only smaller adjustments in the steps are needed.

3.3 State Estimation and Closed-loop Walk

The walk can only be robust if it reacts to unexpected errors in the motion. These can be caused by external disturbances, inaccurate joint control or also by the errors in the model. Many walk motions control the joints using a high hardness and try to move the joints exactly along the planned trajectories. If the joints move this way with high force, errors in the model would only move the ZMP inside the support polygon. Neither the joint angles, nor the torso angle would give information about the error unless the error is already too large, the ZMP reaches the edge of the support polygon and the robots starts to fall. Force sensors could be used to detect the error, but they can be unreliable on the NAO. Therefore, we use a lower hardness in the ankles and hip. This allow the joints to not move exactly to the set angles. If the generated motion is not correct for the current state (e.g. accelerating too fast), the differences in the reached joint angles and the torso angle will give more information. In other words, we do not try to force the robot to execute an exact motion. However, this requires a good tracking of the robots state to detect any errors and adjust the planned steps quickly

The positions of the feet can directly be observed from the measured joint angles. To obtain the complete robot state including the mass velocity, we combine the foot positions, measured torso rotations and the pendulum model by using a particle filter to estimate the most likely mass velocity. Each particle is a hypothesis for the walk state consisting of the foot positions relative to the mass, the mass velocity and the torso angle. This filter is based on the idea to estimate the mass velocity not only from IMU values, but to combine those values with the knowledge of the model and the observed joint angles. Even though the model is expected to have errors and the point of estimating the

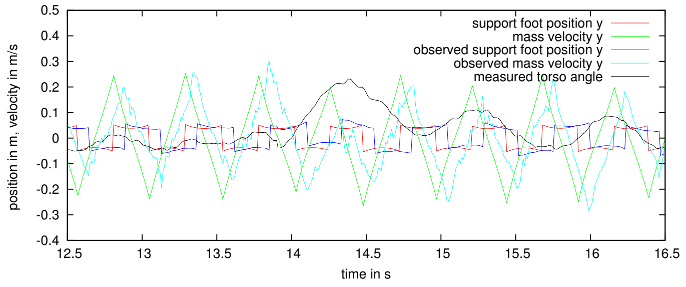


Fig. 3: The support foot positions and velocities in y direction (lateral) using the closed loop walk. The support foot position and velocity set by the walk are now adjusted slightly, if the observed state differs from the expected values.

state is to react to unexpected errors and differences to the model, the robot will not move completely different. Therefore, we update the particles state using the model, but add high noise values to not loose track of the state in the case of disturbances. Additionally, the directly measured To update a particle state using the model we have to decide which foot is the supporting foot which is used as pendulum origin. We use the support foot of the internal state used for generating the motion, but we have to take the control delay into account. Every time the step planning changes the supporting foot, it will take several control cycles until the effect can be observed in the measurements. Instead of setting the delay manually, we added a delay parameter to the estimated state in the filter which represents the number of cycles control delay. This way the filter finds the delay that fits best to the model and observations (and is adjusted automatically, e.g. for the simulator or a physical robot).

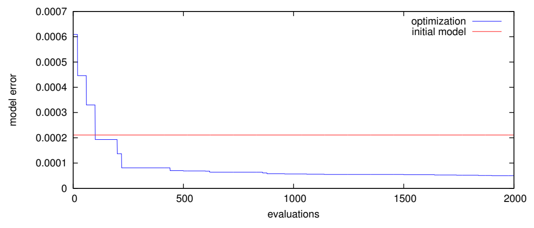
We use the estimated velocity, measured foot positions and torso angles as observed walk state. From the observed walk state we predict the current walk state using the estimated delay. This state reacts very quickly to disturbances, but is also noisy. Therefore, it is merged into the internal walk state of the motion generation using small factors ($f{*}\text{observation}+(1-f){*}\text{internalState}$). These factors are set manually such that the robot reacts fast enough to disturbances without reacting to much to noise in the sensor values. We use values around 0.01 for the foot positions and mass velocity (which seems low, but this update is done 100 times per second on the NAO) and 0.5 for the torso angle.

