DeepQ Stepper: A framework for reactive dynamic walking on uneven terrain

Avadesh Meduri[1], Majid Khadiv[2] and Ludovic Righetti[1,2]

***Abstract*— Reactive stepping and push recovery for biped robots is often restricted to flat terrains because of the difficulty in computing capture regions for nonlinear dynamic models. In this paper, we address this limitation by proposing a novel 3D reactive stepper, the DeepQ stepper, that can approximately learn the 3D capture regions of both simplified and full robot dynamic models using reinforcement learning, which can then be used to find optimal steps. The stepper can take into account the entire dynamics of the robot, ignored in most reactive steppers, leading to a significant improvement in performance. The DeepQ stepper can handle nonconvex terrain with obstacles, walk on restricted surfaces like stepping stones while tracking different velocities, and recover from external disturbances for a constant low computational cost.**

I. INTRODUCTION

The development of fast contact and motion planning algorithms has been a topic of interest in robotics for many years. Fast contact planning allows a legged robot to quickly make or break contacts with its environment to move to desired locations, recover from an external disturbance or react to a change in its surroundings. Fast motion planning enables robots to rapidly move their body to reach a desired contact configuration while remaining balanced.

Most reactive motion planning algorithms, i.e. used for real-time trajectory generation, are based on simplified models such as the Linear Inverted Pendulum Model (LIPM) [1]. The LIPM along with predefined footstep locations was used to make a humanoid robot walk in [2], [3]. Since footstep locations are computed beforehand, these approaches cannot handle external disturbances that require the robot to adapt its footsteps to stabilize itself.

To resolve this limitation, two important concepts, the instantaneous capture point (ICP) [4] and capture region [5], were introduced to decide step locations for recovering from external disturbances. An equivalent concept to the ICP, called the divergent component of motion (DCM), was proposed in parallel to generate walking motions on flat ground [6] and was extended to the non planar ground in [7]. In [8], the notion of DCM offset was introduced to generate walking patterns with desired velocities while adapting both step location and timing online. However, the approach, based on the LIPM, is restricted to flat ground walking.

This work was supported by New York University, the European Union's Horizon 2020 research and innovation program (grant agreement 780684 and European Research Councils grant 637935) and the National Science Foundation (grants 1825993 and 1925079).

[1]Tandon School of Engineering, New York University (NYU), USA. am9789@nyu.edu, ludovic.righetti@nyu.edu

[2]Max-Planck Institute for Intelligent Systems, Tuebingen, Germany. majid.khadiv@tuebingen.mpg.de

Contact planning algorithms that can handle more complex terrains are computationally expensive. In [9], [10], the full-body motion and contact selection problems are formulated as single nonlinear optimization problems. While they can in principle find complex contact sequences, these approaches are computationally too expensive to be used in realtime. In [11], a slightly more efficient phase-based formulation of contact planning is proposed. In [12], mixed integer programming is used to formulate the contact planning problem. The optimizer only considers kinematic constraints to generate a feasible contact plan and the computation time grows exponentially with the number of terrains and steps. [13] extends this approach by introducing centroidal dynamics so that dynamically feasible contact sequences and trajectories can be generated together. [14] generates dynamically feasible contact sequences using a variation of the A^* algorithm with a learned contact feasibility classifier on a centroidal dynamics planner [13]. Dynamic feasibility constraints are also considered in [15] along with learned approximations of the centroidal dynamics to reduce the computation time of the algorithm. Despite the reduced computation time and a certain level of generality of these contact planning algorithms, they are still not quick enough to be used for real-time control.

Given the LIPM assumption, to be able to walk without falling, it is necessary and sufficient to step within the (infinite-step) capturable region [8], [5]. Unfortunately, extending this result to more complicated scenarios including the robot nonlinear dynamics, uneven terrains or non-convex stepping regions is generally not feasible because of the intractability in computing the capture region in such cases.

