Learning Agile Paths from Optimal Control

Anonymous Author(s)
Affiliation
Address
email

Abstract: Efficient motion planning algorithms are of central importance for deploying robots in the real world. Unfortunately, these algorithms often drastically reduce the dimensionality of the problem for the sake of feasibility, thereby foregoing optimal solutions. This limitation is most readily observed in agile robots, where the solution space can have multiple additional dimensions. Optimal control approaches partially solve this problem by finding optimal solutions without sacrificing the complexity of the environment, but do not meet the efficiency demands of real-world applications. This work proposes an approach to resolve these issues simultaneously by training a machine learning model on the outputs of an optimal control approach.

Keywords: Legged Robots, Imitation Learning, Optimal Control

1 Introduction

Autonomous robotic systems are of particular interest for many fields, especially those that can be dangerous for human intervention like search and rescue, and maintenance on rigs. However, motion planning in unstructured environment is still a hard problem for legged robots and their success depends largely on their ability to plan their paths robustly. Moreover, the method in which a controller deals with obstacles has great consequences on the planned trajectory, and these optimizations are quintessential in generating agile motions for real-world robots.

Trajectory optimization is a common practice for generating motion for legged systems [1, 2, 3], since it can produce optimal trajectories which satisfy the physical and environmental constraints of the robot. However, the solution from trajectory optimization is only valid for a particular pair of initial and target positions, and one needs to re-plan if the pair changes. Due to high-dimensionality and complexity, solving such an optimization problem for legged robots is infeasible in real-time.

Previous work simplified the problem by using a reduced-order model [4] and refining the trajectory using model predictive control [5]. However, the issue is exacerbated in the presence of obstacles, since collision avoidance constraints are non-linear algebraic constraints and so harder to solve.

In recent years, imitation learning [6, 7] and reinforcement learning [8, 9] have become the dominant focus in the research community. The data-driven approach offers a global solution and removes the hurdle of re-planning. On the other hand, collecting data for imitation learning is labour intensive work, which can be done by using motion capture [10] or using animal data [11], which is extremely difficult on legged robots. Reinforcement learning does not require any data, but it is extremely time-consuming to learn a policy.

For planning with obstacles, most work focuses on modelling the environment as a 2-dimensional grid that represents the height of the obstacles [12]. The collision avoidance method finds the traversable paths in the plane [13]. However, the paths may be sub-optimal, since completely circumventing an obstacle is time consuming at best, and completely impossible at worst.

To mitigate the limitations of optimal control and imitation learning, we propose a self-supervised learning approach for efficient 3D collision avoidance in real-time. Specifically, we generate a set of motion data from optimal control with a reduced model to create a rough plan and learn a policy that reproduces the motion data. The learned policy is refined through whole-body model predictive control which satisfies the physical constraints of the robot.

2 Background

Let $\mathbf{x}_k, \mathbf{u}_k$ represent the states and actions of the robot at time-step k, the goal of optimal control is to find a trajectory, a set of $\mathbf{x}, \mathbf{u}$, such that a given cost function is minimized. Assuming that $\mathbf{x}^i$, $\mathbf{x}^t$ are the initial and target state of the robot specified by the user, a typical problem can be formulated as the following

$$
\begin{array}{rll}
\min\limits_{\mathbf{x}_0,\ldots\mathbf{x}_N,\mathbf{u}_0,\mathbf{u}_N} & \sum \mathcal{L}(\mathbf{x}_k, \mathbf{u}_k) & \text{cost function} \\
\text{subject to} & \mathbf{x}_0 = \mathbf{x}^i & \text{initial condition} \\
& \mathbf{x}_N = \mathbf{x}^t & \text{terminal condition} \\
& \mathbf{x}_{k+1} = \mathcal{F}(\mathbf{x}_k, \mathbf{u}_k) & \text{forward dynamics} \\
& \vdots & \text{other constraints}
\end{array}
\tag{1}
$$

For legged robots, motion planning is normally done through optimization. It is well-known that the states of legged robots drift, and predicting a long trajectory is not ideal. In addition, trajectory optimization, especially for long horizon, is not feasible in real-time. This is particularly an issue in the presence of obstacles, since collision constraints are generally non-linear and thus require non-linear solvers. Most people combine trajectory optimization for long horizon planning with model predictive control for short horizon planning real time planning and control.

