7 APPENDIX

7.1 Proofs

7.1.1 Proof of Lemma 3.2

By Definition 3.1, $\pi_\theta(s,a)$ should satisfy the following condition:

$$\begin{aligned}\int_{T_\epsilon} q(\epsilon)d\epsilon &= \int_{T_a} \pi_\theta(s,a)da \\ &= \int_{T_\epsilon} \pi_\theta(s, g_\theta(s,\epsilon)) \cdot \Big|\det(\frac{\partial g_\theta(s,\epsilon)}{\partial \epsilon})\Big| d\epsilon \quad \text{(by change of variable).}\end{aligned}$$

We denote the set of continuous distributions that satisfy this condition as P. That is,

$$P_\theta(s,\epsilon) \in P \text{ if } \int_{T_\epsilon} q(\epsilon)d\epsilon = \int_{T_\epsilon} P_\theta(s,\epsilon) \cdot \Big|\det(\frac{\partial g_\theta(s,\epsilon)}{\partial \epsilon})\Big| d\epsilon.$$

Clearly, $\bar{P}_\theta(s,\epsilon) = q(\epsilon) \cdot \Big|\det(\frac{\partial g_\theta(s,\epsilon)}{\partial \epsilon})\Big|^{-1}$ is an element of P, and is well defined, because of Equation 3. In fact, $\bar{P}_\theta(s,\epsilon)$ is the only element of P, as the above condition should be met for any $T_\epsilon \subset \mathbb{R}^n$. That is, if there was another distribution $\tilde{P}_\theta(s,\epsilon) \in P$,

$$\int_{T_\epsilon} (\bar{P}_\theta(s,\epsilon) - \tilde{P}_\theta(s,\epsilon)) \cdot \Big|\det(\frac{\partial g_\theta(s,\epsilon)}{\partial \epsilon})\Big| d\epsilon = 0, \forall T_\epsilon \subset \mathbb{R}^n.$$

Since $\tilde{P}_\theta$ is different from $\bar{P}_\theta$, there exists an open set $\hat{T}_\epsilon$ where $\tilde{P}_\theta < \bar{P}_\theta$. Then

$$\int_{\hat{T}_\epsilon} (\bar{P}_\theta(s,\epsilon) - \tilde{P}_\theta(s,\epsilon)) \cdot \Big|\det(\frac{\partial g_\theta(s,\epsilon)}{\partial \epsilon})\Big| d\epsilon > 0,$$

which is a contradiction. Therefore, $\pi_\theta(s,a) = \pi_\theta(s, g_\theta(s,\epsilon)) = \bar{P}_\theta(s,\epsilon)$. The inverse holds trivially.

7.1.2 Proof of Definition 5

First, we show that f is an injective function by showing that the determinant of Jacobian $\frac{\partial f}{\partial a}$ is always positive:

$$\begin{aligned}\det(\frac{\partial f}{\partial a}) &= \det(I + \alpha \cdot \nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a)) \\ &= \Pi_{i=1}^n (1 + \alpha \cdot \lambda_i(s,a)) > 0 \quad (\because |\alpha| < \frac{1}{\max_{(s,a)}|\lambda_1(s,a)|}),\end{aligned}$$

where $\lambda_i(s,a)$ are the eigenvalues of $\nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a)$ sorted in ascending order.

Then, for an arbitrary open set of action $T \subset A$, π_α selects $\tilde{T} = f(T) \subset A$ with the same probability that the original policy $\pi_{\bar{\theta}}$ selects T.

$$\begin{aligned}\int_{\tilde{T}} \pi_\alpha(s,\tilde{a})d\tilde{a} &= \int_T \frac{\pi_{\bar{\theta}}(s,a)}{|\det(I + \alpha \nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))|} \cdot \Big|\det(\frac{\partial f}{\partial a})\Big| da \\ &= \int_T \frac{\pi_{\bar{\theta}}(s,a)}{|\det(I + \alpha \nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))|} \cdot \Big|\det(I + \alpha \cdot \nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))\Big| da \\ &= \int_T \pi_{\bar{\theta}}(s,a)da.\end{aligned}$$

Therefore, $\pi_\alpha(s,\cdot)$ is a valid probability distribution as $\pi_{\bar{\theta}}(s,\cdot)$.

7.1.3 Proof of Proposition 4.2

For a given state s, we can estimate the (approximate) expected value of the state under π_α as follows.

$$\begin{aligned}
&\int_{\tilde{a}} \pi_\alpha(s,\tilde{a}) A_{\pi_{\bar{\theta}}}(s,\tilde{a}) d\tilde{a} \\
&= \int_a \frac{\pi_{\bar{\theta}}(s,a)}{|\det(I+\alpha\nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))|} A_{\pi_{\bar{\theta}}}(s, a+\alpha\nabla_a A_{\pi_{\bar{\theta}}}(s,a)) |\det(I+\alpha\nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))| da \\
&\approx \int_a \pi_{\bar{\theta}}(s,a) \left[A_{\pi_{\bar{\theta}}}(s,a) + \alpha ||\nabla_a A_{\pi_{\bar{\theta}}}(s,a)||^2 \right] da \\
&= \int_a \pi_{\bar{\theta}}(s,a) A_{\pi_{\bar{\theta}}}(s,a) da + \alpha \int_a \pi_{\bar{\theta}}(s,a) ||\nabla_a A_{\pi_{\bar{\theta}}}(s,a)||^2 da.
\end{aligned}$$

Therefore, we can see the following holds:

$$\begin{aligned}
L_{\pi_{\bar{\theta}}}(\pi_\alpha) &= \int_s \rho_{\pi_{\bar{\theta}}}(s) \int_{\tilde{a}} \pi_\alpha(s,\tilde{a}) A_{\pi_{\bar{\theta}}}(s,\tilde{a}) d\tilde{a} \\
&\approx \eta(\pi_{\bar{\theta}}) + \alpha \int_s \rho_{\pi_{\bar{\theta}}}(s) \int_a \pi_{\bar{\theta}}(s,a) ||\nabla_a A_{\pi_{\bar{\theta}}}(s,a)||^2 da.
\end{aligned}$$

Since $\pi_{\bar{\theta}}(s,a)$ and $||\nabla_a A_{\pi_{\bar{\theta}}}(s,a)||^2$ are positive, $L_{\pi_{\bar{\theta}}}(\pi_\alpha)$ is greater or equal to $\eta(\pi_{\bar{\theta}})$ when $\alpha > 0$. On the contrary, when $\alpha < 0$, $L_{\pi_{\bar{\theta}}}(\pi_\alpha)$ is smaller or equal to $\eta(\pi_{\bar{\theta}})$. Since this argument builds upon local approximation, it holds only when $|\alpha| \ll 1$.