In this paper, we propose a novel reactive stepping approach, the DeepQ stepper, that approximates 3D capture regions for the full robot dynamics using deep reinforcement learning. We formulate the problem of choosing step locations as a continuous-state discrete-action Markov Decision Problem (MDP), affording the use of efficient Deep Q-Learning algorithm [16]. The learned state-action value function implicitly encodes an approximation of the capture region which is then used to plan footsteps online, irrespective of terrain complexity, and recover from external disturbance, while tracking the desired velocity at a constant computational cost. Empirical evidence suggests that the DeepQ stepper learns a good approximation of both 2D & 3D capture regions even for nonlinear dynamic models. Extensive walking simulations with a biped robot with point feet demonstrate the advantages of our approach, which takes into account the full robot dynamics, including swing foot

dynamics, acceleration limits, or joint friction.

II. Background

In this section, we briefly provide the necessary background to introduce the DeepQ stepper.

A. Nonlinear Inverted Pendulum

The nonlinear inverted pendulum dynamics can be derived from the Newton-Euler equations

$$m(\ddot{c} - g) = \sum_{i=1}^{n} F_i, \quad \dot{L} = \sum_{i=1}^{n} (u_i - c) \times F_i \tag{1}$$

where m is the robot mass, c is its CoM location, g is the gravity vector, F_i is the ground reaction force applied on leg i, L is the angular momentum at the CoM and u_i is the foot location (assuming a point contact foot). If we assume that there is no angular momentum around the CoM and only one foot is in contact with the ground at a given instant, the above equations simplify to the nonlinear pendulum dynamics

$$c^{x,y} - \frac{c^z}{\ddot{c}^z + g^z}(\ddot{c}^{x,y} - g^{x,y}) = u^{x,y} \tag{2}$$

$$m(\ddot{c}^z + g^z) = F^z \tag{3}$$

The linear inverted pendulum dynamics [1] is recovered when $\ddot{c}^z = 0$.

B. Capturability and Viability Regions

For a walking robot, the N-step capturable region corresponds to the set of footsteps that would enable the robot to come to a stop after at most N steps. Computing the capture region for a general robot model is generally infeasible. However, the infinite step capture region for the linear inverted pendulum dynamics can be computed analytically [5]. The DCM offset is defined as the distance between the next footstep and the DCM location at the time of the next step. By ensuring that the DCM offset at each step remains in a certain bound [8], it is possible to track the desired COM velocity while remaining capturable.

C. Reinforcement Learning

In this paper, we use the Deep Q Network (DQN) algorithm, a very successful deep reinforcement learning algorithm for continuous state space and discrete actions MDPs [16]. It uses a deep neural network (DNN) to approximate the Q-function and one DNN called the target network which closely follows the Q-function DNN [17] to stabilize the training. The learned Q-function is then used to compute the optimal action for a given state.

In our implementation, we smoothly update the target network weights at each iteration $\theta^- = \tau\theta + (1-\tau)\theta^-$ as in [17], where $\tau \in (0, 1)$. Note that in this work, we minimize a cost as opposed to maximizing a reward function.

III. Approach: the DeepQ Stepper

In this section, we describe our approach to learn an efficient footstep planner along with the trajectory planner and controller used with the robot.

A. DeepQ Stepper

The main idea of the DeepQ Stepper is to learn a Q function associated with each feasible stepping action for a given robot state, where the Q value represents the capability of the robot to either bring itself to rest or track the desired velocity after taking that step. For a given state, the set of possible steps with low Q values will provide an approximation of the capture region. As we use a model-free approach to learn the Q-function, it can directly be used to approximate the capture region of the full robot dynamics.

1) Learning the value of a step: During training, we will use either the (nonlinear) inverted pendulum dynamics (IPM) or the full robot dynamics as the model of the robot. We discretize the possible stepping locations in the x, y direction, such that we can use the DQN approach. The step height (z-direction) is also provided to the DQN (continuous action) during uneven terrain walking and is determined by the terrain at the given step location. At the beginning of each step, the model is allowed to take an action, which corresponds to selecting one of the possible step lengths. The state of the agent at a given time step consists of the distance of the CoM from the current foot location in each direction, the velocity of the CoM in the x and y directions, an index determining which foot is on the ground ($+1$ or -1 if right or left foot is on the ground respectively) and the desired velocity in the x and y directions. That is $x = [c_x - u_x, c_y - u_y, c_z - u_z, \dot{c}_x, \dot{c}_y, n, v_x^{des}, v_y^{des}]$.

