TTR-Based Rewards for Reinforcement Learning with Implicit Model Priors

Xubo Lyu\textsuperscript{1} and Mo Chen\textsuperscript{1}

\textbf{Abstract}— Model-free reinforcement learning (RL) provides an attractive approach for learning control policies directly in high dimensional state spaces. However, many goal-oriented tasks involving sparse rewards remain difficult to solve with state-of-the-art model-free RL algorithms, even in simulation. One of the key difficulties is that deep RL, due to its relatively poor sample complexity, often requires a prohibitive number of trials to obtain a learning signal. We propose a novel, non-sparse reward function for robotic RL tasks by leveraging physical priors in the form of a time-to-reach (TTR) function computed from an approximate system dynamics model. TTR functions come from the optimal control field and measure the minimal time required to move from any state to the goal. However, TTR functions are intractable to compute for complex systems, so we compute it in a lower-dimensional state space, and then do a simple transformation to convert it into a TTR-based reward function for the MDP in RL tasks. Our TTR-based reward function provides highly-informative rewards that accounts for system dynamics. Generally, TTR-based reward functions can be incorporated in any model-free RL algorithm to accelerate training without reducing flexibility and applicability of model-free RL. We evaluate our approach in Gazebo with model-free proximal policy optimization (PPO) algorithm on two goal-oriented tasks for generic car and planar quadrotor respectively, and observe substantial improvement in performance and efficiency compared to using sparse or distance reward functions.

I. INTRODUCTION

Reinforcement learning (RL) is a principled mathematical framework for experience-driven autonomous learning and is a powerful tool for solving sequential decision-making problems by training the agents to maximize the long-term accumulated reward. Generally, RL methods can be divided into two main streams: model-based methods and model-free methods, depending on if the inner working principles (model) between agent and environment are directly employed in learning process \cite{1}. In the context of model-based methods, the optimal policy is derived based on internal simulations of a forward model that corresponds to a representation of the robots dynamics. This characteristic significantly reduces the amount of trials in learning and leads to fast convergence to optimal policy. On the other hand, the main disadvantage of model-based method is the algorithms are heavily dependent on the models capability of accurately representing the transition dynamics.

The downsides of model-based methods have been somewhat alleviated by model-free methods, which do not require a prior model, and the optimal policies are derived by trial-and-error. For example, model-free methods allow the learning of policies that map observations directly to actions, a difficult robotic task for traditional control theoretic tools and model-based methods to solve \cite{2}. Although model-free methods avoid the issue of inaccurate dynamics representation, they often require an impractically large number trials. High sample complexity is a key challenge of model-free RL, and a fundamental barrier impeding its adoption in real-world settings, especially in the context of robotics.

Two main causes of high sample complexity are the use of naive exploration strategies and delayed sparse reward. Naive exploration method such as $\epsilon$-greedy would only simply choose the action which the agent expects to provide the greatest reward as optimal action in higher probability at each timestep. Though theoretical RL literature offers a variety of provably-efficient approaches to deep exploration \cite{3}, most of these are designed for Markov decision processes (MDPs) with small finite state spaces, while others require solving computationally intractable planning tasks \cite{4}. These algorithms are not practical in complex environments with high dimensional states. On the other hand, delayed sparse reward indicates that agent can only have access to fewer reward as feedback even after long-term exploration. Such issue can significantly slow down learning process. Various techniques have been proposed to overcome these challenges. Hierarchical Reinforcement Learning (HRL) involves decomposing a task into sub-tasks and learning to solve each sub-task individually. Curriculum learning \cite{5} offers another attractive strategy by training the agent using simpler tasks first, and increasingly difficult tasks afterwards. While this idea works successfully on a variety of systems \cite{6}, but failed in highly dynamic or unstable system since it’s still not easy to determine reasonable curricula for each learning stage.

The combination of model-free and model-based approaches now attracts more research interests and may pave the way for overcoming the challenge of high sample complexity while maintaining advantages of model-free RL. Vladimir Feinberg \cite{7} proposed model-based value expansion (MVE) to control the uncertainty in the model by
only allowing imagination to fixed depth, thus improving Q-learning by providing higher-quality target values for training. BaRC [8] leverage physical priors in the form of an approximate system dynamics model to design a curriculum for a model-free policy optimization algorithms. In BaRC, the high dimensional input including sensor data as well as motion states are used to learn control policies for robotic systems.