The plot in figure 3 shows an example of the walk on a physical NAO. The error is not increasing and the walk is stable. Towards the end of the plot the robot was pushed from the side and is able to recover from that.

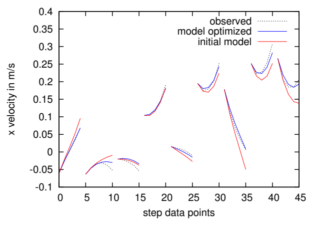
Even though the walk is stable, there are errors in every step even without external disturbances. The robots does not behave exactly as predicted using the model, which prevents it from following the reference trajectory. Therefore, it will not walk with the requested walk speed.

4 Model Optimization

Without external disturbances the errors in the model should be as small as possible for a controlled, stable walk. There will always be a high variance in



(a)



(b)

Fig. 4: a) An example run for the model optimization on a physical NAO. b) The velocities of the observed steps from the physical NAO used for the optimization.

the motion on a physical NAO, but repeating errors might be caused by a bias in the model which can be corrected. Using sequences of observed walk states, we optimize parameters of the model using CMA-ES to improve the predictions. This improves several components such as the state estimation or the step planning. The movement of the inverted pendulum only depends on its length. The height of the center of mass of the NAO could be measured, but there are errors. A shorter or longer pendulum might model the motion of the robot better, since the exact center of mass position is not known. Similarly, an error in the CoM in lateral or sagittal direction might cause errors in the predictions. The model predictions will never match the observed movement, but a bias should be avoided. The pendulum model itself will not be modified by the optimization, but we add parameters to the mapping from the estimated state of the robot to the model. We add offsets for the CoM position in x, y and z direction, offsets for the estimated velocity, factors to adjust the measured foot positions in x direction and a factor for the cycle time. Some of these parameters might not be physically reasonable. However, the point of using a parameter optimization to fit the predictions to the observations is to improve the model without knowing exactly what is causing the errors and which parameters need to be changed. We let the optimization explore which parameters can help with improving the predictions (e.g. as expected the time factor is always very close to 1).

We collect data for the optimization by observing several steps with different mass velocities. The initial state in each recorded state can be used to predict the complete step using the model. The difference between this prediction and the observed positions and velocities are the error that is minimized. The evaluation of parameters does not require to walk using potentially bad parameter values. The optimization is done on the gathered data by calculating several positions on the trajectories using the model and comparing the result positions and velocities to the observed values to calculate an overall mean squared error for the used parameters. This can run on a physical robot in the background with low priority. After a constant amount of iterations it stops and the robot can immediately use the new parameters.

Figure 4a shows an example run of this optimization using data from a NAO robot. The 10 model parameters were optimized using CMA-ES with a population size of 20. Figure 4b shows the velocities of the step observations and the values predicted by the model before and after the optimization. The parameters are only adjusted slightly, but the predictions are improved. This would be very tedious/impossible to do manually for multiple robots.

5 Experiments & Results

We conducted several experiments using different physical and simulated robots to evaluate the effect of the optimization on the walk. The experiments were executed using two different simulators and different physical NAOs. The simulator Webots contains a NAO H21 model, which is very similar to the physical NAOs. The simulator SimSpark uses a robot model similar to the NAO, but with slightly different dimensions and masses. Additionally, it is possible to modify some parameters of the robot model (e.g. longer legs, wider hips). In our experiments, we use the robot types used by the RoboCup 3D Soccer Simulation League. The optimization of the model parameters runs for 20 seconds, then gathers new step data and restarts the optimization. In the experiments using physical robots, it was executed directly on the robot.

