Real-Time Trajectory Generation by Offline Footstep Planning for a Humanoid Soccer Robot

Andreas Schmitz, Marcell Missura, and Sven Behnke

University of Bonn,
Computer Science VI, Autonomous Intelligent Systems
Friedrich-Ebert-Allee 144, 53113 Bonn, Germany
{schmitz4,missura,behnke}@cs.uni-bonn.de
http://ais.uni-bonn.de

Abstract. In recent years, humanoid soccer robots improved considerably. Elementary soccer skills, such as bipedal walking, visual perception, and collision avoidance have matured enough to provide for dynamic and exciting soccer games. While the elementary skills still remain hot research topics, it is time to move forward and address higher level skills, such as motion planning and team play. In this work, we present a new method to generate ball approach trajectories by planning footstep sequences offline and training an online policy to meet the real time requirements of embedded systems with low computational power, as typically used for soccer robots. We compare the results with our current reactive behavior that was used in the last RoboCup competitions and show the improvements we achieved.

Key words: humanoid robots, robot soccer, footstep planning, motion planning, trajectory generation

1 Introduction

Looking back at nearly a decade of history since the introduction of the Humanoid League to RoboCup in the year 2002, an impressive improvement of humanoid robot soccer can be observed. In the beginning, robots were barely able to walk and only penalty kick competitions were possible. Today, elementary soccer skills, such as bipedal walking, visual perception, and collision avoidance have matured enough to provide for dynamic and exciting soccer games played throughout the KidSize, TeenSize and Standard Platform leagues. Guidance and source code for the implementation of low level skills became freely available through a large number of scientific publications and even open source software released by some of the leading teams. While the elementary skills still remain hot research topics, it is time to move forward and address higher-level skills, such as motion planning and team play.

Typically, the control software of humanoid soccer robots is organized in multi-tiered architectures. On top of the fastest sensorimotor control loop, usually a motion layer is used to generate walking, kicking and get-up motions.

These skills are enforced by the qualification requirements for all participants as they are essential for playing soccer. The motions are controlled by a higher behavior layer that covers more complex actions, such as approaching the ball, avoiding obstacles, dribbling the ball towards the goal and aligning for a kick. In the implementation of all leading teams these behaviors are pure reactive [1–3] in the sense that performed actions are direct results of the current sensory input without contemplation on possible future states of the environment. It is remarkable how well robots can already play without planning into the future. We believe it is time to investigate how planning can be incorporated to produce smoother and more intelligent motion trajectories during games to improve the overall performance of soccer robots.

In this work, we demonstrate how ball approach trajectories can be generated by footstep planning with the A* algorithm. As motion planning is a computationally expensive task, we use a set of precalculated footstep sequences to distill an online policy for obstacle-free situations on the soccer field. The recall of the policy is fast, just like a reactive behavior, but it produces trajectories that are implicitly planned into the future.

The remainder of this paper is organized as follows. After reviewing related work, we will outline our implementation of the A*-based footstep planning algorithm. In Section 4, we explain how we used our footstep planning algorithm to precalculate planned trajectories offline for a grid of situations and how we used standard machine learning concepts to learn an online policy that reproduces the planned trajectories. Finally, we integrate the online policy as a ball approach module into our soccer software and compare its performance with our reactive behavior from the last competitions.

