

# A Multiplicative Value Function for Safe and Efficient Reinforcement Learning

Nick Bührer<sup>1</sup>, Zhejun Zhang<sup>1</sup>, Alexander Liniger<sup>1</sup>, Fisher Yu<sup>1</sup>, Luc Van Gool<sup>1,2</sup>

**Abstract**—An emerging field of sequential decision problems is safe Reinforcement Learning (RL), where the objective is to maximize the reward while obeying safety constraints. Being able to handle constraints is essential for deploying RL agents in real-world environments, where constraint violations can harm the agent and the environment. To this end, we propose a safe model-free RL algorithm with a novel multiplicative value function consisting of a safety critic and a reward critic. The safety critic predicts the probability of constraint violation and discounts the reward critic that only estimates constraint-free returns. By splitting responsibilities, we facilitate the learning task leading to increased sample efficiency. We integrate our approach into two popular RL algorithms, Proximal Policy Optimization and Soft Actor-Critic, and evaluate our method in four safety-focused environments, including classical RL benchmarks augmented with safety constraints and robot navigation tasks with images and raw Lidar scans as observations. Finally, we make the zero-shot sim-to-real transfer where a differential drive robot has to navigate through a cluttered room. Our code can be found at <https://github.com/nikeke19/Safe-Mult-RL>.

## I. INTRODUCTION

Reinforcement Learning (RL) has made significant progress in recent years. Breakthroughs have been achieved on playing Atari [1] and board games like Go [2], as well as multi-agent RL on Starcraft [3] and Dota [4]. While some works like Miki et al. [5] deploy RL agents on real-world systems, most of the research is still conducted in simulation [6], [7]. On the one hand, the sim-to-real transfer requires high-fidelity simulators and robust models. On the other hand, there is the safety aspect. In simulation, the agent can perform any action without real consequences. However, when robots are deployed in reality, not every action is admissible. This can be due to damage to the agent, e.g., when a robot crashes into an obstacle, or damage to the environment, e.g., when the obstacle is a human. Thus, it is necessary to consider safety by design. We can formulate safety requirements using Constrained Markov Decision Process (CMDP) [8], where in addition to maximizing the reward, the RL agent has to fulfill constraints on the expected accumulated safety cost.

Usually, the expected safety cost is estimated with a value function, called the safety critic. With a particular choice of safety cost, the cumulative cost constraint can be transformed into a chance constraint and be relaxed with a Lagrange multiplier. In contrast to previous works [9], [10],

<sup>1</sup>All authors are with the Computer Vision Lab, ETH Zurich, Switzerland.  
buehrern@ethz.ch, {zhejun.zhang, alex.liniger, vangool}@vision.ee.ethz.ch, i@yf.io

<sup>2</sup>Luc Van Gool is with PSI, KU Leuven, Belgium.

[11], [12], we additionally propose a novel multiplicative value function where the safety critic explicitly addresses constraints and discounts a reward critic that only estimates constraint-free returns. The multiplicative value function has several advantages. A standard RL algorithm can learn safe behavior by specifying a penalty for constraint-violating actions. However, the performance can be sensitive to the magnitude of the penalty [13]. In contrast, we do not need to specify the magnitude: the reward critic neglects constraint violating returns, and the safety critic learns a binary decision. Moreover, penalties are often large in magnitude to discourage standard RL agents from constraint violating actions. As a result, there can be sharp discontinuities in the value landscape as shown in Fig. 1, which makes it difficult for a regular neural network to learn the value function. In our approach, the reward critic does not have to learn these discontinuities. Instead, the responsibility is shifted to the safety critic that estimates the probability of constraint violation. This makes the model optimization better behaved and leads to faster convergence with increased stability.

We combine our approach with two popular algorithms: Proximal Policy Optimization (PPO) [14] and Soft Actor-Critic (SAC) [15], but in general, it can be combined with any on-policy or off-policy method that relies on a value or advantage function. To evaluate the effectiveness of our approach, we construct four safety-critical environments ranging from low to high dimensional observations based on images and Lidar scans. Finally, we deploy our algorithm on a real robot and perform map-less navigation.

Our contributions are summarized as follows:

- We introduce a novel multiplicative value function, that combines a regular value function and a safety critic in a multiplicative fashion. We integrate this multiplicative value function into PPO and SAC.
- We test our methods on a suit of safety-critical environments and show that our methods outperform competing safe RL methods.
- We conduct experiments on a real-world robot navigation task in a cluttered environment and show zero-shot sim-to-real transfer.

## II. RELATED WORK

An overview of the recent advances in safe RL can be found in the latest surveys [16], [17]. In this section, we will focus on the prior works most related to our approach.

**Primal-dual approaches**, i.e., Lagrangian-based approaches, convert the CMDP into an unconstrained problem by relaxing the safety constraint with a Lagrange multiplier.

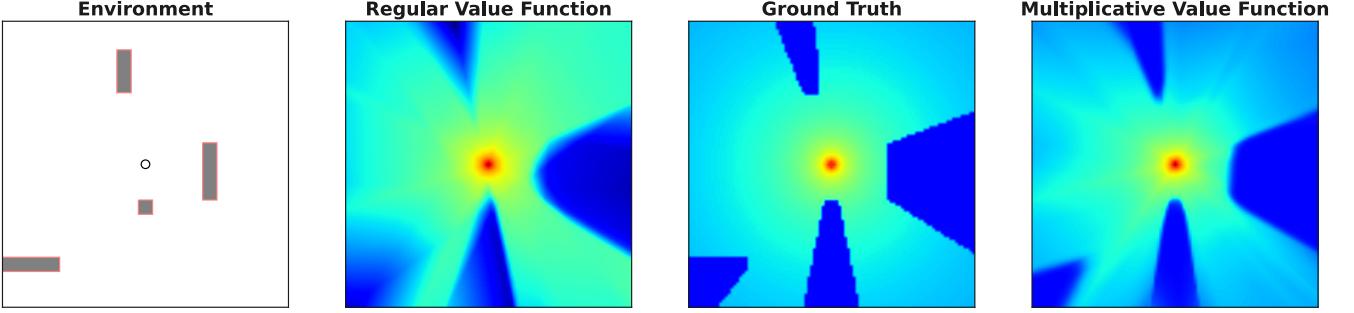


Fig. 1: Linear Quadratic Regulator policy evaluation of a point robot navigating towards the goal in the middle while avoiding the gray obstacles. The ground truth return shows sharp discontinuities that are better replicated by the multiplicative value function.

This is the most straightforward way to solve a CMDP and is also the most related to our work. A Lagrange multiplier with heuristic update rules was first proposed in [9]. More recent works established theoretical groundwork by proving convergence guarantees [10], [18] and zero duality gap [19]. Practically, the Lagrangian approach has been integrated with PPO/TRPO [20] and SAC [21], [22]. Our PPO implementation is similar to [20] except that our multiplicative value function adds a secondary mechanism that improves the learning stability. For our SAC integration, the gradient of the multiplicative value function already naturally results in something similar to a Lagrange multiplier. Commonly, the safety constraint is formulated as a constraint budget over the expected safety cost, e.g., [10], [20], [21], [23], [12], [11]. However, only considering the expected safety at each time step can cause the realized cost to exceed the constraint budget [22]. To provide better constraint satisfaction, worst-case analysis can be performed with the conditional value at risk [10], [22]. We tackle this issue by using reachability analysis and imposing zero constraint violation probability in our experiments. Another line of research improves the stability of the learning process by using derivatives and integrals of the constraint function yielding a PID approach [12], [23]. Similarly, our multiplicative value function improves stability by simplifying the learning task. Lastly, there are works that ensure state-wise safety with a learned Lagrange multiplier [21] and introduce safety transfer learning [11].

**Primal approaches** solve the CMDP by computing a policy gradient that satisfies the constraints. Prior works have achieved this in different ways, for example by searching the feasible policy in the trust region [13], projecting the unconstrained policy [24], projecting the optimum of the constrained non-parametric policy optimization [25], restricting the policy via log-barrier functions [26], alternating between objective improvement and constraint satisfaction [27] or deriving an equivalent unconstrained problem [28], [29]. Since the primal algorithms are harder to implement but not superior in terms of performance, they are less popular than the primal-dual algorithms.

**Lyapunov approaches** address the CMDP by leveraging Lyapunov functions, which are used in control theory to prove the stability of a dynamical system. In terms of

model-based RL, the Lyapunov function can be used to guarantee that an agent can be brought back to a region of attraction [30]. More recently, this approach has been applied to model-free RL [31] and extended in [32] by an exploratory policy that maximizes its knowledge about safety. In [33], the policy optimization is constrained on the Lyapunov decrease condition, which is then relaxed with a Lagrange multiplier.

**Intervention approaches** use backup policies to ensure safe actions. Wagener et al. [34] defines an intervention rule based on the safety advantage between the action proposed by the backup policy and the RL agent. Another possibility is to construct a safe set using model-based or learning-based approaches, or a combination of both. Examples are control barrier functions [35], [36], reachability methods [37], [38] or predictive safe set algorithms [39], [40], [41].

### III. PRELIMINARIES

**Lagrangian methods.** Let us consider the optimization problem  $\min_x f(x)$ , st.  $g(x) \leq c$ , using Lagrangian primal-dual methods this problem can be cast to an unconstrained problem

$$(x^*, \lambda^*) = \min_x \max_{\lambda \geq 0} f(x) + \lambda(g(x) - c),$$

where  $\lambda$  denotes the dual variable or Lagrange multiplier [42].