7.1.4 Proof of Lemma 4.3

When $a = g_{\bar{\theta}}(s,\epsilon)$,

$$\begin{aligned}
g_\alpha(s,\epsilon) &= g_{\bar{\theta}}(s,\epsilon) + \alpha \cdot \nabla_{g_{\bar{\theta}}(s,\epsilon)} A_{\pi_{\bar{\theta}}}(s, g_{\bar{\theta}}(s,\epsilon)) \\
\Rightarrow \frac{\partial g_\alpha(s,\epsilon)}{\partial \epsilon} &= \frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} + \alpha \cdot \nabla^2_{g_{\bar{\theta}}(s,\epsilon)} A_{\pi_{\bar{\theta}}}(s, g_{\bar{\theta}}(s,\epsilon)) \cdot \frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} \\
&= \left(I + \alpha \cdot \nabla^2_{g_{\bar{\theta}}(s,\epsilon)} A_{\pi_{\bar{\theta}}}(s, g_{\bar{\theta}}(s,\epsilon)) \right) \cdot \frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} \\
\Rightarrow \det\left(\frac{\partial g_\alpha(s,\epsilon)}{\partial \epsilon} \right) &= \det\left(I + \alpha \cdot \nabla^2_{g_{\bar{\theta}}(s,\epsilon)} A_{\pi_{\bar{\theta}}}(s, g_{\bar{\theta}}(s,\epsilon)) \right) \cdot \det\left(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} \right) \\
&= \det\left(I + \alpha \cdot \nabla^2_a A_{\pi_{\bar{\theta}}}(s,a) \right) \cdot \det\left(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} \right)
\end{aligned}$$

Since $\left| \det\left(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} \right) \right| > 0$ (Equation 3), we can divide both sides of the above equation with $\det\left(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon} \right)$ and prove Lemma 4.3.

7.1.5 Proof of Lemma 4.4

Here we assume $\alpha = 1$, and define $\hat{A}$ for some state-action pair (s_t, a_t) at timestep t as follows,

$$\hat{A}_{\pi_\theta}(s_t,a_t) = \frac{1}{2} \mathbb{E}_{s_t,a_t,\ldots\sim\pi_\theta} \Big[\sum_{k=t}^{\infty} \gamma^k r(s_k,a_k) \Big],$$

Then

$$\nabla_a \hat{A}_{\pi_\theta}(s_t,a_t) = \frac{1}{2} \mathbb{E}_{s_t,a_t,\ldots\sim\pi_\theta} \Big[\sum_{k=t}^{\infty} \gamma^k \frac{\partial r(s_k,a_k)}{\partial a_t} \Big],$$

which can be rewritten using g_θ as follows.

$$\nabla_a \hat{A}_{\pi_\theta}(s_t,a_t) = \frac{1}{2} \mathbb{E}_{s_t,\epsilon_t,\ldots\sim q} \Big[\sum_{k=t}^{\infty} \gamma^k \frac{\partial r(s_k, g_\theta(s_t,\epsilon_t))}{\partial g_\theta(s_t,\epsilon_t)} \Big], \tag{10}$$

Then, by differentiating Equation 9 with respect to θ after plugging in $\hat{A}$, we can get following relationship:

$$\begin{aligned}
\frac{\partial L(\theta)}{\partial \theta} &= \mathbb{E}_{s_0,\epsilon_0,\ldots,\sim q}\Big[\sum_{t=0}^{\infty} 2 \cdot \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \cdot (g_\theta(s_t,\epsilon_t) - g_\alpha(s_t,\epsilon_t))\Big] \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots,\sim q}\Big[\sum_{t=0}^{\infty} 2 \cdot \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \cdot (g_\theta(s_t,\epsilon_t) - (g_\theta(s_t,\epsilon_t) + \nabla_a \hat{A}_{\pi_\theta}(s_t,a_t)))\Big] \\
&(\because \alpha = 1 \text{ by above assumption }) \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots,\sim q}\Big[\sum_{t=0}^{\infty} -2 \cdot \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \cdot \nabla_a \hat{A}_{\pi_\theta}(s_t,a_t)\Big] \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots,\sim q}\Big[\sum_{t=0}^{\infty} -2 \cdot \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \cdot \frac{1}{2} \cdot \sum_{k=t}^{\infty} \gamma^k \frac{\partial r(s_k, g_\theta(s_t,\epsilon_t)}{\partial g_\theta(s_t,\epsilon_t)}\Big] \\
&(\because \text{Equation 10}) \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots,\sim q}\Big[\sum_{t=0}^{\infty}\sum_{k=t}^{\infty} -\gamma^k \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \cdot \frac{\partial r(s_k, g_\theta(s_t,\epsilon_t))}{\partial g_\theta(s_t,\epsilon_t)}\Big],
\end{aligned}$$

which equals to RP gradient in Appendix 7.2.2, but with different sign. However, they turn out to be the same because we minimize $L(\theta)$, but maximize $\eta(\pi_\theta)$. Therefore, we can say that we gain RP gradient as the first gradient when we minimize Equation 9 for particular advantage function $\hat{A}$ and $\alpha = 1$.

7.1.6 Proof of Proposition 4.5

By Definition 5 and Lemma 4.3,

$$\begin{aligned}
\frac{\pi_\alpha(s,\tilde{\alpha})}{\pi_{\bar{\theta}}(s,a)} &= \frac{1}{|\det(I + \alpha \cdot \nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))|} \\
&= |\det(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon})| \cdot |\det(\frac{\partial g_\alpha(s,\epsilon)}{\partial \epsilon})|^{-1},
\end{aligned}$$

where $a = g_\theta(s,\epsilon)$, and thus $\tilde{a} = g_\alpha(s,\epsilon)$. Since $\pi_{\bar{\theta}} \triangleq g_{\bar{\theta}}$, following holds:

$$\begin{aligned}
\pi_\alpha(s,\tilde{\alpha}) &= \pi_{\bar{\theta}}(s,a) \cdot |\det(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon})| \cdot |\det(\frac{\partial g_\alpha(s,\epsilon)}{\partial \epsilon})|^{-1} \\
&= q(\epsilon) \cdot |\det(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon})|^{-1} \cdot |\det(\frac{\partial g_{\bar{\theta}}(s,\epsilon)}{\partial \epsilon})| \cdot |\det(\frac{\partial g_\alpha(s,\epsilon)}{\partial \epsilon})|^{-1} \\
&(\because \text{ Lemma 3.2}) \\
&= q(\epsilon) \cdot |\det(\frac{\partial g_\alpha(s,\epsilon)}{\partial \epsilon})|^{-1},
\end{aligned}$$

which implies $\pi_\alpha \triangleq g_\alpha$.

