Bipedal Walking Energy Minimization by Reinforcement Learning with Evolving Policy Parameterization

Petar Kormushev, Barkan Ugurlu, Sylvain Calinon, Nikolaos G. Tsagarakis, Darwin G. Caldwell

***Abstract*— We present a learning-based approach for minimizing the electric energy consumption during walking of a passively-compliant bipedal robot. The energy consumption is reduced by learning a varying-height center-of-mass trajectory which uses efficiently the robot's passive compliance. To do this, we propose a reinforcement learning method which evolves the policy parameterization dynamically during the learning process and thus manages to find better policies faster than by using fixed parameterization. The method is first tested on a function approximation task, and then applied to the humanoid robot COMAN where it achieves significant energy reduction.**

I. INTRODUCTION

Biological systems, for instance humans, store and release elastic potential energy into/from muscles and tendons during daily activities such as walking [1]. The management of the elastic potential energy that is stored in these biological structures is essential for reducing the energy consumption and for achieving mechanical power peaks. In this matter, vertical center of mass (CoM) movement appears to be a crucial factor in reducing the metabolic cost [2].

Recent advances in robotics and mechatronics have allowed for the creation of a new generation of passively-compliant robots, such as the humanoid robot COMAN (derived from the cCub bipedal robot [3]) shown in Fig. 1. Similar to biological systems, the springs in this robot can store and release energy, which can be extremely helpful if properly used. However, it is difficult to pre-engineer a proper way to utilize the passive compliance for dynamic and variable tasks, such as walking. Robot bipedal walking has been investigated many times before, but not much yet in the context of passive compliance. For a task such as walking energy minimization, one possible way to find a proper use of the passive compliance is via machine learning. In this paper, we present an approach that minimizes the walking energy by learning a varying-CoM-height walking which efficiently uses the passive compliance of the robot.

A. Related work

Various approaches exists for reducing the energy consumption of bipedal walking, such as [3]–[5], but not in the context of learning a varying-height walking, as in this paper. Machine learning has been successfully used before for learning tasks on bipedal robots, such as dynamic balancing, or even in tasks involving also the upper body such as learning to clean vertical surfaces [6]. One especially promising approach for autonomous robot learning is reinforcement learning (RL), as demonstrated in [7]–[11].

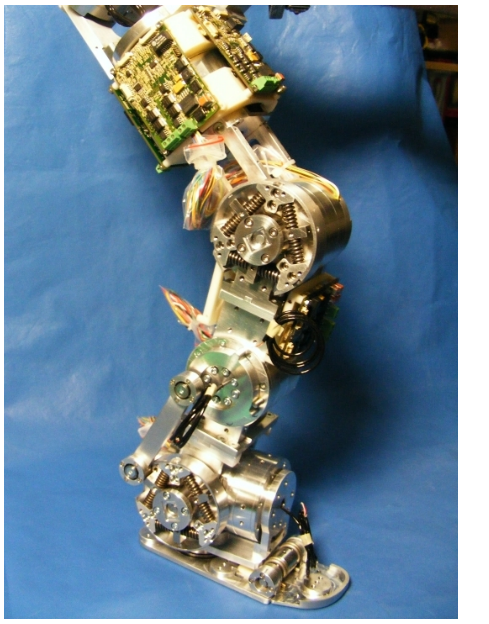


Fig. 1. The right leg of the COMAN robot (**CO**mpliant hu**MAN**oid robot), with uncovered springs which implement the passive compliance in the knee and ankle joints. There are a total of 28 springs in the legs: 6 in each compliant active joint (knee and ankle pitch) on each leg, and 2 for the passive joint at the toe of each foot.

Stulp *et al* presented in [8] a Policy Improvement with Path Integrals (PI^2) RL approach for variable impedance control, where both planned trajectories and gain schedules for each joint are optimized simultaneously. The approach is used to let the robot learn how to push and open a door by minimizing the average stiffness gains controlling the individual joints, with the aim to reduce energy consumption and to increase safety.

Kormushev *et al* presented in [9] the use of Expectation-Maximization-based RL for a pancake flipping task to refine the trajectory of the pan by learning a mixture of proportional-derivative systems with full stiffness matrices. RL in combination with regression yield extremely fast-converging algorithms, such as ARCHER, used by the iCub humanoid robot to quickly learn the skill of archery [10].

Rosenstein *et al* presented in [11] a simple random search approach to increase the payload capacity of a weightlifting

This work was supported by the AMARSi European project under contract FP7-ICT-248311.

P. Kormushev, S. Calinon, N. G. Tsagarakis and D. G. Caldwell are with Dept. of Advanced Robotics, Istituto Italiano di Tecnologia, via Morego 30, 16163 Genova, Italy. B. Ugurlu is with Toyota Technological Institute (TTI), 468-8511 Nagoya, Japan.{petar.kormushev, sylvain.calinon, nikos.tsagarakis, darwin.caldwell}@iit.it; barkanu@toyota-ti.ac.jp.

robot by exploiting the robot's intrinsic dynamics at a synergy level. Via-points are learned by exploration in a first phase of learning. RL and simple random search are then used to refine the joint coordination matrices initially defined as identity gains.