2 Related Work

Footstep planning is a fairly new research topic. The most prominent proposals in [4–6] and also [7] are based on the A* algorithm. By imposing a strong discretization on the state space and using only a small, discrete set of actions, these online solutions plan a few steps ahead and are able to deal with dynamic environments. Uneven floor plans are also considered, so that the footstep plans can include stepping over obstacles and climbing stairs. An intriguing alternative solution has been recently shown in [8]. Here, a short sequence of future footsteps is considered to be a virtual kinematic chain as an extension of the robot. Their location is determined by inverse kinematics. The configuration space and the action space are not discretized, but the algorithm is computationally expensive. A computationally more promising method that can plan in a few milliseconds, if the environment is not too cluttered, has been suggested in [9]. The idea is to solve the footstep planning problem mostly with a path planning algorithm. Actual footstep locations are only given in key points, where the walking speed of the robot has to be zero, for example when stepping over an obstacle. The majority of the footstep locations are laid out along the planned paths by the motion generator developed for HRP-2 [10–12]. The closest related work is [6], where an A*-based footstep planning algorithm was adapted for the humanoid robot ASIMO. As the walking algorithm of ASIMO was not precisely known, the authors were forced to reverse engineer a footstep prediction algorithm from observations with a motion capture system.

Another approach to trajectory planning is the Dynamic Window algorithm [13], but it only plans a small amount of time into the future and therefore it cannot produce optimal trajectories, can get stuck in a local optimum and is likely to produce oscillating behaviors, just like any reactive algorithm.

Since the trajectories described by human walkers appear to have a strong resemblance with those of non-holonomic vehicles [14], global path planning methods that determine a geometrical path that adheres to continuous curvature and minimal turning radius restrictions, such that in principle it can be followed by a vehicle with a steering wheel [15] [16] are also potential candidates to be used for humanoid robot motion planning. But this setting is purely kinematic and ignores important physical aspects of motion planning: velocity and acceleration.

In full-fledged kinodynamic planning [17], the state of a moving object includes not only the Cartesian coordinates and the orientation, but also the translational and angular velocities. Planning is mostly performed directly in control space where velocity and acceleration bounds can inherently be taken into account. The increase in dimensionality makes discrete cell decomposition methods impractical. Thus, randomized approaches [18] are often used that sample possible control inputs and project them into the state space by numerical integration. This way, the open-loop controls to execute a calculated trajectory arise naturally. The avoidance of moving obstacles is also possible, as demonstrated in [19], by extending the state space by the time dimension. The computation times reported by the works cited above do not yet allow the application of kinodynamic planning on embedded systems.

The significant difference between our approach and the approaches discussed above is that we do not attempt online execution of a planning algorithm. Instead, we plan near-optimal solutions for a large number of situations offline and learn a fast-to-evaluate policy. This approach is similar to Plan-Then-Compile architectures [20] in the aspect that the first action of a planned action sequence is “compiled” to a stimulus response of a reactive behavior.

3 Footstep Planing

The foundation of our concept is an implementation of the A* algorithm taylored to the task of planning footsteps for a humanoid robot to reach a goal state by placing its foot on a given target footstep location. In the robot soccer domain this target is in most cases a position behind the ball suitable for kicking. We define the state $s = (s_l, s_v) \in S = L \times V$ in a six dimensional state space S that contains the set of all Cartesian left foot poses $L = \mathbb{R} \times \mathbb{R} \times [-\pi, \pi]$ and velocity vectors $V = [-1, 1]^3$, which satisfy the velocity constraints of the robot. The gait velocity vector describes fractions of the maximum allowed velocities in the sagittal, lateral and rotational directions, respectively. The velocity has a strong influence on the step location of the next state, because the step size of the robot grows with higher velocities. We are able to map the gait velocity vector to a step location in Cartesian space with a motion capture-based transformation that we published last year [21]. To limit the branching factor, we defined only a small discrete set of five actions $A \subset [-1, 1]^3$ as feasible accelerations of the lateral velocity to the left or to the right, the rotational velocity clockwise or counter clockwise, and the sagittal velocity only forward, each with the maximum allowed value as configured for a specific robot. We did not need a sagittal deceleration, because after every action the velocity limits of the robot are enforced and accelerating the lateral or rotational components automatically leads to a decrease of the sagittal velocity. An action is always executed twice, such that after a right-left double step the robot is again in a valid state standing on the left foot. The state transformation $s = t(s, a)$ is given by