7.2 Formulations

7.2.1 Analytic Gradient of Generalized Advantage Estimator (GAE)

GAE [Schulman et al., 2015b] has been widely used in many RL implementations [Schulman et al., 2017, Raffin et al., 2021, Makoviichuk and Makoviychuk, 2022] to estimate advantages. GAE finds a balance between variance and bias of the advantage estimation by computing the exponentially-weighted average of the TD residual terms (δ_t^V) Sutton et al. [1998], which are defined as follows:

$$\delta_t^V = r_t + \gamma V(s_{t+1}) - V(s_t).$$

Then, GAE can be formulated as follows:

$$A_t^{GAE}(\gamma,\lambda) = \sum_{k=0}^{\infty} (\gamma\lambda)^k \delta_{t+k}^V. \tag{11}$$

We can compute the gradients for these terms as:

$$\begin{aligned}
\frac{\partial A_t^{GAE}}{\partial a_t} &= \sum_{k=0}^{\infty} (\gamma\lambda)^k \frac{\partial \delta_{t+k}^V}{\partial a_t} \\
&= (\frac{1}{\gamma\lambda})^t \sum_{k=0}^{\infty} (\gamma\lambda)^{t+k} \frac{\partial \delta_{t+k}^V}{\partial a_t} \\
&= (\frac{1}{\gamma\lambda})^t \left[\sum_{k=0}^{\infty} (\gamma\lambda)^{t+k} \frac{\partial \delta_{t+k}^V}{\partial a_t} + \sum_{k=0}^{t-1} (\gamma\lambda)^k \frac{\partial \delta_k^V}{\partial a_t} \right] \\
(\because \frac{\partial \delta_k^V}{\partial a_t} &= 0 \text{ for } k < t) \\
&= (\frac{1}{\gamma\lambda})^t \sum_{k=0}^{\infty} (\gamma\lambda)^k \frac{\partial \delta_k^V}{\partial a_t} \\
&= (\frac{1}{\gamma\lambda})^t \frac{\partial A_0^{GAE}}{\partial a_t}.
\end{aligned} \tag{12}$$

Based on this relationship, we can compute every $\frac{\partial A_t^{GAE}}{\partial a_t}$ with only one backpropagation of A_0^{GAE}, rather than backpropagating for every A_t^{GAE}.

7.2.2 RP Gradient Formulation

To differentiate Equation 1 with respect to θ, we first rewrite Equation 1 as follows.

$$\begin{aligned}
\eta(\pi_\theta) &= \mathbb{E}_{s_0,a_0,\ldots\sim\pi_\theta} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t,a_t) \right] \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots\sim q} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, g_\theta(s_t,\epsilon_t)) \right].
\end{aligned}$$

Then, we can compute RP gradient as follows.

$$\begin{aligned}
\frac{\partial \eta(\pi_\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta} \mathbb{E}_{s_0,\epsilon_0,\ldots\sim q} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, g_\theta(s_t,\epsilon_t)) \right] \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots\sim q} \left[\frac{\partial}{\partial \theta} \sum_{t=0}^{\infty} \gamma^t r(s_t, g_\theta(s_t,\epsilon_t)) \right] \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots\sim q} \left[\sum_{t=0}^{\infty} \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \sum_{k=t}^{\infty} \gamma^k \frac{\partial r(s_k, g_\theta(s_k,\epsilon_k))}{\partial g_\theta(s_t,\epsilon_t)} \right] \\
(\because \frac{\partial r(s_k, g_\theta(s_k,\epsilon_k))}{\partial g_\theta(s_t,\epsilon_t)} &\neq 0 \text{ only when } k \geq t.) \\
&= \mathbb{E}_{s_0,\epsilon_0,\ldots\sim q} \left[\sum_{t=0}^{\infty} \sum_{k=t}^{\infty} \gamma^k \frac{\partial g_\theta(s_t,\epsilon_t)}{\partial \theta} \frac{\partial r(s_k, g_\theta(s_k,\epsilon_k))}{\partial g_\theta(s_t,\epsilon_t)} \right].
\end{aligned}$$

We can estimate this RP gradient using Monte Carlo sampling, and use it for gradient ascent to maximize Equation 1.

7.2.3 Estimator for Equation 4

Since $\eta(\pi_{\bar{\theta}})$ does not depend on θ, we can ignore the term and rewrite our loss function in Equation 4 using expectation as follows:

$$\begin{aligned} L_{\pi_{\bar{\theta}}}(\pi_\theta) &= \int_s \rho_{\pi_{\bar{\theta}}}(s) \int_a \pi_\theta(s,a) A_{\pi_{\bar{\theta}}}(s,a) \\ &= \mathbb{E}_{s\sim\rho_{\pi_{\bar{\theta}}}, a\sim\pi_\theta(s,\cdot)}\Big[A_{\pi_{\bar{\theta}}}(s,a)\Big]. \end{aligned}$$

However, note that we did not collect trajectories, or experience buffer, using π_θ. Therefore, we use another importance sampling function $q(s,a)$ [Schulman et al., 2015a].

$$L_{\pi_{\bar{\theta}}}(\pi_\theta) = \mathbb{E}_{s\sim\rho_{\pi_{\bar{\theta}}}, a\sim q(s,\cdot)}\Big[\frac{\pi_\theta(s,a)}{q(s,a)} A_{\pi_{\bar{\theta}}}(s,a)\Big].$$

Then, we can estimate this value using Monte Carlo estimation as follows, where B is the experience buffer of size N, which stores state (s_i), action (a_i), and their corresponding advantage value ($A_{\pi_{\bar{\theta}}}(s_i,a_i)$) obtained by following policy $\pi_{\bar{\theta}}$:

$$L_{\pi_{\bar{\theta}}}(\pi_\theta) \approx \frac{1}{N}\sum_{i=1}^{N} \frac{\pi_\theta(s_i,a_i)}{\pi_{\bar{\theta}}(s_i,a_i)} A_{\pi_{\bar{\theta}}}(s_i,a_i). \tag{13}$$

Note that we used $\pi_{\bar{\theta}}$ in the place of q, as it is the most natural importance sampling function we can use [Schulman et al., 2015a, 2017].

7.3 Algorithm

Here we present the entire pseudocode of our algorithm. Before that, we'd like to discuss additional aspects of our algorithm other than those discussed in Section 4.3.