In the proposed work, we will use trajectory optimization to plan a rough path for the robot torso while avoiding collisions with the environment. The outcome of trajectory optimization is generated using an approximated model, which may not be realistic for the robot. Therefore, we use model-predictive control to refine the path from the reduced model.

3 Methods

Let $\mathbf{q}, \dot{\mathbf{q}}, \ddot{\mathbf{q}} \in \mathbb{R}^{12}$ denote the joint positions, velocities, and accelerations of a 12 degree-of-freedom quadruped robot. We assume that the target position of the robot torso is given, and the environment constraints are fully provided. Our goal is find the control torques $\boldsymbol{\tau} \in \mathbb{R}^{12}$ that can reach the given target while avoiding the obstacles. The objective of the proposed framework is to enable robots to learn a novel skill from self-labelled data.

3.1 Autonomous Data Generation

Our approach is to formulate the problem as an optimal control problem. We simplify the problem by focusing on the trajectory of the robot torso.

$$\mathbf{s} = [x, y, z, \theta_z]^T \in \mathbb{R}^4 \tag{2}$$

where x, y, z denote the translation of the torso, θ denotes the rotation about its local z-axis, and roll and pitch are fixed during the optimization. Given the initial position of the robot state $\mathbf{s}^i$, the task is to find the a sequence of states $\{\mathbf{s}_k\}_{k=1}^N$ that guides the robot from its initial pose $\mathbf{s}^i \in \mathbb{R}^4$ to its target pose $\mathbf{s}^t \in \mathbb{R}^4$ while minimizing the time $\mathcal{T}$ and avoiding the obstacles at position $\mathbf{s}^o$.

We formulate this as a trajectory optimization problem, where the state is the positions and velocities $\mathbf{x} = [\mathbf{s}, \dot{\mathbf{s}}]^T \in \mathbb{R}^8$, and the command is the acceleration $\mathbf{u} = \ddot{\mathbf{s}} \in \mathbb{R}^4$. The decision variables are the sequences of N states and commands, as described in Equation 3.

$$
\begin{aligned}
\min_{\mathbf{x}_0,\cdots,\mathbf{x}_N,\mathbf{u}_0,\cdots,\mathbf{u}_N,} \quad & \mathcal{T} && \text{minimum time} \\
\text{subject to} \quad & \mathbf{x}_{k+1} = \mathcal{F}(\mathbf{x}_k, \mathbf{u}_k), \quad \forall k = 0,\ldots,N-1 && \text{forward dynamics} \\
& \mathbf{x}_0 = [\mathbf{s}^i, \mathbf{0}]^T, \mathbf{x}_1 = \mathbf{0} && \text{initial condition} \\
& \mathbf{x}_N = [\mathbf{s}^t, \mathbf{0}]^T, \mathbf{u}_N = \mathbf{0} && \text{terminal condition} \\
& \mathbf{x}_{min} \le \mathbf{x}_k \le \mathbf{x}_{max}, \quad \forall k = 0,\ldots,N && \text{state boundary conditions} \\
& \mathbf{u}_{min} \le \mathbf{u}_k \le \mathbf{u}_{max}, \quad \forall k = 0,\ldots,N && \text{action boundary conditions} \\
& \mathcal{D}(\mathbf{p}_k, \mathbf{p}^o) \ge \varepsilon, \quad \forall k = 0,\ldots,N && \text{collision constraints}
\end{aligned} \tag{3}
$$

where $\mathcal{F}$ defines the dynamic equation of the system, $\mathbf{x}_{min}$, $\mathbf{x}_{max}$, $\mathbf{u}_{min}$,$\mathbf{u}_{max}$ are the lower and upper bounds of states and actions, and $\mathcal{D}(\mathbf{p}_k, \mathbf{p}^o)$ denotes the distance between the robot and the obstacles. This problem is transcribed into a direct collocation problem [14] and solved using CasADi [15].

3.2 Learning a Predictive Model

Assume that data are generated using the formal section as a set of positions $\mathbf{p}$ and velocities $\dot{\mathbf{p}}$, our goal is to learn a mapping $\boldsymbol{\pi}(.) \in \mathbb{R}^4 \to \mathbb{R}^4$ that predicts the most suitable velocity given the current state $\tilde{\dot{\mathbf{p}}}_k = \boldsymbol{\pi}(\mathbf{p}_k))$.

