# Learning to Use Toes in a Humanoid Robot

Klaus Dorer

Hochschule Offenburg, Elektrotechnik-Informationstechnik, Germany,
klaus.dorer@fh-offenburg.de

**Abstract.** In this paper we show that a model-free approach to learn behaviors in joint space can be successfully used to utilize toes of a humanoid robot. Keeping the approach model-free makes it applicable to any kind of humanoid robot, or robot in general. Here we focus on the benefit on robots with toes which is otherwise more difficult to exploit. The task has been to learn different kick behaviors on simulated Nao robots with toes in the RoboCup 3D soccer simulator. As a result, the robot learned to step on its toe for a kick that performs 30% better than learning the same kick without toes.

## 1 INTRODUCTION

Evolution has spent the effort to equip humans with toes. Toes, among other advantages, allow humans to walk smoother and faster with longer steps. Naturally, this has been subject of much research in humanoid robotics.

Passive toe joints have been suggested and used since quite some time to consume and release energy in the toe-off phase of walking. Early examples are the Monroe [8], HRP-2 [15] or Wabian-2R [14] robots.

Active toe joints are less common, but also in use for several years. The H6 robot [11] used toes to reduce angular knee speeds when walking, to increase leg length when climbing and to kneel. The Toni robot [3] improved walking by over-extending the unloading leg using toes. Toyota presented a toed robot running at 7km/h [16]. Lola [4] is equipped with active toe joints and active pelvis joints to research human-like walking. Active toe joints increase the difficulty of leg behaviors in a couple of ways:

- of course they add a degree of freedom (DoF).
- it makes the leg of the robot kinematically redundant. Inverse-kinematics will not find unique solutions which adds effort to deal with, but also offers room for optimization [5].
- moving the toe of a support leg in any case reduces the contact area of the foot, either by stepping on the toe or by lifting the toe off the ground.
- on real robots, they typically increase the complexity of construction and weight of the feet and consume energy.

All examples above for passive or active toes are model-specific either in a sense to be specific to a robot model or to be behavior-specific in calculating desired toe angles.

A couple of more recently built robots avoid the complexity of actuating toes and the inflexibility of flat feet by wearing shoes. Examples are Petman, Poppy [9] or Durus [12; 2] that is able to walk many times more energy-efficiently with this and other improvements. Shoes, or rounded feet in general, improve energy-efficiency of walking, but would not provide the benefits of active toes for any behavior like the kicking behavior demonstrated in this paper.

Learning in combination with toe movement has been employed by Ogura et al. [14]. They used genetic algorithms to optimize some parameters of the ZMP-based foot trajectory calculation for continuous and smooth foot motion. This approach optimizes parameters of an abstract parameter space defined by an underlying model. This has the advantage, that the dimension of the search space is kept relatively small, but it does not generalize to work for any behavior.

Learning to kick is common in many RoboCup leagues. A good overview can be found in [7]. However, none of these report on kicking with toed robots. MacAlpine et al. [10] use a layered learning approach to learn a set of behaviors in the RoboCup 3D soccer simulator used in this work. They learn keyframes for kicking behaviors and parameters for a model-based walking. Although the work is not focused on learning to use toes, they report on an up to 60% improvement in overall soccer game play using a Nao robot variation with toes (see Section 2). This demonstrates the ability of their approach to generalize to different robot models, but is still model-based for walking. Abdolmaleki et al.[1] use a keyframe based approach and CMA-ES, as we do, to learn a kick with controlled distance, which achieves longer kick distances than reported here, but also requires considerably longer preparation time. However, their approach is limited to two keyframes resulting in 25 learning parameters, while our approach is not limited to an upper number of keyframes (see Section 3). Also they do not make use of toes. It is interesting to see, that also in their kick, the robot moves to the tip of its support foot to lengthen the support leg. Not using toes, their kick results in a falling robot after each kick.