**CMDP formulation.** The interaction between the agent and the environment can be modeled with a Markov Decision Process (MDP). The MDP is defined by the Tuple  $(\mathcal{S}, \mathcal{A}, r, P, s_0)$ . Here  $\mathcal{S}$  and  $\mathcal{A}$  denote the state and action space respectively, the transition probability is  $P(\cdot|s, a)$  and the reward is  $r(s, a)$ . Finally,  $s_0 \in \mathcal{S}$  is the initial state. For simplicity, we consider deterministic rewards and initial states, but our results can be easily generalized to random initial state distributions and rewards. Extending an MDP with constraints yields a CMDP which is described by the tuple  $(\mathcal{S}, \mathcal{A}, r, P, s_0, r_c, c_{\max})$ . Here,  $r_c$  is a safety cost and  $c_{\max} \in \mathbb{R}_{\geq 0}$  is an upper bound on the expected cumulative safety cost. This yields the safety constraint  $\mathbb{E}^\pi [\sum_{t=0}^{\infty} \gamma_c^t r_c(s_t)|s_0] \leq c_{\max}$ . The objective of the CMDP

is to find an optimal policy  $\pi^*$  according to

$$\begin{aligned} \max_{\pi \in \Delta} \quad & \mathbb{E}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) \middle| s_0 \right], \\ \text{s.t.} \quad & \mathbb{E}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t r_c(s_t) \middle| s_0 \right] \leq c_{\max}. \end{aligned} \quad (1)$$

If the feasibility set induced by the safety constraint in Eq. 1 is non-empty, then there exists an optimal policy  $\pi^*$  in the class of stationary Markovian policies  $\Delta$  [8].

**Reachability.** To make the safety constraint in Eq. 1 more tangible, we specify the (un)safety as  $P(\exists k : s_k \in \mathcal{C} | s_0 = s_t)$ , i.e., the probability of visiting states in the constraint set  $\mathcal{C} \subseteq \mathcal{S}$ . We consider constraint violations as catastrophic, thus states  $s \in \mathcal{C}$  are terminal states. Coming back to the CMDP, the reachability problem can be cast to a value function by setting  $r_c(s_t) = \mathbb{1}_{\mathcal{C}}(s_t)$ ,

$$\begin{aligned} P(\exists k : s_k \in \mathcal{C} | s_0 = s_t) &:= \Phi^\pi(s_t) \\ &= \mathbb{E}^\pi \left[ \sum_{k=0}^{\infty} \gamma_c^k \mathbb{1}_{\mathcal{C}}(s_k) \middle| s_0 = s_t \right], \end{aligned} \quad (2)$$

where  $\mathbb{1}_{\mathcal{C}}$  is the indicator function of the constraint set  $\mathcal{C}$ .

For the proof, we closely follow [9]. First, we note that the sum  $R_c := \sum_{k=0}^{\infty} \gamma_c^k r_c(s_k)$  is finite and at most 1, namely when a constraint violation occurs and we reach a terminal state. Thus, when setting  $\gamma_c = 1$ , it holds that  $R_c = 1$  if  $\exists t : s_t \in \mathcal{C}$  else 0. We note that  $R_c$  is a Bernoulli Random variable and define  $P(R_c = 1) := q$ . From the Bernoulli distribution we know that  $\mathbb{E}[R_c] = q = \Phi^\pi(s_t)$ .

Practically, a lower discount factor can increase the learning stability when used in an RL setup [9]. Furthermore, we denote  $\Phi^\pi(s_t)$  as the safety critic and similar to Eq. 2, we define the action value safety critic as  $\Psi^\pi(s_t, a_t)$ .

#### IV. METHODS

**Environment structure.** We assume the following bounded reward structure of the environment

$$r(s_t, a_t) = \begin{cases} r_{\text{constraint}} & \text{if } s_t \in \mathcal{C} \\ r_{\text{constraint\_free}}(s_t, a_t) & \text{else} \end{cases}, \quad (3)$$

with  $r_{\text{constraint}} \ll \min_{s,a} r_{\text{constraint\_free}}(s, a)$  and the constraint set  $\mathcal{C}$  being a terminal state. The low reward for violating the constraint discourages standard RL agents from executing constraint violating actions. However, as shown in Fig. 1, the difference of magnitude between  $r_{\text{constraint}}$  and  $r_{\text{constraint\_free}}$  can cause discontinuities in the value landscape that are difficult to learn.

**Multiplicative value function.** The motivation behind our multiplicative value function is to facilitate the learning by splitting responsibilities. The safety critic  $\Phi^\pi(s_t)$  explicitly handles constraints, whereas the reward critic  $\bar{V}^\pi(s_t)$  only estimates constraint-free returns. As argued in Sec. I, it is neither favorable to specify a magnitude for  $r_{\text{constraint}}$  nor to learn  $r_{\text{constraint}}$  with the reward critic  $\bar{V}^\pi(s_t)$ . Instead, we

propose to clip the reward in Eq. 3, and learn the reward critic with this constrain neglecting reward,

$$\begin{aligned} \bar{r}(s_t, a_t) &= \begin{cases} \min_{s,a} r_{\text{constraint\_free}}(s, a) & \text{if } s_t \in \mathcal{C} \\ r_{\text{constraint\_free}}(s_t, a_t) & \text{else} \end{cases}, \\ \bar{V}^\pi(s_t) &= \mathbb{E}^\pi \left[ \sum_{k=0}^{\infty} \gamma^k \bar{r}(s_k, a_k) \middle| s_0 = s_t \right]. \end{aligned}$$

By taking the minimum in case of a constraint violation, we lightly discourage the policy from constraint violating actions. This can be especially useful when approximation errors cause the safety critic to be overly optimistic. Finally, the multiplicative value function  $V_{\text{mult}}^\pi(s_t)$  is obtained by discounting the reward critic with the probability of constrained satisfaction:

$$V_{\text{mult}}^\pi(s_t) := (\bar{V}^\pi(s_t) - \bar{v}_{\min}) \cdot (1 - \Phi^\pi(s_t)) + \bar{v}_{\min},$$

where  $\bar{v}_{\min} := \min_s \bar{V}^\pi(s)$  is the lower bound on the reward critic, such that  $(\bar{V}^\pi(s_t) - \bar{v}_{\min}) \geq 0$ . Practically, we set  $\bar{v}_{\min}$  to the minimum encountered  $\bar{V}^\pi$  value during training. The multiplicative combination of the two critics allows a hyperparameter-free fusion, where a constraint violating state is associated with the value of  $\bar{v}_{\min}$  and for save states, the value is  $\bar{V}^\pi(s_t)$ . Similarly, we define  $Q_{\text{mult}}^\pi(s_t, a_t)$  with  $\bar{q}_{\min}$  as the multiplicative action value function:

$$\begin{aligned} Q_{\text{mult}}^\pi(s_t, a_t) &:= [\bar{Q}^\pi(s_t, a_t) - \bar{q}_{\min}] \cdot (1 - \Psi^\pi(s_t, a_t)) + \bar{q}_{\min}. \end{aligned} \quad (4)$$

Note that the offset terms  $\bar{v}_{\min}$  and  $\bar{q}_{\min}$  could be avoided by assuming positive rewards. Nevertheless, introducing this term allows our formulation to handle arbitrary reward functions.

**Multiplicative advantage.** We also want to consider advantage-based policy gradient methods. The advantage  $A^\pi(s_t, a_t)$  is defined as,

$$A^\pi = Q^\pi - V^\pi = [r(s_t, a_t) + \gamma V^\pi(s_{t+1})] - V^\pi(s_t). \quad (5)$$

From this, we derive three versions of the multiplicative advantage  $A_{\text{mult}}^\pi$ :

$$\begin{aligned} \text{V1: } & [\bar{r}_t + \gamma V_{\text{mult}}^\pi(s_{t+1})] - V_{\text{mult}}^\pi(s_t) \\ \text{V2: } & Q_{\text{mult}}^\pi(s_t, a_t) - V_{\text{mult}}^\pi(s_t) \\ \text{V3: } & [\bar{Q}^\pi(s_t, a_t) - \bar{q}_{\min}] [1 - (r_{c,t} + \gamma_c \Phi^\pi(s_{t+1}))] \\ & + \bar{q}_{\min} - V_{\text{mult}}^\pi(s_t) \end{aligned} \quad (6)$$

In V1 and V2, we consider Eq. 5 and replace all value functions with their multiplicative counterparts. Finally, V3 is similar to V2, but in Eq. 4 we use temporal difference bootstrapping for the safety critic.

**Integration into SAC.** We integrate the multiplicative value function  $Q_{\text{mult}}^\pi$  into the actor objective of SAC by replacing  $Q^\pi$  with  $Q_{\text{mult}}^\pi$ ,

$$\max_{\theta} \mathbb{E}_{a_\theta \sim \pi_\theta} [Q_{\text{mult}}^{\pi_\theta}(s, a_\theta) - \alpha \log \pi_\theta(a_\theta | s)]. \quad (7)$$

We call this version SAC Mult. For a compact notation, we drop the dependency on  $(s, a_\theta, \pi)$  in the following. To get

a better intuition about Eq. 7, we investigate the gradient of the SAC Mult objective

$$\nabla_{\theta} Q_{\text{mult}} = (1 - \Psi) \cdot \nabla_{\theta} \bar{Q} - (\bar{Q} - \bar{q}_{\min}) \cdot \nabla_{\theta} \Psi,$$

which has two terms. The first term is the gradient of the  $\bar{Q}$ -function discounted by the probability of constraint satisfaction. The second term is the gradient of the safety critic  $\nabla_{\theta} \Psi$  discounted by  $(\bar{Q} - \bar{q}_{\min})$ , which can be understood as q-weighted multiplier. The disadvantage of this formulation is that in states where  $\bar{Q}$  is high, the q-weighted multiplier becomes large and the gradient of the safety critic dominates the overall gradient. This can yield overly conservative behaviors. To mitigate this issue, we additionally propose two heuristics, SAC Mult Clipped

$$\nabla_{\theta} Q_{\text{mult}} \approx (1 - \Psi) \cdot \nabla_{\theta} \bar{Q} - \min(\bar{Q} - \bar{q}_{\min}, \lambda_{\max}) \cdot \nabla_{\theta} \Psi$$

and SAC Mult Lagrange

$$\nabla_{\theta} Q_{\text{mult}} \approx (1 - \Psi) \cdot \nabla_{\theta} \bar{Q} - \lambda \cdot \nabla_{\theta} \Psi.$$

In Mult Clipped, we limit the magnitude of the multiplier with  $\lambda_{\max}$  which is a hyperparameter. In Mult Lagrange, we replace the q-weighted multiplier with a Lagrange multiplier that is optimized using primal-dual optimization. Under mild assumptions, this is guaranteed to converge to a local optimum of the CMDP [10]. The Mult Lagrange objective is  $\max_{\theta} \min_{\lambda \geq 0} \mathbb{E}_{a_{\theta} \sim \pi_{\theta}} [(1 - \Psi(s, a_{\theta})) \cdot \text{detach}() \cdot \bar{Q}(s, a_{\theta}) - \alpha \log \pi_{\theta}(a_{\theta} | s) - \lambda \cdot (\Psi(s, a_{\theta}) - c_{\max})]$ , where `.detach()` denotes the operation of detaching the variable from the computational graph. For the exact implementation, we refer to our code.