The goal is to step so as to track a desired walking velocity, while trying to be as close as possible to the hip at the beginning of the next step and choosing as small step lengths as possible without falling down. The episode terminates after n steps or if the robot falls down before the n steps. The agent is considered to have fallen down if the kinematic constraint (maximum allowed leg length) is violated. We use the following cost function

$$\begin{aligned} c(x, a) = w_1(|h^x - u^x| + |h^y - u^y|) + w_2(|\dot{c}_x - v_x^{des}| \\ + |\dot{c}_x - v_x^{des}|) + w_3(|a_x| + |a_y| + |a_z|) + I(x) \end{aligned}$$

where w_1, w_2, w_3 are weights, $c_{x,y,z}$ is the location of the CoM at the start of the next step, $h_{x,y,z}$ is the hip location of the leg whose foot is on the ground. $u_{x,y,z} = u^0_{x,y,z} + a_{x,y,z}$ is the foot location at the start of the next step, $u^0_{x,y,z}$ is the foot location at the start of the current step, $v^{des}_{x,y,z}$ is the desired velocity, $a_{x,y,z}$ is the chosen action at the start of the step and $I(x)$ is an indicator function that returns a constant value (here 100) if the kinematic constraint is violated otherwise it returns zero.

The DeepQ stepper architecture is designed to accept the state of the agent at the starting of each step along with an action (step length in the 3 directions) and return the corresponding Q-value. The optimal policy for the agent is to choose the action with the lowest Q-value.

The DeepQ stepper is trained using the DQN algorithm described in the previous section with $w_1 = 0.5$, $w_2 = 3.0$, $w_3 = 1.5$. To accelerate learning, we bias the exploration at the beginning of learning. We use a LIPM-based stepper to

select actions 80 % of the time to first fill the replay buffer. After the desired buffer size is reached, we use an $\epsilon-$greedy strategy with $\epsilon = 0.2$. When sampling the mini-batch, we take 20 % of the mini-batch to be the latest state-action pairs and the rest is sampled uniformly from the replay buffer. In our experiments, this significantly improved the rate of convergence of the algorithm especially because the rewards are sparse for the problem.

2) The DeepQ Stepper as a reactive foot-step planner: During execution, the state of the agent is computed at the beginning of each step and provided to the trained DeepQ stepper. We compute the optimal step length (lowest Q-value) only using admissible actions given the current allowable stepping regions, which is possible due to the discrete-action space formulation. Consequently, a constant number of network evaluations are needed for the stepper independent of the stepping region, which provides a significant computational advantage. The agent then takes the optimal action at the end of the step. This process is repeated indefinitely to generate walking motions on different terrain.

B. Trajectory generation and control approach

We generate CoM trajectories using the nonlinear inverted pendulum dynamics, Eqs. (2) and (3) given the choice of the foot step locations. This approach is used during both learning and evaluation. Initially, a trajectory optimization problem is solved for the dynamics governing the z direction independently, such that the height of the inverted pendulum moves from its current height to the desired height by the end of the trajectory (duration of one step) while also reaching a zero velocity [18], [19]. The QP problem solved for the z direction trajectory is

$$\begin{aligned} &\min . && \sum_{i=0}^{T} wF_t^2 \\ s.t. \quad & c^z_{t+1} = c^z_t + \delta t \dot{c}^z_t, \quad \dot{c}^z_{t+1} = \dot{c}^z_t + \delta t(F_t/m - g) \\ & c^z_0 = h_0, \quad \dot{c}^z_0 = 0, \quad c^z_T = h_T, \quad \dot{c}^z_T = 0 \end{aligned}$$

where c^z_t is the height at the time step t. h_0, h_T are initial and final height of the inverted pendulum and T is the number of collocation points in the trajectory optimization problem. The z trajectory obtained from the motion planner is then plugged into inverted pendulum Eq. (2) to integrate the x and y dynamics of the system to obtain a dynamically feasible CoM trajectory.

This method of trajectory planning is adopted to bypass the nonlinearity in the dynamics and subsequently generate feasible trajectories very efficiently. This makes it possible to replan CoM trajectories while walking on uneven ground in the presence of external forces on the robot in real-time. Further, in the case where the height of the inverted pendulum is kept a constant, the motion plans from the solver coincide with the linear inverted pendulum, which will facilitate our experimental comparisons later. Note that any other trajectory generator [20] could also be used with the DeepQ stepper since it's a model-free approach.

