K-Score: Kalman Filter as a Principled Alternative to Reward Normalization in Reinforcement Learning

Zixuan Xia **Quanxi Li**

Abstract

We propose a simple yet effective alternative to reward normalization in policy gradient reinforcement learning by integrating a 1D Kalman filter for online reward estimation. Instead of relying on fixed heuristics, our method recursively estimates the latent reward mean, smoothing high-variance returns and adapting to non-stationary environments. This approach incurs minimal overhead and requires no modification to existing policy architectures. Experiments on *LunarLander* and *CartPole* demonstrate that Kalman-filtered rewards significantly accelerate convergence and reduce training variance compared to standard normalization techniques.

1 Introduction

Policy gradient methods are a central class of algorithms in reinforcement learning (RL), particularly effective in continuous action domains [8]. A critical component of their success lies in controlling the variance of the reward signal to stabilize training. Among the commonly adopted heuristics, *reward normalization*—typically via Z-score standardization—has proven useful for mitigating the instability of high-variance returns [4, 2]. However, such normalization schemes often rely on the assumption of stationarity in the reward distribution and offer limited interpretability or adaptability in dynamic environments.

This paper investigates a principled alternative: replacing reward normalization with Kalman filtering [1], a Bayesian recursive estimation technique. Kalman filters have been widely used in robotics and control for state estimation in noisy environments [10], yet remain underexplored in the context of reward estimation in RL. By leveraging the Kalman filter's ability to infer the latent mean of a noisy and potentially non-stationary reward signal, we aim to construct a more robust and adaptive mechanism for policy updates.

We propose a lightweight 1D Kalman filter to estimate the expected reward at each time step, thereby smoothing out high-variance observations and enabling faster and more stable convergence. This approach is especially promising in environments with fluctuating reward dynamics or partial observability. Compared to heuristic normalization, the Kalman filter provides a theoretically grounded mechanism that adapts naturally to changes in the reward signal.

To evaluate the efficacy of our method, we conduct experiments on classic control tasks—*LunarLander* and *CartPole*—from OpenAI Gym[1]. Our results demonstrate that Kalman-filtered reward estimation improves learning stability and accelerates convergence compared to traditional normalization strategies.

[1] `https://gymnasium.farama.org/`

Preprint. Under review.

2 Related Work

2.1 Policy Gradient Methods

Policy gradient algorithms directly optimize the expected return by computing gradients of the policy with respect to its parameters [8]. The REINFORCE algorithm [11] provides an unbiased but high-variance estimator, while actor-critic methods reduce variance by introducing a learned value function as a baseline. Generalized Advantage Estimation (GAE) further improves training stability by providing a more flexible bias-variance trade-off in estimating advantages. Proximal Policy Optimization (PPO) [6] builds on these ideas by enforcing conservative policy updates through clipped objective functions.

2.2 Reward Normalization

Reward normalization is widely adopted to reduce the variance of gradient estimates and speed up convergence in policy-based RL. Z-score normalization is a common approach, where rewards are standardized using a running estimate of their mean and standard deviation [2]. More recent work explores adaptive normalization strategies, such as PopArt [9], which normalizes value function targets while preserving output scale. However, these methods typically assume stationarity in the reward distribution and lack principled probabilistic foundations, making them brittle in non-stationary environments.

2.3 Kalman Filtering

Kalman filtering is a classic Bayesian technique for online state estimation under noise [1, 10]. Several works have explored the integration of Kalman filtering into reinforcement learning, primarily for value estimation and uncertainty modeling. Shashua and Mannor [7] introduced a Kalman-based update mechanism for trust region policy optimization, where the filter was used to track the uncertainty in value function approximations. Similarly, Parr et al. [3] proposed kernel-based reinforcement learning frameworks that leverage Kalman filters for local value estimation in continuous state spaces. More recently, filtering-based approaches have been applied in partially observable settings, where Kalman filtering is used for hidden state inference and belief updating [9].