**Integration into PPO.** Given the three versions of the multiplicative advantage  $A_{\text{mult}}^{\pi}$  in Eq. 6, we can integrate them into the actor objective of PPO and extend it with a Lagrange multiplier

$$\begin{aligned} \max_{\theta} \min_{\lambda \geq 0} \mathbb{E}_{s, a \sim \pi_{\theta_k}} & [\min \left\{ \frac{\pi_{\theta}(a | s)}{\pi_{\theta_k}(a | s)} A_{\text{mult}}^{\pi_{\theta_k}}, g(\epsilon, A_{\text{mult}}^{\pi_{\theta_k}}) \right\} \\ & - \lambda \cdot \mathbb{E}_{a_{\theta} \sim \pi_{\theta}} [\Psi^{\pi}(s, a_{\theta}) - c_{\max}]], \end{aligned}$$

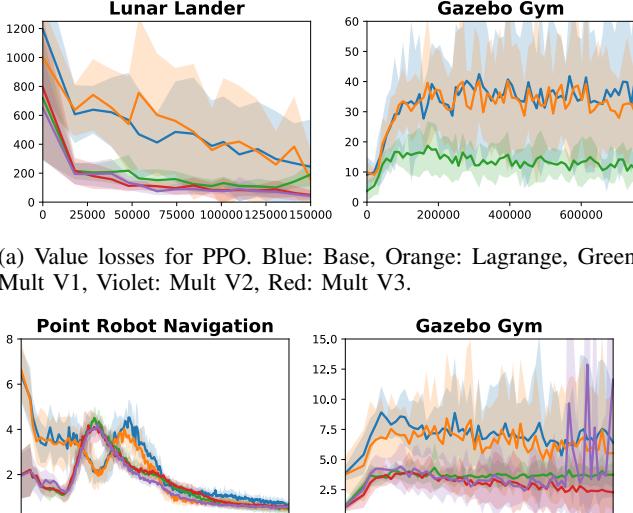
with  $g(\epsilon, A) = (1 + \epsilon)A \cdot \mathbf{1}_{A \geq 0} + (1 - \epsilon)A \cdot \mathbf{1}_{A < 0}$ . For the optimization of the Lagrange multiplier, we again proceed as in [10] to guarantee convergence to a locally optimal policy.

## V. EXPERIMENTAL RESULTS

We evaluate our methods in four environments: Lunar Lander Safe, Car Racing Safe, Point Robot Navigation, and Gazebo Gym. The first two are derived from OpenAI’s gym [44] and extended with safety constraints. For Lunar Lander Safe, we impose the constraint that the agent can only land within the landing zone. In Car Racing Safe, we test how our algorithm deals with both hard and soft constraints. We set the hard constraint that the agent is not allowed to leave the track. For the soft constraint, we encourage the agent to drive below 50 km/h. If the soft constraint is violated, the agent gets a small negative reward, but the episode still continues. The agent observes the environment via an agent centered bird’s-eye-view image and a vector containing information

TABLE I: SAC and PPO evaluation results. Overall, PPO Mult V1 delivers most consistently good performance across environments. This might be because V1 is based on the Generalized Advantage Estimation [43] which has shown to work particularly well for standard PPO. Among the SAC derivatives, Mult Clipped and Mult Lagrange perform the most consistent. For SAC Mult, the q-weighted safety multiplier yields overly conservative behavior in Lunar Lander, where the agent rarely lands, but instead times out. Consequently the reward is low.

	Reward ↑	% Constraint violations ↓	Reward ↑	% Constraint violations ↓
Lunar Lander Safe				
SAC	50k		150k	
SAC base	90 ± 108	10 ± 16	181 ± 117	3 ± 6
Lagrange	111 ± 105	17 ± 13	184 ± 128	2 ± 3
Mult	-35 ± 27	<b>3 ± 5</b>	-34 ± 22	3 ± 4
Mult Clipped	<b>134 ± 94</b>	14 ± 13	243 ± 49	8 ± 15
Mult Lagrange	125 ± 59	29 ± 15	<b>251 ± 20</b>	<b>2 ± 2</b>
PPO	50k		150k	
PPO base	-126 ± 158	77 ± 29	225 ± 100	10 ± 30
Lagrange	-24 ± 146	54 ± 39	204 ± 116	12 ± 24
V1	101 ± 84	41 ± 19	205 ± 78	7 ± 16
V2	89 ± 122	44 ± 34	251 ± 28	5 ± 9
V3	<b>144 ± 4</b>	<b>26 ± 22</b>	<b>264 ± 5</b>	<b>1 ± 2</b>
FOCOPS	-129 ± 21	64 ± 24	117 ± 80	30 ± 19
Point Robot Navigation				
SAC	50k		100k	
SAC base	22 ± 19	1 ± 1	<b>38 ± 1</b>	1 ± 1
Lagrange	29 ± 8	<b>0 ± 0</b>	<b>38 ± 1</b>	<b>0 ± 0</b>
Mult	34 ± 3	<b>0 ± 0</b>	<b>38 ± 1</b>	<b>0 ± 0</b>
Mult Clipped	33 ± 4	<b>0 ± 0</b>	<b>38 ± 1</b>	<b>0 ± 0</b>
Mult Lagrange	<b>37 ± 2</b>	<b>0 ± 0</b>	37 ± 2	<b>0 ± 0</b>
PPO	250k		500k	
PPO base	15 ± 15	3 ± 3	31 ± 3	3 ± 2
Lagrange	15 ± 16	3 ± 3	24 ± 5	3 ± 2
V1	25 ± 5	3 ± 3	30 ± 4	2 ± 1
V2	11 ± 21	3 ± 3	17 ± 15	3 ± 1
V3	<b>27 ± 5</b>	5 ± 2	29 ± 3	3 ± 3
FOCOPS	17 ± 11	<b>0 ± 0</b>	<b>33 ± 2</b>	<b>0 ± 0</b>
Gazebo Gym				
SAC	100k		200k	
SAC base	34 ± 6	7 ± 9	35 ± 5	6 ± 8
Lagrange	30 ± 7	11 ± 12	35 ± 6	5 ± 10
Mult	34 ± 11	5 ± 12	35 ± 6	<b>3 ± 7</b>
Mult Clipped	<b>36 ± 5</b>	<b>3 ± 8</b>	<b>36 ± 5</b>	3 ± 8
Mult Lagrange	36 ± 6	4 ± 9	<b>36 ± 5</b>	3 ± 8
PPO	250k		750k	
PPO base	27 ± 6	18 ± 11	<b>31 ± 5</b>	12 ± 8
Lagrange	26 ± 5	19 ± 9	31 ± 6	12 ± 9
V1	<b>30 ± 6</b>	<b>14 ± 9</b>	<b>31 ± 5</b>	<b>11 ± 8</b>
FOCOPS	13 ± 12	19 ± 8	29 ± 5	15 ± 9
Car Racing Safe				
PPO	500k		1000k	
PPO base	43 ± 23	28 ± 25	89 ± 25	9 ± 15
Lagrange	43 ± 23	28 ± 25	89 ± 25	9 ± 15
V1	<b>74 ± 19</b>	<b>17 ± 14</b>	98 ± 13	<b>9 ± 7</b>
V2	65 ± 24	26 ± 15	<b>100 ± 9</b>	10 ± 6
V3	73 ± 21	25 ± 15	78 ± 17	22 ± 16
FOCOPS	-2 ± 2	93 ± 2	-2 ± 2	93 ± 2



(a) Value losses for PPO. Blue: Base, Orange: Lagrange, Green: Mult V1, Violet: Mult V2, Red: Mult V3.

(b) Value losses for SAC. Blue: Base, Orange: Lagrange, Green: Mult, Red: Mult Clipped, Violet: Mult Lagrange.

Fig. 2: Qualitative Results

about the steering angle, yawing rate, and velocity. The last two environments focus on robotic navigation of ground robots. In Point Robot Navigation, the agent is a point robot that has to navigate towards the goal while avoiding obstacles. At each iteration, a new set of random obstacles is spawned and the agent starts at a random position. The agent perceives its environment via a local occupancy grid and the vector from the current position to the goal. The Gazebo Gym is similar but the agent is a Jackal differential drive robot modeled in Gazebo [45] and the occupancy grid is replaced by a 1D-Lidar scan. Again, the task is to navigate to the goal that lies in the middle of a cluttered room.

For the tuning, we use a sequential grid search on the learning rate, entropy coefficient, the number of optimization steps and experiment with the Beta distribution for the policy. For FOCOPS, we additionally tune lambda, the batch size and the KL divergence target. Furthermore, we tune the KL target and the clip range for PPO and the training frequency for SAC. After the baseline tuning, we keep the same hyperparameters for our multiplicative versions and additionally tune the safety discount factor  $y_c$  and the initial value of the Lagrange multiplier  $\lambda_{\text{init}}$ .

#### A. Results and Comparisons

With our experiments, we want to answer two questions: Firstly, can the integration of the multiplicative value function facilitate the learning, leading to faster convergence and improved stability? For this, we integrate our approach into the SAC and PPO and compare against its Lagrangian counterpart. Secondly, we ask, can the integration of the multiplicative value function into Lagrangian approaches yield comparable performance to recent approaches, e.g., FOCOPS [25]? All the results are shown in Table I, where we evaluate each model at an intermediate checkpoint and at the end of the training. Each evaluation is over 10 seeds

with 100 episodes.

**Increased sample efficiency.** One of the main motivations for the multiplicative value function is to simplify the learning task. This is supported by Fig. 2, where we observe a lower mean value loss and reduced variance across environments for both SAC and PPO. Having a simpler learning task allows our multiplicative versions to achieve significantly fewer constraint violations and higher rewards at the first evaluation checkpoint, indicating greater sampling efficiency. At the final evaluation checkpoint, our method achieves similar constraint satisfaction as the Lagrangian baseline. This is expected since both are Lagrangian methods and with enough training samples and model capacity, the regular value function can properly learn the value landscape.

