The magmaOffenburg 2018 RoboCup 3D Simulation Team

Martin Baur, Kim Christmann, Klaus Dorer, Jens Fischer, Marcel Grüßinger, Carmen Schmider, David Weiler[1]

Hochschule Offenburg, Elektrotechnik-Informationstechnik, Germany

Abstract. In this TDP we describe our framework for learning new behaviors from scratch. It has been the core feature to advance from 5th to 2nd place in RoboCup 2017. The content is based on a paper presented at the RoboCup Symposium 2017 [1], but adds some new results achieved after publishing the paper.

1 Introduction

Learning to kick is common in many RoboCup leagues. A good overview can be found in [4]. However, none of these report on kicking with toed robots. MacAlpine et al. [5] use a layered learning approach to learn a set of behaviors in the RoboCup 3D soccer simulator. 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. This demonstrates the ability of their approach to generalize to different robot models, but is still model-based for walking. Abdolmaleki et al.[2] 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 2). 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 framework used in our team 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.

2 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.

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 [5] 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 where the motor does not follow the commanded speed exactly. 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) [3]. 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, ...)

3 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. In this case that means when 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 1. 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 resulting movement of the robot is shown in Figure 2. The robot learned to improve the kick considerably by stepping on the toes of its 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

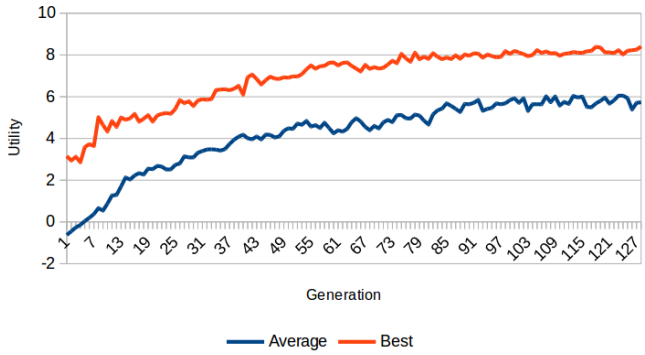

Fig. 1. Learning curve of plain genetic learning with NaoToe.

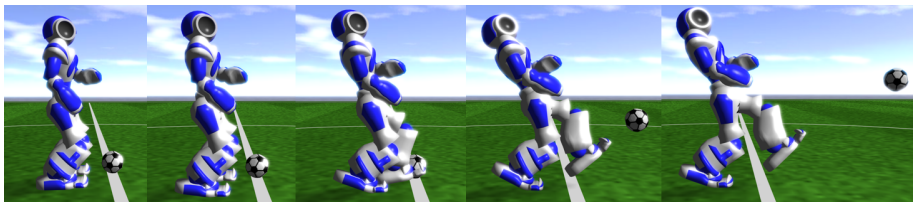

Fig. 2. Sequence of movement when kicking with NaoToe.

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 learning has been performed from a fixed starting position with respect to the ball. Figure 3 shows the dependency of the kick's success to the initial position of the player. As can be seen, a reasonable area is covered in which the kick can be performed. The same area will be covered symmetrically by a left leg kick. The x-axis and y-axis are the x and y coordinate of the player in the ball's coordinate system. Colors are utilities which map approximately to kick distance in meters. Distances to the ball of less than 0.16m result in the robot touching the ball before the kick. Each point on the heatmap is the average of 50 kicks from the corresponding starting position of the NaoToe.

Learning in the meantime showed that this model-free approach is able to learn other behaviors. With the same framework and by simply changing the utility function we learned to kick sidewards, backwards and diagonally. We also started to learn to kick while walking towards the ball.

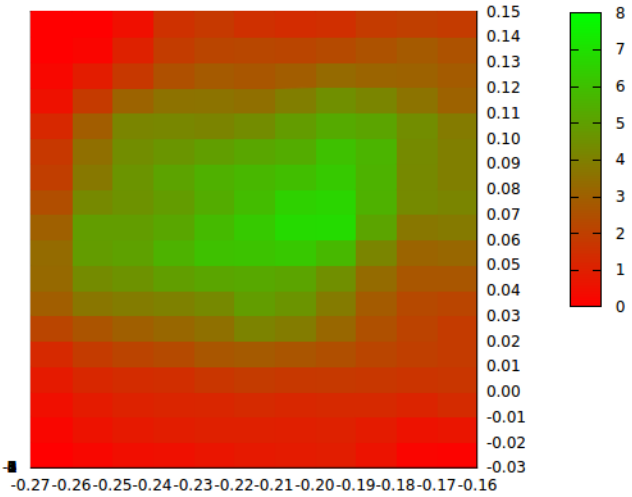

Fig. 3. Heatmap of the influence of the ball position on kick success.

When trying to learn to kick a far ball, the robot did not learn to stretch its leg accordingly, as we would have expected, but learned to walk another step closer to the ball before kicking. We knew that learning such a behavior is possible with our approach in principle, but were surprised that the algorithm is able to find such a solution in this huge search space.

A similarly unexpected behavior resulted from learning to kick a ball that is positioned to a far left (or right, 0.2m) position in front of the robot. Instead of learning to stretch its leg to the side, the robot learned to stop from full speed walking with a single step and then walked sideways to the ball. Both results show that there is huge potential in this approach of model-free learning of behaviors.

References

1. Dorer K (2017) Learning to Use Toes in a Humanoid Robot. Accepted for: RoboCup Symposium 2017, Nagoya, Japan.
2. 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.
3. 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.
4. Jouandeau N, Hugel V: *Optimization of Parametrised Kicking Motion for Humanoid Soccer Player*. In Autonomous Robot Systems and Competitions (ICARSC), 2014.
5. 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.