Optimal Control-Based Baseline for Guided Exploration in Policy Gradient Methods

Xubo Lyu[1], Site Li[1], Seth Siriya[2], Ye Pu[2] and Mo Chen[1]

***Abstract*— In this paper, a novel optimal control-based baseline is presented for policy gradient methods in deep reinforcement learning (RL). The baseline is obtained by computing the value function of an optimal control problem, which is formed to be closely associated with the original RL task. In contrast to the traditional baseline aimed at variance reduction of policy gradient estimates, our work utilizes the optimal control value function to introduce a novel aspect to the role of baseline – providing guided exploration during policy learning. This aspect is less discussed in prior work. We validate the baseline on robot learning tasks, showing its effectiveness in guiding exploration, particularly in sparse reward environments.**

I. Introduction

Deep reinforcement learning has achieved remarkable success in robotics [1]–[4]. One of the key techniques behind the success is the policy gradient method. It uses gradient descent to directly optimize a parameterized control policy with sampled task data, enabling effective learning of high-dimensional and continuous policies while yielding gradient estimates with high variance.

Baseline is a commonly-used component in the policy gradient method to mitigate its high variance of gradient estimates [5], [6]. A baseline is typically a state-dependent function that can be subtracted from the observed total reward, also called return, resulting in a shifted return. This shifted return yields estimated gradient with reduced variance and unchanged expectation. A baseline can have various choices, such as the value function [1], [2], [7], the gradient norm-based [8], and trajectory-based forms [9].

Previous research on baseline [7]–[11] has primarily focused on mitigating gradient variance. While reducing variance is essential, it may not be adequate to ensure policy success, given the challenges of insufficient guidance and exploration in sparse feedback environments. Although other approaches exist to tackle these challenges, there is limited research that addresses them specifically from the perspective of baseline. Therefore, we aim to investigate a novel aspect of baseline, particularly its role in addressing the problem of inefficient exploration in RL.

In this paper, we introduce a novel, optimal control-based baseline to provide guided exploration for policy gradient methods. The baseline is obtained by solving a value function for an optimal control problem. This problem stems from the original RL task, from where a cost function and a coarse mathematical model are identified to approximately describe the objective and the robot system involved in the RL task. This allows the use of optimal control techniques for efficient value computation. Then the value function of the optimal control problem can be used as the baseline for the original RL task. The key insight here is the optimal control value function can offer essential priors related to the RL task, such as the robot system dynamics, thus providing guided exploration for policy gradient RL.

We conduct empirical evaluations of our baseline with the typical policy gradient algorithm on robot learning tasks that involve different types of robots and provide a comprehensive analysis of the baseline's effect on guided exploration, particularly under sparse reward setups.

II. Related Work

A. Baselines in Policy Gradient RL

Various baselines have been proposed in policy gradient RL to mitigate its high variance of gradient estimate. A foundational study [5] established the theoretical bounds of estimated gradient variance induced by commonly used baselines such as the value function. To further lower the variance, recent studies have expanded the range of baseline forms, such as separate baselines for every coefficient of the gradient [8], Taylor expansion of the off-policy critic as baseline [7], state-action dependent baselines [10], [12], and trajectory correlation based baseline [9]. However, most baselines are designed for variance reduction while the potential role of baseline in other aspects, such as providing guided exploration for policy learning, is less studied.

B. Robot System Models from Control Theory

Classical control theory provides mathematical models for a range of robots based on their physical dynamics. For example, the Dubins car model simplifies the motion of vehicles by assuming the constant speed and position-steering alignment while the extended Dubins car model allows variable speeds and turning rates, providing a more realistic representation. Both models are often used to describe differential-wheel robots and four-wheeled vehicles. Besides, the planar quadrotor model simplifies the quadrotor motion from 3D to 2D plane and the attitude model focuses solely on 3-axis rotational behavior. Various other models, such as those for landing robots, pendulum-like systems, and robot manipulators, are available. These models provide simplified abstractions of complex real-world robot models but still capture vital system priors with the potential to enhance modern robot learning methods, such as RL.