**Constraint satisfaction.** In simpler environments, like Lunar Lander and Point Robot Navigation, our approach nearly achieves the target of zero constraint violations with PPO and SAC. In Car Racing Safe, the Lagrangian baseline and PPO V1 achieve 91% constraint satisfaction. The imperfect performance could be caused by the challenging environment setup where minor driving errors can result in a crash. Due to the long run-time, we stopped the Gazebo Gym experiments in Table I before convergence. In fact, SAC Mult Lagrange trained for 2 days as done for the sim-to-real transfer in Sec. V-B achieves a constraint satisfaction of 100%.

**PPO vs. SAC.** Overall, we achieve better constraint satisfaction with SAC agents in the navigation tasks Point Robot Navigation and Gazebo Gym. The only caveat is that we were not able to successfully train any SAC (nor FOCOPS) algorithm on Car Racing Safe because the agents never “make” the first corner. Overall for PPO, we observe an increased variance in the training reward if the maximum allowed KL divergence is not explicitly tuned. The tuning is necessary because the multiplicative value function together with the Lagrange multiplier yields more aggressive policy updates which can cause instabilities.

**Soft constraint satisfaction.** In Car Racing Safe, we additionally imposed the soft constraint to keep the velocity below 50 km/h. Even though PPO V1 and V2 achieve similar constraint satisfaction to the Lagrangian baseline, the reward is higher. This is because V1 and V2 violate the soft constraint with 20% and 12% respectively, whereas the Lagrangian Baseline violates the constraint in 32% of the steps. We credit the multiplicative value function for the improved soft-constraint satisfaction. By facilitating the learning, the reward critic has more capacity to learn the fine details in the reward structure, like the soft constraint on the velocity.

**Increased stability.** We observe better training stability across seeds with the multiplicative value function. This is most prominent in Lunar Lander Safe, where we observe a large variance in the Lagrangian baseline rewards. This is because the Lagrangian agent only lands in 80% of the seeds reliably, both for SAC and PPO. In contrast, SAC Mult Lagrange, Clipped and PPO Mult V2, V3 agents manage to land in all seeds, PPO Mult V1 in 90% of the seeds. The poor

performance of SAC Mult is not due to missing stability, in fact, SAC Mult performs badly across seeds. The issue is caused by the potentially large q-weighted multiplier, which makes the policy updates overly conservative leading to high timeout rates without ever landing.

**Qualitative results.** In Fig. 3a, we show the qualitative results for Point Robot Navigation. Of most interest is the multiplicative value function, which can better represent the obstacles highlighted by red boxes. Furthermore, the trajectories of SAC Mult Clipped seem more directed towards the goal compared to the Lagrangian baseline. In Lunar Lander Safe, we observe the Mult agents land faster than the Lagrangian agents by having greater downward speeds high above the landing pad, while at lower altitudes, landing as cautiously as the Lagrangians.

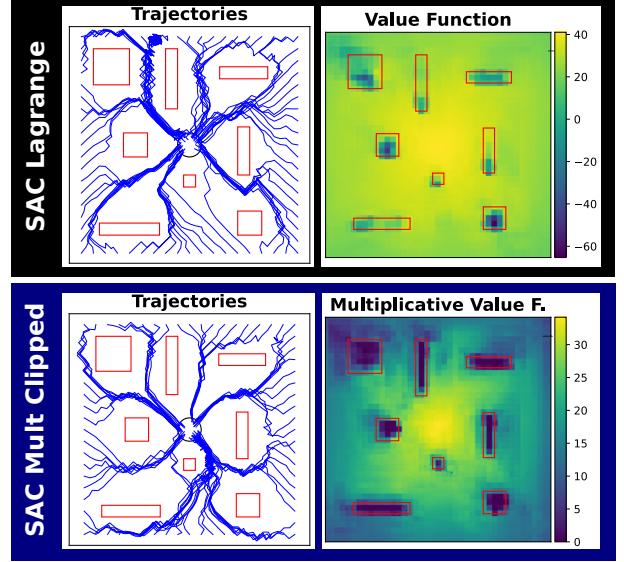
**Theoretical guarantees vs. heuristics.** Based on [10], we have theoretical safety guarantees for all PPO Multiplicative versions as well as for SAC Mult Lagrange. The SAC Mult and Clipped inhibit a q-weighted multiplier from the gradient of the multiplicative value function. However, this multiplier is not a Lagrangian multiplier, thus has no theoretical guarantees. Practically, we observe that SAC Mult and Mult Clipped have similar constraint satisfaction as SAC Mult Lagrange.

**Comparison to FOCOPS.** For Lunar Lander, the FOCOPS evaluation result at 150k steps is significantly worse than for any PPO Mult algorithm. We suspect this is caused by poor sample efficiency. Therefore, we train FOCOPS up to 300k steps where it obtains a reward of  $215 \pm 65$  and a constraint violation rate of  $13 \pm 18\%$ . This is still worse than any Mult algorithm at 150k steps and is due to two seeds showing high constraint violation rates of 53% and 38%. In Point Robot Navigation, our algorithms converge faster but finally, FOCOPS outperforms all PPO Mult agents and is only beaten by SAC. In Gazebo Gym, FOCOPS performs worse than PPO in both evaluations, however, longer training could yield more comparable performance. Finally, in Car Racing Safe the FOCOPS agent never gets passed the first corner similar to SAC.

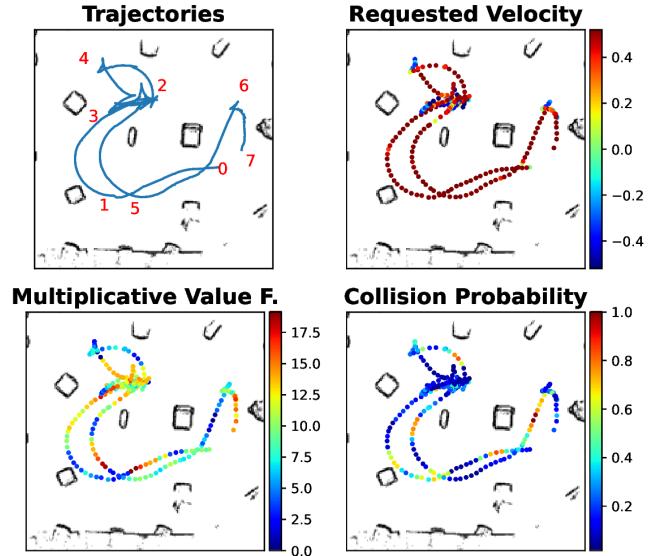
## B. Real-World Experiments

Based on the good performance of SAC Mult Lagrange on Gazebo Gym, the question arises of how the learned policy performs on a real Jackal robot? Can we tackle navigation, one of the fundamental robotic problems, in a safe way while only given sparse Lidar observations and a direction to the goal? To this end, we trained the policy for 4M steps in simulation with action noise, a smaller goal region of 0.3m and noisy Lidar observations. Furthermore, we included a cross-attention encoder for both the policy and value nets as depicted in Fig 4. This attention mechanism allows the networks to focus on the latent representation of certain Lidar rays, for example the rays that are pointing forward. Those changes helped the policy to achieve a 100% success rate with 0% constraint violations in the simulation.

One of the main differences between simulation and reality is that the ground friction in the real world is larger and the



(a) Evaluation of SAC Mult Lagrange and the Lagrangian baseline after 100k steps on Point Robot Navigation. The multiplicative value function better represents obstacles.



(b) Trajectories of real-world experiment with a differential drive robot and SAC Mult Lagrange. Goal regions are marked with numbers 1-7. Starting point is 0. Here, success rate is 100%.

Fig. 3: Qualitative results on robot navigation environments.

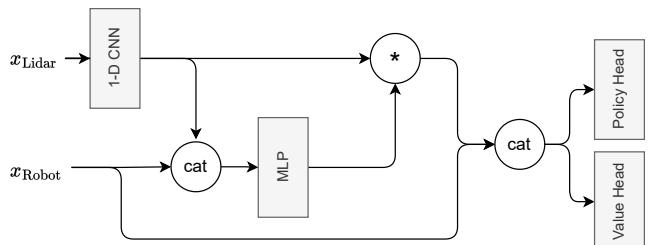


Fig. 4: Attention Encoder used for the real world experiments. Here  $x_{\text{Robot}}$  is the vector from current position to the goal and  $x_{\text{Lidar}}$  is the measurement from the 1D-Lidar.

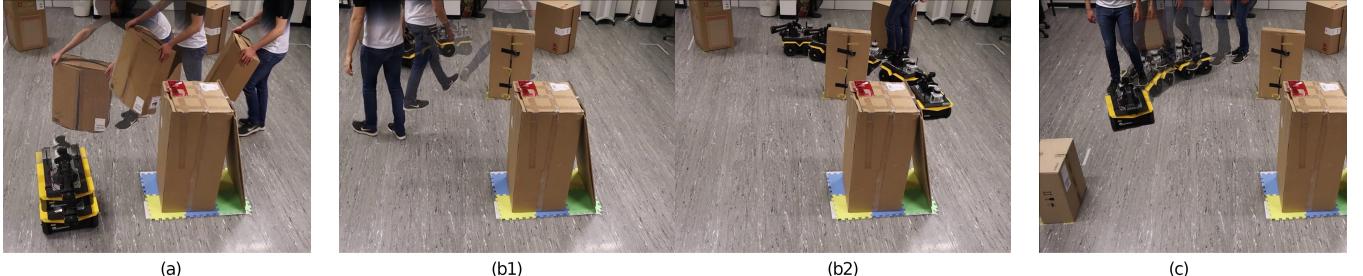


Fig. 5: (a): We dynamically put an obstacle in the trajectory of the robot which causes it to stop. The robot then waits until the obstacle is removed and continues its trajectory. (b1, b2): Dynamic interactions with the robot by first walking next to it (b1), then overtaking it and standing to the side of the box (b2). While the person walks next to it the robot continues its trajectory. When the person overtook the robot, it swings to the right to avoid the collision. (c): The person walks towards the robot. To avoid the collision, the robot drives backwards. The complete video is available at <https://youtu.be/gAcETwOWTM4>.

velocity controller behaves differently, i.e., given the same velocity command, the robot moves faster in simulation. In reality, if the robot is stationary and the velocity command is chosen too small, the robot can remain stationary due to the increased friction and different behavior of the velocity controller. Another real-world difficulty is the delay of 0.3s from the Lidar rays being recorded, send to the off-board computer, and the policy commands being sent back and executed on the robot. However, the fact that the real robot moves slower mitigates the time delay to an extent.