Swing foot trajectories are generated for the robot at the beginning of each step after obtaining the desired next step location from the DeepQ stepper. The end-effector trajectory starts from the current location of the foot and ends at the next foot location. A 5th-degree polynomial is used to parameterize the foot trajectory. The trajectories generated by the CoM planner and the swing foot trajectory generator are tracked using the whole-body controller proposed in [21].

IV. Experiments

We now present extensive simulation results to investigate the capabilities of the approach. In the first set of experiments, we evaluate the DeepQ stepper to control the simplest dynamic model, a 1D LIPM. This enables its analysis in light of exact results for the capturability region [5]. Then, we evaluate the approach to generate walking for a 3D nonlinear inverted pendulum model. Finally, we demonstrate the ability of the approach to learn footstep planning using the full dynamics of a biped robot with point feet. In particular, we show that the stepper is capable of learning a better approximation of the capture region that more adequately reflects the true capabilities of the robot. This leads to an improvement in walking performance when compared to typical LIPM-based approaches.

A. Experimental Setup

For the simulations, we use Bolt (Fig. 4), a newly designed open source biped robot [22], as we eventually aim to implement our approach on the real robot. Bolt is a torque-controlled biped robot with 6 active DoFs [21], [22] capable of very dynamic walking. It has passive ankles which we model with point feet. Each leg is 0.4m long, its base is 0.13m wide and 0.078m high. It easily can step every 0.2 seconds [22]. All the modeling choices such as step time or nominal CoM height described below are based on the model of the real robot and its capabilities.

The DeepQ stepper is trained using the procedure discussed in section III-A. The DNN of the DeepQ stepper contains 7 layers each containing 512 neurons followed by a final layer with one neuron. All layers except the last one are activated with a ReLU function. The stepper is trained with a learning rate of 10^{-4}, $\tau = 10^{-3}$, and a buffer size of 8000. Each episode during training is terminated after 10 steps or when the agent falls down. The DNN architecture and training parameters are kept the same throughout all the experiments. All the training instances converged consistently for all the experiments we conducted.

For all the experiments, we used a Dell precision 5820 tower machine with a 3.7 GHz Intel Xeon processor. The accompanying video[1] illustrates the simulation results.

B. 1D LIPM

A 1D LIPM model is used to initially test the performance of the DeepQ stepper. The model has its CoM at a constant height of 0.35 m above the ground and is allowed to take a step every 0.2 seconds. The state of the agent is $x =$

[1] https://www.youtube.com/watch?v=aITooZlm-WY

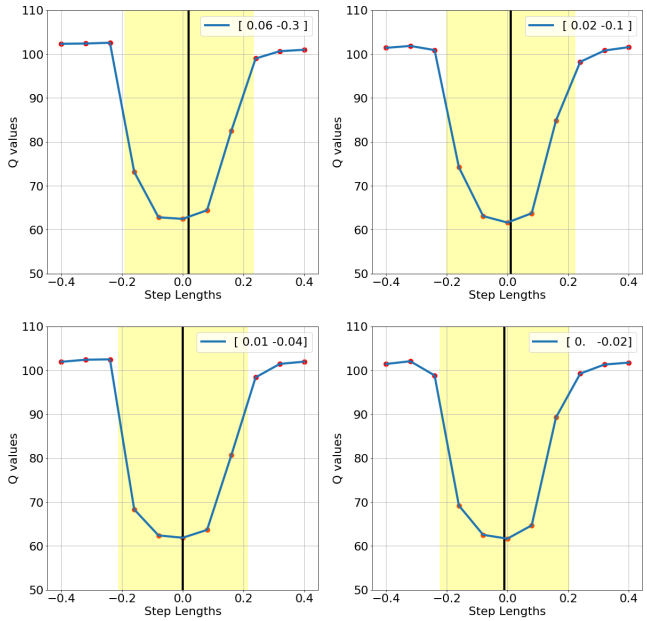

Fig. 1: Action values along with the location of the ICP/DCM (vertical black line) and the infinite step capture region (yellow region) for a 1D LIPM in different states (blue label).

$[c_x - u_x, \dot{c_x}]$. We discretize the action space into 11 equally spaced step lengths starting from -0.4 m to 0.4 m (i.e. an 8 cm discretization step).