The work presented here generates raw output data during learning without using an underlying robot- or behavior-model. To demonstrate this, we present learning results on different robot models (focusing on a simulated Nao robot with toes) and two very different kick behaviors without an underlying model.

The remainder of the paper is organized as follows: In Section 2 we provide some details of the simulation environment used. Section 3 explains our model-free approach to learning behaviors using toes. It is followed by experimental results in Section 4 before we conclude and indicate future work in Section 5.

## 2 DOMAIN

The robots used in this work are robots of the RoboCup 3D soccer simulation which is based on SimSpark[1] and initially initiated by [13]. It uses the ODE physics engine[2] and runs at a speed of 50Hz. The simulator provides variations

[1] http://simspark.sourceforge.net/
[2] http://www.ode.org/

of Aldebaran Nao robots with 22 DoF for the robot types without toes and 24 DoF for the type with toes, NaoToe henceforth. More specifically, the robot has 6 (7) DoF in each leg, 4 in each arm and 2 in its neck. There are several simplifications in the simulation compared to the real Nao:

- all motors of the simulated Nao are of equal strength whereas the real Nao has weaker motors in the arms and different gears in the leg pitch motors.
- joints do not experience extensive backlash
- rotation axes of the hip yaw part of the hip are identical in both robots, but the simulated robot can move hip yaw for each leg independently, whereas for the real Nao, left and right hip yaw are coupled
- the simulated Naos do not have hands
- the touch model of the ground is softer and therefore more forgiving to stronger ground touches in the simulation
- energy consumption and heat is not simulated
- masses are assumed to be point masses in the center of each body part

The feet of NaoToe are modeled as rectangular body parts of size 8cm x 12cm x 2cm for the foot and 8cm x 4cm x 1cm for the toes (see Figure 1). The two body parts are connected with a hinge joint that can move from -1 degrees (downward) to 70 degrees.

All joints can move at an angular speed of at most 7.02 degrees per 20ms. The simulation server expects to get the desired speed at 50 Hz for each joint. If no speeds are sent to the server it will continue movement of the joint with the last speed received. Joint angles are noiselessly perceived at 50Hz, but with a delay of 40ms compared to sent actions. So only after two cycles the robot knows the result of a triggered action. A controller provided for each joint inside the server tries to achieve the requested speed, but is subject to maximum torque, maximum angular speed and maximum joint angles.

The simulator is able to run 22 simulated Naos in real-time on reasonable CPUs. It is used as competition platform for the RoboCup 3D soccer simulation league[3]. In this context, only a single agent was running in the simulator.

## 3 APPROACH

The guiding goal behind our approach is to create a framework that is model-free. With model-free we depict an approach that does not make any assumptions about a robot's architecture nor the task to be performed. Thus, from the viewpoint of learning, our model consists of a set of flat parameters. These parameters are later grounded inside the domain. In our case, the grounding would mean to create 50 joint angles or angular speed values per second for each of the 24 joints of NaoToe. This would result in 1200 values to learn for a behavior with one second duration assuming the 50 Hz frequency of the simulator. This seemed unreasonable for time being and led to some steps of relaxing the ultimate goal.

[3] http://www.robocup.org/robocup-soccer/simulation/

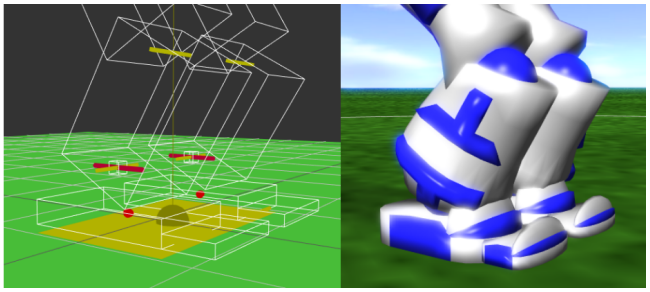

**Fig. 1.** Wire model of the Nao with toes (left) and how it is visualized (right).

As a first step, the search space has been limited to the leg joints only. This effectively limits the current implementation of the approach to leg behaviors, excluding, for example, behaviors to get up. Also, instead of providing 50 values per second for each joint, we make use of the fact that output values of a joint over time are not independent. Therefore, we learn keyframes, i.e. all joint angles for discrete phases of movement together with the duration of the phase from keyframe to keyframe. The experiments described in this paper used two to eight of such phases. The number of phases is variable between learning runs, but not subject to learning for now, except for skipping phases by learning a zero duration for it.