[1]School of Computing Science, Simon Fraser University, BC, Canada {xlv, sitel}@sfu.ca, mochen@cs.sfu.ca; [2]Department of Electrical and Electronic Engineering, University of Melbourne, Vic, Australia ssiriya@student.unimelb.edu.au, ye.pu@unimelb.edu.au

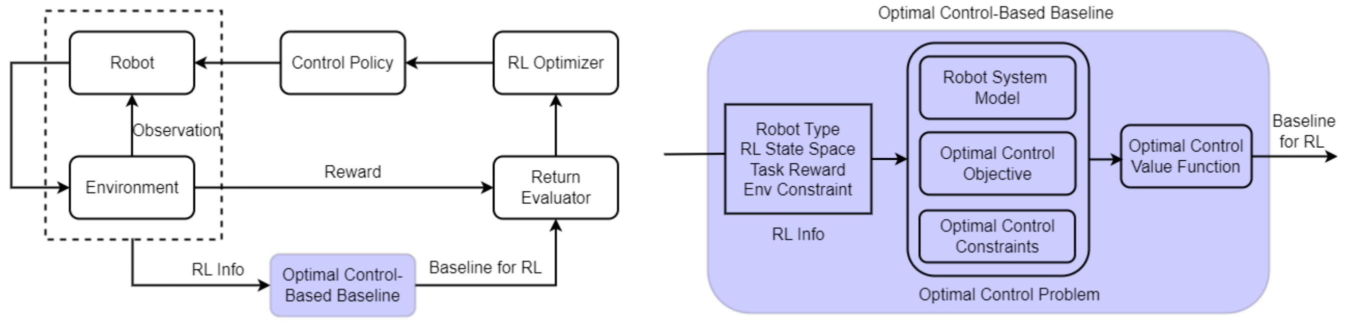


Fig. 1: Overview of our method. **Left**: We propose a novel baseline for the policy gradient RL. Given an RL problem, the RL info is extracted from the robot and environment to form an associated optimal control problem and the optimal control value function is computed to be used back in the RL as a baseline. **Right**: The RL info includes the key attributes of the RL problem such as the robot type, RL state and reward, and can be used to form the main optimal control components: robot system model, objective and constraints. Techniques like MPC and value iteration are used to compute the value function that can be directly used as an RL baseline.

C. *Relations to Our Work*

Our work falls into the realm of introducing a novel baseline for policy gradient RL. Unlike prior work focused on reducing gradient variance, this new baseline prioritizes providing valuable guidance during policy learning. Furthermore, our baseline is formed by utilizing the value function derived from an associated optimal control problem. It innovatively leverages well-developed robot system models from control theory to incorporate essential robot priors to provide guidance exploration for policy gradient RL.

III. Optimal Control-Based Baseline

This section describes the process of constructing an optimal control-based baseline. It starts with extracting the key RL info from the RL problem and using it to form an associated optimal control problem. Then the value function of this optimal control problem is computed and serves as a baseline for the original RL problem.

A. *Extraction of RL Info*

A given RL problem often has key information characterizing its core aspects, which is denoted as RL Info in this work. There are in total four common components of RL info: robot type, state space, task reward, and environment constraints. This info can be extracted from the given robot and environment setup.

1) Robot Type. Robots can have diverse types, like cars, quadrotors, landing robots, etc.
2) RL State Space. RL full state often includes the physical state of the robot such as its position and velocity as well as sensory observation such as pixels or LiDAR readings. This state could be high-dimensional.
3) Task Reward. It is numerical feedback provided to the robot at each time step to evaluate the robot's actions.
4) Environment Constraints. Typical constraints include factors such as maximum episode length, the valid range of state and action spaces, and physical limitations such as obstacles in the navigation task.

Robot Type	Robot System Model
Car-like system	3D Dubins car [13], 5D extended Dubins car [14], 4D bicycle model [15]
Quadrotor	6D planar quadrotor [16], 6D rotation model [17] 3D point mass model [18]

TABLE I: A list of available robot system models. A 3D Dubins car has three states – position (x, y) and its heading angle (θ), describing the dynamics of a simple differential-wheeled robot. Its 5D extended version involves speed v and turn rate ω as additional states. A 4D bicycle model describes the motion of a four-wheeled vehicle with positions, heading and specifically steering angle. For quadrotors, the 6D planar model focuses on 2D plane motion with states of positions, velocities, and 1-axis rotation while the 6D rotation model focuses on the 3-axis rotation, disregarding translation. For simplicity, a quadrotor can be simplified as a point mass, focusing solely on 3D translation while ignoring rotational dynamics.

The RL info is used to choose an appropriate robot system model and to form an associated optimal control problem in the following steps.

B. *Formation of Optimal Control Problem*

Once the RL info mentioned above is obtained, an associated optimal control problem can be formed, which consists of three primary components: robot system model, optimal control objective, and related constraints.