The work that we present in this paper shares a similar view that the above papers by providing the robot with exploration capabilities to optimize energy consumption while reproducing a desired task. We consider the challenging task of natural walking with compliant robot, which would take too many iterations to be learned only by imitation and reinforcement learning, but where a good compromise between state-of-the-art control approaches and optimization of policy parameters through RL is possible.

Adaptive resolution in state space has been studied in RL (see e.g. [12]). In [13], Moore *et al* employed a decision-tree partitioning of state-space and applies techniques from game-theory and computational geometry to efficiently and adaptively concentrate high resolution on critical areas. They address the pitfalls of discretization during reinforcement learning, and note that in high dimensions it is essential that learning does not plan uniformly over a state-space. However, in the context of RL, adaptive resolution in the policy parameterization remains largely unexplored so far.

Miyamoto *et al* presented in [14] an actor-critic reinforcement learning scheme with via-point trajectory representation for a cart-pole swing up task simulation. The actor incrementally generates via-points at a coarse time scale, while a trajectory generator transforms via-points to primitive action at the lower level.

Morimoto *et al* presented in [15] a walking gait learning approach in which via-points are detected from the observed walking trajectories, and RL modulates the via-points to optimize the walking pattern. The system is applied to a biped robot fixed to a boom that constrains the robot to the sagittal plane. Exploration tries to minimize the torques while keeping the robot above a desired height (i.e., not falling).

Wada and Sumita presented in [16] a via-points acquisition algorithm based on actor-critic reinforcement learning, where handwriting patterns are reproduced by iterative and sequential generation of short movements. The approach finds a set of via-points to mimic a reference trajectory by iterative learning using evaluation values of generated movement pattern.

Our approach differs from the above work by combining the efficiency of EM-based RL with incremental adaptive resolution in the policy parameterization to solve a complex real-world walking optimization task.

B. Novelty

In this paper we develop an integrated approach for learning how to minimize the energy required for walking of a passively-compliant bipedal robot. The energy minimization problem is challenging because it is nearly impossible to be solved analytically, due to the difficulty in modeling accurately the properties of the springs, the dynamics of the whole robot and various nonlinearities, such as stiction.

The novelty of the paper spreads in three directions: evolving policy parameterization, variable CoM-height walking, and passive compliance for reduction of energy consumption. First, we introduce a novel reinforcement learning technique which makes possible to use changeable policy parameterization over time. We call it evolving policy parameterization, and show one possible way to implement it using splines. Second, we develop a variable CoM-height ZMP-based walk generator and demonstrate various walking gaits on a bipedal robot. Third, we exploit the passive compliance built into our bipedal robot, in order to minimize the energy needed for walking, using the proposed reinforcement learning algorithm to find the optimal CoM trajectory which minimizes the consumed energy.

The proposed method is tested on the lower body of the compliant humanoid robot COMAN. The robot's legs have passive compliance (via springs) in the two pitch joints (knee and ankle) of each leg, as shown in Fig. 1.

II. PROPOSED APPROACH

We propose a RL method which learns optimal trajectory for the CoM of the robot to be used during walking, in order to minimize the energy consumption. The proposed approach consists of 3 interacting components: reinforcement learning, walk generation, and real-world rollout execution. Fig. 2 shows a high-level outline of the approach.

A. Reinforcement Learning

The conventional state-action based reinforcement learning approaches suffer severely from the curse of dimensionality. To overcome this problem, policy-based reinforcement learning approaches were developed, which instead of working in the huge state/action spaces, use a smaller policy space, which contains all possible policies representable with a certain choice of policy parameterization. Thus, the dimensionality is drastically reduced, and the convergence speed is much faster.

In order to find a good solution, i.e. a policy which produces a reward very close to the optimum/desired one, the policy parameterization has to be powerful enough to represent a big enough policy space, so that a good candidate solution is present in it. If the policy parameterization is very simple, with only a few parameters, then the convergence is quick, but often a sub-optimal solution is reached. If the policy parameterization is overly complex, the convergence is slow, and there is a higher possibility that the learning algorithm will converge to some local optimum, possibly much worse than the global optimum. The level of sophistication of the policy parameterization should be just the right amount, in order to provide both fast convergence and good enough solution.