7.3.1 Out-of-range-ratio

As discussed in Section 2, PPO achieves its stable learning by updating policy only in the trust region near the current policy. Note that TRPO [Schulman et al., 2015a] achieves this by computing the KL divergence between the current policy and the original policy, and restraining it below certain threshold. However, PPO [Schulman et al., 2017] uses a much simpler approach, to restrict the ratio of probabilities $\pi_\theta(s_i,a_i)$ and $\pi_{\bar{\theta}}(s_i,a_i)$ for every state-action pair (s_i,a_i) in our buffer as follows,

$$1-\epsilon_{clip} < \frac{\pi_\theta(s_i,a_i)}{\pi_{\bar{\theta}}(s_i,a_i)} < 1+\epsilon_{clip},$$

where ϵ_{clip} is a constant. To be specific, PPO uses following surrogate loss function to mandate this:

$$L_{PPO}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \min\Big(\frac{\pi_\theta(s_i,a_i)}{\pi_{\bar{\theta}}(s_i,a_i)} A_{\pi_{\bar{\theta}}}(s_i,a_i), clip\Big(\frac{\pi_\theta(s_i,a_i)}{\pi_{\bar{\theta}}(s_i,a_i)}, 1-\epsilon_{clip}, 1+\epsilon_{clip}\Big) A_{\pi_{\bar{\theta}}}(s_i,a_i)\Big) \tag{14}$$

Note that some of the ratios could go out of bounds while optimizing this loss function (see Schulman et al. [2017] for details). However, the spirit of PPO lies in restricting this ratio.

As we discussed in Section 4.3, we first update our policy towards π_α by minimizing Equation 9 before performing PPO update. If α is sufficiently small, most of the ratios $\frac{\pi_\theta(s_i,a_i)}{\pi_{\bar{\theta}}(s_i,a_i)}$ will stay near 1, which does not harm the PPO's assumption. However, if α is big, many of the ratios could go out of bounds. Therefore, we compute the out-of-range-ratio after updating our policy to π_α as follows,

$$\text{out-of-range-ratio} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\Big(\Big|\frac{\pi_\theta(s_i,a_i)}{\pi_{\bar{\theta}}(s_i,a_i)} - 1\Big| > \epsilon_{clip}\Big), \tag{15}$$

where N is the size of the buffer, and $\mathbb{I}$ is the indicator function. Note that this ratio depends on ϵ_{clip}. If this out-of-range-ratio is larger than a pre-defined threshold, we decrease α.

7.3.2 Importance sampling function

After updating our policy to π_α, if we do not take additional measure and maximize surrogate loss function in Equation 14, the updated policy can totally ignore π_α, which is undesirable. To avoid such situation, we use another importance sampling function π_h defined as follows:

$$\pi_h(s,a) = \frac{1}{2}(\pi_{\bar{\theta}}(s,a) + \pi_\alpha(s,a)).$$

By using this importance sampling function and making our updated function to stay near π_h, we can guarantee that the updated function stays close to π_α. Then our final surrogate loss function is

$$L_{GIPPO}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \min(\frac{\pi_\theta(s_i,a_i)}{\pi_h(s_i,a_i)} A_{\pi_{\bar{\theta}}}(s_i,a_i), clip(\frac{\pi_\theta(s_i,a_i)}{\pi_h(s_i,a_i)}, 1-\epsilon_{clip}, 1+\epsilon_{clip}) A_{\pi_{\bar{\theta}}}(s_i,a_i)). \tag{16}$$

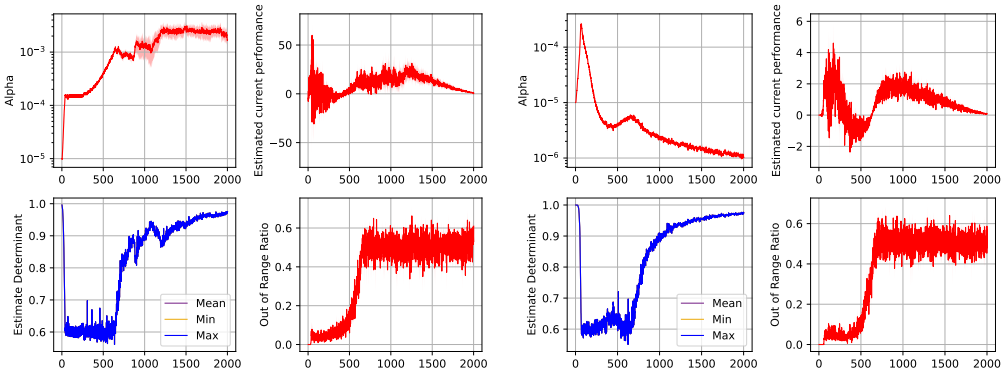
See how it differs from the original loss function in Equation 14.

7.3.3 Pseudocode

In Algorithm 1, we present pseudocode that illustrates the outline of our algorithm, GI-PPO. Note that we used three criteria (variance, bias, out-of-range-ratio) to control α after we update our policy to the α-policy. If there is no reason to reduce α, we increase it, because we can expect higher expected return with bigger α as shown in Proposition 4.2. Also note that there are five hyperparameters here, which are $\alpha_0, \beta, \delta_{det}, \delta_{oorr}$, and $\max(\alpha)$. We present specific values for these hyperparameters for our experiments in Appendix 7.4.

Algorithm 1 GI-PPO

$\alpha \leftarrow \alpha_0$, Initial value
$\beta \leftarrow$ Constant multiplier larger than 1 for α
$\delta_{det}, \delta_{oorr} \leftarrow$ Constant thresholds
$B \leftarrow$ Experience buffer
while *Training not ended* **do**
  Clear B
  while *Not collected enough experience* **do**
    Collect experience $\{s_t, \epsilon_t, a_t, r_t, s_{t+1}\} \rightarrow B$
  end
  Estimate advantage A for every (s_i, a_i) in B using Eq. 11
  Estimate advantage gradient $\frac{\partial A}{\partial a}$ for every (s_i, a_i) in B using Eq. 12
  For current α, approximate α-policy by minimizing loss in Equation 9
  // Variance
  For each ϵ_i and its corresponding state-action pair (s_i, a_i), estimate $\det(I + \alpha \cdot \nabla_a^2 A_{\pi_{\bar{\theta}}}(s,a))$ by Lemma 4.3 and get its sample minimum (ψ_{min}) and maximum (ψ_{max})
  // Bias
  Evaluate expected additional return in Equation 13 with our current policy to get R_α
  // Out-of-range-ratio
  Evaluate out-of-range-ratio in Equation 15 with our current policy to get R_{oorr}
  if $\psi_{min} < 1 - \delta_{det}$ *or* $\psi_{max} > 1 + \delta_{det}$ *or* $R_\alpha < 0$ *or* $R_{oorr} > \delta_{oorr}$ **then**
    $\alpha = \alpha / \beta$
  end
  else
    $\alpha = \alpha \times \beta$
  end
  $\alpha = clip(\alpha, 0, \max(\alpha))$
  Do PPO update by maximizing the surrogate loss in Equation 16
end



(a) De Jong's Function (64)

(b) Ackley's Function (64)

Figure 6: Change of α during solving function optimization problems in Section 5.1. Since these problems are one-step problems, and observations are all the same, there is no difference between mean, min, and max of estimated determinant.

