Foothold Selection Using Direct State-to-Action Mapping

Jemin Hwangbo, Marco Hutter, Christian Gehring, Dario Bellicoso, Peter Fankhauser and Roland Siegwart
Autonomous Systems Lab, ETHZ, Switzerland
jhwangbo@ethz.ch

1 Introduction

We introduce a method of constructing a foothold selection policy for a high Degrees-Of-Freedom legged robot by building a direct state-to-action mapping. Our method starts with optimizing multiple trajectories with different velocity commands and initial configurations. By collecting the optimized trajectories, we get rich data about the underlying optimal policy function. Using Optimally Pruned Extreme Learning Machine (OP-ELM) [1], we find a compact regression form of the data which can be stored in a small memory size and evaluated in a short time.

2 Problem Definition

We are interested in building a foodhold selection policy for a quadruped robot StarlETH [2]. StarlETH is about 25 kg in weight and 50 cm in height. The desired policy should output an optimal foothold location for a given human command considering the full state of the robot. Such policy can be written as $\pi^* : x \rightarrow a^*$, where x is the state and a is the corresponding action. Our state can be written as $x = [x_{int}, x_{com}]^T$, where x_{int} is the state of the robot and x_{com} is the state of the human command.

The human operator sends a velocity command $x_{com} = [v_x, \omega_z]$ which is composed of the desired heading velocity and the desired turning rate. To make the system simple, we use only a single policy that can handle different velocity commands. This policy should output the desired foothold location in order to follow the command from the human. In this way, the controller can also work with a high level planner which plans the global path of the robot.

The gait chosen for this task is walking trot. The gait pattern generator defines the contact/lift-off timing and virtual model controller [4] stabilizes the torso.

3 Optimization of Individual Trajectories

We use the optimization framework used in [3]. The framework is rollout-based and requires many forward simulations of the dynamics. Currently it is running on four cores of an Intel i7-3740QM and the total optimization time for this work is 3 days.

We generate initial conditions by disturbing the robot with impulses. We apply $-20 < I_x < 20$ kgm/s and $0 < I_y < 15$ kgm/s randomly where x is the heading axis and y is the lateral axis. We also vary the command as $0 < v_x < 0.6$ m/s and $0 < \omega < 0.2$ rad/s. For each cases we can find an optimal trajectory respect to a certain cost function. We use a cost function that penalizes energy consumption, instability and degree of violation from the human command.

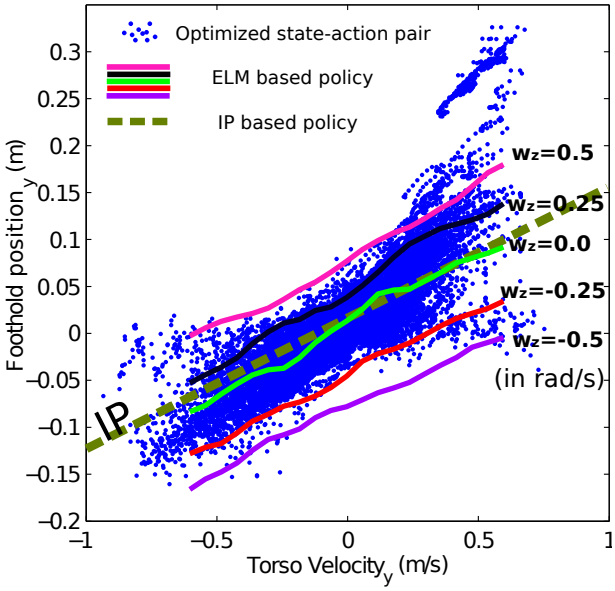

Figure 1: Data obtained from single trajectory optimization projected on 2D plane formed by y-axis of the torso velocity and y-axis of the foothold position is plotted (blue dots). Inverted pendulum based foothold position is plotted in dark green dotted line. In the case of our optimized policy, we vary both yaw rate and lateral velocity and plotted the output with solid lines with different colors.

4 Building a Policy with Optimized Footholds

Since the robot is symmetric about both the sagittal and the coronal plane, we only build one model and use it for all four feet. This ensures that we minimize the training time and obtain a more compact policy representation.

