Guided Learning of Robust Hurdling Policies with Curricular Trajectory Optimization

Gautam Salhotra, Shashank Hegde, Sumeet Batra, Peter Englert, Gaurav S. Sukhatme*
University of Southern California

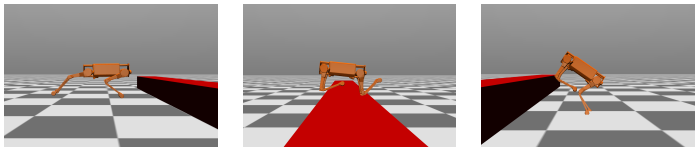


Fig. 1: **Screenshots of the Laikago hurdling task.** Based on a policy produced by our method, CTO-RL, the robot jumps onto the obstacle, and performs a second jump onto the floor to continue running. CTO-RL guides the learning with trajectories obtained from curricular trajectory optimization (CTO) to a reinforcement learning (RL) algorithm.

***Abstract*— We combine analytical and learning-based techniques to help researchers solve challenging robot locomotion problems. Specifically, we explore the combination of curricular trajectory optimization (CTO) and deep reinforcement learning (RL) for quadruped hurdling tasks. Our framework enables engineers and researchers to get the generalization capabilities of learned policies and the efficiency of trajectory optimization. We generate trajectories from a curricular optimization algorithm, as an imitation learning supervisor to an RL algorithm. We evaluate our approach on various robot hurdling tasks where the robot needs to jump over an obstacle of varying size and location. We achieve greater sample efficiency than state-of-the-art reinforcement learning when solving the task, and significantly greater performance than the original trajectories. Results can be seen at https://sites.google.com/usc.edu/cto-rl.**

Robot locomotion tasks such as running and jumping are challenging as they are highly dynamic. Often, we do not have highly accurate models of the environment or robot, to enable methods like model-predictive control [1]. End-to-end learning is not very robust to large changes in the task or environment, and can be sample inefficient.

Here, our method, CTO-RL, addresses robot hurdling with a robust feedback policy, using a two-step procedure. In the first step, we apply trajectory optimization (we use CMA-ES [2]) on a curriculum of hurdling environments with increasing difficulty. In the second part, we distill these trajectories into a feedback control policy by combining imitation and task rewards similar to DeepMimic [3]. This framework allows us to distill a finite set of trajectories into a feedback policy that generalizes across a continuous task range. Additionally, we remove the need for motion priors for highly dynamic skills.

The main contributions of this work are 'CTO' to generate trajectories, and the 'CTO-RL' algorithm that uses these trajectories to train robust feedback policies that outperforms state-of-the-art RL on hurdling tasks.

salhotra|khegde|ssbatra|penglert|gaurav@usc.edu
* Gaurav Sukhatme holds concurrent appointments as a Professor at USC and as an Amazon Scholar. This paper describes work performed at USC and is not associated with Amazon.

I. Guided Learning with Curricular Trajectory Optimization

In curricular trajectory optimization, we solve M tasks sequentially starting from the easiest to the hardest. We store the solution with the highest objective of each task in the buffer $\mathcal{B} = \{(\mathbf{s}_k, \mathbf{u}_k, v_k)\}_{k=0}^{M-1}$, that consists of state trajectories $\mathbf{s}_k$, action trajectories $\mathbf{u}_k$ and task variables $v_k \sim \mathcal{V}$.

We use $\mathcal{B}$ as reference trajectories for training a deep neural network policy $u_t = \pi_\theta(o_t)$. The total reward at time step t is a weighted sum of an imitation reward r_t^{imi} and a task reward r_t^{task}. i.e., $r_t = w_I r_t^{\text{imi}} + (1 - w_I) r_t^{\text{task}}$. Our imitation reward is primarily for imitating parts of the robot state s_t. When the robot is not near the start state of the imitation trajectory, we compare against the first reference state, $r_t^{\text{imi}} = r_t^{\text{imi}}(s_t, \hat{s}_0)$. The imitation reward incentivizes the agent to match the joint positions q_t, velocities $\dot{q}_t$, and base position to the reference motion p_t. During training, we sample task variable v_i and choose an imitation trajectory as the stage that is the closest trajectory that is harder to solve.

II. Experiments

We show experiments for two tasks with a Laikago robot: **Jump up**, (jump on top of a block), and **Hurdle** (get over a block). The block can be of any height in a given range, at any location along the path, and is only sensed by the robot when within the sensing range. The sensor readings contain height ($\leq 0.4m$), width ($0.6m$ for hurdling) and distance from the block (sensing range $4m$). A rollout is successful if the robot crosses 2 meters past the edge of the obstacle without overturning or rolling. The task is solved when a policy achieves a success rate $\geq 95\%$, averaged over all seeds.

The task reward incentivises a positive x-velocity $\dot{p}_t^x$, and penalises control input u_t & deviation of torso orientation β_t from the standing position $\hat{\beta}_t$. This is the same across curricular trajectory optimization (CTO) and RL and is the same for both tasks and all task variations. During training, we create a curriculum of $M = 9$ interpolated environments between 0 and $0.4m$ height with a difference of $0.05m$ between successive environments. At every reset, we sample a block height $h_s \sim [0, 0.4]$, and x-position $x_s \sim [0, 10]$ in metres. We choose a corresponding imitation trajectory with the smallest step height larger than h_s.