7.3.4 Example: Function Optimization Problems (Section 5.1)

Here we present how α changes as we solve function optimization problems in Section 5.1 with Figure 6. There are 4 plots for each of the problems.

- Alpha: Shows change of α over training epoch.
- Estimated current performance: Shows expected additional return, which corresponds to Equation 13 (R_α in Algorithm 1).
- Estimate Determinant: Shows statistics of estimated $\det(I + \alpha \cdot \nabla_a^2 A_{\pi_\theta}(s, a))$.
- Out of Range Ratio: Shows out-of-range-ratio in Equation 15.

For these problems, we have set $\alpha_0 = 10^{-5}$, $\delta_{det} = 0.4$, and $\delta_{oorr} = 0.5$.

For De Jong's function (Figure 6a), we can observe that α shows steady increase over time, but it is mainly upper bounded by variance criterion and out-of-range-ratio. In the end, α stabilizes around 10^{-3}. In contrast, for Ackley's function (Figure 6b) we can see that α increases rapidly at first, but it soon decreases due to the variance and bias criteria. Compare these graphs with optimization curve in Figure 3d. Then we would be able to observe that this large α contributes to the faster convergence of GI-PPO compared to PPO at the early stage. However, after that, α decreases slowly due to the out-of-range-ratio. In the end, α reaches approximately 10^{-6}, which is much lower than 10^{-3} of De Jong's function. These observations align well with our intuition that RP gradient would play bigger role for De Jong's function than Ackley's function, because De Jong's function exhibits RP gradients with much lower variance than those of Ackley's.

7.4 Experimental Details

7.4.1 Baseline Methods

Here we explain implementation details of the baseline methods that were used for comparisons in Section 5. First, we'd like to point out that we used relatively a short time horizon to collect experience, and used the critic network for bootstrapping, instead of collecting experience for the whole trajectory for all the methods. Therefore, the collected experience could be biased. However, we chose to implement in this way, because it allows faster learning than collecting the whole trajectory in terms of number of simulation steps.

LR We can estimate original LR gradient as follows, using log-derivative trick [Williams and Peng, 1989, Glynn, 1990]

$$\begin{aligned}\frac{\partial \eta(\pi_\theta)}{\partial \theta} &= \frac{\partial}{\partial \theta}\mathbb{E}_{s_0,a_0,\ldots\sim\pi_\theta}\Big[L(s_0,a_0,\ldots)\sum_{t=0}^{\infty}\log\pi_\theta(s_0,a_0)\Big] \\ &= \frac{\partial}{\partial \theta}\mathbb{E}_{s_0,a_0,\ldots\sim\pi_\theta}\Big[\sum_{t=0}^{\infty}L(s_0,a_0,\ldots)\log\pi_\theta(s_0,a_0)\Big],\end{aligned}$$

where $L(s_0,a_0,\ldots)=\sum_{t=0}^{\infty}\gamma^t r(s_t,a_t)$ is the discounted accumulated return. However, this LR gradient often suffers from high variance, because L varies a lot from trajectory to trajectory. To reduce such variance and faithfully compare our method to LR gradient-based method, we decided to use advantage function in the place of L, and particularly used GAE [Schulman et al., 2015b] as the advantage function as we used for PPO. After calculating this LR gradient, we take a gradient ascent step with pre-defined learning rate.

RP By using RP gradient in Appendix 7.2.2, we can perform gradient ascent as we did in LR. Also, because we use short time horizon and critic network, this method is very similar to SHAC [Xu et al., 2022]. Please refer to it for more details.

PPO Our PPO implementation is based on that of RL Games [Makoviichuk and Makoviychuk, 2021]. However, to be as fair as possible, we used the same critic network, instead of using that implemented already. Also, we omitted several other additional losses that are not vital to PPO's formulation, such as entropy loss.

LR+RP After we compute LR (∇_{LR}) and RP (∇_{RP}) gradient as shown above, we can interpolate them using their sample variance as follows [Parmas et al., 2018].

$$\begin{aligned}\nabla_{LR+RP} &= \nabla_{LR}\cdot\kappa_{LR}+\nabla_{RP}\cdot(1-\kappa_{LR}), \\ \kappa_{LR} &= \frac{\sigma_{RP}^2}{\sigma_{RP}^2+\sigma_{LR}^2},\end{aligned}$$

where σ_{RP} and σ_{LR} are sample standard deviation of RP and LR gradients. We gain these terms by computing trace of covariance matrix of the sample gradients from different trajectories.

Since we have to compute this sample statistics, we have to do multiple different backpropagations for different trajectories, which incur a lot of computation time. Also, we found out that computing covariance matrix is also time consuming when controller has a large number of parameters. Therefore, we decided to use only limited number of sample gradients (16) to compute sample variance, and also truncate the gradient to smaller length (512) to facilitate computation.

PE We tried to faithfully re-implement policy enhancement scheme of [Qiao et al., 2021].

7.4.2 Network architecture and Hyperparameters

In this section, we provide network architectures and hyperparameters that we used for experiments in Section 5. For each of the experiments, we used the same network architectures, the same length of time horizons before policy update, and the same optimization procedure for critic updates, etc. We present these common settings first for each of the problems.

For GI-PPO, there are hyperparameters for update towards α-policy, and those for PPO update. We denote the hyperparameters for α-policy update by appending (α), and those for PPO update by appending (PPO). For the definition of hyperparameters for α-policy update, please see Algorithm 1 for details.

Function Optimization Problems (Section 5.1) For these problems, common settings are as follows.

- Actor Network: MLP with $[32, 32]$ layers and ELU activation function

- Critic Network: MLP with $[32, 32]$ layers and ELU activation function
- Critic Hyperparameters: Learning rate = 10^{-3}, Iterations = 16, Batch Size = 4
- Number of parallel environments: 64
- Horizon Length: 1
- γ (Discount factor): 0.99
- τ (GAE): 0.95

Hyperparameters for **LR** are as follows.