Deciding what policy parameterization to use, and how simple/complex it should be, is a very difficult task, often performed via trial-and-error manually by the researchers. This additional overhead is usually not even mentioned in reinforcement learning papers, and falls into the category of "empirically tuned" parameters, together with the reward

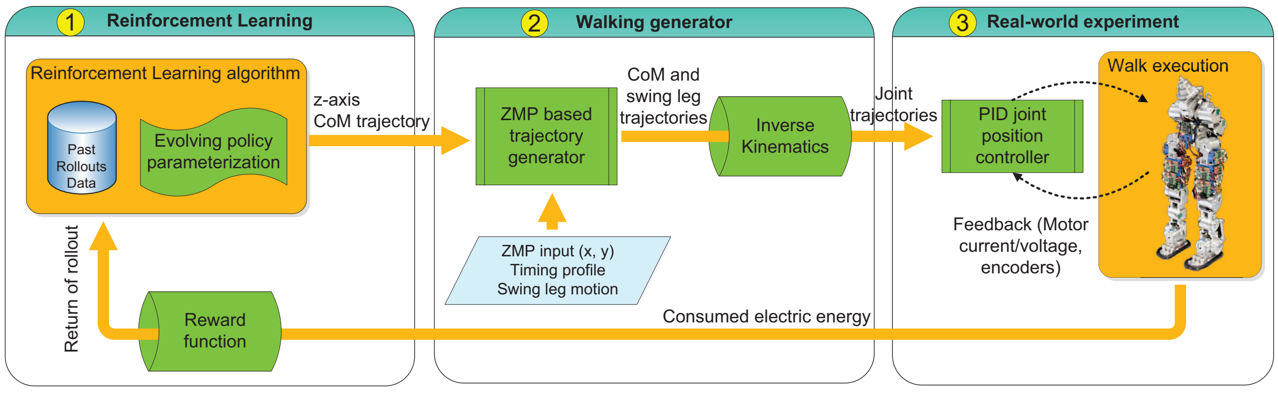


Fig. 2. Outline of the proposed approach, showing details about each of the three components: reinforcement learning, walk generation, and real-world rollout execution.

function, decay factor, exploration noise, weights, etc. Since changing the policy parameterization requires to restart the learning from scratch, throwing away all accumulated data, this process is slow and inefficient. As a consequence, the search for new solutions often cannot be done directly on a real-world robot system, and requires instead the use of simulations.

In this paper, we propose an approach that allows to change the complexity of the policy representation dynamically while the reinforcement learning is running, without losing any of the collected data, and without having to restart the learning. What we propose is a mechanism which can incrementally ”evolve” the policy parameterization as necessary, starting from a very simple parameterization and gradually increasing its complexity and thus, its representational power. The goal is to create an adaptive policy parameterization, which can automatically ”grow” to accommodate increasingly more complex policies and get closer to the global optimum. A very desirable side effect of this is that the tendency of converging to a sub-optimal solution will be reduced, because in the lower-dimensional representations this effect is less exhibited, and gradually increasing the complexity of the parameterization helps not to get caught in a poor local optimum.

The main difficulty to be solved is providing backward compatibility, i.e. how to design the subsequent policy representations in such a way, that they are backward-compatible with the previously collected data, such as past rollouts and their corresponding policies and rewards. In general, it is possible to consider cases in which simplifying the policy parameterization might be useful, but in this work we are going to assume that we only want to increase the complexity of the policy over time.

One of the simplest representations which have the property of backward compatibility, are the geometric splines. For example, if we have a cubic spline with K knots (or via-points), and then we increase the number of knots, we can still preserve the exact shape of the generated curve (trajectory) by the spline. In fact, if we put one additional knot between every two consecutive knots of the original spline, we end up with a $2K-1$ knots and a spline which coincides with the original spline. Based on this, the idea we propose is to use the spline knots as a policy parameterization, and use the spline backward compatibility property for evolving the policy parameterization without losing the previously collected data. In order to do this, we need to define an algorithm for evolving the parameterization from K to L knots ($L > K$), which is formulated in Algorithm 1. Without loss of generality, the values of the policy parameters are normalized in the range [0, 1], and appropriately scaled/shifted as necessary later upon use.

Algorithm 1 EvolvePolicy(current policy: $P_{current}$, desired new number of parameters: L)

```
1:  $K \leftarrow P_{current}.numberOfParameters$
2:  $X_{current} \leftarrow [0, \frac{1}{K-1}, \frac{2}{K-1}, ..., 1]$
3:  $Y_{current} \leftarrow P_{current}.parameterValues$
4:  $S_{current} \leftarrow$ ConstructSpline($X_{current}$, $Y_{current}$)
5:  $X_{new} \leftarrow [0, \frac{1}{L-1}, \frac{2}{L-1}, ..., 1]$
6:  $Y_{new} \leftarrow$ EvaluateSplineAtKnots($S_{current}$, $X_{new}$)
7:  $S_{new} \leftarrow$ ConstructSpline($X_{new}$, $Y_{new}$)
8:  $P_{new}.numberOfParameters \leftarrow L$
9:  $P_{new}.parameterValues \leftarrow S_{new}.Y_{new}$
10: return $P_{new}$
```

The proposed technique for evolving the policy parameterization can be used with any policy-based RL algorithm. In this paper, we use the EM-based RL algorithm PoWER [17], due to its low number of parameters that need tuning.

Fig.3 illustrates the process of using spline representation for the evolving policy parameterization. Fig.4 shows an example for a reinforcement learning process using evolving policy parameterization to approximate an unknown function (further explained in Section III-A).