We compare some results to the previously used walking engine based on the walk of the RoboCup team B-Human [6], which is a very stable closed-loop walk based on the inverted pendulum model. However, integrated in the RoboCanes agent, it depends highly on an accurate calibration of the NAO and on several parameters that can be set manually. We found a configuration that is very stable, but the acceleration is limited and the robots can not react quick enough in many situations. Sometimes the balancing ignored the requested walk speed completely to walk stable. The results used for comparison are created using the best configuration we found which was used by the team RoboCanes in several competitions.

In the first experiment, the robot walks on a spot (the requested speed is 0) for several minutes. The lateral movement is most important in this experiment, which depends on the step timing and selection of step target positions. The robot should make steps such that it maintains a stable oscillation from left to

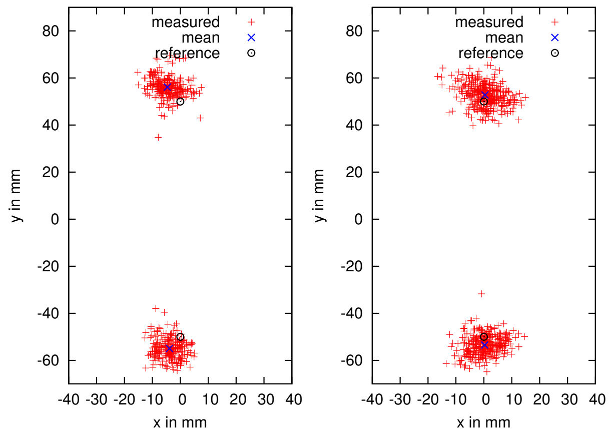


Fig. 5: Variance in the positions of the support foot at the end of each step on a physical NAO without (left) and with the optimization (right).

right without much variance. The model parameters are optimized using information from only a few steps. If better parameters are found, the walking engine directly starts using these parameters, which decreases the average prediction error of the model. The better the predictions for the movement of the mass are, the better are the step target positions that are chosen in the beginning of each step and the step target position does not have to be updated much during the step.

If the robot model would be perfect, the robot would be able to choose each step such that the robot's state at the end of each step is the state defined by the reference trajectory. Figure 5 shows the variance in the support foot positions at the end of the steps made by a physical NAO. Without the optimization the robot walked slowly forward (the feet move toward negative x) and the feet are further apart than the position from the reference trajectory (which is based on the 50 mm hip offset of the NAO's legs). Using the optimized model, there is still a similar variance in the step targets but the bias is reduced and the robot does not walk forward anymore.

For the next experiments the robots receive walk requests that change from 150 mm/s forward to 150 mm/s backward every 3 seconds. The walking engine is expected to accelerate towards the requested walk speed as fast as possible, but without falling. The resulting values are averaged over 5 minutes.

Table 1 shows that the optimization reduces the overall prediction error. In this experiment, we use odometry information from the motion of the torso (position and angle) relative to the feet to calculate the approximate current walk speed. If errors in the predictions prevent the robot from walking with the desired speed, the robot measures that but consistently chooses wrong step target positions which do not change the walk speed correctly. Therefore, a smaller error in the predictions allows the robot to follow the requested walk speeds more accurately. For 0.5 seconds before each change in the walk direction, we comparing the measured walk speed with the requested speed for the average speed error shown in table 1. The optimization reduces the error and standard deviation of the walk speed.

Without changing any parameters and without optimization, the new walking engine walks too fast backwards, such that it can not slow down and falls. For

Table 1: Observed values for the forward/backward walk with 150 mm/s with different robots. The second row for each robot contains the results using the optimization.