- Learning Rate: Dejong(1), Dejong(64) = 10^{-3}, Ackley(1) = 10^{-4}, Ackley(64) = $3 \cdot 10^{-4}$
- Learning Rate Scheduler: Linear[1]

Hyperparameters for **RP** are as follows.

- Learning Rate: Dejong(1), Dejong(64) = 10^{-2}, Ackley(1), Ackley(64) = 10^{-3}
- Learning Rate Scheduler: Linear

Hyperparameters for **LR+RP** are as follows.

- Learning Rate: Dejong(1), Dejong(64) = 10^{-3}, Ackley(1) = 10^{-4}, Ackley(64) = $3 \cdot 10^{-4}$
- Learning Rate Scheduler: Linear

Hyperparameters for **PPO** are as follows.

- Learning Rate: Dejong(1), Ackley(1) = 10^{-4}, Dejong(64), Ackley(64) = 10^{-2}
- Learning Rate Scheduler: Constant
- Batch Size for Actor Update: 64
- Number of Epochs for Actor Update: 5
- ϵ_{clip}: 0.2

Hyperparameters for **GI-PPO** are as follows.

- (α) Learning Rate: 10^{-3}
- (α) Batch Size for Actor Update: 64
- (α) Number of Epochs for Actor Update: 16
- (α) α_0: 10^{-5}
- (α) $\max(\alpha)$: 1.0
- (α) β: 1.1
- (α) δ_{det}: 0.4
- (α) δ_{oorr}: 0.5
- (PPO) Learning Rate: Dejong(1), Ackley(1) = 10^{-4}, Dejong(64), Ackley(64) = 10^{-2}
- (PPO) Learning Rate Scheduler: Constant
- (PPO) Batch Size for Actor Update: 64
- (PPO) Number of Epochs for Actor Update: 5
- (PPO) ϵ_{clip}: 0.2

[1]Learning rate decreases linearly to the minimum value as learning progresses.

Differentiable Physics Problems (Section 5.2) For these problems, common settings are as follows.

- Actor Network: MLP with $[64, 64]$ (Cartpole), and $[128, 64, 32]$ (Ant, Hopper) layers and ELU activation function
- Critic Network: MLP with $[64, 64]$ layers and ELU activation function
- Critic Hyperparameters: Learning rate = 10^{-3}(Cartpole), $2{\cdot}10^{-3}$(Ant), and $2{\cdot}10^{-4}$(Hopper), Iterations = 16, Batch Size = 4
- Number of parallel environments: 64(Cartpole, Ant), 256(Hopper)
- Horizon Length: 32
- γ (Discount factor): 0.99
- τ (GAE): 0.95

Hyperparameters for **LR** are as follows.

- Learning Rate: 10^{-4}
- Learning Rate Scheduler: Linear

Hyperparameters for **RP** are as follows.

- Learning Rate: Cartpole = 10^{-2}, Ant, Hopper = $2 \cdot 10^{-3}$
- Learning Rate Scheduler: Linear

Hyperparameters for **PPO** are as follows.

- Learning Rate: Cartpole = $3 \cdot 10^{-4}$, Ant, Hopper = 10^{-4}
- Learning Rate Scheduler: Cartpole = Adaptive[2], Ant, Hopper = Constant
- Batch Size for Actor Update: 2048
- Number of Epochs for Actor Update: 5
- ϵ_{clip}: 0.2

Hyperparameters for **GI-PPO** are as follows.

- (α) Learning Rate: Cartpole = 10^{-2}, Ant = $5 \cdot 10^{-4}$, Hopper = $5 \cdot 10^{-3}$
- (α) Batch Size for Actor Update: Cartpole, Ant = 2048, Hopper = 8192
- (α) Number of Epochs for Actor Update: 16
- (α) α_0: Cartpole, Ant = $5 \cdot 10^{-1}$, Hopper = $5 \cdot 10^{-3}$
- (α) $\max(\alpha)$: 1.0
- (α) β: 1.02
- (α) δ_{det}: 0.4
- (α) δ_{oorr}: Cartpole, Hopper = 0.75, Ant = 0.5
- (PPO) Learning Rate: Cartpole = $3 \cdot 10^{-4}$, Ant, Hopper = 10^{-4}
- (PPO) Learning Rate Scheduler: Cartpole = Adaptive, Ant, Hopper = Constant
- (PPO) Batch Size for Actor Update: 2048
- (PPO) Number of Epochs for Actor Update: 5
- (PPO) ϵ_{clip}: 0.2

[2]Learning rate is adaptively controlled, so that the KL divergence between the updated policy and the original policy is maintained at certain value, 0.008 in this case.

Traffic Problems (Section 5.3) For these problems, common settings are as follows.

- Actor Network: MLP with $[512, 64, 64]$ layers and ELU activation function
- Critic Network: MLP with $[64, 64]$ layers and ELU activation function
- Critic Hyperparameters: Learning rate = 10^{-3}, Iterations = 16, Batch Size = 4
- Number of parallel environments: 64
- Horizon Length: 32
- γ (Discount factor): 0.99
- τ (GAE): 0.95

Hyperparameters for **LR** are as follows.

- Learning Rate: $3 \cdot 10^{-4}$
- Learning Rate Scheduler: Linear

Hyperparameters for **RP** are as follows.

- Learning Rate: 10^{-3}
- Learning Rate Scheduler: Linear

Hyperparameters for **LR+RP** are as follows.

- Learning Rate: $3 \cdot 10^{-4}$
- Learning Rate Scheduler: Linear

Hyperparameters for **PPO** are as follows.

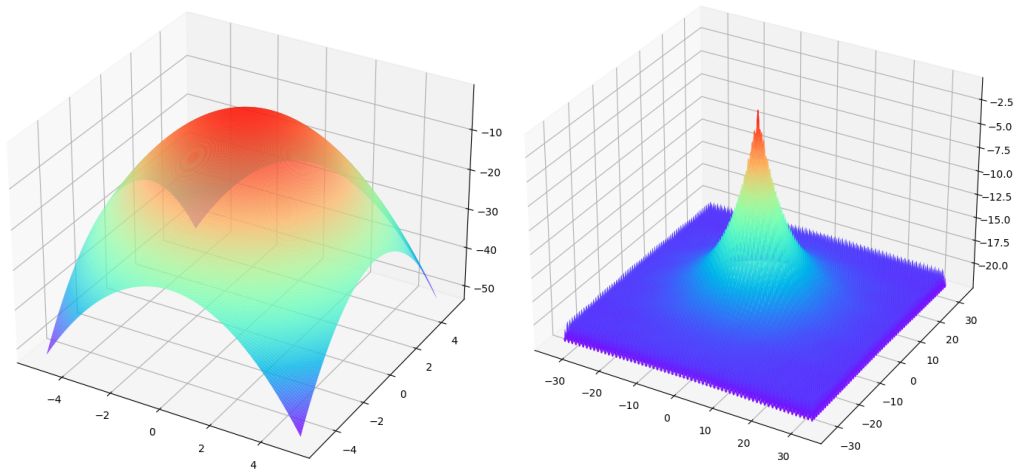
- Learning Rate: $3 \cdot 10^{-4}$
- Learning Rate Scheduler: Constant
- Batch Size for Actor Update: 2048
- Number of Epochs for Actor Update: 5
- ϵ_{clip}: 0.2

Hyperparameters for **GI-PPO** are as follows.