$$s_l = m(m(s_l, s_v + a), s_v + 2a) \tag{1}$$

$$s_v = s_v + 2a, \tag{2}$$

where s_l is the Cartesian location of the left foot, s_v is the current velocity of the robot, $a \in A$ is the action to perform, and $m(l, v)$ is the gait control vector to step location transformation as described in [21]. We chose the double step as a unit action to cut the depth of the search tree in half and allow the A* algorithm to find results in shorter time, but as we will show later on, this does not necessarily mean that the robot has to reach the goal with the left foot. Please note that since the state includes the current velocity of the robot and the actions are accelerations, we are performing kinodynamic planning that takes the dynamic properties of the robot into account.

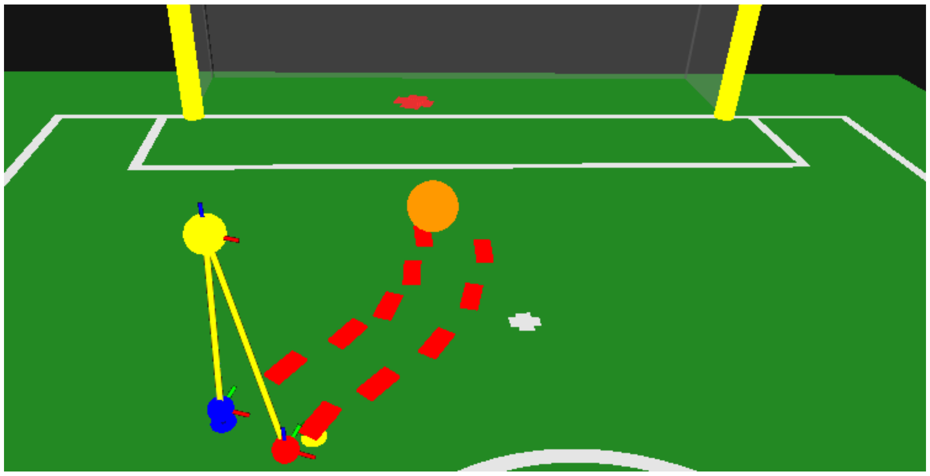

Fig. 1. Visualization of a planned footstep sequence that leads the robot to the ball aiming at the center of the yellow goal.

The goal is to reach the set of target states $T \subseteq S$ that are contained by a ball with a radius of 0.1 around the target state s^*

$$T = \{s \in S, \|s - s^*\| \leq 0.1\}. \tag{3}$$

The cost function for every state is simply the number of steps that had to be taken so far. As a heuristic $h(s)$ we used the Euclidean distance on the ground plane between the state s and the target state s^* divided by the largest possible step size d.

$$h(s) = \frac{\|(s_x, s_y) - (s_x^*, s_y^*)\|}{d}. \tag{4}$$

The A* algorithm defined above is able to find nearly optimal dynamic ball approach sequences. An example is illustrated in Figure 1. The problem with this approach is that in many cases its runtime exceeds several minutes of computation and can require gigabytes of memory to keep track of open states. Clearly, it is not suitable to be run on embedded systems with low computational power along with the real time requirements of a soccer game. Thus, we only use it to compute a large set of trajectories offline that we use as training examples to learn a policy.

4 Policy Learning

Using the A* algorithm described in the previous section, we calculate the footstep plans for a set of start states defined as follows. The ball is located in the origin of the coordinate system and the y-axis points towards the target direction (the goal), as illustrated in Figure 2. The target velocity at the ball is set to the maximum forward speed of the robot. We distribute the start states

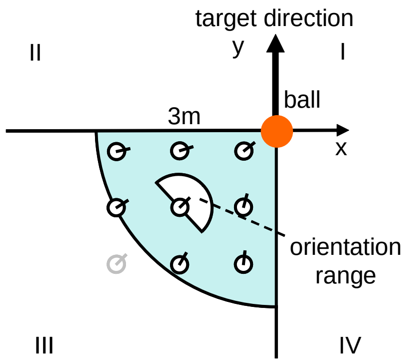

Fig. 2. The domain of start positions. The coordinates of the start positions are distributed in a grid in the third quadrant, from which a circular area with a radius of three meters is selected. Each of these positions has an orientation in the range $\pm\pi$ centered around the direct line to the ball.