In this paper, we also apply the high-level idea of combining model-based and model-free method and propose Time-To-Reach (TTR) reward shaping. TTR function is traditionally used in the field of optimal control and is defined as a function of the robot states. It represents the minimum arrival time to the goal from any state under specific dynamics constraints. Here we adopt TTR function as the reward elements in RL tasks since minimal arriving time is far more reasonable to measure "how good" the state is. Also in our method the TTR function is computed by implicitly leveraging approximate physical model while not altering the flexibility of model-free RL. The TTR function is hard to be computed for high-dimensional systems, which is why we use an approximate, lower dimensional model of the robot to compute it. As will demonstrated in our experiments, the approximate physical model is sufficient for the purpose of guiding policy learning because normally we don’t require highly accurate reward signals in reinforcement learning.

Our method has distinct features on addressing sample complexity issue. Unlike curriculum learning, there is no requirement of designing curricula in TTR-based reward shaping method because the learning process does not depend on curricula but on globally shaped reward. Moreover, the TTR function only needs to be computed once for entire state space. The advantage of our approach is that TTR function captures an essential aspect of a task: the TTR-based reward measures “how good” a state is by the minimal arrival time of a simplified system model. In addition, it only requires approximate TTR function, so a low-fidelity approximate physical model can be used to compute them while not affecting forward learning. So our method acts like a wrapper that augments any model-free RL algorithm and is capable of working on highly dynamic systems in which the forward and backward time dynamics vastly differ.

By using prior knowledge of the system through approximate dynamics, our approach drastically reduce sample complexity while still maintaining the potential of model-free method to learn high quality policies with complicated inputs.

We demonstrate our approach on a generic car environment and a planar quadrotor environment, and observe substantial improvements in sample efficiency compared to sparse or distance reward settings.

Contributions: We propose a TTR-based reward function that leverages physical model priors to efficiently guide policy learning in model-free RL tasks. In our method, an approximate system model with lower-dimensional state space is selected first. Once an approximate model is chosen, there is very little hand-tuning needed to compute the TTR function. After it is computed, we perform a simple transformation to convert the TTR function for the approximate model to a TTR-based reward function for the MDP in order to guide the policy learning in RL tasks. Moreover, our new approach can be incorporated in any model-free algorithms for solving robotic tasks with different system dynamics in highly flexible way. In particular, by effectively incorporating system dynamics in an implicit and compatible manner with deep RL, we retain the ability to learn policies that map robotic sensor inputs directly to actions. Thus, our approach represents a bridge between traditional analytical optimal control approaches and the modern data-driven RL, and inherits benefits of both. We evaluate our approach on two common robotic tasks and obtain significant improvements in learning efficiency and performance.

II. RELATED WORK

Integration of RL and neural networks has a long history [9][10]. With recent exciting achievements of deep learning [11], deep neural network has been prevailing in RL in different areas such as games, robotics and NLP. The deep Q-network, which was developed to learn to play a range of Atari 2600 video games at a superhuman level directly from image pixels, provides solutions for instability of function approximation techniques in the class of value-based RL method [12]. Policy-based methods, including Guided policy search (GPS) [13]. Trust region policy optimization (TRPO) [14] and proximal policy optimization (PPO) [15] aims to directly find policies by means of gradient-free or gradient-based method. As shown in our work, we evaluate our reward shaping method on two robotic tasks with policy-based model-free learning algorithm and illustrates competitive performance.

Exploration inefficiency, as the root cause of high sample complexity, remains a major challenge in RL [16]. “Deep exploration” strategy address this problem by maintaining a distribution over possible values and bootstrapping from it with random initialization [16]. Curriculum-based approaches paved another possible path by first training on easier sub-problems, and increasing the difficulty of sub-problems as training progresses until the original task becomes appropriate to train on directly [5]. Hierarchical RL framework tries to improve exploration efficiency by build modules with different level, where top-level module learns a policy over options (subgoals) and the bottom-level module learns policies to accomplish the objective of each option [17][19].