3 Methods

3.1 Kalman Filtering for Reward Estimation

We model the true expected reward at each time step as a latent variable r_t that evolves over time. Instead of normalizing observed rewards using running statistics, we use a Kalman filter to maintain an adaptive estimate of the underlying reward signal.

Let the true reward evolve as a random walk:

$$r_t = r_{t-1} + w_t, \quad w_t \sim \mathcal{N}(0, Q), \tag{1}$$

where Q is the process noise variance capturing the potential drift in the true reward over time. The observed reward at time t is modeled as:

$$\hat{r}_t = r_t + v_t, \quad v_t \sim \mathcal{N}(0, R), \tag{2}$$

where v_t is zero-mean measurement noise with variance R.

The Kalman filter computes a recursive posterior estimate of r_t via a two-step update:

Predict step:

$$r_{t|t-1} = r_{t-1}, \tag{3}$$

$$P_{t|t-1} = P_{t-1} + Q, \tag{4}$$

Update step:

$$K_t = \frac{P_{t|t-1}}{P_{t|t-1} + R}, \tag{5}$$

$$r_t = r_{t|t-1} + K_t(\hat{r}_t - r_{t|t-1}), \tag{6}$$

$$P_t = (1 - K_t)P_{t|t-1}, \tag{7}$$

where $r_{t|t-1}$ is the prior estimate, P_t is the posterior variance, and K_t is the Kalman gain determining how much to trust the new observation.

This filter behaves like an adaptive exponential moving average: when the observed rewards are noisy (R is large), the filter relies more on past estimates; when the observations are more reliable, the filter quickly adapts to new data. Crucially, unlike sample means or Z-score normalization, this formulation explicitly tracks uncertainty and adjusts its confidence over time.

We replace the standard normalized rewards (e.g., Z-scored returns) with the filtered reward estimate r_t in all downstream computations, including returns and advantage estimates.

3.2 Variance Behavior and Convergence Analysis

We compare the error behavior of the Kalman filter with that of a standard sample mean estimator under the same noisy reward observation model. Let the observed reward at time t be $\hat{r}_t = r_{\text{true}} + v_t$, where $v_t \sim \mathcal{N}(0, \sigma^2)$ and r_{true} is constant. The sample mean after t steps is:

$$\bar{r}_t = \frac{1}{t}\sum_{i=1}^{t} \hat{r}_i, \tag{8}$$

with a mean squared error (MSE) of:

$$\text{MSE}(\bar{r}_t) = \frac{\sigma^2}{t}. \tag{9}$$

This estimator converges slowly and is highly sensitive to early outliers when t is small, which is often the case in early RL training.

In contrast, the Kalman filter maintains an adaptive variance estimate P_t that converges to a steady-state value P_∞:

$$P_\infty = \frac{\sqrt{Q^2 + 4QR} - Q}{2} \approx cQ \tag{10}$$

where Q and R are the process and measurement noise variances, and c is some constant, respectively. This implies that the Kalman filter's error does not indefinitely shrink with more samples, but it reaches a reliable equilibrium that balances prior stability and new evidence.

The Kalman estimator outperforms the sample mean when:

$$\frac{\sigma^2}{t} > P_\infty \quad \Rightarrow \quad t < \frac{\sigma^2}{P_\infty} \tag{11}$$

i.e., during the early stages of training when sample sizes are small. This regime is particularly important in reinforcement learning, where the agent often begins with high reward noise and limited data.

3.3 Integration into Policy Gradient Algorithms

We integrate our adaptive Kalman filtering module directly into the return processing stage of general policy gradient methods. Instead of computing running statistics (mean and variance) over the return-to-go G_t, we recursively estimate a filtered reward baseline using a Kalman filter. This provides an online, adaptive normalization of rewards, which helps reduce variance and stabilize training, especially in early-stage or sparse-reward regimes.