- (α) Learning Rate: 10^{-5}
- (α) Batch Size for Actor Update: 2048
- (α) Number of Epochs for Actor Update: 16
- (α) α_0: 10^{-1}
- (α) $\max(\alpha)$: 1.0
- (α) β: 1.1
- (α) δ_{det}: 0.4
- (α) δ_{oorr}: 0.5
- (PPO) Learning Rate: $3 \cdot 10^{-4}$
- (PPO) Learning Rate Scheduler: Constant
- (PPO) Batch Size for Actor Update: 2048
- (PPO) Number of Epochs for Actor Update: 5
- (PPO) ϵ_{clip}: 0.2

7.5 Problem Definitions

Here we present details about the problems we suggested in Section 5.



(a) De Jong's Function (b) Ackley's Function

Figure 7: Landscape of target functions in 2 dimensions.

7.5.1 Function Optimization Problems (Section 5.1)

(N-dimensional) De Jong's function (F_D) and Ackley's function (F_A) are defined for n-dimensional vector x and are formulated as follows [Molga and Smutnicki, 2005]:

$$F_D(x) = \sum_{i=1}^{n} {x_i}^2,$$

$$F_A(x) = -20 \cdot exp(-0.2 \cdot \sqrt{\frac{1}{n}\sum_{i=1}^{n} {x_i}^2}) - exp(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)) + 20 + exp(1).$$

As we mentioned in Section 5.1, we multiply -1 to these functions to make these problems maximization problems. Also, even though these two functions have their optimum at $x = 0$, they exhibit very different landscape as shown in Figure 7.

When it comes to the formal definition as RL environments, we can define these problems as follows.

- Episode Length: 1
- Observation: $[0]$
- Action: n-dimensional vector $\mathbf{x}$, all of which element is in $[-1, 1]$.
- Reward: De Jong's Function = $F_D(5.12\mathbf{x})$, Ackley's Function = $F_A(32.768\mathbf{x})$

7.5.2 Traffic Problems (Section 5.3)

Before introducing pace car problem, which we specifically discussed in this paper, we'd like to briefly point out the traffic model that we used to simulate motions of individual vehicles in the traffic environment.

Traffic Model In our traffic simulation, the state $s \in \mathbb{R}^{2N}$ is defined as a concatenation of all vehicle states, $q_n \in \mathbb{R}^2$, where $1 \leq n \leq N$ stands for vehicle ID. q_n can be represented simply with the vehicle's position (x_n) and velocity (v_n).

$$q_n = \begin{bmatrix} x_n \\ v_n \end{bmatrix}$$
$$s = \left[q_1^T, \ldots, q_N^T\right]^T,$$

where n is the vehicle ID. Since this traffic state s changes over time, we represent the traffic state at timestep t as s_t. Then our differentiable traffic simulator is capable of providing the gradient of the simulation state at the next timestep $t+1$ with respect to the state at the current timestep t, or $\frac{ds_{t+1}}{ds_t}$.

This is possible because we use the Intelligent Driver Model (IDM) to model car-following behavior in our simulator Treiber et al. [2000], which is differentiable. IDM describes vehicle speed $\dot{x}$ and acceleration $\dot{v}$ as a function of desired velocity v_0, safe time headway in meters T, maximum acceleration a, comfortable deceleration b, the minimum distance between vehicles in meters δ, vehicle length l, and difference in speed with the vehicle in front Δv_α as follows:

$$\begin{aligned}
\dot{x}_\alpha &= \frac{\mathrm{d}x_\alpha}{\mathrm{d}t} = v_\alpha, \\
\dot{v}_\alpha &= \frac{\mathrm{d}v_\alpha}{\mathrm{d}t} = a\left(1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right), \\
s_\alpha &= x_{\alpha-1} - x_\alpha - l_{\alpha-1}, \\
s^*(v_\alpha, \Delta v_\alpha) &= s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{a\,b}}, \\
\Delta v_\alpha &= v_\alpha - v_{\alpha-1},
\end{aligned}$$

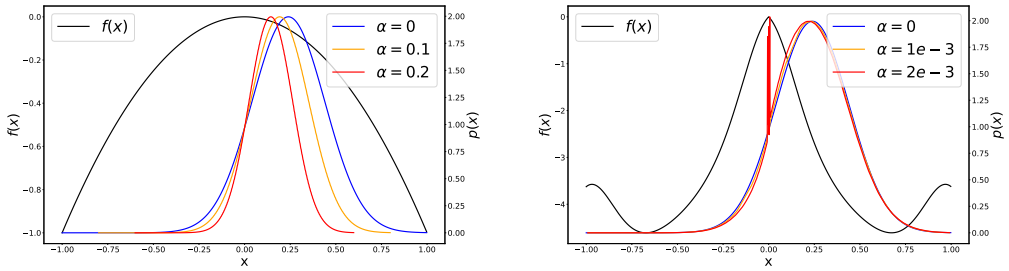
where α means the order of the vehicle in the lane. Therefore, $(\alpha-1)$-th vehicle runs right in front of α-th vehicle, and this relationship plays an important role in IDM. Also note that the computed acceleration term $(\dot{v})$ is differentiable with respect to the other IDM variables.

IDM is just one of many car-following models used in traffic simulation literature. We choose IDM for its prevalence in previous mixed autonomy literature, however, any ODE-based car-following model will also work in our simulator as far as it is differentiable. In our simulator, automatic differentiation governs the gradient computation.

Lane Under IDM, lane membership is one of the most vital variables for simulation. This is because IDM builds on the leader-follower relationship. When a vehicle changes its lane with lateral movement, as shown as red arrows in Figure 9, such relationships could change, and our gradients do not convey any information about it, because lane membership is a discrete variable in nature. Note that this lateral movement also affects the longitudinal movement, rendered in green arrows in Figure 9, of a vehicle, and gradient from it is valid as far as the lane membership does not change. However, when the lane membership changes, even if our simulator gives gradient, it tells us only "partial" information in the sense that it only gives information about longitudinal behavior. Therefore, we could say that analytical gradients we get from this environment is "incomplete" (not useless at all, because it still gives us information about longitudinal movement), and thus biased.