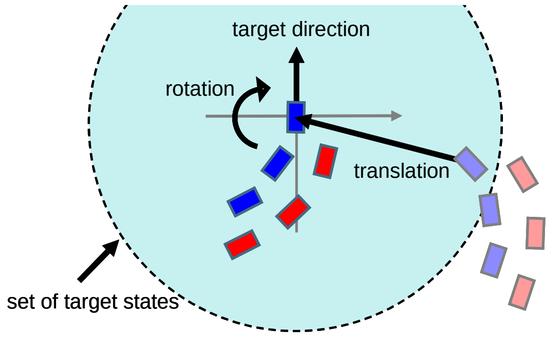

Fig. 3. The last step of precalculated paths does not hit the target exactly. A transformation can correct these inaccuracies. The size of the set of target states is strongly exaggerated in this illustration.

in the third quadrant in a six-dimensional grid layout with ten intersections in each dimension. From these we select only those with a distance no greater than three meters to the ball. The orientations are taken from a range of $\pm\pi$ centered around the direct line to the ball. Altogether we have somewhat less than 10^6 start states due to the selection of the circular area. Any situation can be mirrored between the third and the fourth quadrant, so the fourth quadrant does not need to be calculated explicitly. The first and the second quadrant are excluded from our considerations.

With 24 CPU cores, we managed to calculate approximately 17,000 paths randomly sampled from the defined grid in two days time totaling approximately 400,000 single steps. In post processing we corrected the error A* makes by applying a translation and a rotation to each of the entire footstep sequences such that the final footstep precisely hits the target. This is depicted in Figure 3. Applying a mirroring technique, we also produced example footstep plans that finish with the right foot instead of the left foot and doubled the size of the training set. The idea of designing the policy is that no more than the next single step is needed to be known at any time. After the robot has executed the step, it will evaluate the policy again and obtain a new step. By successive recall of the policy, the same footstep sequence will be reproduced that was planned with the A* algorithm requiring only very little computational power. The policy $\pi : a(s) \in A$ we aim to learn is a mapping from a six dimensional state space to a three dimensional action space. Here the state s is expressed in a reference frame centered on the target. Consequently, we do not need to deal with entire footstep sequences, but every single step can be used as a training example. A convenient property of the method we used to calculate the training data is that since all A* paths lead to the target, the density of the steps increases automatically in the vicinity of the ball. Since the training set contains both, the robot is standing on the right foot and the robot is standing on the left foot states, we can now discard the double step restriction that we introduced to accelerate the A* algorithm. In fact we distinguish four different situations. The robot is standing on the right foot and wants to hit the ball with the right foot denoted as RR, the robot is standing on the left foot and wants to hit the ball with the right foot denoted as LR, and respectively RL and LL situations. For each of the four combinations we train a separate policy. Please note that the A* precalculation produced only RL and LL results, but in post processing we generated training examples for RR and LR situations as well. The choice whether the robot wants to hit the ball with the left foot or the right foot can be made once at the beginning of a ball approach either with a heuristic or by calculating the expected number of steps for both cases using the learned policy and deciding for the lesser.