The RoboCup server requires robots to send the actual angular speed of each joint as a command. So the first representation used in this work is to directly learn the speed values to be sent to the simulator. This requires to learn 15 parameters per phase (14 joints + 1 for the duration of the phase) resulting in 30, 60, 90 and 120 parameters for the 2, 4, 6, 8 phases worked with. The disadvantage of this approach is, that the speed will be constant during one phase and will especially not adapt to discrepancies of the commanded and the true motor movement.

The second representation therefore interpreted the parameters as angular values to reach at the end of a phase as is done in [10] for kicking behaviors. A simple controller divided the difference of the current angle and the goal angle of each joint by the duration and sent a speed accordingly. The two representations differ in cases, when the motor does not exactly follow the commanded speed. Using keyframes of angles will adjust the speeds to this situation.

A third representation used a combination of angular value and the maximum amount of angular speed each joint should have. The direction of movement is entirely encoded in the angular values, but the speed is a combination of representation one and two above. If the amount of angular speed does not allow to reach the angular value, the joint behaves like in version 1. If the amount of angular speed is bigger, the joint behaves like version 2. This almost doubles the amount of parameters to learn, but the co-domain of values for the speed values is half the size, since here we only require an absolute amount of angular speed.

Interpolation between keyframes of angles is linear for now. It could be changed to polynoms or splines, but we do not expect a big difference since the resulting behavior is anyhow smoothened by the inertia of body parts and since phases can be and are learned to be short in time if the difference from linear to polynomial matters.

Learning is done using plain genetic algorithms and covariance matrix adaptation evolutionary strategies (CMA-ES) [6]. Feedback from the domain is provided by a fitness function that defines the utility of a robot. Currently implemented fitness functions use ball position, robot orientation and position during or at the end of a run. The decision maker to trigger the behavior also uses foot force sensors.

To summarize, the following domain knowledge is built into our approach:

- the system has to provide values at 50Hz (used as angular speeds of the joints)
- there are up to 232 free parameters for which we know the range of reasonable values (defined by minimum and maximum joint angles and angular speeds and maximum phase duration)
- a fitness function using domain information gives feedback about the utility of a parameter set
- a kicking behavior is made possible by moving the player near the vicinity of the ball.

The following domain knowledge is not required or built into the system:

- geometry, mass of body parts
- position or axis of joints
- the robot type (humanoid, four-legged, ...)

## 4 RESULTS

The first behaviors to learn were kicks. Experiments have been conducted as follows. The robot connects to the server and is beamed to a position favorable for kicking. It then starts to step in place. Kicking without stepping is easier to achieve, but does not resemble typical situations during a real game. The step in place is a model-based, inverse-kinematic walk and is not subject to this learning for now. After 1.4s, the agent is free to decide to kick. It will then decide to kick as soon as foot pressure sensors show a favorable condition. Here it means if kicking with the right leg, the left leg had to just touch the ground and the right leg had to leave the ground. After the kick behavior, the agent continues to step in place until the overall run took five seconds. A run is stopped earlier, if the agent falls (z component of torso's up-vector $< 0.5$). This will typically be the case for most of the individuals of the initial random population.

Table 1 shows the influence of the three representations used for the learning parameters as well as the influence of using different amounts of phases. Each value in the table is the result of 400.000 kicks performed using genetic

algorithms. The best kick has been learned using version 3 and 4 phases ending in a kick that is more than 8m on average. Due to the long learning times, no oversampling for the results in the table has been used, so there is certainly some noise in the single values. However, in all of the runs the agent was able to learn at least a reasonable kick. Averaging over the four runs for different phases, version 3 (learning angular values and speeds) had the highest utility. The value for 8 phases just learning angular speeds is missing due to a problem with the simulator that lets the robot explode in some situations. This happened too often in the 8 phase angular speed scenario to create sensible learning data (see below).