In the lab, we constructed a cluttered obstacle course and directly deployed the policy trained in simulation on the robot. We defined seven goal regions and let the robot pass them four times. One of the four runs is depicted in Fig. 3b. The robot drives with a direct trajectory from start 0 to goals 1 and 2, where it needs to reverse its direction by 180 degrees to approach goal 3. Interestingly, the policy did not learn to turn on the spot a differential drive robot would allow, but instead, the trajectories from goal 2-4 resemble more a kinematic bicycle model. This unfortunately caused the robot to get stuck once between goals 2-3, however, without violating a safety constraint. Overall, we report a success rate of 96% with 100% constraint satisfaction meaning that no box was touched.

Encouraged by the demonstrated safety of our algorithm, we wanted to see if our agent can generalize from static box obstacles as encountered in the simulation to moving obstacles like humans in reality. For this, we arranged the goals in a circle and sequentially let the robot pass them. In the first experiment, we suddenly put a box in front of the agent as shown in Fig. 5 (a). The robot reacted in a safe manner and came to an immediate stop. After a few seconds, we removed the box and the robot continued its original trajectory. In the second experiment shown in Fig. 5 (b) we wanted to go a step further and see how the robot deals with obstacle shapes it has never seen before, i.e., human legs. Additionally, we wanted to know if the robot could naturally interact with a human walking next to it. For this, we started behind the robot, and then walked next to it at a certain safety distance, see Fig. 5 (b1). This did not visibly influence the robot’s trajectory. When we overtook the robot,

we positioned ourselves close to a box such that we were in the trajectory of the robot, see Fig. 5 (b2). Because of that, the robot corrected its trajectory and steered away from us. The final interaction is shown in Fig. 5 (c). Here we wanted to investigate what happens if we actively provoke a collision by moving towards the robot. In that case, the robot started to move backward to keep a certain safety distance from us. The only drawback is that when relentlessly chasing the robot one can cause a rear collision with another obstacle. An explanation for this is that the robot has a Lidar blind spot in the back due to the mounted robot arm. All the interactions can be found in the supplementary video.

## VI. CONCLUSIONS AND LIMITATIONS

In this work, we introduced a safety critic to yield a multiplicative value function. We started with the CMDP formulation, derived the safety critic from reachability analysis and integrated our approach into the SAC and PPO framework. We proposed several versions of SAC and PPO using our multiplicative value function and showed increased sample efficiency and stability compared to the Lagrangian and FOCOPS baselines. Furthermore, the multiplicative value function can help to learn the fine details in the reward structure, like soft constraints. To show the real-world potential of our method, we took a SAC Mult Lagrange agent trained in simulation and successfully deployed the policy on a real robot in a zero-shot sim-to-real fashion. The robot showed safe behavior and was able to generalize to dynamic obstacles of novel shape. In future work, we would like to investigate further theoretical justification for our multiplicative value function.

**Limitations.** One of the main limitations is that our Lagrangian approach encourages safety during training but cannot guarantee it. This issue can be mitigated by adding an intervention mechanism, which could however cause problems when learning the safety critic as it requires reaching the constraint set. Future work will investigate the feasibility of using the intervention as a terminal state and more theoretical analysis of the multiplicative value function. Moreover, our method adds the initial value of the Lagrange multiplier and the safety discount factor  $\gamma_c$  as hyperparameters to which the algorithm can be sensitive in some environments.

## REFERENCES

- [1] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing atari with deep reinforcement learning,” *arXiv*, 2013.
- [2] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, *et al.*, “Mastering the game of go without human knowledge,” *nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [3] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, *et al.*, “Grandmaster level in starcraft ii using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [4] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, *et al.*, “Dota 2 with large scale deep reinforcement learning,” *arXiv*, 2019.
- [5] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter, “Learning robust perceptive locomotion for quadrupedal robots in the wild,” *Science Robotics*, vol. 7, no. 62, p. eabk2822, 2022.
- [6] F. Fuchs, Y. Song, E. Kaufmann, D. Scaramuzza, and P. Dürr, “Super-human performance in gran turismo sport using deep reinforcement learning,” *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4257–4264, 2021.
- [7] Z. Zhang, A. Liniger, D. Dai, F. Yu, and L. Van Gool, “End-to-end urban driving by imitating a reinforcement learning coach,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15222–15232.
- [8] E. Altman, *Constrained Markov Decision Processes*. CRC Press, 1999.
- [9] P. Geibel and F. Wysotski, “Risk-sensitive reinforcement learning applied to control under constraints,” *Journal of Artificial Intelligence Research*, vol. 24, pp. 81–108, 2005.
- [10] Y. Chow, M. Ghavamzadeh, L. Janson, and M. Pavone, “Risk-constrained reinforcement learning with percentile risk criteria,” *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6070–6120, 2017.
- [11] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn, “Learning to be safe: Deep rl with a safety critic,” *arXiv*, 2020.
- [12] A. Stoote, J. Achiam, and P. Abbeel, “Responsive safety in reinforcement learning by pid lagrangian methods,” in *ICML*. PMLR, 2020, pp. 9133–9143.
- [13] J. Achiam, D. Held, A. Tamar, and P. Abbeel, “Constrained policy optimization,” in *ICML*. PMLR, 2017, pp. 22–31.
- [14] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv*, 2017.
- [15] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, “Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor,” in *ICML*. PMLR, 2018, pp. 1861–1870.
- [16] Y. Liu, A. Halev, and X. Liu, “Policy learning with constraints in model-free reinforcement learning: A survey,” in *IJCAI*, 2021, pp. 4508–4515.
- [17] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, Y. Yang, and A. Knoll, “A review of safe reinforcement learning: Methods, theory and applications,” *arXiv*, 2022.
- [18] C. Tessler, D. J. Mankowitz, and S. Mannor, “Reward constrained policy optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.
- [19] S. Paternain, L. Chamon, M. Calvo-Fullana, and A. Ribeiro, “Constrained reinforcement learning has zero duality gap,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.
- [20] A. Ray, J. Achiam, and D. Amodei, “Benchmarking Safe Exploration in Deep Reinforcement Learning,” 2019.
- [21] H. Ma, Y. Guan, S. E. Li, X. Zhang, S. Zheng, and J. Chen, “Feasible actor-critic: Constrained reinforcement learning for ensuring statewise safety,” *arXiv preprint arXiv:2105.10682*, 2021.
- [22] Q. Yang, T. D. Simão, S. H. Tindemans, and M. T. Spaan, “Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning,” in *AAAI*, 2021, pp. 10639–10646.
- [23] B. Peng, Y. Mu, J. Duan, Y. Guan, S. E. Li, and J. Chen, “Separated proportional-integral lagrangian for chance constrained reinforcement learning,” in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 193–199.
- [24] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge, “Projection-based constrained policy optimization,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.
- [25] Y. Zhang, Q. Vuong, and K. Ross, “First order constrained optimization in policy space,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 33, pp. 15338–15349, 2020.
- [26] Y. Liu, J. Ding, and X. Liu, “Ipo: Interior-point policy optimization under constraints,” in *AAAI*, vol. 34, no. 04, 2020, pp. 4940–4947.
- [27] T. Xu, Y. Liang, and G. Lan, “Crpo: A new approach for safe reinforcement learning with convergence guarantee,” in *ICML*. PMLR, 2021, pp. 11480–11491.
- [28] L. Zhang, L. Shen, L. Yang, S. Chen, X. Wang, B. Yuan, and D. Tao, “Penalized proximal policy optimization for safe reinforcement learning,” in *IJCAI*, 2022.
- [29] A. Sootla, A. I. Cowen-Rivers, T. Jafferjee, Z. Wang, D. H. Mgundi, J. Wang, and H. Ammar, “Sauté rl: Almost surely safe reinforcement learning using state augmentation,” in *ICML*. PMLR, 2022, pp. 20423–20443.
- [30] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause, “Safe model-based reinforcement learning with stability guarantees,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.
- [31] Y. Chow, O. Nachum, A. Faust, E. Duenez-Guzman, and M. Ghavamzadeh, “Safe policy learning for continuous control,” in *Conference on Robot Learning*, 2020.
- [32] S. Hub and I. Yang, “Safe reinforcement learning for probabilistic reachability and safety specifications: A lyapunov-based approach,” *arXiv*, 2020.
- [33] L. Zhang, R. Zhang, T. Wu, R. Weng, M. Han, and Y. Zhao, “Safe reinforcement learning with stability guarantee for motion planning of autonomous vehicles,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 12, pp. 5435–5444, 2021.
- [34] N. C. Wagener, B. Boots, and C.-A. Cheng, “Safe reinforcement learning using advantage-based intervention,” in *ICML*. PMLR, 2021, pp. 10630–10640.
- [35] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick, “End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks,” in *AAAI*, vol. 33, no. 01, 2019, pp. 3387–3395.
- [36] K. P. Wabersich and M. N. Zeilinger, “Predictive control barrier functions: Enhanced safety mechanisms for learning-based control,” *IEEE Transactions on Automatic Control*, 2022.
- [37] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin, “A general safety framework for learning-based control in uncertain robotic systems,” *IEEE Transactions on Automatic Control*, vol. 64, no. 7, pp. 2737–2752, 2018.
- [38] K.-C. Hsu, A. Z. Ren, D. P. Nguyen, A. Majumdar, and J. F. Fisac, “Sim-to-lab-to-real: Safe reinforcement learning with shielding and generalization guarantees,” *arXiv*, 2022.
- [39] W. Zhao, T. He, and C. Liu, “Model-free safe control for zero-violation reinforcement learning,” in *Conference on Robot Learning*, 2021.
- [40] K. P. Wabersich and M. N. Zeilinger, “A predictive safety filter for learning-based control of constrained nonlinear dynamical systems,” *Automatica*, vol. 129, p. 109597, 2021.
- [41] A. Wachi and Y. Sui, “Safe reinforcement learning in constrained markov decision processes,” in *ICML*. PMLR, 2020, pp. 9797–9806.
- [42] D. P. Bertsekas, *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.
- [43] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-dimensional continuous control using generalized advantage estimation,” in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.
- [44] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, “Openai gym,” *arXiv*, 2016.
- [45] N. Koenig and A. Howard, “Design and use paradigms for gazebo, an open-source multi-robot simulator,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2004.
- [46] A. Raffin, A. Hill, A. Gleave, A. Kanervisto, M. Ernestus, and N. Dormann, “Stable-baselines3: Reliable reinforcement learning implementations,” *Journal of Machine Learning Research*, 2021.
- [47] A. Raffin, “Rl baselines3 zoo,” <https://github.com/DLR-RM/rl-baselines3-zoo>, 2020.
- [48] A. Raffin, J. Kober, and F. Stulp, “Smooth exploration for robotic reinforcement learning,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1634–1644.