		mean prediction error				mean step change		walk speed err. in mm/s	
		x	$\dot{x}$	y	$\dot{y}$	x	y	mean	std.dev.
Webots	NAO	-0.2	1.5	2.0	-12.0	0.25	0.81	-37.0	23.8
		-0.4	3.2	1.1	-3.7	-0.4	0.25	-28.5	18.2
SimSpark	type0	-0.2	1.4	0.8	-4.8	-0.1	0.4	-24.4	16.5
		0.4	-2	0.6	-5.6	0.08	0.5	-3.8	20.8
	type3	-0.1	0.4	0.9	-5.3	-0.08	0.51	-12.9	15.7
		0.2	-0.4	0.4	-1.1	0.04	-0.01	-8.6	19.1
physical	Nao1	2.7	-16.8	4.6	-29.8	1.05	2.48	61.0	106.3
		1.3	-4.7	1.4	-0.8	0.32	-0.16	38.0	45.7
	Nao2	1.6	-10.3	4.9	-32.0	1.26	2.94	64.8	102.9
		0.3	-1.2	0.5	8.8	0.32	-0.15	24.8	77.5
physical	old walk	-	-	-	-	-	-	-51.8	88.6

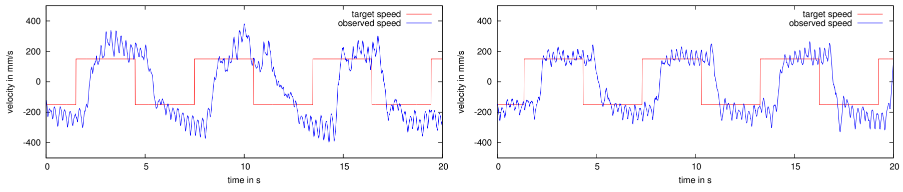


Fig. 6: Forward/backward walk with 150 mm/s on a physical robot using the new walk without optimization (left) and with optimization (right).

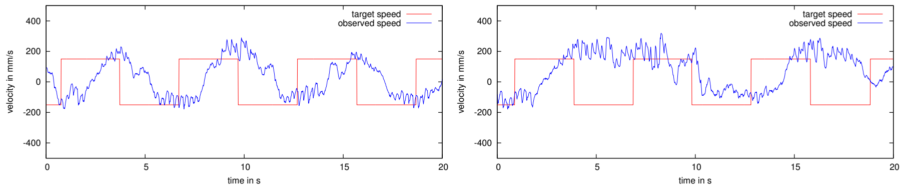


Fig. 7: Results for the forward/backward walk using the old walking engine. With our current parameters this walk accelerates slowly to be stable (left). The old walk with a higher acceleration limit (right).

the experiments on the physical robot without the optimization, it was already necessary to manually set a mass offset of 5 mm in the model. Figure 6 shows measured walk speeds using this offset and no optimization on a physical robot. The walk speed varies and it walks too fast. The velocity backwards is too high, such that it sometimes takes longer to slow down and change the direction. With optimized parameters, the improved prediction yields a more stable walk that maintains a more constant velocity. The backwards velocity is still too high, but not as much as without the optimization and the robot does not fall.

Since the observed walk speed does not depend on information from the walking engine, we can compare these results with the speeds of the previously used walking engine. This walk is very stable, but figure 7 shows that it accelerates much slower and the forward velocity is faster than backwards. One of the parameters of this walk controls the maximum velocity change. However, increasing this parameter does not necessarily improve the walk. With a higher allowed acceleration the walk is less controlled and the torso starts oscillating more. When the walk is too unstable, the old walk even ignores the walk request completely to only balance as shown in figure 7.

We conducted several more experiments, all showing that the optimization yields a more controlled, more stable walk. On hard surfaces (e.g. thin carpet in RoboCup competitions) the walk can be very fast, but it can also walk in different environments without manual parameter tuning. By automatically adjusting the model it is able to walk stable on soft artificial grass or outside on pavements with shoes that add extra weight. [1] As a byproduct, the lower hardness in the joint control reduces the sum of the currents in the hip and knee joints by approximately 10% compared to walking with full hardness.

[1] Video material: http://www.cs.miami.edu/home/visser/tmp/walking/

6 Conclusion

We have implemented a closed-loop LIPM-based dynamic gait for the NAO. Components such as the reference trajectory generation and step planning use a LIPM to generate the motion. We added several parameters to the model that can be modified to improve the model predictions. These parameters are simple offsets and scaling factors that can compensate systematic errors caused by variances in the hardware of physical robots.