Pace Car Problem In this problem, there is a single autonomous vehicle that we have to control to regulate other human driven vehicles, which follow IDM, to run at the given target speed. Since human driven vehicles control their speeds based on the relationship to their leading vehicles, our autonomous vehicle has to control itself to run in front of those human driven vehicles and adjust their speeds to the target speed. The number of lanes and human driven vehicles varies across different environments as follows.

- Single Lane: Number of lanes = 1, Number of vehicles per lane = 1
- 2 Lanes: Number of lanes = 2, Number of vehicles per lane = 2
- 4 Lanes: Number of lanes = 4, Number of vehicles per lane = 4
- 10 Lanes: Number of lanes = 10, Number of vehicles per lane = 1



(a) De Jong's Function

(b) Ackley's Function

Figure 8: α-policies for 1-dimensional (a) De Jong's function and (b) Ackley's function [Molga and Smutnicki, 2005].

See Figure 10 for 10-lane environment.

Then we can formally define RL environments as follows, where N is the number of human driven vehicles in total, and v_{tgt} is the target speed.

- Episode Length: 1000
- Observation: $s \in \mathbb{R}^{2N}$
- Action: $\mathbf{a} \in \mathbb{R}^2$ ($\mathbf{a}_0$ = Acceleration, $\mathbf{a}_1$ = Steering)
- Reward: $1 - \frac{1}{N}\sum_{i=1}^{N} \min(\frac{|v_i - v_{tgt}|}{v_{tgt}}, 1)$

However, to facilitate learning, we additionally adopted following termination conditions to finish unpromising trials early.

- Terminate when autonomous vehicle collides with human driven vehicle
- Terminate when autonomous vehicle goes out of lane
- Terminate when autonomous vehicle runs too far away from the other vehicles
- Terminate when autonomous vehicle falls behind the other vehicles

When one of these conditions is met, it gets reward of -1 and the trial terminates.

7.6 Renderings

7.6.1 α-policies

Here we illustrate α-policies for De Jong and Ackley's function in Section 5.1. In Figure 8, the original policy is rendered in blue, and alpha policies for different α values are rendered in orange and red. Those α-policies are obtained from analytical gradients ($\frac{\partial f(x)}{\partial x}$) of the given function $f(x)$, which is rendered in black. Note that singularities arise in Ackley's α-policy near 0 when $\alpha = 2 \cdot 10^{-3}$, as it has a steep change of gradients there. This is explains one of the reasons why we need to keep α value sufficiently small.

7.6.2 Traffic Simulation

Gradient flow As described in Appendix 7.5.2, analytical gradients are only valid along longitudinal direction, as illustrated with green arrows in Figure 9. When a vehicle changes lane with lateral movement, as shown with red arrows, even though such change would incur different behaviors of following vehicles in multiple lanes, analytical gradients cannot convey such information. However, they still tell us information about longitudinal behavior - which is one of the motivations of our research, to use analytical gradients even in these biased environments.

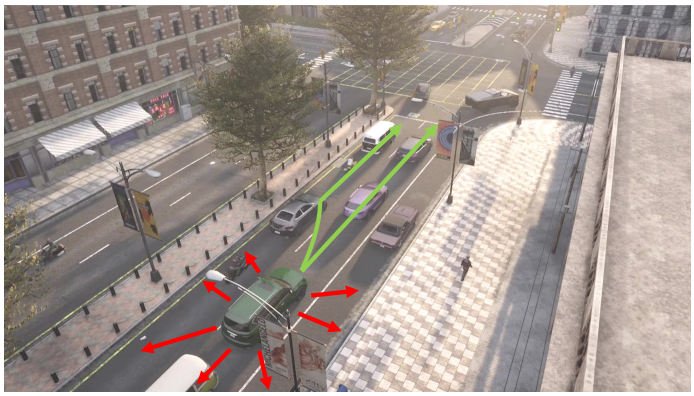
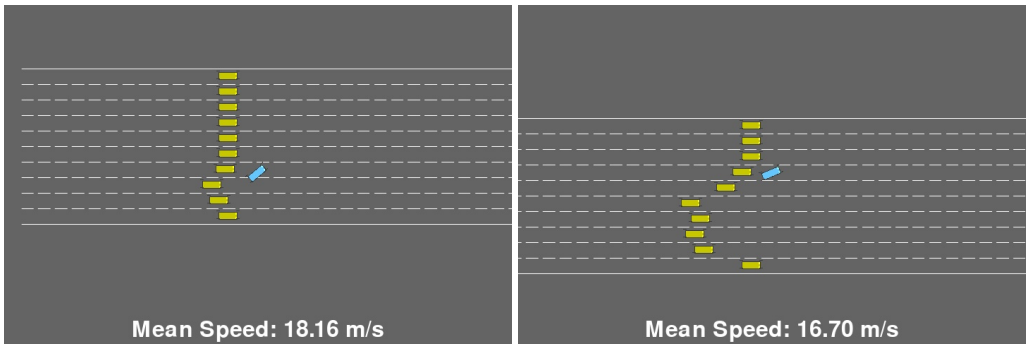


Figure 9: **Traffic Environment.** In traffic flows, only partial gradients of forward simulation are known; acceleration along a (longitudinal) traffic lane is continuous and admits gradient flow (green), while (lateral) lane-changing is a discrete event and thus prohibits gradient flow (red). Analytical gradients convey only partial information about traffic dynamics.



(a) 0-th frame (b) 150-th frame

(c) 300-th frame (d) 450-th frame

Figure 10: 10-Lane pace car environment. The autonomous vehicle (Blue) has to interfere paths of the other human driven vehicles (Yellow) to limit their speeds to $10m/s$. Even though it is a hard problem to achieve high score, our method achieved far better score than the baseline methods.

Pace Car Problem We additionally provide renderings of the 10-lane pace car problem defined in Appendix 7.5.2. In Figure 10, we can observe 10 parallel lanes, 10 human driven vehicles rendered in yellow, and the autonomous vehicle rendered in blue that we have to control. As shown there, human driven vehicles adjust their speeds based on IDM when the autonomous vehicle blocks their way. Therefore, the autonomous vehicle has to learn how to change their lanes, and also how to run in appropriate speed to regulate the following vehicles. Our experimental results in Section 5.3 show that our method achieves better score than the baseline methods by adopting biased analytical gradients in PPO.