The whole training process of the DeepQ stepper takes about 15 to 20 minutes (approximately 5000 episodes). A visualization of the Q values for different states of the agent at the start of the step is shown in Fig. 1. The location of the instantaneous capture point (ICP) at the end of the corresponding step along with the infinite step capture region is also shown. We notice that the optimal stepping strategy found by our approach is close to the location of the ICP for different states, which is the optimal stepping strategy to stop, for the LIPM model. Further, the value assigned to each action by the learned Q-function is such that the ones with the lower values lie within the capture region while those with higher values lie outside the capture region. This suggests that the Q-function has implicitly learned the capture region. During execution, the value of each step can be evaluated to select the next best possible step to take, to continue walking, in case the optimal action cannot be chosen (i.e. by choosing the step with the lowest value that is also feasible in the current environment).

C. 3D Inverted Pendulum

We now use our approach with a nonlinear inverted pendulum model (IPM) (Sec. II-A), for which no analytic capture region characterization exists, to learn to walk. The model contains a 0.13m-width base and a maximum leg length is 0.4m (same as the real robot), which is used as a kinematic constraint in the model to terminate the episode when violated. The model is expected to keep its CoM at a nominal height of 0.35 m above the ground at the start of each step and it follows the trajectory generated

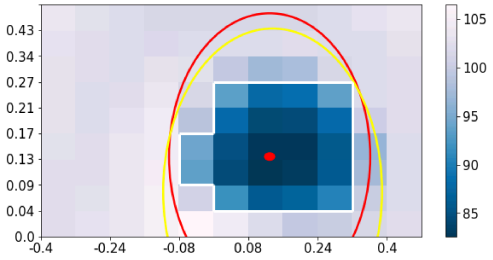

Fig. 2: A visualization of the action values while walking on flat ground. The infinite step capture region for the model at the end of the step is depicted by the red circle. The red dot shows the location of the ICP/DCM at the end of the step. The yellow circle represents the kinematic constraints at the end of the step.

by the inverted pendulum motion planner (Sec. III-B). The model takes a step every 0.2 seconds and its state is the same as described in Sec. III-A. The action space now consists of 99 step lengths with 11 equally spaced step lengths in the forward direction (x axis) between $\pm 0.4m$ and 9 geometrically spaced (geometric progression) step lengths in the lateral direction (y-axis) starting from $0m$ to $0.43m$. Since the minimum step length allowed in the lateral direction is zero, the model is not allowed to step to the right of the right foot and the left of the left foot. This reduces the possibility of foot collisions. However, the DeepQ stepper can be trained with an action space that allows for crossing of the feet, since the stepper can handle nonconvex reachable spaces efficiently.

The DeepQ stepper takes approximately 6000 episodes to learn to walk without falling while tracking the desired velocity ranging from $\pm 0.7m/s$. After learning, the stepper can recover or balance without falling when an episode is initialized with velocities ranging from $\pm 1.0m/s$.

A visualization of the Q values for a state = [0, 0.065, 0.35, 0.16, -0.16, 1, 0, 0] (The agent has a velocity of 0.16 m/s and -0.16 m/s in the x and y direction respectively, while the CoM is at a height of 0.35 m above the ground and 0.065 m to the left of the right foot) at the start of the step, on flat ground, along with the theoretical infinite step capture region, ICP/DCM and kinematic constraints for the model at the end of the step are shown in Fig. 2. Stepping on flat ground is shown here because the dynamics match the LIPM, which enables comparison with theoretical results. Darker squares correspond to actions with lower cost (the darkest square is the optimal action according to the Q-function). The white box encloses the steps from which the robot can recover (determined empirically). That is if the IPM is forced to take any step within the white box (which need not be the best action) and after which it is allowed to take optimal actions, the DeepQ stepper can bring its CoM to rest without falling, i.e. is captured. This shows that the DeepQ stepper not only learns an optimal location to step for a given state (that is close to the ICP/DCM), but it is also able to learn an approximation of the true capture region for the nonlinear inverted pendulum dynamics.