## APPENDIX

### A. Supplementary Video

The supplementary video for the paper can be found here at <https://youtu.be/gAcETwOWTM4>. This video demonstrates the qualitative outcomes of the interaction experiment with the Jackal differential drive robot. In the first part of the video, we dynamically block the path of the agent with a box. Encouraged by the demonstrated safety, we try interactions with a human in the second part of the video. This is challenging in two ways: First, the agent only perceives the legs of the human, which are significantly thinner than the obstacles encountered in training. Second, the agent only encountered static objects in training while the human is a dynamic obstacle. Still, the agent shows safe behavior.

### B. Complete Algorithms for SAC and PPO Multiplicative

Algorithm 1 is the complete version of SAC Multiplicative, whereas Algorithm 2 is the complete version of the PPO Multiplicative. The differences between our proposed methods and the standard algorithms are highlighted in blue.

### C. Hyperparameter Tuning

For the tuning, we start with the default parameters of [46] or the suggested parameters of [47] for Car Racing and Lunar Lander. We then tune the PPO and SAC baseline algorithms by varying the parameters of the initial learning rate and schedule, entropy coefficient and schedule, state-dependent exploration [48] vs. standard action sampling, and Gaussian vs. Beta distribution for the actor policy. For PPO, we additionally tune the clip range, the KL divergence target, the initial variance parameter of the Gaussian policy, and the number of epochs of optimization. As for SAC, we vary the training frequency, the number of gradient steps, how many samples to collect before starting the training, and the buffer size. Once satisfied with the baseline performance, we keep the same hyperparameters for our multiplicative versions and additionally tune the safety discount factor  $y_c$  and the initial value of the Lagrange multiplier  $\lambda_{\text{init}}$ . To best compare the effect of the multiplicative value function, we use the same initial Lagrange multiplier  $\lambda_{\text{init}}$  for our multiplicative versions and the Lagrange baseline.

### D. Detailed Experimental Description

To recapitulate, we assume the following reward structure of the environment

$$r(s_t, a_t) = \begin{cases} r_{\text{constraint}} & \text{if } s_t \in \mathcal{C} \\ r_{\text{constraint\_free}}(s_t, a_t) & \text{else} \end{cases}.$$

Here, we associate  $r_{\text{constraint}}$  with hard constraints which cause the episode to end in case of constraint violations. Additionally, there can be soft constraints encoded in  $r_{\text{constraint\_free}}$ . Violating a soft constraint causes a negative reward but the episode continues.

**Lunar Lander Safe** is a continuous control task where the agent has to land a rocket on the moon's surface [44]. Once the rocket lands, the episode ends. The agent receives

a reward for minimizing the distance to the landing pad and it can land anywhere as long as its down velocity is slow enough when touching the ground. The observation is

$$\vec{x}_{\text{observation}} = [\vec{d}, \vec{v}, \phi, \dot{\phi}, \mathbb{1}_{\text{contact.left}}, \mathbb{1}_{\text{contact.right}}],$$

where  $\vec{d}$  denotes the vector from current position to landing pad,  $\vec{v}$  is the linear velocity of the agent,  $\phi$  the roll angle and  $\mathbb{1}_{\text{contact}}$  denotes if the corresponding left or right leg has ground contact. We want to make the environment more safety-critical and go by the concept “the floor is lava”. This means, we keep the original constraint on the maximum allowed landing velocity but expand the constraint set  $\mathcal{C}$  such that landing outside the landing pad is not allowed anymore. The rewards are

$$\begin{aligned} r_{\text{constraint\_free}} = & -100 \cdot (\|\vec{d}_t\|_2 - \|\vec{d}_{t-1}\|_2) \\ & - 100 \cdot (\|\vec{v}_t\|_2 - \|\vec{v}_{t-1}\|_2) \\ & - 100 \cdot (\phi_t - \phi_{t-1}) \\ & + 10 \cdot (\mathbb{1}_{\text{contact.left,t}} - \mathbb{1}_{\text{contact.left,t-1}}) \\ & + 10 \cdot (\mathbb{1}_{\text{contact.right,t}} - \mathbb{1}_{\text{contact.right,t-1}}) \\ & - \text{fuel\_spend} \\ & + 100 \cdot \mathbb{1}_{\text{contact.right,t}} \mathbb{1}_{\text{contact.left,t}} \mathbb{1}_{\|\vec{v}_t\|_2 < 0.01}, \\ r_{\text{constraint}} = & -100, \end{aligned}$$

where the last line in  $r_{\text{constraint\_free}}$  denotes the state of a successful landing. Furthermore, we introduce a timeout if the agent cannot land within 1000 steps. Since no measure of time is present in the observation, the agent does not know about the timeout. The action space is two dimensional, representing the impulse the agent can apply to the left/right and up/down using “rocket engines”.

**Car Racing Safe** environment is based on CarRacing from OpenAI [44] where the agent is rewarded for driving around a race track. Each iteration, a new random track is spawned. The episode terminates if the agent has visited all track tiles or leaves the playground, which extends far beyond the track. Again, we want to make this environment more safety critic. For this, we tighten the hard constraint such that the agent enters the constraint set  $\mathcal{C}$  if it leaves the racetrack. Also, the agent has to drive faster than 0.1 km/h after an initial time period. Additionally, we impose a soft constraint that encourages the agent to drive below 50 km/h, which is encoded in the reward

$$\begin{aligned} r_{\text{constraint\_free}} = & \frac{0.3}{n_{\text{tiles}}} \cdot \mathbb{1}_{\text{new.track}} + \begin{cases} 0.005v & \text{if } v < 50 \\ -0.01v & \text{if } v \geq 50 \end{cases}, \\ r_{\text{constraint}} = & -10, \end{aligned}$$

where  $v$  denotes the longitudinal velocity. The observation space of the agent consists of an agent-centered bird's-eye-view image, longitudinal velocity  $v$ , yawing rate  $\psi$  and steering angle of the wheels  $\delta$ ,

$$\vec{x}_{\text{observation}} = [\text{img}, v, \psi, \delta].$$

The action space is continuous and two-dimensional. The actions are between accelerating/braking and steering to the

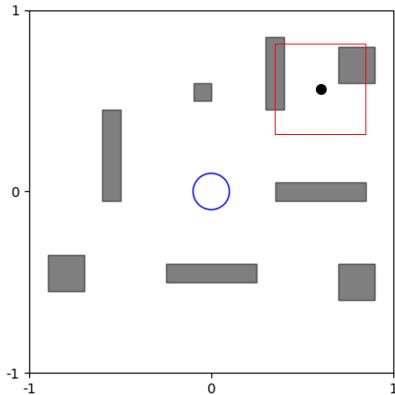


Fig. 6: Point Robot Navigation. The obstacles are shown in gray, the goal region in blue, the agent is a black dot, and the field of view of the local occupancy grid is marked by the red box. The environment has the boundaries  $[-1, 1]$  m.

left/right. We use the CNN structure of [1] to process the birds-eye-view image.

#### E. Point Robot Navigation

In the Point Robot Navigation environment, the agent has to navigate to a goal region of 0.1 m while avoiding obstacles and staying inside the environment boundaries. At each iteration, both the starting position of the robot and the set of obstacles are random. An example of the environment can be taken from Fig. 6. As the observation, the agent receives the vector  $\vec{d}$  from the current position to the goal and an agent-centered occupancy grid,

$$\vec{x}_{\text{observation}} = [\vec{d}, \text{oc\_grid}].$$

The action space is two dimensional, representing the percentage of a step size the agent can travel in x and y-direction. The reward is

$$r_{\text{constraint\_free}} = \begin{cases} 40 & \text{if } \|\vec{d}\|_2 < 0.1 \\ -0.1 & \text{else} \end{cases}, \quad (8)$$

$$r_{\text{constraint}} = -20.$$

This corresponds to a sparse reward setting. The step penalty of  $-0.1$  encourages the agent to reach the goal in a low number of steps. Furthermore, we choose the reward for reaching the goal higher than the penalty for constraint violation. With a lower goal reward, we have experienced baseline agents that try to crash immediately into obstacles to avoid the accumulation of step penalties. To process the occupancy grid, we use a small CNN encoder. When initially training SAC and PPO baselines, we experienced instability. This was related to the model not understanding the occupancy grid. To mitigate the issue, we added a decoder module to the CNN and posed an auxiliary loss on the reconstruction error. This improved the performance and stability of PPO. For SAC, we had to additionally use separate CNN encoders for actor, reward and safety critic.

**Gazebo Gym** is similar to the Point Robot Navigation meaning that it shares the same task and constraint set  $\mathcal{C}$

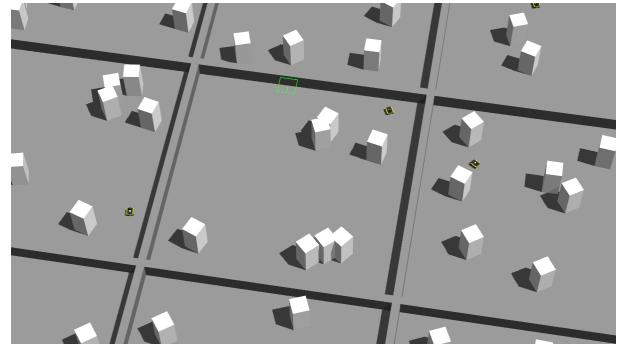


Fig. 7: Gazebo Gym. This is an 8x8 m world where the task is to navigate a cluttered environment. The picture shows the training of PPO.