We use a neural network to encode this relationship. The architecture consists of six fully connected layers, each separated by a tanh activation function. The network is trained using stochastic gradient descent to optimize the mean squared error between the generated $\dot{\mathbf{p}}_k$ and the predicted velocity $\tilde{\dot{\mathbf{p}}}_k$.

3.3 Whole-Body Model Predictive Control

Assuming that $\mathbf{x}_k^{ref}$ is the reference state produced by the learnt model in Sec. 3.2, the whole-body model predictive control [16] computes the torques $\boldsymbol{\tau} \in \mathbb{R}^{12}$ that track the desired inputs $\mathbf{x}_k^{ref}$. This component minimizes the ground reaction force $\mathbf{F} = [\mathbf{f}_1, \ldots, \mathbf{f}_K]^T$ for K stance legs while satisfying the physical constraints of the robot and the friction cone constraints, which prevent slippage.

$$
\begin{aligned}
\min_{\mathbf{F}_1,\mathbf{F}_2,\ldots,\mathbf{F}_M} \quad & \sum ||\mathbf{x}_k - \mathbf{x}_k^{ref}|| + ||\mathbf{F}_k|| && \text{loss function} \\
\text{subject to} \quad & \mathbf{x}_{k+1} = \mathcal{F}(\mathbf{x}_k, \mathbf{u}_k) && \text{forward dynamics} \\
& \mu \lambda_z \ge \sqrt{\lambda_x^2 + \lambda_y^2} && \text{friction cone constraints} \\
& \boldsymbol{\tau}^{min} \le \boldsymbol{\tau}_k \le \boldsymbol{\tau}^{max} && \text{torque limit constraints} \\
& \mathbf{q}^{min} \le \mathbf{q}_k \le \mathbf{q}^{max} && \text{joint limit constraints}
\end{aligned} \tag{4}
$$

Here, the horizon M is a relatively small number. Once the $\mathbf{F}$ is found, the first solution $\mathbf{F}_0$ is taken and the rest are discarded. The ground reaction forces are converted into the equivalent torques.

The swing leg motion is independent from the model predictive control. Given the desired base velocity, we use a simple footstep planner which reads the desired base velocity and generate the next footstep position [17]. A simple interpolation is applied between the current footstep position and the next footstep position. Then, this is tracked by standard PD control.

Finally, a flowchart of the proposed framework is summarized in Fig. 1.

4 Experiments

The experiments were carried out on a quadruped robot in PyBullet [18]. We created a simulated world where the robot needs to move from its initial position to its target position with an obstacle between them. The nominal height of the robot torso is 28 cm, and the height of the table is 25 cm. The robot can crawl under the obstacle only if it lowers its torso height.

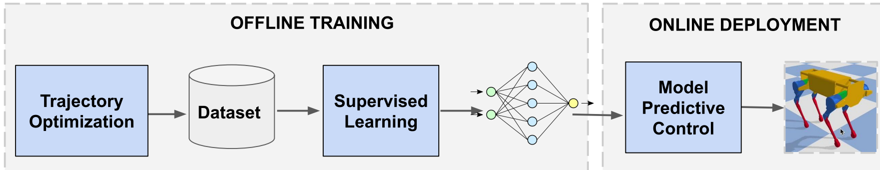

Figure 1: The pipeline of self-supervised collision avoidance planning

Fig. 2 illustrates the simulated setup. The robot starts at the origin and must move to the black arrow at $(3,0)$. The red curve shows the path generated by planning the 2D motion, and the green curve shows the path generated by the 3D optimal control approach.

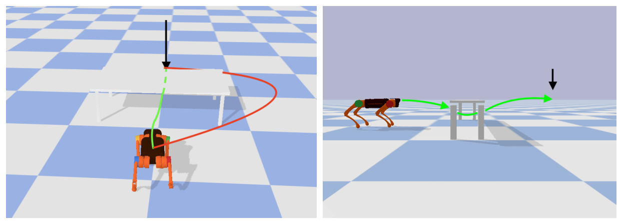

Figure 2: The experimental setup from the front and the side view. The robot starts at the origin and must move to the target (black arrow). The red curve shows the path generated by planning motion in 2D, and the green curve shows the path generated via 3D optimal control.

The target position is $(3,0)$, the table is placed at $(1.5,0)$, and the initial positions of the robot are randomly drawn from $\mathbf{p}^i \sim \mathcal{N}([0.5,0.066,0.026],[0.5,0.66,0.02])$. We use the trajectory optimization method discussed in Sec. 3.1 to generate a path for each initial position, which yields 10000 trajectories, each with ≈ 100 data points.