When compared to vanilla optimization, neither regular CMA-ES (no curriculum) nor direct trajectory optimization are able to solve this task with the same amount of iterations. We compare our method against an APPO baseline with the same implementation and hyperparameters, but without a trajectory buffer or imitation. The baseline does not completely solve either task. The difference in success rates for the difficult tasks can be seen when we break down success rates by height, in Fig. 3.

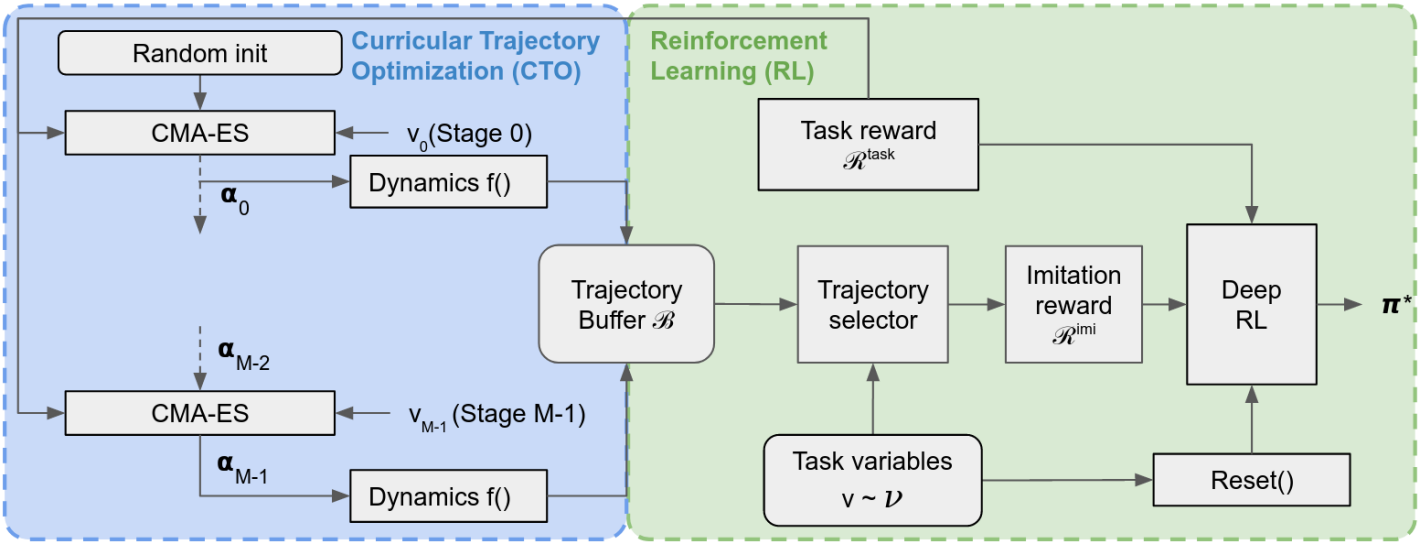
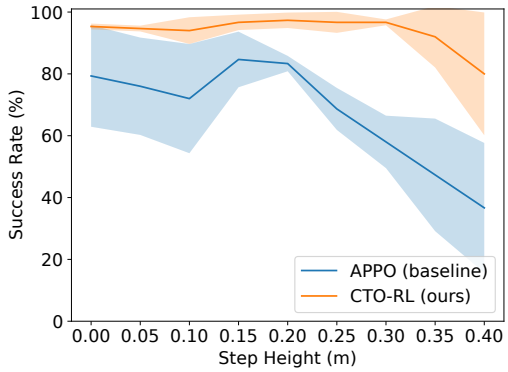
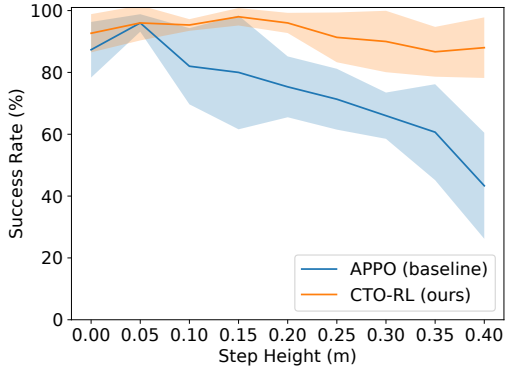


Fig. 2: **Schematic of CTO-RL.** Left: CMA-ES is used to generate reference trajectories for M different environments. These trajectories are stored in the trajectory buffer $\mathcal{B}$. Right: Reference trajectories are sampled from the buffer and used for the imitation reward. The RL agent uses the combined IL and Task rewards to update its parameters and find the optimal policy.



(a) Jump up



(b) Hurdle

Fig. 3: **Success rate measured over step height**, with the trained models for the jump up and hurdle tasks. Both the baseline and our method have a high success rate for small step heights. However, the baseline performance worsens considerably as the task gets harder.

Smaller step heights are easier to jump over, giving higher baseline success rates. As training proceeds, our algorithm naturally learns to move away from the IL component and favor the RL component without explicit hand-crafted interpolation. Our approach achieves an average velocity of about 2-3x faster than the original trajectories.

References

[1] Y. Shi, P. Wang, M. Li, X. Wang, Z. Jiang, and Z. Li, "Model predictive control for motion planning of quadrupedal locomotion," in *2019 IEEE 4th International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2019, pp. 87–92.

[2] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[3] X. B. Peng, P. Abbeel, S. Levine, and M. van de Panne, "Deep mimic: Example-guided deep reinforcement learning of physics-based character skills," *ACM Transactions on Graphics*, vol. 37, no. 4, p. 1–14, Aug 2018. [Online]. Available: http://dx.doi.org/10.1145/3197517.3201311