During training, the IPM is also exposed to scenarios

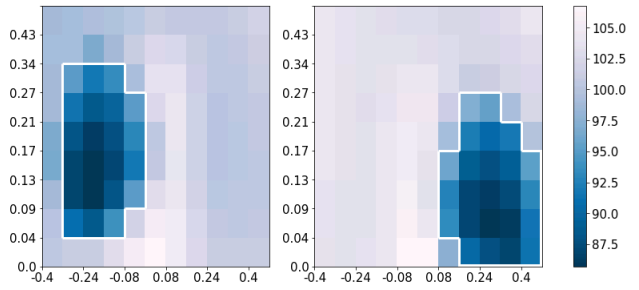

Fig. 3: Heatmaps representing learned 3D capture region for two different states. The left and right figures are when the agent is to step on a terrain that is 0.05 m below and 0.08 m above the current location of the foot respectively.

.

where the terrain varies randomly with heights ranging from $\pm 0.07m$ (20 % of the leg height of the robot) so that the DeepQ stepper approximately learns the 3D capture region, which is the region in 3D space the IPM can step to remain capturable. Using the learned 3D capture region, the robot can navigate complicated terrain by accounting for terrain height while choosing optimal actions. Figure 3 shows a visualization of the learned 3D capture region for two different scenarios. Since the infinite step capture region for a nonlinear inverted pendulum cannot be computed analytically, it is not possible to compare the learned capture regions with the ground truth. However, we verified that the robot can walk on uneven terrain when steps within the white box of the heatmaps are taken, as in the case of 2D stepping discussed previously. It shows that the Q-function can be used to choose among a set of possible steps, at no additional cost, therefore approximating a 3D capture region and enabling footstep selection on complex terrains.

D. 3D walking with Bolt Robot

Now we train the DeepQ stepper directly with a simulation of the complete robot to demonstrate that our algorithm can take into account its full dynamics such as inertia of the base, friction in the joints, swing foot dynamics, possible tracking errors in the controller, etc. All of which are ignored in the nonlinear inverted pendulum model.

1) Comparison between the IPM and full-robot DeepQ stepper: We compare the stepper learned with the 3D IPM with the stepper learned in the full robot simulation to control the full robot. Our goal is to show the improvement in footstep selection when taking into account the complete robot dynamics. The 3D IPM stepper was not able to generate robust walking gaits with the real robot dynamics, due to significant effects of the swing leg. A similar issue appeared when a LIPM-based reactive planner, that ignores the swing foot dynamics, was transferred to the real robot [22]. We, therefore, set a simulation where we reduced the mass of the legs to 10% of the mass of the real robot to increase similarity with the LIPM dynamics.

The 3D IPM DeepQ stepper can generate walking gaits while tracking velocities from ± 0.7 m/s on flat terrain and bring the robot to rest (capturable) from velocities up to 0.8 m/s without falling down.

The DeepQ stepper trained directly in the simulation environment with the modified Bolt robot (lighter legs) takes approximately 6000 episodes to be able to generate robust walking gaits of up to $\pm 1m/s$. The robot is controlled with the same control framework described above.

Fig. 5 shows a comparison between the learned capture regions of the DeepQ stepper trained directly in simulation and with the inverted pendulum environment for the same state. The white box in the two heat maps encloses the step locations from which Bolt can recover by taking optimal actions after, when it starts from the same initial condition, and uses the two DeepQ steppers in simulation. As can be seen, the DeepQ stepper trained in simulation is able to learn a larger capture region as compared to the inverted pendulum stepper. Also, the orange box encloses actions that the IPM stepper believes are good actions to take, but in reality, these actions lie outside the kinematic limits and can not be taken by the robot. This discrepancy arises because the rate of divergence of the CoM of the robot is different from the simplified IPM for the same state. Subsequently, actions that are viable with the IPM for the same state, are not feasible on the robot. In contrast, the simulation stepper returns high Q values for actions that lie outside the kinematic region, since it learns to step using the full robot dynamics.