B. Bipedal Walking Generator

To be able to generate real-time bipedal walking patterns, we adapted the resolution method explained in [18], using Thomas Algorithm [19]. Fig. 5 illustrates the one mass model and ZMP, both on x-z plane and y-z plane. In this figure,

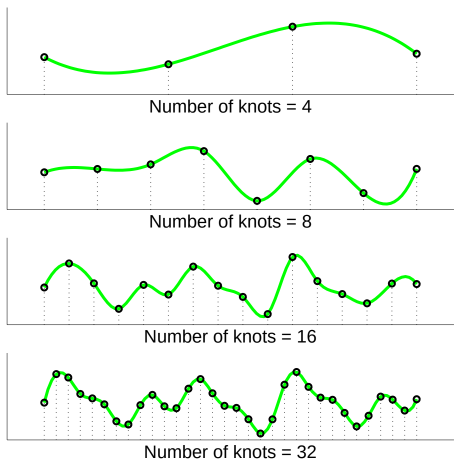


Fig. 3. An example for an evolving policy parameterization based on spline representation of the policy. The set of spline knots is the policy parameterization. The spline values at the knots are the actual policy parameter values. The parameterization starts from 4 knots and evolves up to 32 knots, thus gradually increasing the resolution of the policy.

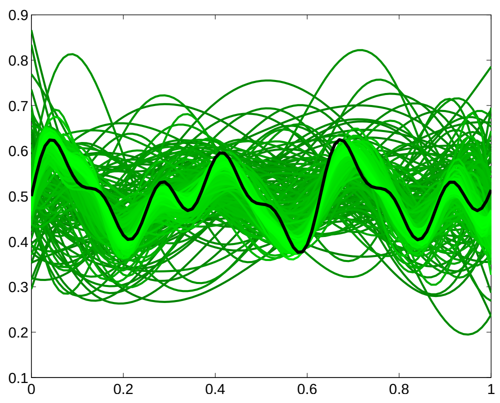


Fig. 4. An example for a reinforcement learning process using evolving policy parameterization. The black trajectory is the unknown global optimum which the RL algorithm is trying to approximate. The RL policy is encoded using a spline and parameterized by the spline knots. The policy evolves by gradually increasing the number of spline knots. Each green trajectory represents one intermediate policy during the learning process. The policy evolution is depicted by changing the color from dark green (for the older policies) to bright green (for the newer ones). The color gradient shows how the evolving policy gradually converges toward the global optimum.

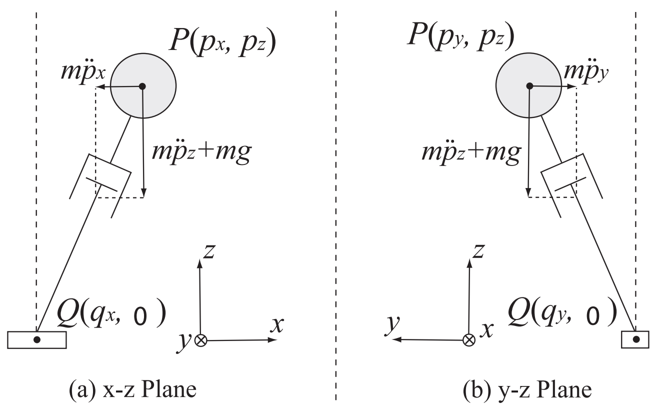


Fig. 5. One Mass Models on x-z and y-z planes. The walking direction coincides with the x axis. (a) Showing the sagittal plane during single support, which is the same for either the left or the right leg. (b) Showing the frontal plane during single support on the left leg.

$P = (p_x, p_y, p_z)$ is the CoM (Center of Mass) position while $Q = (q_x, q_y, 0)$ symbolizes ZMP position. For the purpose of simple bipedal walking planning, angular momentum rate changes are omitted. Therefore, the x-axis ZMP equation takes the form:

$$q_x = p_x - \frac{\ddot{p}_x}{\ddot{p}_z + g} p_z, \tag{1}$$

where g is the gravitational acceleration. The vertical CoM position (p_z) and acceleration ($\ddot{p}_z$) are provided by the learning algorithm for all times. As next step, (1) is discretized for p_x as follows:

$$\ddot{p}_x(t) = \frac{p_x(i+1) - 2p_x(i) + p_x(i-1)}{\Delta t^2}, \tag{2}$$

where Δt is the sampling period, i is the discrete event. i starts from 0 to n which is the total number of discrete events. Inserting (2) into (1), we may obtain the following:

$$p_x(i+1) = \frac{b(i)}{c(i)} p_x(i) - p_x(i-1) + \frac{q_x(i)}{c(i)} \tag{3}$$

$$b(i) = 1 - 2c(i); \qquad c(i) = -\frac{p_z(i)}{(\ddot{p}_z(i) + g)\Delta t^2} \tag{4}$$