Robot System Model: In control theory, there are many known models developed based on the physical dynamics of robot systems. Despite being abstracted and simplified from true dynamics, these models allow efficient optimal control computation and have the potential to be used in RL problems to mitigate its data inefficiency especially when the RL environment model is unknown and hard to learn. Thus, we aim to leverage those control theory models to calculate an optimal control value function, which can be used as a baseline for the RL problem to provide guidance.

The robot system model is a mathematical equation describing robot physical dynamics and can be represented by a state-space Ordinary Differential Equation (ODE), denoted by $\dot{\mathrm{x}}(t) = g(\mathrm{x}(t), \mathrm{u}(t))$ where $\mathrm{x}(t)$ and $\mathrm{u}(t)$ are the state and control of an optimal control problem at continuous time t and $g(\cdot, \cdot)$ is the model function. In practice, we typically discretize it using finite difference schemes like the Forward Euler method, resulting in a discrete version $\mathrm{x}_{k+1} = g(\mathrm{x}_k, \mathrm{u}_k)$, where k is discrete time steps.

This model can be heuristically selected from control theory based on RL info. We first categorize the robot system by its type (e.g., car-like, quadrotor or others) and assess candidate models with two criteria: (1) x components should be accessible from the RL full state s, where $\mathrm{x} \subset s$; (2) x is able to determine the RL task reward, defined as $r(s) \triangleq r(\mathrm{x})$.

Optimal Control Objective: The optimal control objective specifies the desired outcome that an optimal control problem aims to achieve while satisfying the robot system model. It is often described in Eq. (1) as the minimization of cumulative cost subject to a model constraint,

$$\begin{aligned} \min_{\mathrm{u}_{0:T-1}} \sum_{k=0}^{T-1} c(\mathrm{x}_k, \mathrm{u}_k) \\ \text{s.t. } \mathrm{x}_{k+1} = g(\mathrm{x}_k, \mathrm{u}_k) \text{ for } k = 0, \ldots, T-1 \end{aligned} \tag{1}$$

where $\mathcal{L}$ is the objective function, $c(\mathrm{x}_k, \mathrm{u}_k)$ is the step cost function defined over optimal control state x_k and control input u_k at each discrete time step k, and $\mathrm{x}_{k+1} = g(\mathrm{x}_k, \mathrm{u}_k)$ is the discrete-time robot system model. The key to deriving this objective $\mathcal{L}$ is to find the appropriate cost function.

The cost function $c(\mathrm{x}_k, \mathrm{u}_k)$ is designed by penalizing the error between the current and desired state whilst optionally minimizing the control effort. In practice, the error function can be in different forms, such as Euclidean distance of positions in navigation tasks.

$$c(\mathrm{x}_k, \mathrm{u}_k) = \alpha * e(\mathrm{x}_k, \mathrm{x}^*) + \beta * \|\mathrm{u}_k\|^2 \tag{2}$$

To use this cost function, we assume the existence of a goal state x^* from the original RL problem, which is a subset of RL full state s that specifically defines the goal of the RL problem. Notably, x^* must also be within the state space of the chosen robot system model, adhering to $\mathrm{x}^* \subseteq \mathrm{x}_k \subseteq s_k$. The cost function coefficients α, β can be flexibly adapted to obtain optimal control solutions.

Optimal Control Constraints: It refers to the environmental limitations or conditions that must be obeyed when solving an optimal control problem. Typical constraints include the range of state space, the initial state, and the positions of obstacles in a collision-avoidance scenario. Those constraints can be added to the optimal control objective to further rectify the solution.

C. Calculation of Optimal Control Value Function

With the formed optimal control problem, we solve it depending on the cost function and obtain the associated optimal control value function. MPC is a widely used control technique that repeatedly solves the optimization problem in an iterative manner. We use MPC based on the objective function, system model, and initial state to generate trajectories which include state and control at every step.

We first uniformly generate a number of initial states from the optimal control state space. With these initial states and Eq. (1), MPC produces a set of feasible trajectories, denoted by $\{\tau_j | j = 0, 1, 2, ..., N\}$, where N is the number of trajectories and each trajectory has a horizon of length H_j. From each trajectory τ_j, we select every state x_m^j, which is the mth state of the jth trajectory and compute the optimal control value function $V^{\mathrm{oc}}(\mathrm{x}_m^j)$ following Eq. (3).

$$V^{\mathrm{oc}}\left(\mathrm{x}_m^j\right) = \sum_{m=0}^{H_j} \gamma^m r(\mathrm{x}_m^j) \tag{3}$$

The value function sums up the discounted reward of each state along the whole trajectory span, where the γ^m means the discount factor of mth step of state.