Our work is mostly based on reward shaping, a technique that provides additional localized feedback based on prior knowledge to guide the learning process [20]. Reward shaping has been proven to be a simple but powerful method for reducing sample complexity and speed up many RL tasks. Randlov and Alstrom [21] apply a shaping function to learn control policy for riding a bicycle to a goal. However, the function requires significant engineering efforts to hand-tune. Ng, Harada, and Russell [22] raised up potential-based reward shaping (PBRS), proving the conditions that an arbitrary externally specified reward shaping function must satisfy if they could be included in a RL system without modifying its optimal policy. What’s more, Vieuwiora [23] proposed the potential-based advice (PBA) framework that allows the potential function to also include actions while still restricted with PBRS conditions.

Inverse reinforcement learning (IRL) can be used to learning a reward function from the expert demonstrations, thereby avoiding the manual specification [24], [25]. Recently, incorporating prior model knowledge into learning
acquire some success. [8] apply physical model dynamics as prior knowledge for stable curricula generation to accelerate learning process. [26] allow the full transition information as prior beliefs to provide shaping rewards directly without learning through interactions with environment. Our work is inspired by the high-level idea of utilizing physical system dynamics to improve deep exploration thus speed up learning process. Specifically, we leverage model dynamics as prior to offer shaping reward directly without any engineering effort or interactions with environment.

Time-to-reach (TTR) problems are an important class of optimal control problems, and involves finding the minimum time it takes a system to reach some goal starting from any initial state. The common approach for solving TTR problem is solving a Hamilton-Jacobi (HJ) partial differential equation (PDE). Normally the computational cost of TTR is too expensive for system with high dimensional state. However, model simplification and system decomposition techniques make a significant leap in alleviating the computational burden in a variety of problem setups [27]–[29]. As we will demonstrate in our method, approximating the TTR via simplifying the model dynamics or applying system decomposition is sufficient to generate effective reward map for guiding policy learning.

III. PRELIMINARIES

The objective of our work is to implicitly utilize the system dynamics as prior to generate dynamics-informed reward function automatically to accelerate model-free RL.

A. Markov Decision Processes

Consider a Markov decision process (MDP) whose reward function have finite variance, defined by a tuple \( M = (S, A, P, r, \rho_0, \gamma, T) \), where \( S \) is a finite set of states, and \( A \) is a finite set of actions. \( P : S \times A \times S \to [0, 1] \) are the transition probabilities for transitioning to state \( \tilde{s} \) conditional on the agent being in state \( s \) and taking action \( a. r : S \to \mathbb{R} \) is the deterministic reward function. \( \rho_0 : S \to \mathbb{R} \) is start state distribution, and \( \gamma \in [0, 1] \) is a discount rate. In context of model-free deep RL (DRL), we utilize neural network represent a policy \( \pi_\theta : S \times A \to \mathbb{R}_+ \), describing the probability of taking action \( a \) conditional on being in state \( s. \theta \) is the parameters of neural network. Policy \( \pi_\theta \) can be either deterministic or stochastic. The objective of an RL agent is to find an optimal policy \( \pi^* \) that maximize the expected return: \( J_{\rho_0}(\pi_\theta) = E_{s_0 \sim \rho_0} R(\pi, s_0) \), where \( R(\pi, s_0) = \mathbb{E}_{e_{s_0}^{T\rightarrow 0}} \sum_{t=0}^{T} r(s_t, a_t) \) is the expected cumulative reward starting from states \( s_0 \) \( \rho_0. \tau = (s_0, a_0, \ldots, a_{T-1}, S_T) \) denotes a complete trajectory with \( a_t \sim \pi_\theta(a_t|s_t) \) and \( s_{t+1} \sim P(s_{t+1}|s_t, a_t) \).