**Table 1.** Influence of representation and number of phases.

| Phases | 2 | 4 | 6 | 8 | Average |
|---|---|---|---|---|---|
| Just Speed | 6.4 | 3.5 | 5.4 | - | 5.1 |
| Just Angles | 4.2 | 3.7 | 4.4 | 5.0 | 4.3 |
| Angles and Speed | 4.1 | 8.1 | 7.2 | 5.3 | 6.2 |
| Average | 4.9 | 5.7 | 5.7 | 5.15 | 5.2 |

The result of the learning process for NaoToe learning angles and speeds with four phases is shown in Figure 2. It was achieved using a plain genetic algorithm with

- population size: 200
- genders: 2
- parents per individual: 2
- individual mutation probability: 0.1
- gene mutation probability: 0.1
- selection: Monte-Carlo + take over best 10%
- recombination: Multi-Crossover.

Utility function is the kick distance (positive x-coordinate) minus the absolute amount of y-deviation of a straight kick minus a penalty of 2 for falling. The factor two has been chosen as a result of initial experiments. A much lower penalty resulted in better kick distances but the robot falling regularly. A much higher penalty did not result in good kicks, as if it is easier to learn to kick first and to then try keeping upright as opposed to the other way around.

The figure shows the average fitness of a generation and the best individual of 128 generations measured with 10-fold oversampling. It combines the results of $200 * 10 * 128 = 256,000$ kicks performed during learning in about two days of simulation. As can be seen, the learning resulted in a kick distance of more than 8m.

The noise in the curve, especially points with decreasing utility compared to the predecessor generation, is partially due to mutation of the previous best

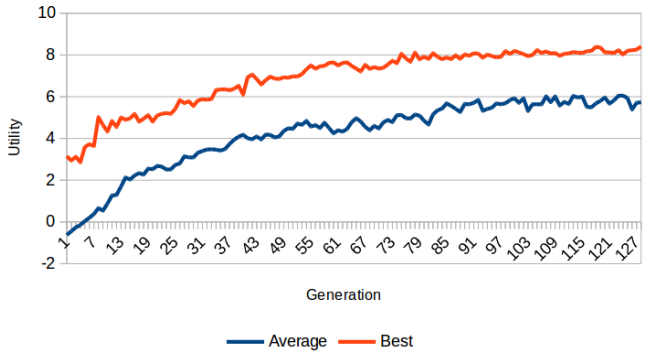


**Fig. 2.** Learning curve of plain genetic learning with NaoToe.

individual, but mainly it is the non-determinism of the simulator. It could be decreased by a higher number of oversampling runs, lengthening the learning process even more.

The resulting initial movement of the robot is shown in Figure 6. The robot learned to improve the kick considerably by stepping on its toes of the support leg, seen especially in sub-images three and four. Not shown in the figure: after the kick, the robot also learned to get back into position to successfully continue stepping in place.

The overall kick takes eight simulation cycles, which takes 0.16s to perform. A comparable hand-tuned kick reaching 8m takes about 1s, a kick reaching 15m takes about 2s. With a speed of approximately 1m/s for the fastest teams, this means we can perform the new kick before opponents reach the ball in situations where opponents are almost one (two) meter(s) closer compared to comparable kicks in the league. Although the fitness function does not explicitly contain a preference for short times, implicitly quick kicks are preferred by the penalty for falling. The longer the robot is on one leg, the higher the likeliness of falling down. The detailed movement of the toes is shown in Figure 3.

The same learning has been performed for the Nao without toes. As can be seen in Figure 4, without the availability of toes, the learning curve flattens out at 5.5m.