In order to solve this tridiagonal equation efficiently, we employ Thomas Algorithm [19]. In this algorithm, we need to define initial and final position of x-axis CoM position, p_x. Therefore, for a given set of reference ZMP trajectory, initial conditions and final conditions, we are able to calculate CoM trajectory. For that purpose, we can re-arrange the tridiagonal equation as below:

$$p_x(i) = e(i+1)p_x(i+1) + f(i+1) \tag{5}$$

In (5), $e(i+1)$ and $f(i+1)$ can be defined as follows:

$$e(i+1) = -\frac{c(i)}{c(i)e(i) + b(i)} \tag{6}$$

$$f(i+1) = \frac{q_x(i) - c(i)f(i)}{c(i)e(i) + b(i)} \tag{7}$$

Combining (5), (6) and (7) we obtain (8):

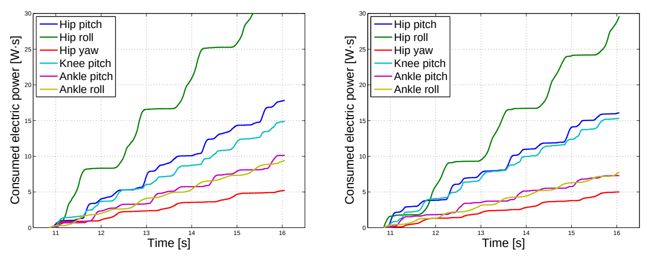
$$p_x(i) = -\frac{c(i)}{c(i)e(i) + b(i)} p_x(i+1) + \frac{q_x(i) - c(i)f(i)}{c(i)e(i) + b(i)} \tag{8}$$

Keeping in mind that $p_x(0) = x_0$ and $p_x(n) = x_n$, $e(1)$ and $f(1)$ are determined as 0 and x_0, respectively.

Utilizing the Thomas Algorithm for the solution of this tridiagonal equation, we can obtain the CoM trajectory's x-axis component. If an identical approach is also executed for y-axis CoM position, we could derive all the components of the CoM trajectory in real-time since vertical CoM position is previously determined.

C. Evaluation of Walking Energy Consumption

There are many ways in which energy can be measured. One possible approach is to estimate the mechanical energy from motor torque measurements and angular velocities. But the problem with this approach is that it incorrectly includes the work done by gravity, and can only infer indirectly



(a) Left leg (b) Right leg

Fig. 6. Electric energy consumption of each leg of COMAN during a 4-cycle walk (i.e. 8 steps).

TABLE I
SEQUENCE OF WALKING PHASES

No.	Phase description	Start time[s]	Duration[s]
1	Wait 1	0.00	1.00
2	Knee bend	1.00	1.00
3	Wait 2	2.00	5.00
4	Transfer phase (Double)	7.00	0.60
5	Right single	7.60	0.50
6	Double	8.10	0.15
7	Left single	8.25	0.50
8	Double	8.75	0.15
9	Right single	8.90	0.50
10	Double	9.40	0.15

the actual electric power used for walking. Also, electrical energy may be used by the motors even when the mechanical energy is zero, e.g. when there is no movement.

We propose, what we think is the best approach, to directly measure the electrical energy used by all the motors of the robot, which allows us to explicitly measure the value that we are trying to minimize. We use the formula $P = IU$, linking the electric power P to the electric current I and the voltage U, and we integrate over time to calculate the consumed electric energy in Joules. The COMAN robot is equipped with both current and voltage sensors at each motor, so we can accurately measure these values. Fig. 6 shows the accumulated consumed electric energy values for the motor of each individual joint of COMAN, calculated as:

$$E_j(t_1, t_2) = \int_{t_1}^{t_2} I_j(t) U_j(t)\, dt, \tag{9}$$

where j is a selected joint for which the energy consumption is calculated, and $[t_1, t_2]$ is the time interval.

For evaluation of a whole walking rollout, we define the energy consumption measure of a rollout τ to be the average electric energy consumed per walking cycle, and estimate it using the formula:

$$E(\tau) = \frac{1}{c} \sum_{j \in J} E_j(t_1, t_2), \tag{10}$$

where J is the set of joints in the sagittal plane (hip, knee, and ankle pitch of both legs, 6 in total) whose energy consumption we try to minimize.

In order to reduce the effect of noise on the measurement, for each rollout we make the robot walk for 16 seconds, and we take the electric current measurement of the $c = 4$ consecutive walk cycles (4 repetitions of phases 7 to 10 in Table I), which contain a total of 8 steps. Therefore, the value of t_1 is the start of phase 7, and the time t_2 is the end of phase 10 in the fourth cycle. Then, we average the energy consumption and use this value as the estimate of the electric energy used for this walking rollout.

Since the installed springs can only help to reduce the energy in the sagittal plane, in our evaluation metric we use the sum of all electric energy consumed by the motors controlling the motion in the sagittal plane, i.e. the hip, knee, and ankle pitch joints on both legs. Fig 6 shows that, in fact, the biggest amount of electric energy is used for the hip roll joint motor, because it has to sustain half of the body mass during the single support phase. This cannot be minimized without using an upper body to counter-balance the swing leg (and also because there are no springs in the hip roll joint), that is why we do not include the hip roll in the evaluation metric.

Finally, we define the return of a rollout τ as:

$$R(\tau) = e^{-kE(\tau)}, \tag{11}$$

where k is a scaling constant. The lower the energy consumed, the higher the reward is.