In the standard policy gradient formulation, the gradient of the expected return is:

$$\nabla_\theta J(\theta) = \mathbb{E}_\pi \left[\nabla_\theta \log \pi_\theta(a_t|s_t) \cdot G_t\right] \tag{12}$$

where G_t denotes the return-to-go. We propose to replace G_t with a normalized version:

$$\hat{G}_t = \frac{G_t - x_t}{\sqrt{P_t + \epsilon}} \tag{13}$$

where x_t and P_t are the estimated mean and variance of G_t obtained from a 1D Kalman filter. This normalization smooths the reward signal while adaptively adjusting to changing reward dynamics.

The full training procedure with Kalman-based return normalization is shown in Algorithm 1

Algorithm 1 Policy Gradient with Kalman-Normalized Returns

Initialize policy parameters θ, Kalman state (x_0, P_0), noise parameters Q, R, decay factor α
for each iteration $t = 1$ to T **do**
  Sample trajectory $\tau = \{(s_t, a_t, r_t)\}_{t=1}^{N}$ using policy π_θ
  Compute discounted returns: $G_t = \sum_{k=t}^{N} \gamma^{k-t} r_k$
  Initialize Kalman filter $\mathcal{K}$ with (x_0, P_0)
  for each timestep t in τ **do**
    $(x_t, P_t) \leftarrow \mathcal{K}$.update$(G_t)$
    Normalize return: $\hat{G}_t \leftarrow \dfrac{G_t - x_t}{\sqrt{P_t + \epsilon}}$
  end for
  Compute policy gradient estimate:

$$g = \frac{1}{N} \sum_{t=1}^{N} \nabla_\theta \log \pi_\theta(a_t|s_t) \cdot \hat{G}_t \tag{14}$$

  Update policy: $\theta \leftarrow \theta + \eta \cdot g$
end for

return final policy π_θ

This modification introduces no changes to the policy network architecture and incurs negligible computational overhead. Since the Kalman filter operates on scalar returns and maintains only a few internal variables, it can be efficiently applied at each training step. Compared to ad-hoc normalization techniques, our method provides a principled and adaptive alternative for stabilizing policy optimization.

4 Experiment

4.1 Experimental Setup

We evaluate the effectiveness of our Kalman-based reward normalization across two classic control environments from OpenAI Gym:

- **CartPole-v1**: A simple balancing task with a discrete action space and dense reward. The agent receives $+1$ for every timestep it balances the pole, up to a maximum episode length of 500. The environment is considered solved when the average reward over 100 episodes exceeds 475.
- **LunarLander-v2**: A more challenging control task with a larger discrete action space and sparse, high-variance rewards. The task is considered solved when the agent achieves an average reward above 200.

All algorithms are implemented using PyTorch and trained using the standard OpenAI Gym interface. Each model is trained for a fixed number of episodes or until it reaches the predefined reward threshold. Due to time constraints and in order to reduce randomness to some content, we set up both a training environment and a separate evaluation environment for each baseline. These environments share identical configurations but are initialized with different random seeds. Training is terminated once the agent reaches the predefined `REWARD_THRESHOLD` in the evaluation environment.

For our Kalman reward normalization, we implement two variants:

- **Simple Kalman**: uses fixed process and measurement noise values (Q, R) selected via grid search for each model and environment.
- **Adaptive Kalman**: adaptively updates the measurement noise R_t using an exponential moving average of the squared residuals, with update rule

$$R_t = \alpha R_{t-1} + (1 - \alpha)(z_t - x_{t-1})^2 \tag{15}$$

  where $\alpha = 0.9$ unless otherwise specified.

These variants are used in both baseline comparisons and ablation studies to evaluate the impact of dynamic versus fixed noise modeling.