B. Model-free optimization

Policy-based algorithms, coupled with nonlinear approximator have been prevailing in the area of model-free optimization in RL. policy-based methods have better convergence properties since it’s optimized by gradient. Moreover, policy gradients are more effective in high dimensional action spaces, or when using continuous actions. For these reasons, the learning algorithm we apply in our task is "Proximal policy optimization (PPO)" [15], which belongs to the category of policy-based method. Policy-based methods do not need to maintain a value function, but directly search for an optimal policy \( \pi^* \). Typically, a parameterized policy \( \pi_\theta \) is chosen, whose parameters are updated to maximize the expected return \( E[R|\theta] \) using gradient-based optimization. Gradients provide a strong learning signal as to how to improve a parameterized policy. However, to compute the expected return we need to average over plausible trajectories induced by the current policy parameterization. Techniques such as REINFORCE [30] or the "re-parameterization trick" [31] provide neural network the ability to back-propagate through stochastic function. Intuitively, gradient ascent using the estimator increases the log probability of the sampled action, weighted by the return. More formally, the REINFORCE rule can be used to compute the gradient of an expectation over a function \( f \) of a random variable \( X \) with respect to parameters \( \theta \):

\[
\nabla_\theta E_X[f(X; \theta)] = E_X[f(X; \theta)\nabla_\theta \log p(X)]
\]

Searching directly for a policy represented by a neural network with very many parameters can be difficult and the resulting policy may be stuck in local minima. Trust region is a common used method to overcome such problem by restricting optimization steps to lie within a region where the approximation of the true cost function still holds. One of the newer algorithms in this line of work, trust region policy optimization (TRPO), has been shown to be relatively robust and applicable to domains with high-dimensional inputs [14]. However, the constrained optimization of TRPO requires calculating second order gradients, limiting its applicability. Proximal policy optimisation (PPO) algorithm performs unconstrained optimization, requiring only first-order gradient information while still retaining the performance of TRPO. These characteristic means PPO is gaining popularity for a range of RL tasks [15], [32].

C. Approximate System Model and Time-to-Reach Functions

Given the following dynamical system in \( \mathbb{R}^n \)

\[
\dot{s}(t) = f(s(t), a(t))
\]

where \( \dot{s} \) is the state of approximated system model and \( a \) is control action, and \( A \subseteq \mathbb{R}^n_A \) is the feasible set of control. The TTR problem involves finding the minimum time it takes to reach a goal from any initial state \( s \), subject to the system dynamics in (2). We assume that \( f \) is Lipschitz continuous. Under these assumptions, the dynamical system has a unique solution.

Mathematically, the time it takes to reach a goal \( \Gamma \subseteq \mathbb{R}^n \) using a control policy \( a(\cdot) \):
Detailed derivations and discussions are presented in [33], [34]. Well-studied numerical techniques [27], [28], [35] based on level set methods have been developed to solve [5].

IV. APPROACH

In RL, an agent learns to optimize its behaviors in an unknown environment by executing control policies, experiencing the consequent rewards, and improving the policy based on the rewards. Without sufficiently strong reward signals, learning is very difficult. For many interesting problems, the state space is far too large to be exhaustively explored, and the reward function is extremely sparse, leading to poor learning performance. Our method addresses this disparity by assigning a higher reward to states with lower arriving time at each training time step, we can dramatically improve learning performance without changing the model-free RL algorithm. Intuitively, to many forms of life, agent (animals or human) would learn faster when fed with more localized feedback resulting in progress toward a target behavior.

In many goal-oriented RL tasks with sparse reward setting, the best behaviors we want is to reach the goal as fast as possible. Such requirement is naturally in accordance with the features of TTR-based reward because it offers a convenient choice as localized feedback by indicating the minimal arriving time from any state to goal region. Intuitively, the states with lower arriving time is a better state because it’s easier to get into goal region, thus it should be assigned with a higher reward. In this way, we build the connection between the meaning of TTR function and reward signal and obtain the non-sparse continuous reward function. Note that unlike the common shaping function such as reward because the state \( s \rightarrow \bar{R} \) is the reward function that captures the information of state in all feasible dimensions in a dynamically-informed manner. By applying the TTR-based reward to state at each training time step, we can dramatically improve learning performance without changing the model-free RL algorithm. The rest of this section describes our method in detail.