We use the methods from Sec. 3.2 for learning a predictive model. The network architecture is [256,1024,1024,1024,1024,256] in the hidden layers, and it took 80 seconds to train a model. This was done with batch sizes of 1024 data points for 20 epochs with the stochastic gradient descent optimizer and an initial learning rate of 0.5. The train-validate-test size proportions were $80\% - 10\% - 10\%$. The results of the model can achieve average mean squared error of 10^{-5}.

Fig. 3 shows the snapshot of an example trajectory generated using the proposed method. We can see that the robot lower is body in order to crawl under the table.

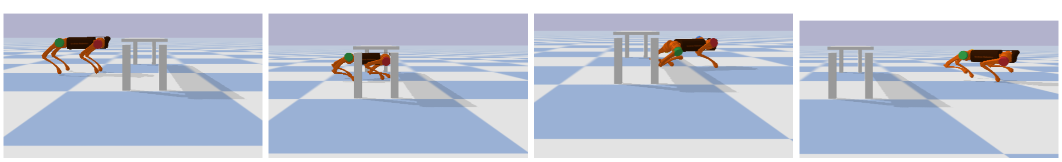

Figure 3: A snapshot of motion generated from learned model

5 Conclusion

This work proposes a self-supervised learning approach to learn a rough plan for a quadruped robot maneuver around obstacles. We use optimal control to generate a rough plan and then use supervised learning to learn a predictive model. The learned model provides the desired base motion and then it is refined using model predictive control for whole-body control. Further improvements include relaxing more control variables to include the pitch and roll of the base and incorporating cameras and LiDARs for perceiving the environment.

References

[1] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal. Stomp: Stochastic trajectory optimization for motion planning. In *IEEE international conference on robotics and automation*, pages 4569–4574, 2011.

[2] M. Posa, S. Kuindersma, and R. Tedrake. Optimization and stabilization of trajectories for constrained dynamical systems. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1366–1373, 2016.

[3] M. Bjelonic, P. K. Sankar, C. D. Bellicoso, H. Vallery, and M. Hutter. Rolling in the deep–hybrid locomotion for wheeled-legged robots using online trajectory optimization. *IEEE Robotics and Automation Letters*, 5(2):3626–3633, 2020.

[4] T. Apgar, P. Clary, K. Green, A. Fern, and J. W. Hurst. Fast online trajectory optimization for the bipedal robot cassie. In *Robotics: Science and Systems*, volume 101, page 14, 2018.

[5] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9, 2018.

[6] S. Schaal. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

[7] Y. Duan, M. Andrychowicz, B. Stadie, O. Jonathan Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning. *Advances in neural information processing systems*, 30.

[8] Y. Yang, T. Zhang, E. Coumans, J. Tan, and B. Boots. Fast and efficient locomotion via learned gait transitions. In *Conference on Robot Learning*, pages 773–783, 2022.

[9] T. Li, H. Geyer, C. G. Atkeson, and A. Rai. Using deep reinforcement learning to learn high-level policies on the atrias biped. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 263–269, 2019. doi:10.1109/ICRA.2019.8793864.

[10] H.-C. Lin, M. Howard, and S. Vijayakumar. A novel approach for representing and generalising periodic gaits. *Robotica*, 32(8):1225–1244, 2014.

[11] X. B. Peng, E. Coumans, T. Zhang, T.-W. E. Lee, J. Tan, and S. Levine. Learning agile robotic locomotion skills by imitating animals. In *Robotics: Science and Systems*, 2020.

[12] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020.

[13] M. V. Gasparino, A. N. Sivakumar, Y. Liu, A. E. Velasquez, V. A. Higuti, J. Rogers, H. Tran, and G. Chowdhary. Wayfast: Navigation with predictive traversability in the field. *IEEE Robotics and Automation Letters*, 7(4):10651–10658, 2022.

[14] O. Von Stryk. *Numerical solution of optimal control problems by direct collocation*. Springer, 1993.

[15] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, 11(1):1–36, 2019.

[16] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim. Dynamic locomotion in the mit cheetah 3 through convex model-predictive control. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1–9, 2018.

[17] M. H. Raibert. *Legged robots that balance*. MIT press, 1986.

[18] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning, 2016.