IV. Results

In this section, we present results on the performance of the proposed method.

A. Demonstrate example: differential drive car navigation

We first evaluate our approach on a simulated TurtleBot, which resembles a differential drive car navigating in an unknown environment with multiple static obstacles. The car aims to learn a collision-free control policy via trial and error to move from a starting area to a specific goal area. We use the Gazebo simulator to build the robotic system and environment for the target RL problem.

Following the steps described in the methodology, the original RL problem provides info for optimal control baseline construction:(1) robot type is car, (2) state and observations $s = (x, y, \theta, v, \omega, d_1, ..., d_8)$, which are position, heading angle, current speed, turn rate, and eight-dimensional lidar sensory data. (3) Task reward as in where G refers to the set of goal states, C the set of collision states and I the set of other states which involve neither collision nor goal.

$$r(s) = \begin{cases} 0 & (x, y) \in \mathbf{I} \\ +1000 & (x, y) \in \mathbf{G} \\ -400 & (x, y) \in \mathbf{C} \end{cases} \tag{4}$$

Environment constraints: start area, goal area, obstacles, and map size. According to the key components of RL reward (x,y), the most suitable optimal control model can be selected as the 3D Dubins car[13], which consists of three items,(x,y) of target position, and heading angle theta. Then, the objective function could be defined as:

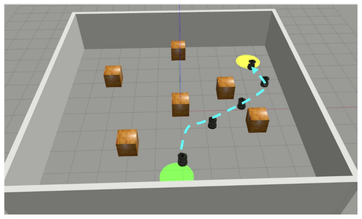
$$\dot{x} = g(x, u) = \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} v\cos(\theta) \\ v\sin(\theta) \\ \omega \end{bmatrix}$$

where state $\mathrm{x} = [x, y, \theta]^\top$ and control input $u = [v, \omega]^\top$, v is constant value.

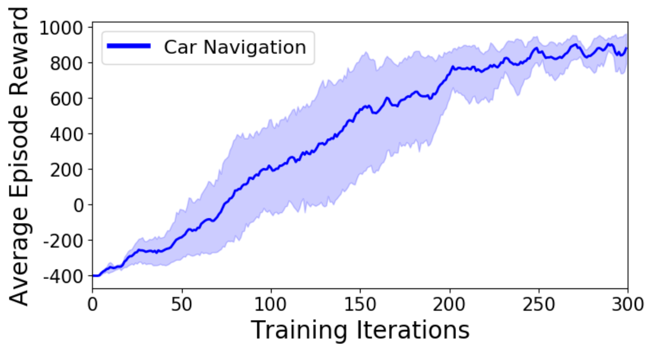
Then we construct the objective function according to equation 1. The key step in getting the objective function is figuring out the cost function $c(\mathrm{x}_k, \mathrm{u}_k)$. In the MPC method, the cost function should be designed to refer to the original goal of the RL problem and its reward function. In this car problem, the goal is to navigate the car to a goal position. The cost function is a function that is proportional to the distance to the goal. As the distance to the goal decreases, the cost becomes smaller, but as the distance to the goal increases, the cost becomes larger. Therefore, according to the commonly used description in optimal control, the cost function formula can be expressed in the form of $A^\top QA + B^\top RB$, where A and represent vectors, Q and R are learnable parameters. Then we can get the objective function written as:

$$\begin{aligned} \min \sum_{k=0}^{T-1} & \left[(x - x_*)^\top \gamma^{k+1} Q (x - x_*) + u^\top R u \right] \\ \text{s.t.} \quad & \tilde{x} = g(x, u) \\ & \text{initial } x_0 \in \text{start area} \\ & x_T \in \text{goal area} \\ & (x, y) \notin \text{obstacles} \\ & \forall x_k \in \text{map area}. \end{aligned} \tag{5}$$

Then the value function can be calculated With the well-defined cost function and objective function. In this example, we used the TPO-PT method to get the trajectories $\{\tau_j | j = 0, 1, 2, ..., N\}$, for every state x_m^j of every single trajectory in τ_j, we can calculate its value function following Eq. (3). Then a continuous baseline function can be regressed from the pair set of $(\mathrm{x}_m^j, V^{\mathrm{oc}}(\mathrm{x}_m^j))$.



(a) Car Navigation



(b) Car Navigation Performance

Fig. 2: Robots learn collision-free navigational policy with hi-dim sensory input (e.g. lidar) with lo-dim model-based baseline.

B. Applied to more robot tasks

C. Compare with other baseline methods

D. Ablation study

E. Real robot experiment

V. Conclusion

References

[1] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning (ICML)*, 2015.