To perform the actual learning task, we divided the training data into appropriate sets for the RR, LR, RL, and LL policies, each of them containing 200,000 examples. We evaluated three different machine learning approaches: k-nearest neighbor interpolation, piecewise linear approximation and multi-layer perceptrons. First we tried k-nearest neighbor interpolation by retrieving the 200 nearest neighbors for a query point, fitting a six-dimensional hyperplane with a least squares method into the point cloud of the neighbors for each of the three output dimensions. The hyperplanes are evaluated at the location of the query. We found that this method produced smooth step sequences that were able to hit the ball reliably in simulation. However, the policy recall times were not satisfactory due to the overhead of the nearest neighbor retrieval and the hyperplane fitting with each query. Therefore we transformed the data set to a regular grid by evaluating each grid node with the aforementioned k-nearest neighbor technique. In a regular grid the nearest neighbors can be found instantly and no hyperplane fitting is needed, since we can just interpolate between the corners of a grid cell (a six-dimensional grid cell has 64 corners). Using the regular grid we achieved efficient policy recall times without sacrificing precision, but it still has some undesirable memory requirements. We used a grid with 11 intersections in each of the six dimensions resulting in 80 MB of data in double precision. As a third alternative we trained-multi-layer perceptrons. We used a separate network for each of the three output dimensions with one output neuron, 20 sigmoid neurons in the hidden layer and six input neurons. The networks learned the desired functions with a precision comparable to the other two approaches. The evaluation of a network is even faster than the piecewise linear approximation and the memory requirements for storing the network weights are minimal. We present two types of accuracy measures in Figures 4 and 5. The single step accuracy in Figure 4 shows the mean error and standard deviation of the gait velocities produced by the learned policies compared to the corrected output of the A* algorithm. The errors were determined using a test set of approximately 10,000 examples that were not used for training. The single step errors do not accumulate over an entire path, because the policy recall after each step has an error correcting property. Most important is the accuracy at the target point at the end of the entire footstep plan. This is shown in Figure 5. The policy recall times are shown in Figure 6. In light of the low hardware requirements of the neural network we decided to use it for all subsequent experiments.

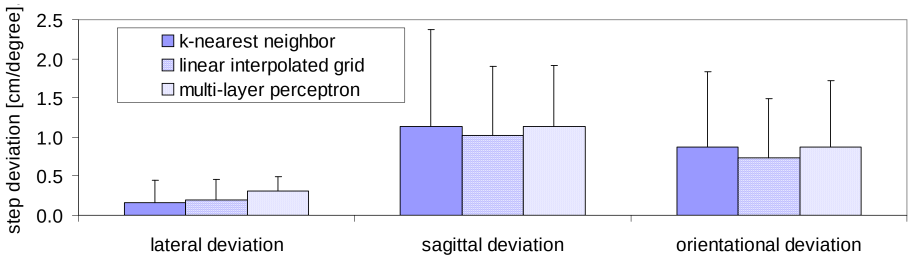

Fig. 4. Evaluation of the single step accuracy of the three machine learning methods. The bar chart represents the mean error and standard deviation of the gait velocities produced by the learned policies compared to the output of the A* algorithm.

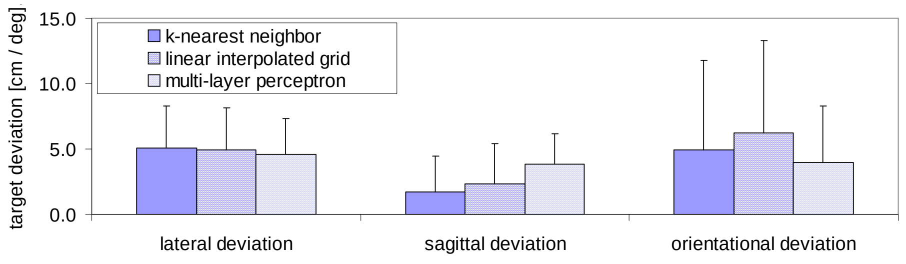

Fig. 5. Evaluation of the path accuracy. Mean errors and standard deviations are measured after the last step of a successive policy recall for an entire footstep plan.