We optimize the model parameters using CMA-ES to fit the model to observations of the robot's movement. The experiments show, that the optimization reduces bias in the predictions by approximately 50% to 90% (varies depending on walk requests, robot, environment). It is able to improve the walk, even if it uses only a few observed steps to adjust the parameters and runs with limited computational resources, e.g. on the 1.6 GHz Atom CPU of a physical NAO. The optimization does not require an explicit training and takes less than a minute on a NAO such that the parameters can be updated while the robot is executing other behaviors. Our experiments show that the reduced model errors yield a more controlled, faster and energy-efficient walk that works on different robots and environments without manual parameter tuning.

References

1. Aftab, Z., Robert, T., Wieber, P.B.: Ankle, Hip and Stepping Strategies for Humanoid Balance Recovery with a Single Model Predictive Control Scheme. In: Proc. of the IEEE-RAS International Conference on Humanoid Robots, 2012 (2012)
2. Chernova, S., Veloso, M.: An Evolutionary Approach to Gait Learning for Four-Legged Robots. In: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2004. vol. 3, pp. 2562–2567. IEEE (2004)
3. Dallali, H., Kormushev, P., Li, Z., Caldwell, D.: On Global Optimization of Walking Gaits for the Compliant Humanoid Robot, COMAN Using Reinforcement Learning. Journal of Cybernetics and IT vol. 12, no. 3, pp. 39–52 (2012)
4. Faber, F., Behnke, S.: Stochastic Optimization of Bipedal Walking using Gyro Feedback and Phase Resetting. In: 7th IEEE-RAS International Conference on Humanoid Robots, 2007. pp. 203–209. IEEE (2007)
5. Gökçe, B., Akin, H.L.: Parameter Optimization of a Signal-Based Omni-directional Biped Locomotion using Evolutionary Strategies. In: RoboCup 2010: Robot Soccer World Cup XIV, pp. 362–373. Springer (2011)
6. Graf, C., Röfer, T.: A Center of Mass Observing 3D-LIPM Gait for the RoboCup Standard Platform League Humanoid. In: RoboCup 2011: Robot Soccer World Cup XV. pp. 101–112. Lecture Notes in Artificial Intelligence (2011)
7. Hansen, N., Müller, S.D., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). Evolutionary computation 11(1), 1–18 (2003)
8. Hebbel, M., Kosse, R., Nistico, W.: Modeling and Learning Walking Gaits of Biped Robots. In: Proceedings of the Workshop on Humanoid Soccer Robots of the IEEE-RAS International Conference on Humanoid Robots. pp. 40–48 (2006)
9. Huang, W., Chew, C.M., Zheng, Y., Hong, G.S.: Pattern Generation for Bipedal Walking on Slopes and Stairs. In: 8th IEEE-RAS International Conference on Humanoid Robots, 2008. pp. 205–210. IEEE (2008)
10. Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H.: Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In: Proceedings of IEEE International Conference on Robotics and Automation (ICRA), 2003. vol. 2, pp. 1620–1626 vol.2 (2003)