Table 2 shows a summary of the results for 10-fold and 50-fold oversampling. The difference of the two oversampling columns shows that there is still quite some noise in measuring utilities, even using 10-fold oversampling. NaoToe performs more than 30% better than Nao. It fell 3 out of 50 times, which is unfavorable and indicates that the penalty for falling should have been chosen slightly higher.

Most interesting are the cross-parameterization runs in row three and four. Using the kick learned by NaoToe on a Nao without toes (ignoring the toe parameters) results in falling 98% of the time and a kick utility of 1.275. Considering the fall penalty of the fitness function, the average kick distance would only be

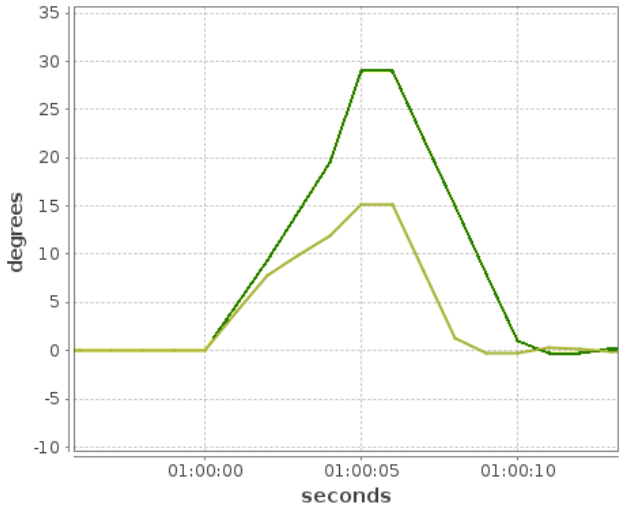


**Fig. 3.** Movement of the toes (support leg in dark).

approximately 3.25m. As can be seen in Figure 6, the robot is able to lean backwards considerably by stepping on its toe. Without the toe available in Nao, the robots falls backwards most of the time.

Similar results were measured when running the parameterization learned by Nao on NaoToe. Toe movement has been kept at zero in this run. Although the robot remains standing in all 50 tries, the result is a mere 1.5m. Deeper analysis shows that the robot hits the ground with the heel of its kicking leg most of the time. This is surprising and not yet completely understood. One reason could be the slightly different mass distribution in the foot of NaoToe compared to Nao. Another reason could be that although the toes are not actively moving, they could be slightly bend by the force acting on them. However, the amount has been confirmed to be less than 0.01 degrees and is therefore unlikely to cause such an effect. Also, foot force sensors are used to decide when to trigger the behavior. It could be that the force sensors in the feet show zero values, while the toes still touch the ground.

Finally, row five of Table 2 shows the impact of not moving the toes for the NaoToe parameterization on a NaoToe. The robot is four times more likely to fall than with moving toes. The average kick distance of approximately 5.4m is also considerably less than when using the toes.

To show that this model-free approach is able to learn other behaviors, the utility function was changed to measure the ball movement to the side in order to learn a sidekick. Sidekicks are more difficult since only the hip roll and yaw-pitch joints allow to create sidewise movements. Both joints are limited not to move to the side of the other leg in order to avoid leg-to-leg collisions. Nevertheless, NaoToe was able to learn a kick with more than 5m to the side. The toes,