1) Model Selection: “Model selection”\(^1\) here refers to that we need to pick up a physical model in order to compute TTR function. The physical model here should be the ordinary differential equation (ODE) which describe the approximate system dynamics. For example, we choose (7) as our approximate model for generic car RL task and (8) for planar quadrotor RL task. Normally, the state domain of the approximate physical model is a subset of full MDP state domain in order to make TTR function be tractably computed. One feature of our method is that we don’t have to know or select the same accurate MDP model we follow in learning process since the TTR-based reward computed via the approximate physical model is sufficient to provide discriminative local feedback for guiding policy searching. For example, Gazebo simulator has a built-in physical engine as MDP model which we do not know exactly. However, the TTR-based reward still could be used to guide policy learning even though it’s computed based on the approximate physical model we have chosen below.

Moreover, it’s rather important to decide which states should be included in the approximate model. As demonstrated in our experiment, the information about pitch angles and angular velocity would be very useful for describing the motion of quadrotor. Therefore, if we include angle elements as part of states in the approximate model to compute TTR-based reward, the TTR-based reward will guide the quadrotor to learn the policy which always choose the best pitch angle in order to reach the goal region as soon as possible.

\[
\dot{s} = \begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta}
\end{bmatrix} = \begin{bmatrix}
\nu \cos \theta \\
\nu \sin \theta \\
\omega
\end{bmatrix},
\]

As an example, for the first experiment involving a simple car, we used the approximate model in (7), where the position \((x, y)\), heading \(\theta\) are the car’s states. The control variable is the angular velocity \(\omega\).

For our second experiment involving a quadrotor, we used the approximate model in [8], with \(T \leq T_1, T_2 \leq T\), and where the state is given by the position in the vertical plane \((x, z)\), velocity \((v_x, v_z)\), pitch \(\phi\), and pitch rate \(\omega\). The control variables are the thrusts \(T_1, T_2\). In addition, the quadrotor has mass \(m\), moment of inertia \(I_y\), half-length \(l\). Furthermore, \(g\) denotes the acceleration due to gravity, \(C_D\) the translational drag coefficient, and \(C_D\) the rotational drag coefficient.

2) TTR Function as Approximate Reward: Consider the real reward function in formal MDP framework: \(r(s) : S \rightarrow \mathbb{R}\) is the reward function that governs the reward signal when in state \(s\). Note that here the state \(s\) in MDP could be high dimensional states that is usually regard as input of model-free learning algorithm. Now we define \(\tilde{r}(\tilde{s}) : \tilde{S} \rightarrow \mathbb{R}\) as the approximate reward function which only embraces a subset of full MDP state. Note that here \(\tilde{s}\) refers to a subset of full state \(s\) and is the set of state variables in the approximate physical model we have chosen in (7) or (8). In our method, the TTR function \(\tilde{r}(\tilde{s})\) defined in (4) is used as the approximate reward function \(\tilde{r}(\tilde{s})\).

\[
r(s) \approx \tilde{r}(\tilde{s}) = -\phi(\bar{\tilde{s}})
\]

By definition, \(\phi(\bar{\tilde{s}})\) is non-negative and \(\phi(\bar{\tilde{s}}) = 0\) if and only if \(\bar{\tilde{s}} \in \Gamma\). Here we use \(-\phi(\cdot)\) as reward because the state with less arriving time to the goal should be given higher reward. The reason that we apply TTR function \(\phi(\cdot)\) as approximate reward in RL process is three-folds: firstly, the reward function in learning process doesn’t need to be highly accurate, thus the TTR function would be a good approximation of reward only if it is able to capture key system behaviors and that is sufficient for guiding policy learning; secondly, the high dimensional state in most RL task may include sensor data which is usually not directly relevant to the reward; thirdly, TTR function can be efficiently computed for a simplified system model that contains a subset of the MDP states.

\(^1\)Note that the meaning of “model selection” here is different from that in machine learning.
To reduce the computational complexity of solving the PDE on complicated system dynamics (such as quadrotor), we may apply system decomposition methods [27]–[29] to obtain approximate TTR function without significantly impacting the overall policy training time. Particularly, we first decomposed the entire system into several sub-systems with overlapping components of state variables, thus each sub-system can be solved efficiently. Specifically, we use the open source helperOC\textsuperscript{2} and Level Set Methods toolboxes [27] in MATLAB.