4.2 Baselines and Variants

To evaluate the effect of Kalman-based reward normalization, we compare its performance against several widely used policy gradient methods with standard return or advantage estimation techniques. The baselines include:

- **Actor-Critic (AC)**: A standard policy gradient method where the advantage is computed as $A_t = G_t - V(s_t)$ using a learned value function as a baseline.
- **Generalized Advantage Estimation (GAE)** [5]: Computes a bias-variance tradeoff approximation of the advantage by using an exponentially weighted sum of temporal differences.
- **Proximal Policy Optimization (PPO)** [6]: A robust on-policy optimization method using clipped surrogate objectives and value function baselines.

We augment each of these methods with our proposed Kalman-based return normalization. Specifically, we apply a 1D Kalman filter to the computed return G_t before it is used in policy gradient or critic updates. This gives rise to two variants:

- **Simple Kalman**: Uses a fixed noise Kalman filter for reward normalization, with (Q, R) selected via grid search.
- **Adaptive Kalman**: Updates the measurement noise R_t dynamically using an exponential moving average of residuals, as described in Section 4.1.

These variants allow us to examine the effect of reward normalization independently of the underlying optimization algorithm, and to study the benefit of adaptive uncertainty modeling in high-variance environments.

4.3 Evaluation Metric

To evaluate and compare convergence speed across different algorithms and normalization strategies, we adopt a threshold-based metric. For each environment, we define a fixed `REWARD_THRESHOLD` and measure the number of training episodes required for the agent to reach or exceed this threshold in the evaluation environment.

- **CartPole-v1**: `REWARD_THRESHOLD` is set to 475, consistent with the standard OpenAI Gym success criterion.
- **LunarLander-v2**: `REWARD_THRESHOLD` is set to 200, indicating successful landing behavior over multiple trials.

For each method, we record the number of episodes needed until the evaluation environment achieves an average return above the threshold. As described in Section 4.1, training is terminated as soon as this condition is met. This setup reflects a practical goal in real-world RL applications: reaching reliable performance with minimal training steps.

The reported metric is the episodes to convergence, which can represent the learning efficiency, rather than final asymptotic performance, and highlights the potential of Kalman-based reward normalization to accelerate policy optimization.

4.4 Results and Analysis

We present our experimental results in terms of convergence speed, i.e., the number of episodes required to reach the predefined reward threshold on the evaluation environment.

CartPole: Actor-Critic and GAE Figure 1 and Figure 2 compare reward curves on CartPole using two normalization methods under the Actor-Critic and GAE frameworks, respectively. In both cases, the left subplot shows results with standard mean-std normalization, while the right uses Kalman-based reward filtering. Kalman normalization consistently achieves faster convergence and lower variance, particularly during the early training phase.

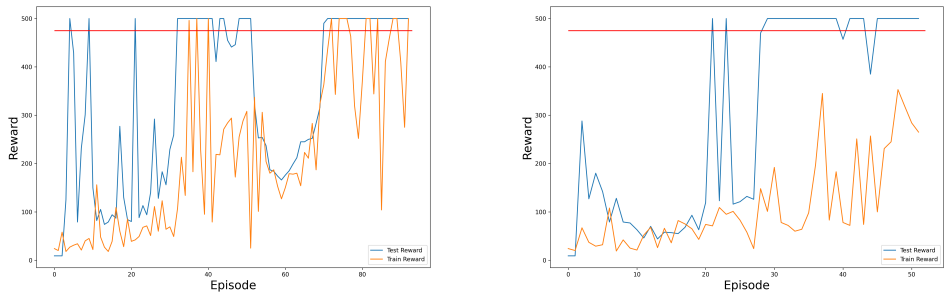

Figure 1: Left: Actor-Critic on CartPole using Mean-Std normalization. Right: Actor-Critic on CartPole using Kalman normalization. Kalman filtering achieves faster and more stable convergence.