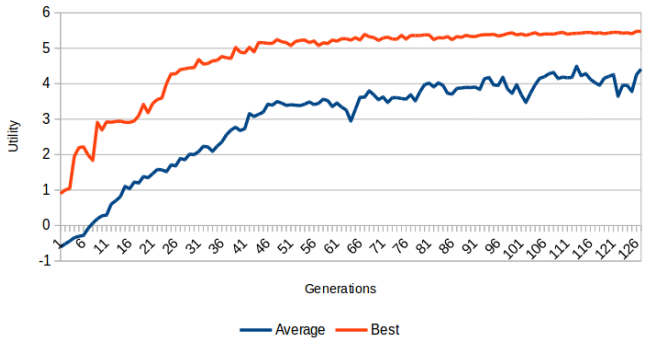


**Fig. 4.** Learning curve of plain genetic learning with Nao.

**Table 2.** Utilities of learned kicks and of cross-parameterization.

| Utility measures | 10 fold | 50 fold | falls (in %) |
|---|---|---|---|
| Nao | 5.469 | 5.161 | 0 |
| NaoToe | 8.407 | 7.736 | 6 |
| Nao with NaoToe parameters | 1.135 | 1.275 | 98 |
| NaoToe with Nao parameters | 1.635 | 1.515 | 0 |
| NaoToe without moving toes | 5.627 | 5.226 | 24 |

unsurprisingly, did not provide a benefit for this kick. Videos of the kicks are available here[4].

Initial experiments with learning to walk this way by performing a learned double step during model-based walking results in a walk that is up to now 86% the speed of pure model-based walking. Unfortunately, attempts to learn walking using NaoToe are hampered by a bug in the simulator that lets the robot explode in some cases of toe movement. For kick learning this did in most cases not exhibit a problem, since such parameter settings did not result in ball movement and were eliminated by the genetic search. For walking, however, the genetic algorithm does exploit this bug by learning to explode in situations were the torso fragment of the robot is boosting forward.

## 5 CONCLUSIONS and FUTURE WORK

With the model-free approach presented in this paper we were able to learn different behaviors on different robot type variations. In particular, the advantages of a robot with toes could be exploited in a quick forward-kick without the need to care for problems with, for example, inverse kinematics.

[4] https://www.dropbox.com/s/u3k3117zc0ptg1m/RoboCupSymposium.mp4?dl=0

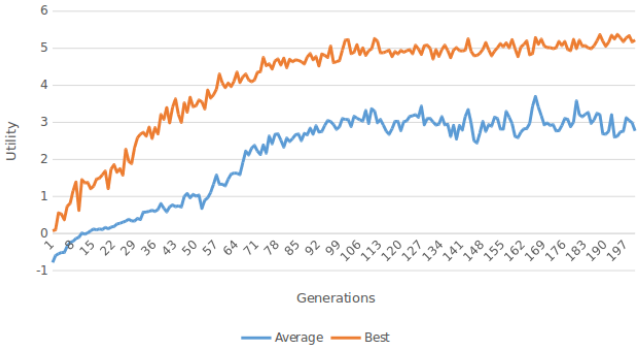


**Fig. 5.** Learning curve of learning a sidekick with NaoToe.

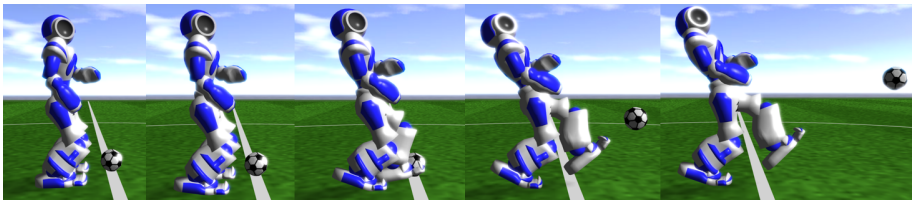

**Fig. 6.** Sequence of movement when kicking with NaoToe.

We have not been successful in transferring a learned kick on Nao to a useful kick on NaoToe so far using CMA-ES. All attempts ended in kicks of NaoToe with a distance of two to four meters only. The opposite, learning a kick on Nao from parameters learned for NaoToe did work. This deserves further investigations since learning times could hopefully be reduced considerably compared to learning from scratch each time. CMA-ES is, however, able to adjust a learned kick of e.g. NaoToe to different ball positions with respect to the robot.

The examples of learned behaviors in this paper are all kicks. The model-free approach, however, should allow to learn any behavior, at least for legs in its current implementation. We are currently investigating to learn a walk using toes. This is more difficult since the duration of a walk behavior before repeating itself is longer than for the kicks shown here, so the parameter space is increasing. First results are promising, but hampered by a problem of the currently used simulator.