V. EXPERIMENTS

We perform simulation experiments on two dynamical systems: generic car system and planar quadrotor system, using Gazebo, a robot simulator with robust physical engine [36]. Also we evaluate our reward shaping method on a variety of measurements relating to performance and efficiency. We apply PPO [15] as our model-free policy optimization method.

A. Simple Car Model

A five dimensional car model is used in our first experiment. Car model is widely used as standard test environment in motion planning [36] and RL [37]. The state of car is represented as a vector with five elements: \( s = (x, y, \theta, v, \omega) \), where \( x \) and \( y \) denote position in the plane, \( \theta \) denotes heading angle, \( v \) denotes linear velocity, and \( \omega \) denotes angular velocity. The learning agent receives an observation that is augmented based on 5D state with 8 range and bearing measurements, which may be obtained through a laser rangefinder. These measurements provide the distance and direction from the vehicle to the nearest obstacle, and the learning agent must learn to map the augmented state, which includes the 5 internal states \( (x, y, \theta, v, \omega) \) augmented with the 8 measurements. The control input of this system is \( a = [v', \omega'] \), where \( v' \) is the linear velocity of car, \( \omega' \) is the angular velocity. The car starts from some initial condition, and aims to reach some goal region without colliding with obstacles. We set the goal state as a point at upper-right corner with \( x = 4, y = 4, \theta = 0 \). The states that are within 0.3m in positional distance and 0.3 radian in heading angle of the goal state are considered to have reached the goal.

We apply three different kinds of reward functions on same problem setting as comparison: one is sparse reward, where agent obtain reward 1000 at goal states, reward \(-400\) when collision happens, and 0 reward at any other states. The second one is the commonly-used distance reward which measures the Euclidean distance from any position to goal while agent obtain same reward at goal states and collision states. The last one is TTR-based reward, and agent also obtain same reward at goal states and collision states, but negative TTR value as reward at any other states. Note that TTR value is computed based on the system dynamics we’ve selected, so at any state, there will be a corresponding dynamics-informed TTR-based reward indicating “how good” current state is in terms of arriving time to goal state.

Figures 2-4 show the performance of standard PPO algorithm [15] on the car model with TTR-based reward, distance reward and sparse reward. All statistics are based on five different runs with the background shadow representing the

\textsuperscript{2}helperOC: https://github.com/HJReachability/helperOC
95 percentage confidence interval. Note that in Figure 3, one algorithm iteration contains 30 PPO iterations in Figure 2. Figure 3 indicates the success percentage of agent reaching goal versus training iterations of outer loop. Note that the success percentage is measured by the percent between the number of episodes ending with goal state and the total number of episodes in one out loop iteration. As we can see, TTR-based reward shaping achieves fast convergence and high success rate based on all five experiments in under ten algorithm iterations. In contrast, with sparse reward it hardly learn any useful policy to reach the goal since the occurrence of strong reward signals is too rare to support valid policy optimization. For distance reward, even though sometimes it will perform a little better than sparse reward, it’s very unstable across different runs of same experiment especially when the optimal trajectory to the goal position includes large curvature.

Figure 2 shows the average reward during training process. The average reward is computed among all sampled episodes in one single PPO iteration. As a result, with the similar tendency as success percentage, TTR-based reward shaping rapidly reaches a high and stable cumulative reward level after around 50 or 60 PPO iterations while there are a few oscillations during exploration at beginning. We also evaluating three trained policies in the identical environment with same sparse reward setting and the results are shown in figure 4. It’s easy to notice that the control policy trained with TTR-based reward is performing well among all episodes. Nevertheless, the control policies trained with distance and sparse perform much worse since the agent never learns how to deliberately arrive to the goal.