III. EXPERIMENTS

A. Simulation experiment

In order to evaluate the proposed reinforcement learning with evolving policy parameterization, we conduct an experiment in simulation. The goal is to compare the proposed method with a conventional fixed policy parameterization method that uses the same reinforcement learning algorithm as a baseline. The following synthetic function is used as the (unknown) goal for the optimization process:

$$\begin{split} \tilde{\tau}(t) = 0.5 + 0.2\sin(10t) + 0.07\sin(20t) + \\ + 0.04\sin(30t) + 0.04\sin(50t), \end{split} \tag{12}$$

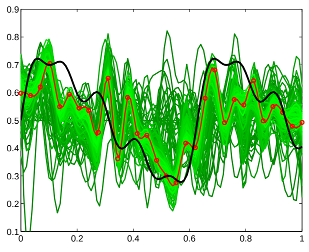
where $\tilde{\tau}(t)$ is a synthetic function which the learning algorithm is trying to approximate by minimizing the difference between it and the policy-generated trajectory, with domain $t \in [0, 1]$, and range $\tilde{\tau}(t) \in [0, 1]$.

The reward function used for the simulated experiment is defined as follows:

$$R(\tau) = e^{-\int_0^1 [\tau(t) - \tilde{\tau}(t)]^2 dt}, \tag{13}$$

where $R(\tau)$ is the return of a rollout (trajectory) τ; $\tilde{\tau}$ is the unknown (to the learning algorithm) function that the algorithm is trying to approximate.

Fig. 7 shows a comparison of the generated policy output produced by the proposed evolving policy parameterization method, compared with the output from the conventional fixed policy parameterization method. Fig. 8 shows that the convergence properties of the proposed method are significantly better than the conventional approach.


(a) Fixed policy parameterization

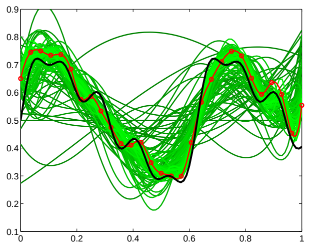

(b) Evolving policy parameterization

Fig. 7. Comparison of the policy output from RL with fixed policy parameterization (30-knot spline) versus evolving policy parameterization (from 4- to 30-knot spline). In black, the unknown to the algorithm global optimum which it is trying to approximate. In green, all the rollouts performed during the learning process. In red, the current locally-optimal discovered policy by each RL algorithm. Due to the lower policy-space dimensionality at the beginning, the evolving policy parameterization approaches much faster the shape of the globally-optimal trajectory. The fixed policy parameterization suffers from inefficient exploration due to the high dimensionality, as well as from overfitting problems, as seen by the high-frequency oscillations of the discovered policies.

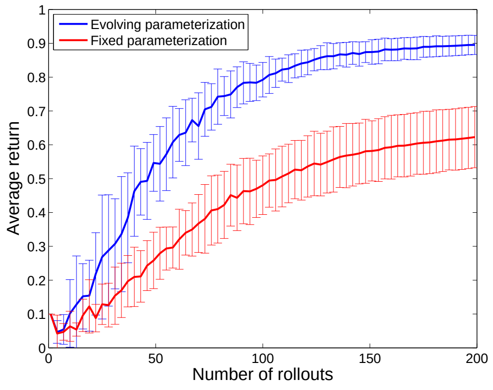


Fig. 8. Comparison of the convergence of the RL algorithm with fixed policy parameterization (30-knot spline) versus evolving policy parameterization (from 4- to 30-knot spline). The results are averaged over 20 runs of each of the two algorithms in simulation. The standard deviation is indicated with error bars. In addition to faster convergence and higher achieved rewards, the evolving policy parameterization also achieves lower variance compared to the fixed policy parameterization.

B. Real-world bipedal walking experiment

Based on the results from the simulation experiment, the proposed evolving policy parameterization method is chosen for the real-world walking experiment. The experimental setup is shown in Fig. 9.

We use the lower body of the passively-compliant humanoid robot COMAN that we developed earlier to explore compliant humanoid characteristics. The lower body has a total of 17 DoF (degrees of freedom): 6 active and 1 passive (toe joint) DoF in each leg, plus 3 active DoF at the waist. Each joint incorporates three position sensors (2 absolute and 1 relative) and one torque sensor. Fig. 10 shows the COMAN robot's lower body and its kinematic configuration. Fig. 1 shows the springs that implement the passive compliance of the legs.

Fig. 11 shows the convergence results from the walking experiments. Fig. 12 visualizes the 180 rollouts that were performed and the discovered optimal policy by the RL algorithm. A video of the experiment is available at [20].