## ACKNOWLEDGMENT

Thanks to the magmaOffenburg team for providing the agent software and tools used in this paper, as well as to the contributors of the RoboCup 3D soccer simulator and RoboViz visualizer.

## References


1. Abdolmaleki A, Simoes D, Lau N, Reis L P, Neumann G: *Learning a Humanoid Kick With Controlled Distance.* Proceedings of the 20th RoboCup International Symposium, Leipzig, Germany, 2016.
2. Ackerman E: *DURUS Brings Human-Like Gait (and Fancy Shoes) to Hyper-Efficient Robots.* http://spectrum.ieee.org/automaton/robotics/humanoids/durus-brings-humanlike-gait-and-fancy-shoes-to-hyperefficient-robots, 2016.
3. Behnke S: *Human-Like Walking using Toes Joint and Straight Stance Leg.* In Proceedings of 3rd International Symposium on Adaptive Motion in Animals and Machines (AMAM), Ilmenau, 2005.
4. Buschmann T, Schwienbacher M, Favot V, Ewald A and Ulbrich H: *The Biped Walking Robot Lola – Hardware Design and Walking Control.* Journal of the Robotics Society of Japan 30, 2012.
5. Buschmann T: *Simulation and control of biped walking robots.* Ph.D. dissertation, Technische Universität München, 2010.
6. Hansen N, Müller S D and Koumoutsakos P: Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES). Evolutionary Computation Volume 11 Issue 1, 2003.
7. Jouandeau N, Hugel V: *Optimization of Parametrised Kicking Motion for Humanoid Soccer Player.* In Autonomous Robot Systems and Competitions (ICARSC), 2014.
8. Kumagai M and Emura T: *Sensor-based walking of human type biped robot that has 14 degree of freedoms.* Annual Conference on Mechatronics and Machine Vision in Practice(M2VIP-97)), pp. 112, 1997.
9. Lapeyre M, Rouanet P and Oudeyer P: *The Poppy Humanoid Robot: Leg Design for Biped Locomotion.* 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, November 3-7, 2013.
10. MacAlpine P, Depinet M, Stone P: *UT Austin Villa 2014: RoboCup 3D Simulation League Champion via Overlapping Layered Learning.* Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI), vol 4, pp. 2842–48, 2015.
11. Nishiwaki K, Kagami S, Kuniyoshi Y, Inaba M and Inoue H: *Toe Joints that Enhance Bipedal and Fullbody Motion of Humanoid Robots.* Proceedings of the 2002 IEEE International Conference on Robotiks and Automation Washington, DC May 2002
12. Reher E A, Cousineau A, Hereid C, Hubicki M and Ames A D: *Realizing Dynamic and Efficient Bipedal Locomotion on the Humanoid Robot DURUS.* To appear in: International Conference on Robotics and Automation (ICRA), 2016.
13. Obst O and Rollmann M: *SPARK - A Generic Simulator for Physical Multiagent Simulations* Computer Systems Science and Engineering, 20(5), September 2005.
14. Ogura Y, Shimomura K, Kondo H, Morishima A, Okubo T, Momoki S, Lim H and Takanishi A: *Human-like Walking with Knee Stretched, Heel-contact and Toe-off Motion by a Humanoid Robot* Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), pp. 3976–81, 2006.

15. Sellaouti R, Stasse O, Kajita S, Yokoi K and Kheddar A: *Faster and Smoother Walking of Humanoid HRP-2 with Passive Toe Joints.* Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, October 9 - 15, 2006.
16. Tajima R, Honda D and Suga K: *Fast Running Experiments Involving a Humanoid Robot.* Proc. IEEE Int. Conference on Robotics and Automation (ICRA), pp. 1571–6. 2009.

## APPENDIX

Parameter values learned by Nao and NaoToe for the forward kick behavior. The abbreviations are L/R = left/right, H/K/F/T = hip/knee/foot/toe, Y/R/P = yaw/roll/pitch. Times are in cycles, angles in degrees, speeds in degrees per cycle of 50Hz.