B. Planar Quadrotor Model

Our second experiment is conducted on a simulated planar quadrotor model in Gazebo to evaluate the performance of our TTR-based reward shaping method on highly dynamic and unstable system. This planar quadrotor is a standard test problem in the control literature [36], [37]. "Planar" means the quadrotor would only fly in vertical plane without changing the roll and yaw angle. The system has the state $s = (x, v_x, z, v_z, \phi, \omega)$, where $x, z, \phi$ denote the planar coordinates and pitch, and $v_x, v_z, \omega$ denote their time derivatives. Quadrotor’s movement is controlled by the force difference between two motor thrusts, $T_1$ and $T_2$. Similar as the task on car model, the observation of agent is augmented based on state by 8 laser rays averagely extracted from “Hokoyu utm30lx” ranging sensor for detecting nearest obstacles. The goal region is defined as $S_g = \{s : 3.5 \leq x \leq 4.5, 8.5 \leq z \leq 9.5, 0.4 \leq \phi \leq 1.0\}$ and the states in this region are considered to have reached the goal. For PPO training, the output of network is the mean and variance of Gaussian distribution depicting the distribution of continuous action in stochastic sense, thus we could sample the action from this Gaussian distribution.

Fig. 5: Performance visualization of car in gazebo simulator. The picture is a combination of the same car at different time snapshots.

Fig. 6: The average cumulative reward obtained from three different reward functions during training on quadrotor model

Fig. 7: The success rate on reaching the goal obtained from three different reward functions during training on quadrotor model

Similar to generic car problem, we apply three different reward function for comparison. For sparse reward setting, agent receives a reward of 1000 for reaching goal region and
a reward of $-400$ for collision penalty. At any other states, agent receives 0 reward. Distance reward setting measures the Euclidean distance from any state position to the goal while agent obtain same reward at goal states and collision states. TTR-based reward function preserve exactly same reward information for goal region and collisions but assign negative TTR value as reward to quadrotor at each timestep. The episode runs for at most 512 timesteps and the maximum thrust of quadrotor is set to be 1.5 times the weight of itself to simulate reality as much as possible. As is shown in our result, with the extremely sparse reward associated with reaching the goal on unstable highly dynamic system, it’s extremely difficult to learn a good policy. However, with a simple replacement with dynamics-informed TTR-based reward function, not only the performance but efficiency of agent’s learning is greatly improved.

Figures 6–8 show the performance of learning when use TTR-based reward function as well as the sparse reward and distance reward. Sparse reward, due to the extremely lack of effective feedback signal, never guide the agent reach the goal and thus cannot learn to deliberately move toward the goal region. Instead, it learns a locally optimal policy which hovers in small safe region to avoid any collision. The distance reward only consider the absolute distance between the agent and goal instead of the angle information, thus quadrotor always tries to reach the neighborhood area of goal region in a near-hover configuration. However, in our environment setting, quadrotor has much more difficulties in arriving the goal without any tilting motion. Thus quadrotor receiving distance reward will finally stay hovering nearby the goal but never produce any deliberate behaviors tilting towards the goal.

In contrast, as shown in Figure 9, the TTR-based reward performs very well in guiding quadrotor reaching the goal along shortest trajectories based on its’ start state in highly tilting status. During the training process, the TTR-based reward leads the quadrotor approach the goal area not only considering the distance between their positions but also the angular tendency. Gradually, the quadrotor find out that it would accumulate higher reward if it manages to arrive the goal within less time with highly tilting motions. The quadrotor can learn such complex behavior using TTR-based reward because it has captured the model dynamics in all state dimensions including pitch angles.

Specifically, figure 6 and 7 plots the training curves in terms of cumulative reward and success rate versus training iterations. As we can see, even though the learning process via TTR-based reward has comparatively large variance, it will finally search for a proper trajectory to the goal thus receive highest reward and success rate compared to the distance and sparse reward. This phenomenon illustrates that the TTR-based reward is able to encourage the agent to explore even more efficiently to avoid being stuck. And the evaluation figure 8 tells the same story that the policy learned from TTR-based reward will keep performing good and definitely beats the policy learned via distance or sparse reward especially when the task requires highly complicated motions, like quadrotor flying in a tilting manner.

VI. CONCLUSION

In this paper, we proposed TTR-based reward shaping method, which addresses the challenge of high sample complexity in model-free reinforcement learning (RL) by using TTR function to provide reward at each timestep for guiding the policy learning process. In our approach, we first pick up an system model which could be the same one as forward learning process or a simplified model preserving main elements describing reward function. By computing TTR function based on the model we have chosen at the beginning of learning process and integrate it into training as reward at each timestep, the agent receive more dynamics-informed feedback and learns better and faster. TTR-based reward shaping is an effective way to leverage physical priors to dramatically increase the speed of training for model-free algorithms without affecting their flexibility to a large variety of complicated RL problems.