Figure 6 shows a comparison of episode costs over 50 episodes in the simulation environment. During each episode, the robot is initialized with random conditions (different CoM states, desired tracking velocities, and uneven terrain), and both controllers are executed for 15 steps. The simulation stepper shows a better performance about 77% of the time (lower episode cost) when tested over 1000 episodes. The IPM stepper often fails to capture the robot in scenarios with high CoM velocities ($\geq$ 0.8 m/s) where the swing foot dynamics becomes prominent because this requires taking large steps to recover. Consequently, the stepper does not account for this and chooses steps that are longer than necessary which destabilizes the robot. On the other hand, the simulation DeepQ stepper is able to account for these dynamics and learn a richer representation of the capture region (the Q values are quite different within the white box, as one moves away from the optimal action), where it prefers to take smaller steps whenever possible to improve the stability of the robot and reduce the effects of the swing foot dynamics and still recover from high initial CoM velocities. In addition, the simulation DeepQ stepper is more robust while walking on uneven terrain and is able to track desired velocities better as compared to the IPM stepper. This can also be seen in Fig 6 where the bullet stepper incurs lower episode cost in episodes where both the steppers can generate walking without falling down (episodes with cost under 100).

2) DeepQ stepper with robot dynamics: The DeepQ stepper is now trained with real robot dynamics. During training, the simulation environment is initialized with a sparsely generated terrain consisting of random step heights up to 0.05

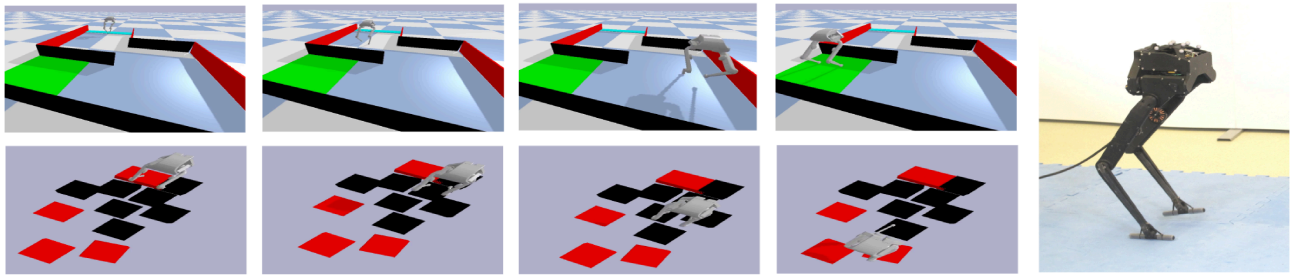

Fig. 4: Bolt navigating challenging environments with obstacles and stepping stones. Sequence of motion ordered starting from left to right.

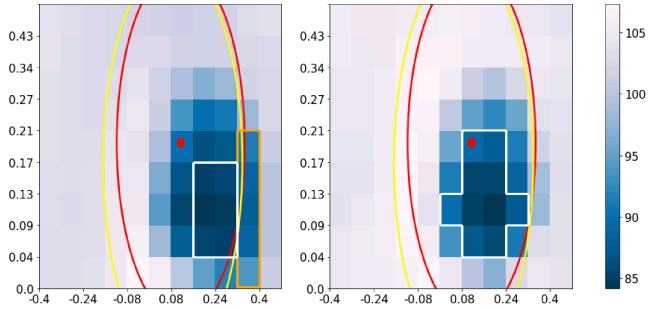

Fig. 5: Comparison of 2D capture regions for DeepQ stepper trained in simulation (figure to the right) and in the inverted pendulum environment (figure to the left).

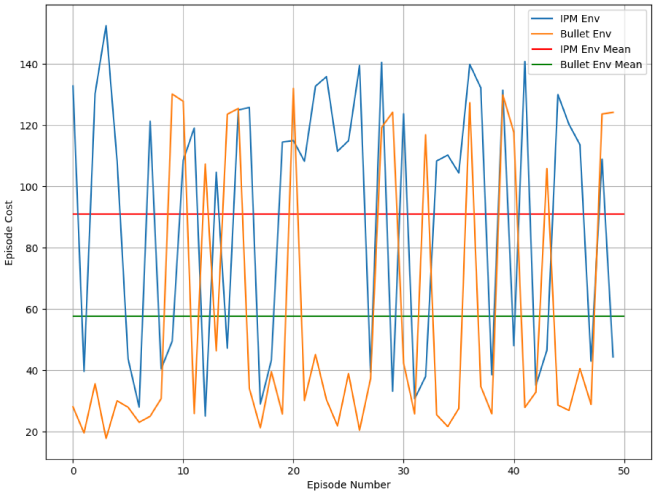

Fig. 6: Comparison of episode cost in simulation.

m (contains both flat and nonconvex terrain). After every 500 episodes of training more nonconvex steps are placed randomly into the environment to increase complexity. The DeepQ stepper converges in about 6000 episodes.