**Table 3.** Parameter values learned by Nao and NaoToe.

| | Nao | | | | | | | | NaoToe | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Phase | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| TIME | 3.3 | 1.5 | 0.2 | 1.5 | - | - | - | - | 2.2 | 2.3 | 0.7 | 0.9 | - | - | - | - |
| | Angle | | | | Speed | | | | Angle | | | | Speed | | | |
| LHYP | 0.3 | -21.3 | -5.4 | -22.7 | 5.8 | 0.0 | 5.2 | 3.0 | -5.7 | 0.9 | -43.6 | -18.5 | 0.6 | 1.3 | 5.9 | 5.6 |
| LHR | -7.0 | -5.4 | 0.2 | 7.6 | 4.9 | 3.1 | 4.7 | 0.9 | 5.2 | -10.0 | 0.6 | 12.0 | 4.3 | 3.4 | 4.9 | 4.0 |
| LHP | -11.7 | 19.4 | -36.3 | -57.3 | 6.4 | 0.3 | 1.5 | 1.5 | -28.6 | -8.7 | 18.8 | 12.8 | 6.7 | 5.7 | 3.4 | 6.9 |
| LKP | -12.6 | -33.9 | -71.1 | -57.7 | 3.7 | 5.6 | 7.2 | 2.5 | -74.0 | -78.8 | -61.7 | -58.4 | 2.9 | 2.1 | 6.2 | 5.4 |
| LFP | 7.5 | -12.3 | -18.2 | -11.2 | 0.1 | 5.1 | 0.6 | 7.1 | 18.8 | -13.9 | -11.0 | -11.0 | 6.4 | 0.3 | 4.5 | 4.2 |
| LFR | 13.9 | 15.8 | 6.2 | 9.3 | 2.3 | 6.5 | 2.5 | 0.7 | 2.9 | 11.8 | -3.1 | -11.8 | 6.0 | 3.2 | 2.2 | 1.7 |
| LTP | - | - | - | - | - | - | - | - | 68.3 | 21.9 | 25.2 | 63.9 | 4.1 | 4.4 | 4.1 | 4.2 |
| RHYP | -28.3 | -25.7 | -43.6 | -31.9 | 6.6 | 1.2 | 6.9 | 3.1 | -20.3 | -26.7 | -7.6 | -40.5 | 0.9 | 6.0 | 3.3 | 3.6 |
| RHR | 16.8 | 9.8 | -16.0 | -3.0 | 6.1 | 5.6 | 3.9 | 3.9 | -1.3 | 15.5 | 14.1 | 11.5 | 4.9 | 6.6 | 2.3 | 2.9 |
| RHP | 74.5 | 66.4 | 21.7 | -38.8 | 6.5 | 2.8 | 1.0 | 5.1 | 48.0 | 57.0 | -5.8 | -34.1 | 1.7 | 6.2 | 6.4 | 0.1 |
| RKP | -78.4 | -45.0 | -94.7 | -79.7 | 0.3 | 0.1 | 1.0 | 4.7 | -76.9 | -53.3 | -97.3 | -25.2 | 5.2 | 5.3 | 6.8 | 6.9 |
| RFP | 30.9 | 31.1 | -38.0 | -7.6 | 6.8 | 0.6 | 2.6 | 2.5 | -52.7 | 26.8 | -16.2 | -45.7 | 7.1 | 7.1 | 7.1 | 1.9 |
| RFR | -17.8 | -1.1 | -10.3 | 4.8 | 4.2 | 6.1 | 1.0 | 7.1 | -12.2 | -13.0 | 14.1 | -6.0 | 4.8 | 6.5 | 0.3 | 6.3 |
| RTP | - | - | - | - | - | - | - | - | 20.3 | 44.6 | 6.8 | 37.1 | 3.4 | 1.8 | 2.0 | 4.9 |