There are still several possible choices of future work. Since usually there will be obstacles in the scene of goal-

3A short video demo of our work: https://youtu.be/KdA2UGr6T4g
oriented RL tasks, it would be of interest if the info of obstacles (position, shape) could be incorporated as part of model dynamics to develop more informative TTR-based reward for learning policies. Moreover, TTR-based reward shaping method has potential to replace the sparse reward setting when physical model could be accessible and could be integrated with other deep exploration techniques like hierarchical reinforcement learning to obtain even better experimental performance.

REFERENCES

[1] A. S. Polydoros and L. Nalpantidias, “Survey of model-based reinforcement learning: Applications on robotics,” J. Intelligent & Robotic Systems, vol. 86, no. 2, pp. 153–173, May 2017.

[2] A. m. Farahmand, A. Shademan, M. Jagersand, and C. Szepesvari, “Model-based and model-free reinforcement learning for visual servoing,” in Proc. IEEE Int. Conf. Robotics and Automation, 2009.

[3] P. Auer, T. Jakobs, and R. Ortner, “Near-optimal regret bounds for reinforcement learning,” in Advances in Neural Information Processing Systems 21, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 89–96.

[4] A. Guez, D. Silver, and P. Dayan, “Efficient bayes-adaptive reinforcement learning using sample-based search,” in Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1025–1033.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, “Curriculum learning,” in Proc. Annual Int. Conf. Machine Learning, 2009.

[6] C. Florensa, D. Held, M. Wulfmeier, and P. Abbeel, “Reverse curriculum generation for reinforcement learning,” CoRR, 2017. [Online]. Available: http://arxiv.org/abs/1707.05300

[7] Y. Bengio, O. Klimov, “Proximal policy optimization algorithms,” CoRR, vol. abs/1707.06347, 2017.

[8] M. Bardi and P. Soravia, “Hamilton-jacobi equations with singular Hamiltonians and optimal control problems,” Transactions of the American Mathematical Society, vol. 325, no. 1, pp. 205–229, 1991.

[9] A. Y. Ng, D. Harada, and S. J. Russell, “Algorithms for inverse reinforcement learning,” in Proc. Annual Conf. Computational Learning Theory, 1998.

[10] A. Y. Ng and S. J. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in Proc. Annual Int. Conf. Machine Learning, 1999.

[11] Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” J. Machine Learning Research, vol. 15, pp. 2673–2694, 2014.

[12] J. Schmidhuber, “Deep learning in neural networks: An overview,” Proc. IEEE Int. Conf. Machine Learning and Comput., 2013.

[13] S. Russell, “Learning agents for uncertain environments (extended abstract),” in Proc. Ann. Conf. Neural Information Processing Systems, 1995.

[14] S. Russell and A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.

[15] M. Chen, S. L. Herbert, M. S. Vashishtha, S. Bansal, and C. J. Tomlin, “Decomposition of reachable sets and tubes for a class of nonlinear systems,” IEEE Transactions on Automatic Control, vol. 63, no. 11, pp. 3675–3688, Nov 2018.

[16] R. J. Williams, “Simple statistical gradient-following algorithms for connectionist reinforcement learning,” Machine Learning, vol. 8, no. 3, pp. 229–256, May 1992.

[17] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, “Hierarchical reinforcement learning,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots and Syst., 1999.

[18] R. S. Sutton, D. Precup, and S. Singh, “Between mdps and semi-mdps: Reinforcement learning in partially observable environments,” in Proc. AAAI Conf. Artificial Intelligence, 1999.

[19] T. G. Dietterich, “The maxq method for hierarchical reinforcement learning,” in Proc. IEEE Int. Conf. Robotics and Automation, 2000.

[20] A. Y. Ng, D. Harada, and S. J. Russell, “Policy invariance under reward transformations: Theory and application to reward shaping,” in Proc. Annual Int. Conf. Machine Learning, 1999.