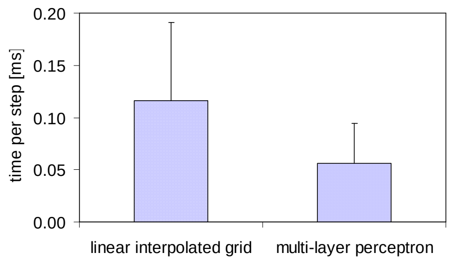

Fig. 6. Policy recall times for a single step when using the regular grid (left) or the multi-layer perceptron (right). The k-nearest neighbor method is omitted from this comparison, because its recall times are much higher. The times were measured on an Intel Core 2 Duo T8300 2.4GHz CPU with Windows 7.

5 Experimental Results

We integrated the trained policies into our soccer software and performed an experiment with a real robot. We compared the performance of the new trained footstep policies with our reactive dribbling behavior that played an important role in our success of winning the TeenSize competitions last year. The same robot "Dynaped" was used to evaluate both ball approach strategies. The experimental setup is illustrated in Figure 7.

Starting from the poses numbered from one through five, dribbling ball approaches were performed with both methods such that the robot would hit the ball located in one of the positions a, b or c with maximum possible speed. The target direction is the center of the goal to the left. Each start pose and ball position combination was repeated seven or eight times, so that at least 100 ball approaches were performed with each, the footstep policy and the reactive behavior. The robot and the ball were equipped with reflective markers and we recorded every approach with a motion capture device. Using the motion capture data we were able to reconstruct the ball approach trajectories and identified the footstep locations of single steps. For a qualitative comparison we determined figures such as the number of steps taken, the ball velocity, and the rolling direction of the ball. The footstep policy and the reactive ball approach are compared in accuracy of meeting the target angle, the velocity of the ball after contact and the number of steps taken by the robot to reach the ball. Numbers are presented in Figure 8. These figures are not easy to compare, because they trade off properties of a ball approach. For example, ball velocity can be gained by sacrificing precision. The angle precision of the footstep policy was in average five degrees better, than the reactive behavior. The ball velocity could not be improved. However, approximately the same ball velocity was reached with a higher precision. The average number of steps taken was also improved by the footstep policy and it is worth taking a closer look at Figure 9, where the number of steps is depicted in detail for each start pose and ball position combination. In easy situations, where the robot basically only has to walk straight forward against the ball, the reactive behavior performs just as well. In hard cases, however, the footstep policy outperforms the reactive behavior by five or six steps. There are also cases, where the reactive behavior just walks straight

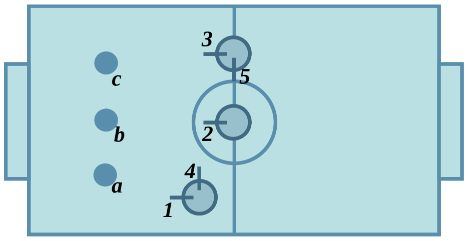

Fig. 7. Arrangement of the robot start poses [1..5] and ball positions [a,b,c].

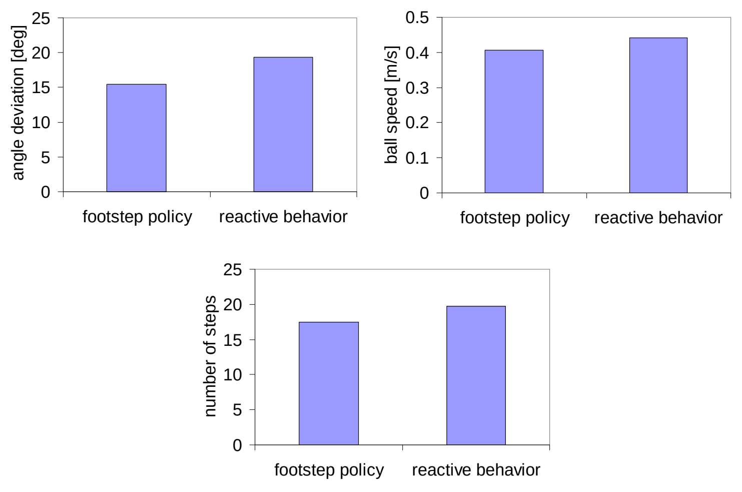

Fig. 8. Comparison of the average angle deviation (a), ball velocity (b), and number of steps (c) of the footstep policy and the reactive behavior.