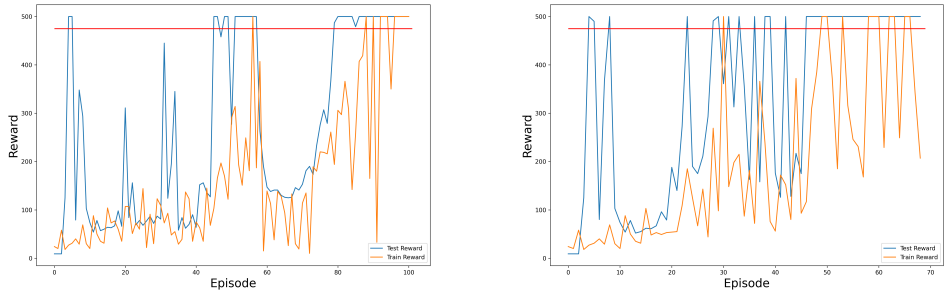

Figure 2: Left: GAE on CartPole using Mean-Std normalization. Right: GAE on CartPole using Kalman normalization. Kalman-based smoothing reduces reward variance and accelerates convergence.

LunarLander: Actor-Critic Comparison Figure 3 shows the training reward per episode on LunarLander using Actor-Critic. Kalman normalization significantly outperforms standard mean-std normalization, reaching the threshold in fewer episodes with higher reward stability.

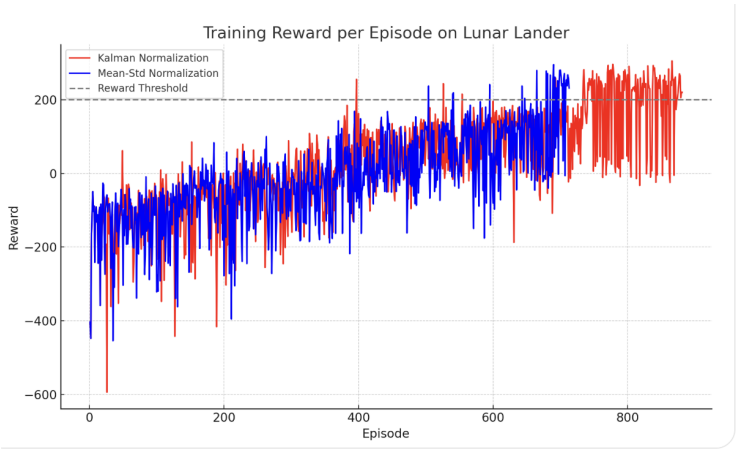

Figure 3: Actor-Critic on LunarLander: Kalman vs. Mean-Std Normalization

PPO on CartPole: Comparing Normalization Methods. To further evaluate the generality of our approach, we apply Proximal Policy Optimization (PPO)[6] to the CartPole environment using three different normalization strategies: adaptive Kalman normalization, simple Kalman normalization, and conventional mean-std normalization.

As shown in Figure 4, the adaptive Kalman variant (left) achieves the fastest and most stable convergence, consistently reaching the threshold within 60 episodes. In contrast, the mean-std normalization baseline (center) exhibits slower convergence and higher variance, often requiring more than 90 episodes to stabilize. The simple Kalman version (right), while still outperforming mean-std normalization, underperforms its adaptive counterpart due to the use of fixed noise parameters.

These results provide additional evidence that adaptively adjusting the noise scale in Kalman filtering leads to superior learning dynamics compared to fixed heuristics or manually chosen hyperparameters.

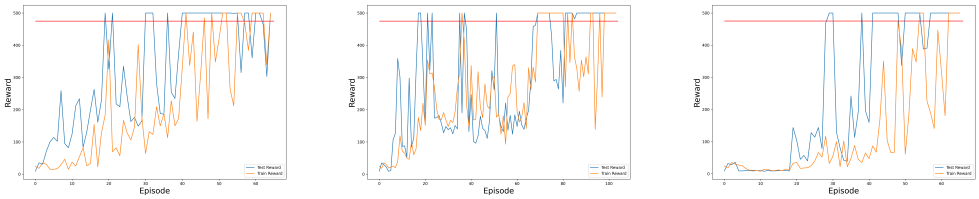

Figure 4: PPO on CartPole using (Left) Adaptive Kalman normalization, (Center) Mean-Std normalization, and (Right) Simple Kalman normalization. Adaptive Kalman normalization achieves faster and more stable convergence.