[2] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.

[3] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. M. O. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” *CoRR*, vol. abs/1509.02971, 2016.

[4] T. Johannink, S. Bahl, A. Nair, J. Luo, A. Kumar, M. Loskyll, J. A. Ojea, E. Solowjow, and S. Levine, “Residual reinforcement learning for robot control,” in *International Conference on Robotics and Automation (ICRA)*, 2019.

[5] E. Greensmith, P. L. Bartlett, and J. Baxter, “Variance reduction techniques for gradient estimates in reinforcement learning,” *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1471–1530, 2004.

[6] T. Hofmann, A. Lucchi, S. Lacoste-Julien, and B. McWilliams, “Variance reduced stochastic gradient descent with neighbors,” *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[7] S. Gu, T. Lillicrap, Z. Ghahramani, R. E. Turner, and S. Levine, “Q-prop: Sample-efficient policy gradient with an off-policy critic,” *arXiv preprint arXiv:1611.02247*, 2016.

[8] J. Peters and S. Schaal, “Reinforcement learning of motor skills with policy gradients,” *Neural networks*, vol. 21, no. 4, pp. 682–697, 2008.

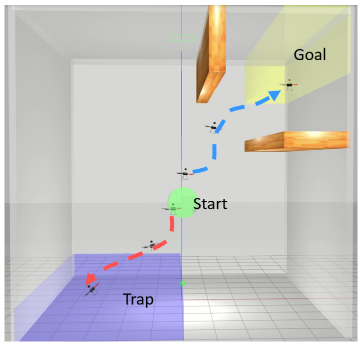
[9] C.-A. Cheng, X. Yan, and B. Boots, “Trajectory-wise control variates for variance reduction in policy gradient methods,” in *Conference on Robot Learning*. PMLR, 2020, pp. 1379–1394.

[10] H. Liu, Y. Feng, Y. Mao, D. Zhou, J. Peng, and Q. Liu, “Action-depedent control variates for policy optimization via stein’s identity,” *arXiv preprint arXiv:1710.11198*, 2017.

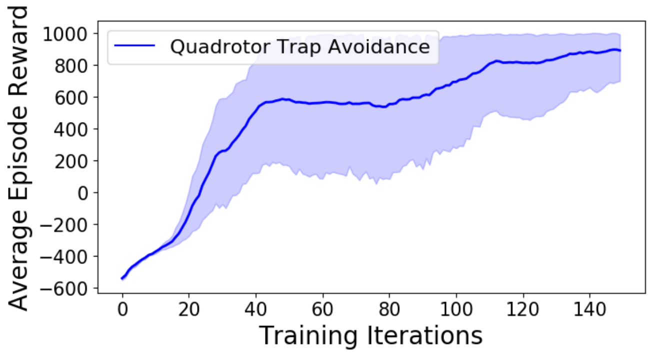
[11] C. Wu, A. Rajeswaran, Y. Duan, V. Kumar, A. M. Bayen, S. Kakade, I. Mordatch, and P. Abbeel, “Variance reduction for policy gradient with action-dependent factorized baselines,” in *International Conference on Learning Representations*, 2018. [Online]. Available: https://openreview.net/forum?id=H1tSsb-AW

[12] A. Wächter and L. T. Biegler, “On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming,” *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.

[13] L. E. Dubins, “On curves of minimal length with a constraint on



(a) Quadrotor trap avoidance



(b) Performence

Fig. 3: Robots learn collision-free navigational policy with hi-dim sensory input (e.g. lidar) with lo-dim model-based baseline.

average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[14] A. Sahraeekhanghah and M. Chen, "Pa-fastrack: Planner-aware real-time guaranteed safe planning," in *2021 60th IEEE Conference on Decision and Control (CDC)*, 2021, pp. 2129–2136.

[15] B. A. Francis, M. Maggiore, B. A. Francis, and M. Maggiore, "Models of mobile robots in the plane," *Flocking and rendezvous in distributed robotics*, pp. 7–23, 2016.

[16] X. Lyu and M. Chen, "Ttr-based reward for reinforcement learning with implicit model priors," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 5484–5489.

[17] Z. Tahir, W. Tahir, and S. A. Liaqat, "State space system modelling of a quad copter uav," *arXiv preprint arXiv:1908.07401*, 2019.

[18] A. Romero, S. Sun, P. Foehn, and D. Scaramuzza, "Model predictive contouring control for time-optimal quadrotor flight," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3340–3356, 2022.