We optimize a total of 880 trajectories and extract about 60,000 state-to-action pairs. These pairs are subset of the underlying optimal policy function which we want to approximate efficiently and accurately. We regress the data using OP-ELM. OP-ELM builds a Single-hidden Layer Feedforward neural Network (SLFN) and reduces the number of hidden layers nodes by removing less efficient ones. This ensures that we have a compact representation of the policy. Raw

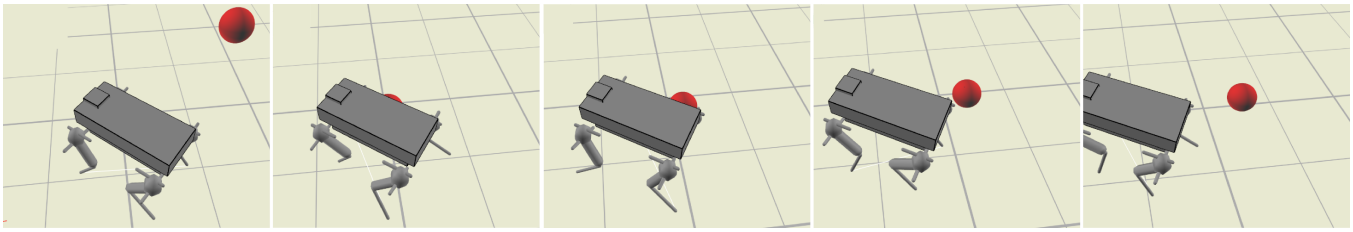

Figure 2: The final foothold controller responds to the disturbance by 4 kg ball thrown at 6 m/s in ODE environment. $v_x = 0.6$ m/s and $\omega_z = 0.25$ rad/s is commanded by the human operator.

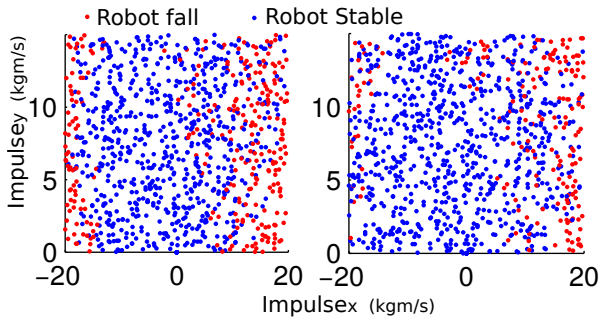

Figure 3: The robot is disturbed at 1,000 different impulse cases with both IP-based controller (left) and optimized controller (right) with the failure rate of 29.3% and 19.1% respectively.

data, inverted pendulum policy and sections of our optimized policy are plotted in Fig. 1. The policy is 600 kb in size in XML format and takes 60 μs to compute.

5 Results

We test our control policy with controlled impulse cases. While commanding constant velocity of $v_x = 0.3$ m/s and $\omega_z = 0.0$ rad/s, we apply randomly sampled impulses to the robot as shown in Fig. 3. Our optimized policy failed 19.1% of the time whereas the inverted pendulum policy failed 29.3% of the time. We also test our policy using free running environment with varying velocity command. A snapshot of the simulation is shown in Fig. 2 and the related video can be found at: http://youtu.be/9h17wxgaIlM

6 Conclusion

The proposed approach generates a higher performance policy compared to simple-model-based approaches and is more computationally efficient than optimizing a trajectory at every control update. Our computation of optimizing trajectories can be carried on a powerful workstation or even on a supercomputer since we do not need to compute it on the real robot. For more complicated systems, it can be used to generate an initial guess for some trajectory optimizers. We plan to build a hierarchical structure of optimized policies such that robot can perform a free-gait where the robot decides when to lift a foot depending on the command and its current state.

References

[1] Y. Miche, A. Sorjamaa, P. Bas, O. Simula, C. Jutten, and A. Lendasse, Op-elm: optimally pruned extreme learning machine, Neural Networks, IEEE Transactions on, vol. 21, no. 1, pp. 158162, 2010.

[2] M. Hutter, C. Gehring, M. Bloesch, M. Hoepflinger, C. D. Remy, and R. Siegwart, "Starleth: A compliant quadrupedal robot for fast, efficient, and versatile locomotion," in Int. Conf. on Climbing and Walking Robots (CLAWAR), 2012.

[3] C. Gehring, S. Coros, M. Hutter, M. Bloesch, P. Fankhauser, M. A. Hoepflinger, and R. Siegwart, "Towards automatic discovery of agile gaits for quadrupedal robots" in International Conference on Robotics and Automation. IEEE, 2014, pp. 42434248.

[4] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, Virtual model control: An intuitive approach for bipedal locomotion, The International Journal of Robotics Research, vol. 20, no. 2, pp. 129143, 2001.