11. Kajita, S., Kanehiro, F., Kaneko, K., Yokoi, K., Hirukawa, H.: The 3D Linear Inverted Pendulum Mode: A Simple Modeling for a Biped Walking Pattern Generation. In: Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2001. vol. 1, pp. 239–246 vol.1 (2001)
12. Kulk, J., Welsh, J.S.: Evaluation of Walk Optimisation Techniques for the NAO Robot. In: 11th IEEE-RAS International Conference on Humanoid Robots, 2011. pp. 306–311. IEEE (2011)
13. Liu, C., Chen, Q.: Proceedings of the 2015 Chinese Intelligent Automation Conference: Intelligent Technology and Systems, chap. Methods Synthesis of Central Pattern Generator Inspired Biped Walking Control, pp. 371–379. Springer Berlin Heidelberg, Berlin, Heidelberg (2015)
14. Liu, J., Urbann, O.: Bipedal walking with dynamic balance that involves three-dimensional upper body motion. Robotics and Auton. Systems 77, 39 – 54 (2016)
15. MacAlpine, P., Barrett, S., Urieli, D., Vu, V., Stone, P.: Design and Optimization of an Omnidirectional Humanoid Walk: A Winning Approach at the RoboCup 2011 3D Simulation Competition. In: AAAI (2012)
16. Missura, M., Behnke, S.: Lateral Capture Steps for Bipedal Walking. In: 11th IEEE-RAS Intern.l Conf. on Humanoid Robots, 2011. pp. 401–408. IEEE (2011)
17. Missura, M., Behnke, S.: Omnidirectional Capture Steps for Bipedal Walking. In: Proc. of IEEE Int. Conf. on Humanoid Robots (Humanoids) (2013)
18. Missura, M., Behnke, S.: Balanced Walking with Capture Steps. In: RoboCup 2014: Robot World Cup XVIII (2015)
19. Niehaus, C., Röfer, T., Laue, T.: Gait Optimization on a Humanoid Robot using Particle Swarm Optimization. In: Zhou, Pagello, Menegatti, Behnke (eds.) Proc. of the Second WS on Humanoid Soccer Robots in conjunction with the 2007 IEEE-RAS Intern. Conf. on Humanoid Robots (2007)
20. Ott, C., Roa, M., Hirzinger, G.: Posture and Balance Control for Biped Robots based on Contact Force Optimization. In: 11th IEEE-RAS International Conference on Humanoid Robots, 2011. pp. 26–33 (2011)
21. Rokbani, N., Benbousaada, E., Ammar, B., Alimi, A.M.: Biped Robot Control using Particle Swarm Optimization. In: IEEE International Conference on Systems Man and Cybernetics (SMC), 2010. pp. 506–512. IEEE (2010)
22. Shafii, N., Lau, N., Reis, L.P.: Learning to Walk Fast: Optimized Hip Height Movement for Sim. and Real Humanoid Robots. J of Intell. & Rob. Sys. pp. 1–17 (2015)
23. Strom, J.H., Slavov, G., Chown, E.: Omnidirectional Walking Using ZMP and Preview Control for the NAO Humanoid Robot. In: RoboCup 2009: Robot Soccer World Cup XIII. pp. 378–389. Lecture Notes in Artificial Intelligence (2009)
24. Torres, E., Garrido, L.: Automated Generation of CPG-Based Locomotion for Robot Nao. In: Robot Soccer World Cup XV, pp. 461–471. Springer (2012)
25. Urbann, O., Hofmann, M.: Modification of foot placement for balancing using a preview controller based humanoid walking algorithm. In: RoboCup 2013: Robot world cup XVII, pp. 420–431. Springer (2013)
26. Wieber, P.B.: Trajectory Free Linear Model Predictive Control for Stable Walking in the Presence of Strong Perturbations. In: 6th IEEE-RAS International Conference on Humanoid Robots, 2006. pp. 137–142 (2006)
27. Wu, S., Pan, G., Yu, L.: Dynamic Walking Gait Designing for Biped Robot Based on Particle Swarm Optimization. In: International Conference on Control Engineering and Communication Technology (ICCECT), 2012. pp. 372–377 (2012)
28. Yi, S.J., Zhang, B.T., Hong, D., Lee, D.: Active Stabilization of a Humanoid Robot for Impact Motions with Unknown Reaction Forces. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012. pp. 4034–4039 (2012)
29. Zhang, Q., Tang, T., Zhang, D., Yang, S., Shao, Y.: Optimized Central Pattern Generator Network for NAO Humanoid Walking Control. In: IEEE Intern. Conf. on Robotics and Biomimetics (ROBIO), 2013. pp. 1486–1490. IEEE (2013)