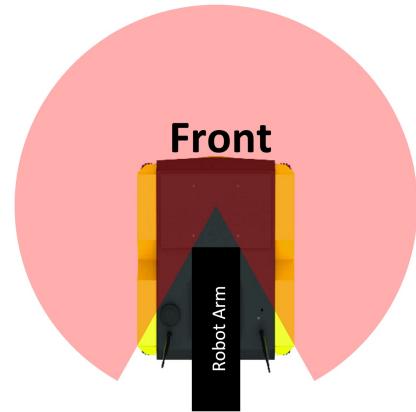


Fig. 8: Top-down view of the Jackal Robot. The laser scan is occluded due to the robot arm yielding a field of view of  $300^\circ$  as depicted with the red circle.

but resembles a realistic environment. The agent is a Jackal differential drive robot. Furthermore, the environment size is increased from 1x1 to 8x8 m. Similarly, the goal region is enlarged to 1 m in the easy and to 0.3 m in the hard setting of Gazebo Gym. The observation of the agent is given as

$$\vec{x}_{\text{observation}} = [\vec{d}, \sin(\psi), \cos(\psi), v_{\text{long}}, \dot{\psi}, \vec{d}_{\text{laser}}], \quad (9)$$

where  $\vec{d}$  denotes the vector from the current robot position to the goal,  $\psi$  the yawing angle of the robot and  $v_{\text{long}}$  the longitudinal velocity, assuming zero slip. The environment is encoded in  $\vec{d}_{\text{laser}}$  which is a 120-dimensional vector containing the 1D range measurements of each laser ray with small additive Gaussian noise. The laser scan has a field of view of  $300^\circ$  as depicted in Fig. 8. We first have been experiencing with the sparse reward setting in Eq. 8 in Point Robot Navigation. However, both SAC and PPO baseline performed poorly with success rates around 10% at 250k steps. To account for the more challenging dynamics of a differential drive robot, we implemented the dense reward

$$r_{\text{constraint\_free}} = \begin{cases} 40 & \text{if } \|\vec{d}\|_2 < 0.1 \\ -0.1\|\vec{d}\|_2 & \text{else} \end{cases}, \quad (10)$$

$$r_{\text{constraint}} = -20.$$

The term  $-0.1\|\vec{d}\|_2$  is a dense signal that directly connects the observation of the robot in Eq. 9 to the reward. The environment is implemented in Gazebo [45]. For PPO, we train 9 agents simultaneously, whereas for SAC, only one agent is used. The sampling time is 0.1s. At each iteration, the agent starts at a random position, and every forty iterations, a new set of obstacles is randomly spawned. In the hard mode, the agent is subject to Gaussian additive action noise. An example of the environment is shown in Fig. 7.

#### F. Additional Experimental Results

We provide the training curves of SAC in Fig. 9 and of PPO in Fig. 10. Compared to the evaluation results in Table 1 and 2 of the main paper, the algorithms violate the constraints more frequently in training. However, this is expected since the randomness of the training policy can cause the agent to reach potentially dangerous states. Furthermore, in Fig. 10, we provide the soft constraint violation rates in Car Racing Safe. Interestingly, the agent drives faster with the deterministic policy such that the soft constraint is violated more frequently in evaluation.

We furthermore present qualitative results for Lunar Lander Safe, Car Racing Safe and Gazebo Gym. In Fig. 11, we compare the landing of SAC Mult Clipped against SAC baseline on Lunar Lander Safe. In the first row, the flights of baseline and Mult Clipped agent look similar. However, in the third frame, we see that the downward velocity of the baseline lander is 1.7 m/s, which is much higher than 0.9 m/s of the multiplicative version. When continuing with the baseline plot, second row, first image, we see that the lander slows down by 0.2 m/s and uses its right leg to establish ground contact. This makes the agent decelerate from 1.5 to 0.3 m/s, shown in the next frame. In contrast, the multiplicative agent lands more conservatively. Starting from the second row, the next five frames show the agent decelerating. Shortly before landing, the agent has a velocity of 0.7 m/s and lands with both legs simultaneously. This is much safer from a real-world perspective: Slowing down from 0.7→0 m/s with two legs compared to 1.5→0.3 m/s with one leg like the baseline.

Next, we regard the first frame of Car Racing Safe in Fig. 12. Both the multiplicative and baseline agents steer to the right to open up the corner. When proceeding to the third frame, we can see that the baseline agent has opened up the corner more but also has a higher velocity of 49 km/h than the multiplicative agent. On the other side the multiplicative agent slows down to 36 km/h and cuts the inside. Continuing with frame four, we see that the velocity of the baseline agent is still high at 45 km/h. This limits the agent’s ability to steer to the left without losing traction. At this point, the safety critic estimates a crashing probability of 40% (note that we train a safety critic also in the baseline version for visualization purposes, but to not use it in any way for the policy training and stop all possible gradients). As we proceed, the agent cannot make the corner and leaves the track. We now regard the multiplicative policy. In the fourth frame, we can see that the velocity is low at 34 km/h.

This allows the agent to steer more aggressively to the left. In the next frame, the agent suggests further steering to the left but also starts to accelerate. This is a real-world racing technique where one starts to accelerate after the apex of a corner to increase traction.

Finally, we switch to the Gazebo Gym results shown in Fig. 13 and start with the best case, where both SAC baseline and Mult Clipped achieve a success rate of 100%. Furthermore, the trajectories of the multiplicative version seem smoother and more natural compared to the baseline. Another difference is that our approach strictly drives with the front forward, whereas the baseline sometimes drives back first. This is shown in the velocity plots, where a velocity smaller than zero means driving back-first. This is potentially dangerous since the agent has a 60° blindspot in the back as shown in Fig. 8. Note that the behavior of driving back or front first is an emerging behavior that was not incentivized by the reward.

In the second-worst case, the behavior of occasionally driving back first causes the baseline to crash, as can be seen in the bottom right corner of the trajectory plot. On the other hand, our approach drives head-first and achieves a success rate of 96%. For the 4% of trajectories in which the agent crashes, there is an anomaly in the reward critic: the estimated return is overly optimistic, especially close to obstacles. The worst case is obtained with seed two, where the combination of driving back first and anomalies in the value functions cause baseline and multiplicative agents to crash every second trajectory. However, since the second-worse case achieves significantly higher performance, we consider this second seed an outlier.

#### G. Additional Ablation Studies

Finally, we want to show the effects of the choice of initial Lagrange multiplier  $\lambda_{\text{init}}$  and safety discount factor  $\gamma_c$  which is depicted in Fig. 14. For SAC in Fig. 14a, the different choices of the initial Lagrange multiplier do not significantly affect the performance at convergence but can result in different sample efficiencies. However, Fig. 14b shows PPO is more sensitive where multipliers with magnitude five and higher cause training instabilities with rising timeout rates. Note that in the stable-baselines3 implementation [46] upon which we build our code, the advantage in PPO is normalized whereas the reward in SAC is not. This means that the different magnitudes of the Lagrange multiplier have a greater effect on PPO than on SAC in Point Robot Navigation.

Furthermore, we experiment with different safety discount factors  $\gamma_c$  shown in Fig. 14c and 14d. Both for SAC and PPO the safety discount factor seems to have little effect on the performance. This can be explained by the fact that the “dynamics” of the point robot allow for an instantaneous change of direction such that a short safety horizon with low  $\gamma_c$  is sufficient for obstacle avoidance.

---

**Algorithm 1** SAC Multiplicative

---

1: **Init:** policy parameters  $\theta$ ,  $\bar{Q}$ -function parameters  $\xi_1, \xi_2$ ,  
**Safety critic parameters**  $\psi_1, \psi_2$ , empty replay buffer  $\mathcal{D}$ .

2: Set target parameters equal to main parameters  $\xi_{\text{targ},i} \leftarrow \xi_i$ ,  $\psi_{\text{targ},i} \leftarrow \psi_i$ , for  $i \in \{1, 2\}$

3: **repeat**

4:   Observe state  $s$  and sample action  $a \sim \pi_\theta(\cdot|s)$ .

5:   Observe next state  $s'$  and done signal  $d$ .

6:   **Observe clipped reward  $\bar{r}$  and constraint cost  $r_c$ .**

7:   Store  $(s, a, \bar{r}, r_c, s', d)$  in replay buffer  $\mathcal{D}$ .

8:   Reset environment states **if**  $s'$  is terminal.

9:   **if** it's time to update **then**

10:     **for**  $j$  in range(however many updates) **do**

11:       Randomly sample a batch of transitions from  $\mathcal{D}$ ,  $B = \{(s, a, \bar{r}, r_c, s', d)\}$ .

12:       Compute targets  $y(\bar{r}, s', d)$  for  $\bar{Q}$ -functions:  

$$\bar{r} + \gamma(1-d)(\min_i \bar{Q}_{\xi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s'))$$
  
where  $\tilde{a}' \sim \pi_\theta(\cdot|s')$

13:       **Compute targets**  $y_c(r_c, s', d)$  for safety critic:  

$$r_c + \gamma_c(1-d)(\max_i \Psi_{\psi_{\text{targ},i}}(s', \tilde{a}'))$$
,  
where  $\tilde{a}' \sim \pi_\theta(\cdot|s')$

14:       Update  $\bar{Q}$ -functions for  $i \in \{1, 2\}$ :  

$$\nabla_{\xi_i} \frac{1}{|B|} \sum_{b \in B} (\bar{Q}_{\xi_i}(s, a) - y(\bar{r}, s', d))^2,$$
  
where  $b = (s, a, \bar{r}, r_c, s', d)$ .

15:       Update safety critic for  $i \in \{1, 2\}$ :  

$$\nabla_{\psi_i} \frac{1}{|B|} \sum_{b \in B} \text{BCE}(\Psi_{\psi_i}(s, a), y_c(r_c, s', d)),$$
  
where BCE denotes the binary cross-entropy.