The DeepQ stepper trained directly in the simulation environment is able to navigate complex terrains (Fig. 4). The robot can track and change desired velocities up to $\pm 1 m/s$ online to avoid obstacles reasonably well, despite having point feet and discrete allowed step lengths (Fig. 7). In addition, since the terrain is not flat, the CoM height is constantly changing while the robot walks. LIPM approximations that are used in existing reactive steppers would not hold in such scenarios and it becomes important to account

Fig. 7: CoM velocity, COM moving average of 0.2 second and desired velocity plot.

for the nonlinear dynamics of the system, which the DeepQ stepper can do. In situations where the robot can only step in certain regions such as stepping stones and the optimal step location is infeasible, the DeepQ stepper is able to choose the next best viable action because it learns a capture region. Subsequently, it becomes possible to plan footstep locations on complex terrain online, with no additional computational cost. Finally, the stepper is able to recover from external disturbance ranging from $\pm 3N$ (push recovery) for up to 0.5 seconds on any terrain. The performance of the DeepQ stepper in different scenarios is shown in the attached video.

Conclusion

In this work, a novel reactive stepping framework, the DeepQ stepper is proposed. The stepper can learn the 3D capture regions of different nonlinear systems for which analytical solutions do not exist. Using this information the DeepQ stepper generates robust walking gaits on uneven terrain and recover from external disturbances. Further, the stepper can handle scenarios like stepping stones by providing the next best feasible actions at no additional computation cost. Finally, when trained with the robot in simulation, the DeepQ stepper is able to improve its overall performance by taking into account the whole dynamics (swing foot, joint frictions, etc) that is ignored in simple dynamic models.

REFERENCES

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)*, vol. 1. IEEE, 2001, pp. 239–246.

[2] P.-B. Wieber, "Trajectory free linear model predictive control for stable walking in the presence of strong perturbations," in *2006 6th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2006, pp. 137–142.

[3] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by using preview control of zero-moment point," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1620–1626.

[4] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *2006 6th IEEE-RAS international conference on humanoid robots*. IEEE, 2006, pp. 200–207.

[5] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The international journal of robotics research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[6] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot-1 st report: Walking gait pattern generation," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 1084–1091.

[7] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control using divergent component of motion," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 2600–2607.

[8] M. Khadiv, A. Herzog, S. A. A. Moosavian, and L. Righetti, "Walking control based on step timing adaptation," *IEEE Transactions on Robotics*, 2020.

[9] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 31, no. 4, pp. 1–8, 2012.

[10] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[11] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.

[12] R. Deits and R. Tedrake, "Footstep planning on uneven terrain with mixed-integer convex optimization," in *2014 IEEE-RAS international conference on humanoid robots*. IEEE, 2014, pp. 279–286.

[13] B. Ponton, M. Khadiv, A. Meduri, and L. Righetti, "Efficient multi-contact pattern generation with sequential convex approximations of the centroidal dynamics," *IEEE Transactions on Robotics*, 2021.

[14] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5280–5286.

[15] P. Fernbach, S. Tonneau, O. Stasse, J. Carpentier, and M. Taïx, "C-croc: Continuous and convex resolution of centroidal dynamic trajectories for legged robots in multicontact scenarios," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 676–691, 2020.

[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[17] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[18] H. Audren, J. Vaillant, A. Kheddar, A. Escande, K. Kaneko, and E. Yoshida, "Model preview control in multi-contact motion-application to a humanoid robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 4030–4035.

[19] R. Mirjalili, A. Yousefi-Korna, F. A. Shirazi, A. Nikkhah, F. Nazemi, and M. Khadiv, "A whole-body model predictive control scheme including external contact forces and com height variations," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–6.

[20] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[21] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, *et al.*, "An open torque-controlled modular robot architecture for legged locomotion research," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3650–3657, 2020.

[22] E. Daneshmand, M. Khadiv, F. Grimminger, and L. Righetti, "Variable horizon mpc with swing foot dynamics for bipedal walking control," *IEEE Robotics and Automation Letters*, 2021.