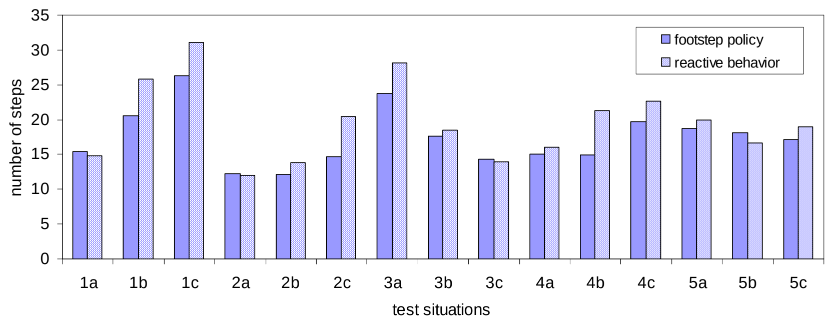

Fig. 9. Comparison of the average number of steps used by the reactive behavior and the footstep policy in the different test situations.

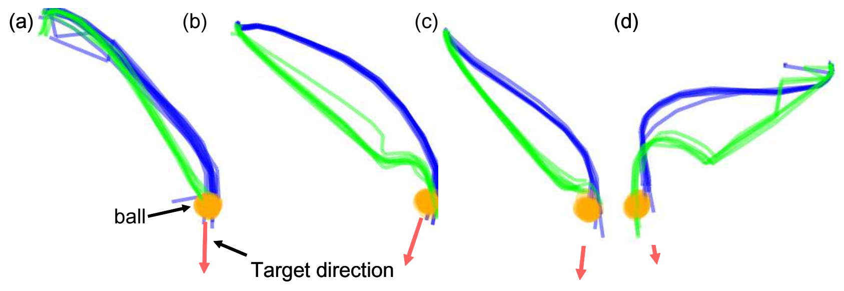

Fig. 10. Reconstructed ball approach trajectories of the reactive behavior (green / light) and the footstep policy (blue / dark).

into the ball without care for the target angle, as shown in Figure 10 part (a). In these cases the footstep policy has difficulties to approach the ball with the same number of steps while giving the target angle a higher priority.

In Figure 10 some of the reconstructed ball approach trajectories are shown. In case (a) the reactive behavior takes only a small amount of steps and moves the ball with a high velocity, but only at the cost of ignoring the target direction. In cases (b), (c), and (d) it can be clearly seen how the reactive behavior heads for the ball first and then tries to align to the goal while the footstep policy produces smooth approach trajectories.

6 Conclusions

We have presented a ball approach trajectory generation method by footstep planning. The plans were calculated offline using the A* algorithm for a set of predefined start situations. The resulting steps were used to train a policy that predicts only the next step. Successive evaluation of the policy reproduces the planned trajectories implicitly. With this method, we were able to outperform our reactive ball approach behavior while maintaining low computational and memory requirements. The current footstep policy does not avoid obstacles. In future work, we are planning to implement obstacle avoiding ball approaches by injecting via points from a higher layer. Another possibility is to include an obstacle in the training data and training the footstep policy to take obstacles into account.