Ablation Study for Adaptive Kalman Filtering To evaluate the impact of noise modeling in Kalman filtering, we conduct an ablation study on CartPole using GAE[5]. We fix the measurement noise $R = 1$ and vary the process noise Q from 10^{-4} to 1, observing how the convergence speed changes. As shown in Table 1, smaller Q values lead to faster convergence, confirming that properly tuned or adaptively learned Kalman parameters can significantly influence learning efficiency. Notice that, although for the standard mean-std normalization, it will take 151 episodes to reach the threshold.

Note: The same agent using standard mean-std normalization requires 104 episodes on average to reach the reward threshold. So we believe that in most cases, if we can select the noise factors in a correct way, we will be more likely to reach the reward threshold in fewer episodes, which means we can get a faster convergence.

Table 1: Effect of process noise Q on convergence speed (PPO on CartPole, $R = 1$ fixed)

Q	R	Episodes to Threshold
1×10^{-4}	1	75
1×10^{-3}	1	71
1×10^{-2}	1	69
1×10^{-1}	1	170
1	1	98

Across all experiments, Kalman-based reward normalization consistently accelerates convergence compared to mean-std normalization. The improvement is particularly pronounced in early training stages and high-variance environments like LunarLander. Furthermore, adaptive adjustment of noise parameters (e.g., R_t) provides additional benefits, supporting our hypothesis that Kalman filters offer a principled and efficient reward normalization strategy.

5 Conclusions, Limitations and Future Work

Conclusions. We proposed a lightweight, principled alternative to reward normalization in policy gradient reinforcement learning based on Kalman filtering. By estimating a smoothed and adaptive baseline from the return signal, our method improves convergence speed and stability across multiple algorithms and environments. Empirical results on CartPole and LunarLander show that Kalman-based normalization outperforms standard mean-std normalization in early training and high-variance scenarios, with particularly strong gains in sample efficiency.

Limitations. Despite these promising results, our approach has several limitations. First, both the simple and adaptive Kalman variants require manual initialization of the process and measurement noise parameters (Q, R). Although these can be tuned per task, the absence of an automated or learned mechanism restricts general applicability. Second, our method uses a one-dimensional Kalman filter applied solely to the scalar return signal G_t. While effective in simple environments, this limits the capacity to capture temporal or structural dependencies present in more complex tasks.

Future Work. Future research can explore multi-dimensional or structured extensions of the Kalman filter, enabling richer representations of uncertainty that consider state dynamics or policy structure. One natural direction is to make the Kalman filter fully differentiable and learnable within an end-to-end training framework, allowing adaptive estimation of (Q, R) directly from data. Additionally, applying Kalman filtering beyond reward normalization—such as to value estimation or advantage calculation—could further improve learning stability. Finally, evaluating this approach in continuous control, offline RL, and safety-critical domains would validate its robustness and broader utility.

References

[1] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[2] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PmLR, 2016.

[3] Christopher Painter-Wakefield and Ronald Parr. Greedy algorithms for sparse reinforcement learning. *arXiv preprint arXiv:1206.6485*, 2012.

[4] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[5] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[6] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[7] Shirli Di-Castro Shashua and Shie Mannor. Trust region value optimization using kalman filtering. *arXiv preprint arXiv:1901.07860*, 2019.

[8] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems*, 12, 1999.

[9] Hado Van Hasselt, Yotam Doron, Florian Strub, Matteo Hessel, Nicolas Sonnerat, and Joseph Modayil. Deep reinforcement learning and the deadly triad. *arXiv preprint arXiv:1812.02648*, 2018.

[10] Greg Welch, Gary Bishop, et al. An introduction to the kalman filter. 1995.

[11] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8:229–256, 1992.