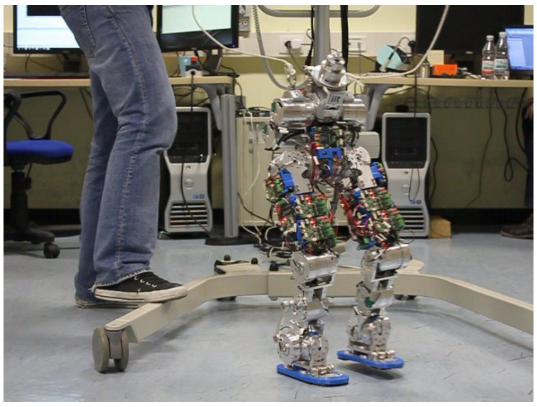


Fig. 9. The experimental setup, showing a snapshot of the COMAN robot's lower body during one walking rollout execution.

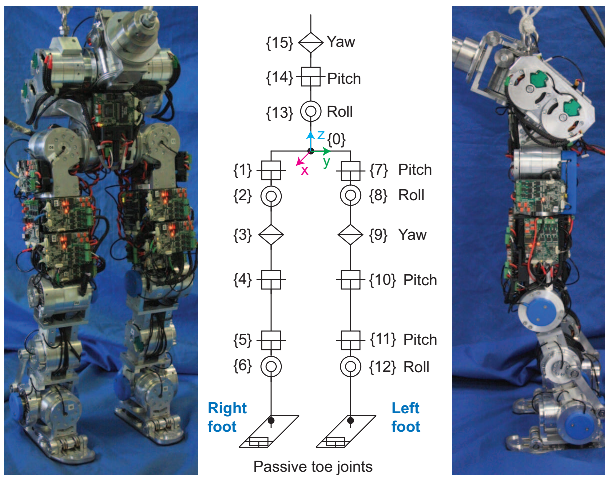


Fig. 10. Lower-body design of the compliant humanoid robot COMAN.

IV. DISCUSSION

The total distance traveled by the robot during our experiments is around 0.5 km. For the evaluation of the energy consumption, we did not include the traveled distance, as the speed of walking was the same for all rollouts, because the stride length was fixed. In the future, we plan to conduct other experiments with varying stride length.

The passive compliance proved to be extremely beneficial also to the stability of the walking. From a total of 180 walking gaits, less than 10 were unstable and resulted in undesirable oscillations. These rollouts were automatically demoted by the learning algorithm due to the low reward they generated.

With respect to the learning, the focus of the paper is not on the encoding scheme (spline), but on the evolving policy parameterization. Spline-based techniques have well-known limitations such as providing a non-autonomous (time-based) control policy, discarding variability and synergy information in the representation, and having difficulty to cope with unforeseen perturbations (see e.g. [21]). Being aware of their limitations, splines provided us with a simple encoding

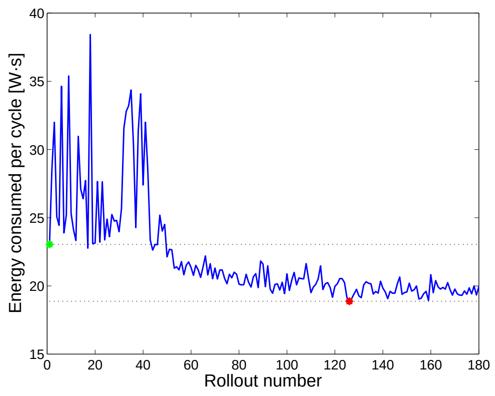


Fig. 11. Results from the real-world minimization of the consumed energy for walking. The figure shows the convergence of the consumed energy over time during the reinforcement learning. Each rollout corresponds to a walking experiment that was executed. For each rollout, the average energy consumed per cycle is shown (averaged over 8 walking steps, i.e. 4 full walk cycles), calculated from Eq. (10). The first value (green dot) is the baseline, which is the average energy consumption of the fixed-CoM-height walk. At rollout number 126 (red dot) the lowest energy consumption was achieved, which is 18% lower than the baseline.

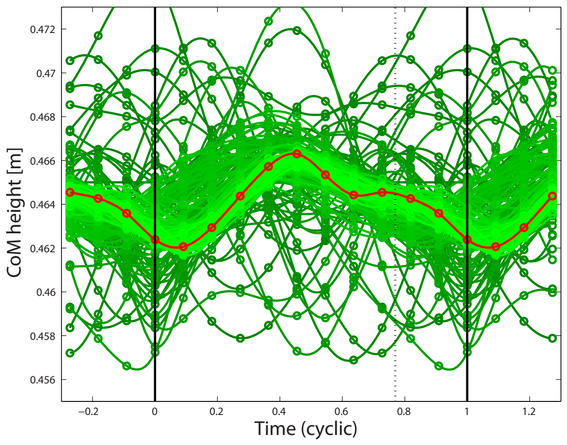


Fig. 12. The discovered optimal policy (in red) by the reinforcement learning, among all tried 180 CoM trajectories (in green), which were executed on the real robot. The RL algorithm discovered by itself the timings for single and double support phases, as well as the moment when the heel strikes the ground (shown with dotted vertical line), and adjusted the trajectory so that the CoM height is bounced off upward in that exact same moment. Note that this is the reference trajectory, which means that the actual trajectory will differ slightly due to the passive compliance of the robot. All trajectories have been made cyclic in time, so that walking can be executed continuously over many cycles.

scheme to be used as a first step to study the possibility to dynamically evolve the policy parameterization during the learning.