References

1. Thomas Röfer and Tim Laue and Judith Müller et.al. B-Human Team Report and Code Release 2010, `http://www.b-human.de/file_download/33/bhuman10_coderelease.pdf`
2. M. Friedmann, J. Kiener, S. Petters, H. Sakamoto, D. Thomas and O. von Stryk, *Versatile, high-quality motions and behavior control of humanoid soccer robots*, In Workshop on Humanoid Soccer Robots of the 2006 Humanoids, 2006, pp. 9-16
3. Sven Behnke and Jörg Stückler, *Hierarchical Reactive Control for Humanoid Soccer Robots* International Journal of Humanoid Robots (IJHR), vol. 5, no. 3, pp. 375-396, September 2008.
4. J. Chestnutt and J. Kuffner, *A Tiered Planning Strategy for Biped Navigation*, In Humanoids, 2004
5. J. Kuffner, S. Kagami, K. Nishiwaki, M. Inaba and H. Inoue, *Online Footstep Planning for Humanoid Robots*, In ICRA, 2003, pp. 932-937
6. J. Chestnutt, M. Lau, K.M. Cheung, J. Kuffner, J.K. Hodgins and T. Kanade, *Footstep Planning for the Honda ASIMO Humanoid*, In ICRA, 2005
7. J.S. Gutmann, M. Fukuchi and M. Fujita, *Real-time path planning for humanoid robot navigation*, In Proc. of 19th Int. Conf. on Artificial intelligence, 2005, pp. 1232-1237
8. O. Kanoun, E. Yoshida and J.P. Laumond, *An Optimization Formulation for Footsteps Planning*, In Humanoids, 2009

9. Y. Ayaz, T. Owa, T. Tsujita, A. Konno, K. Munawar and M. Uchiyama, *Footstep Planning for Humanoid Robots Among Obstacles of Various Types*, In Humanoids, 2009
10. S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada and K. Yokoi, *Biped walking pattern generation by using preview control of zero-moment point*, 2003, In Proc. of the International Conference on Robotics and Automation, 2003, pp. 1620-1626
11. S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi and H. Hirukawa, *The 3D linear inverted pendulum mode: a simple modeling for a bipedwalking pattern generation*, In IROS, 2001, Vol. 1, pp. 239-246
12. K. Kaneko and F. Kanehiro and S. Kajita and H. Hirukawa and T. Kawasaki and M. Hirata and K. Akachi and T. Isozumi, *Humanoid Robot HRP-2*, In ICRA, 2004, pp.1083-1090
13. D. Fox, W. Burgard, S. Thrun, *The Dynamic Window Approach to Collision Avoidance*, IEEE Robotics and Automation Magazine, 4(1):23-33, 1997
14. J.-P. Laumond, G. Arechavaleta, T. V. A. Truong, H. Hicheur, Q. C. Pham, A. *Berthoz, The words of the human locomotion*, In Proceedings of 13th international symposium on robotics research (ISRR-2007), Berlin: Springer
15. F. Lamiraux and J.-P. Laumond, *Smooth motion planning for car-like vehicles*, IEEE Transactions on Robotics and Automation, vol. 17, 2001, pp. 498-502.
16. A. Scheuer, Th. Fraichard, *Collision-Free and Continuous-Curvature Path Planning for Car-Like Robots*, Proc. IEEE Int Conf. on Robotics & Automation, 1997, pp. 867-873.
17. B. Donald, P. Xavier, J. Canny, J. Reif, *Kinodynamic Motion Planning*, J. ACM, vol. 40, 1993, pp. 1048-1066.
18. S. M. LaValle, J. J. Kuffner Jr., *Randomized Kinodynamic Planning*, I. J. Robotic Res., vol. 20, 2001, pp. 378-400.
19. D. Hsu, R. Kindel, Robert, J.-C. Latombe, S. Rock, *Randomized Kinodynamic Motion Planning with Moving Obstacles*, The International Journal of Robotics Research, vol. 21, 2002, pp. 233-255.
20. Tom M. Mitchell *Becoming Increasingly Reactive*, In Proceedings of the eighth National conference on Artificial Intelligence - Volume 2
21. A. Schmitz, M. Missura, S. Behnke, *Learning Footstep Prediction from Motion Capture* In Proceedings of RoboCup International Symposium, Singapore, June 2010.