V. Conclusion

We proposed a reinforcement learning approach that can evolve the policy parameterization dynamically during the learning process. The gradually increasing representational power of the policy parameterization helps to find better policies faster than a fixed parameterization. We successfully applied it to a bipedal walking energy minimization task by developing a variable-CoM-height ZMP-based walk generator. The method achieved 18% reduction of energy consumption in the sagittal plane by learning to use efficiently the passive compliance of the robot.

References

[1] M. Ishikawa, V. Komi, M. J. Grey, V. Lepola, and P. Bruggemann, “Muscle-tendon interaction and elastic energy usage in human walking,” *Journal of Appl. Physiol.*, vol. 99, no. 2, pp. 603–608, 2005.

[2] J. D. Ortega and C. T. Farley, “Minimizing center of mass vertical movement increases metabolic cost in walking,” *Journal of Appl. Physiol.*, vol. 581, no. 9, pp. 2099–2107, 2005.

[3] B. Ugurlu, N. G. Tsagarakis, E. Spyrakos-Papastravridis, and D. G. Caldwell, “Compiant joint modification and real-time dynamic walking implementation on bipedal robot cCub,” in *IEEE Intl. Conf. on Mechatronics*, 2011.

[4] C. A. Amran, B. Ugurlu, and A. Kawamura, “Energy and torque efficient zmp-based bipedal walking with varying center of mass height,” in *IEEE Intl. Workshop on Advanced Motion Control*, 2010, pp. 408 – 413.

[5] M. Uemura, K. Kimura, and S. Kawamura, “Generation of energy saving motion for biped walking robot through resonance-based control method,” in *IROS*, 2009, pp. 2928 – 2933.

[6] P. Kormushev, D. N. Nenchev, S. Calinon, and D. G. Caldwell, “Upper-body kinesthetic teaching of a free-standing humanoid robot,” in *Proc. IEEE Intl Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 3970–3975.

[7] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal, “Skill learning and task outcome prediction for manipulation,” in *Intl Conf. on Robotics and Automation (ICRA)*, Shanghai, China, 2011.

[8] F. Stulp, J. Buchli, E. Theodorou, and S. Schaal, “Reinforcement learning of full-body humanoid motor skills,” in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Nashville, TN, USA, December 2010, pp. 405–410.

[9] P. Kormushev, S. Calinon, and D. G. Caldwell, “Robot motor skill coordination with EM-based reinforcement learning,” in *Proc. IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS)*, Taipei, Taiwan, October 2010, pp. 3232–3237.

[10] P. Kormushev, S. Calinon, R. Saegusa, and G. Metta, “Learning the skill of archery by a humanoid robot iCub,” in *Proc. IEEE Intl Conf. on Humanoid Robots (Humanoids)*, Nashville, TN, USA, December 2010, pp. 417–423.

[11] M. T. Rosenstein, A. G. Barto, and R. E. A. Van Emmerik, “Learning at the level of synergies for a robot weightlifter,” *Robotics and Autonomous Systems*, vol. 54, no. 8, pp. 706–717, 2006.

[12] A. Bernstein and N. Shimkin, “Adaptive-resolution reinforcement learning with polynomial exploration in deterministic domains,” *Machine Learning*, vol. 81, no. 3, pp. 359–397, 2010.

[13] A. W. Moore and C. G. Atkeson, “The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces,” *Machine Learning*, vol. 21, pp. 199–233, December 1995.

[14] H. Miyamoto, J. Morimoto, K. Doya, and M. Kawato, “Reinforcement learning with via-point representation,” *Neural Networks*, vol. 17, pp. 299–305, April 2004.

[15] J. Jun Morimoto and C. G. Atkeson, “Learning biped locomotion: Application of poincare-map-based reinforcement learning,” *IEEE Robotics and Automation Magazine*, vol. 14, no. 2, pp. 41–51, 2007.

[16] Y. Wada and K. Sumita, “A reinforcement learning scheme for acquisition of via-point representation of human motion,” in *Proc. of the IEEE Intl Conference on Neural Networks*, vol. 2, July 2004, pp. 1109–1114.

[17] J. Kober and J. Peters, “Policy search for motor primitives in robotics,” in *Advances in Neural Information Processing Systems*, 2009, vol. 21, pp. 849–856.

[18] S. Kagami, T. Kitagawa, K. Nishiwaki, T. Sugihara, T. Inaba, and H. Inoue, “A fast dynamically equilibrated walking trajectory generation method of humanoid robot,” *Autonomous Robots*, vol. 2, no. 1, pp. 71–82, 2002.

[19] B. Ugurlu, T. Hirabayashi, and A. Kawamura, “A unified control frame for stable bipedal walking,” in *IEEE Intl. Conf. on Industrial Electronics and Control*, Porto, Portugal, 2009, pp. 4167–4172.

[20] Video accompanying this paper. [Online]. Available: http://kormushev.com/research/videos/

[21] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural Networks*, vol. 21, no. 4, pp. 682–697, 2008.