16:       Update policy by one step of gradient ascent:  

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_{\text{mult}, \xi, \psi}(s, \tilde{a}_\theta) - \alpha \log \pi_\theta(\tilde{a}_\theta|s),$$
  
where  $\tilde{a}_\theta(s)$  is sampled from  $\pi_\theta(\cdot|s)$  via reparametrization trick.

17:       Update target networks:  

$$\xi_{\text{targ},i} \leftarrow \rho \cdot \xi_{\text{targ},i} + (1 - \rho) \xi_i,$$
  

$$\psi_{\text{targ},i} \leftarrow \rho_c \cdot \psi_{\text{targ},i} + (1 - \rho_c) \psi_i,$$
  
where  $i \in \{1, 2\}$ .

18:     **end for**

19:   **end if**

20: **until** convergence

---



---

**Algorithm 2** PPO Multiplicative

---

1: **init:** policy parameters  $\theta_0$ , value function parameters  $\xi_0$ ,  
**safety critic parameters**  $\psi_{1,0}, \psi_{2,0}$ , maximum unsafety probability target  $c_{\max}$ , and Lagrange multiplier  $\lambda$ .

2: **for**  $k = 0, 1, 2, \dots$  **do**

3:   Collect set of trajectories  $\mathcal{D}_k = \{\tau_i\}$  by running policy  $\pi_k = \pi(\theta_k)$  in the environment.

4:   Compute **clipped** rewards-to-go  $\hat{R}_t$  and **constraint cost-to-go**  $\hat{C}_t$ .

5:   Compute advantage estimates,  $\hat{A}_{\text{mult}}$  with current value function  $\bar{V}_{\xi_k}$  and **safety critic**  $\Psi_{\psi_k}$ .

6:   **Approximate**  $\hat{\Phi}^{\pi_\theta}(s) := \mathbb{E}_{a_\theta \sim \pi_\theta} [\Psi(s, a_\theta) - c_{\max}]$  the expectation in the PPO Mult objective by sampling from the policy

$$\hat{\Phi}^{\pi_\theta}(s) \approx \frac{1}{N} \sum_{i=0}^N \max_{j=1,2} \Psi_{\psi_j}(s, a_{\theta_i}) - c_{\max},$$

where  $a_{\theta_i} \sim \pi_\theta$ .

7:   Update the policy  $\theta$  by maximizing the PPO-Clip Mult objective:

$$\frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left( \frac{\pi_\theta(a_t|x_t)}{\pi_{\theta_k}(a_t|x_t)} A_{\text{mult}}^{\pi_{\theta_k}}(x, a), g(\epsilon, A_{\text{mult}}^{\pi_{\theta_k}}(x, a)) \right) - \lambda \hat{\Phi}^{\pi_\theta}(s_t),$$

typically via stochastic gradient ascent with Adam.

8:   Update the Lagrange Multiplier by

$$\lambda \leftarrow \max \left( 0, \lambda + \alpha \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \hat{P}_c^{\pi_\theta}(x_t) \right),$$

where  $\alpha$  is a learning rate.

9:   Fit value function using mean-squared error:

$$\xi_{k+1} = \arg \min_{\xi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T (\bar{V}_\xi(s_t) - \hat{R}_t)^2$$

10:   Update safety critic parameter  $\psi_{i,k+1}$  by minimizing the binary cross entropy:

$$\frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \text{BCE}(\Psi_{\psi_i}(x, a), \hat{C}_t),$$

where  $i \in \{1, 2\}$ .

11: **end for**

---

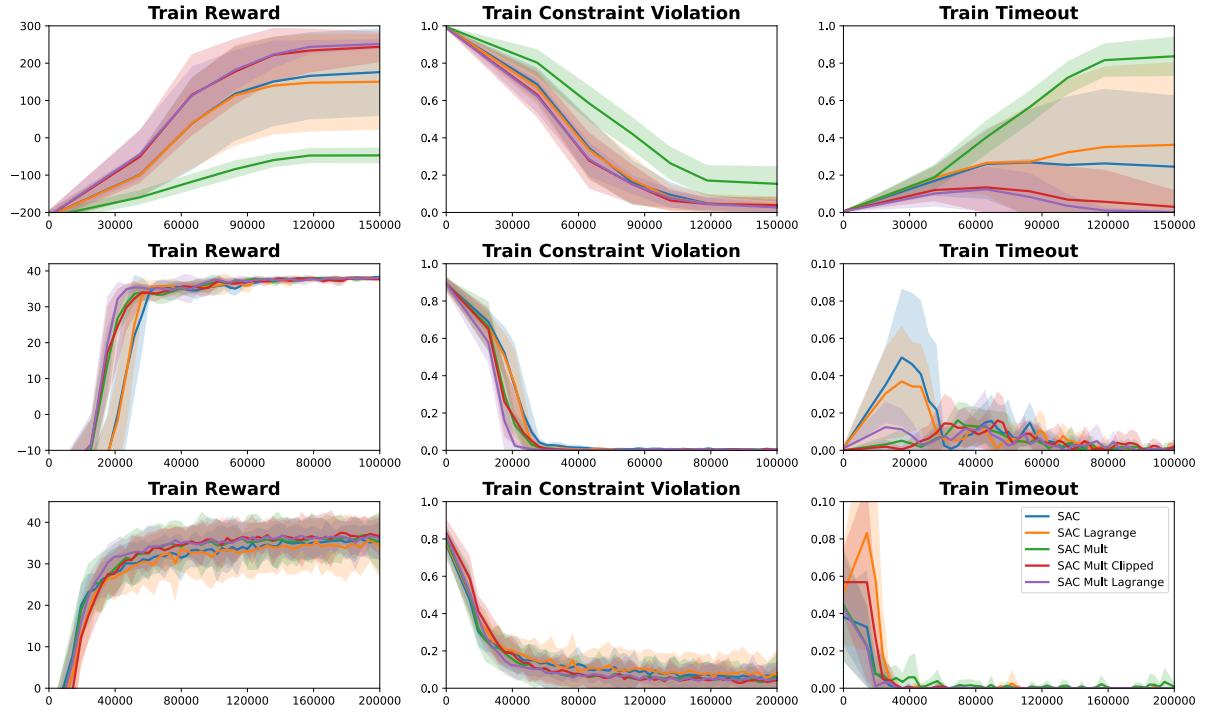


Fig. 9: SAC training curves. Top to bottom: Lunar Lander Safe, Point Robot Navigation, Gazebo Gym.

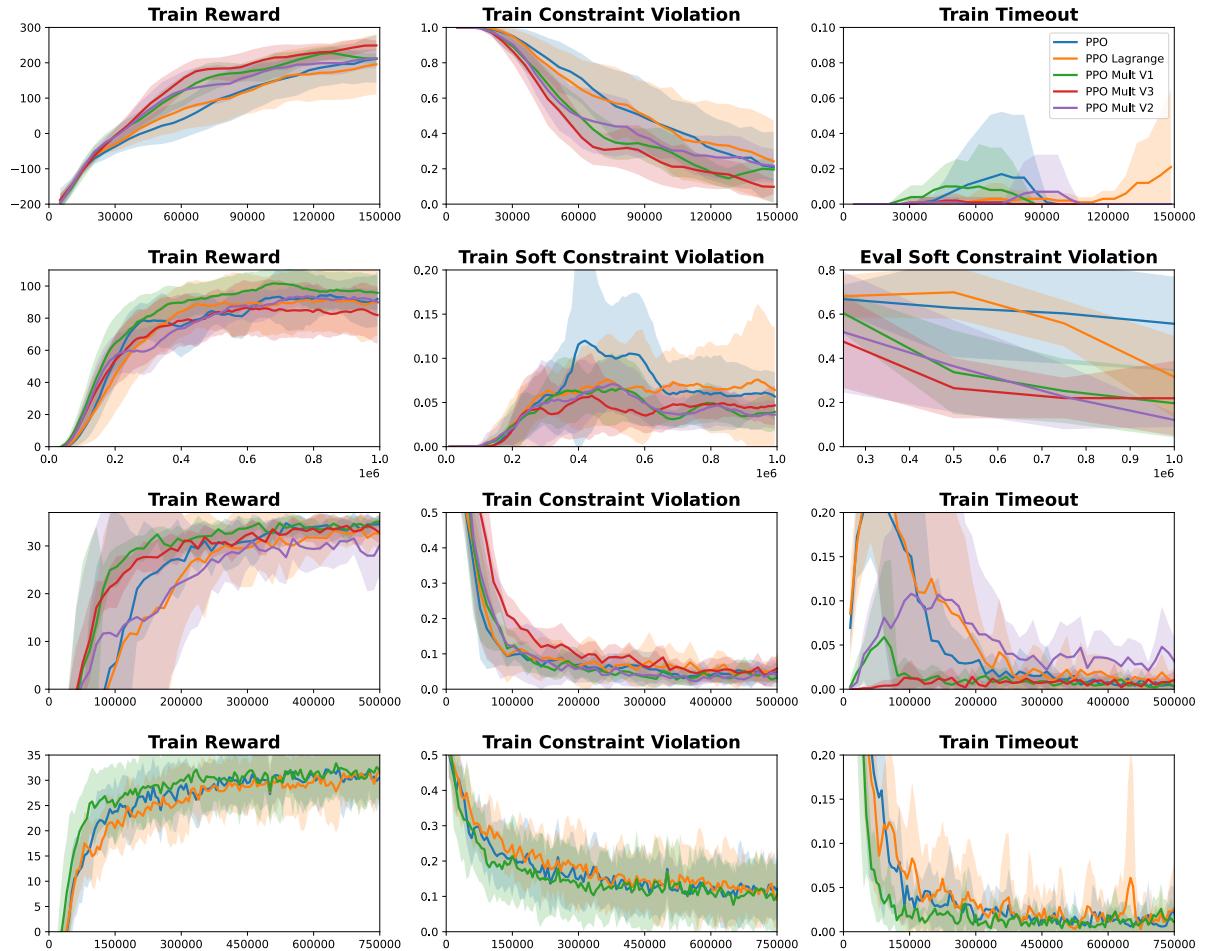


Fig. 10: PPO training curves. Top to bottom: Lunar Lander Safe, Car Racing Safe, Point Robot Navigation, Gazebo Gym.

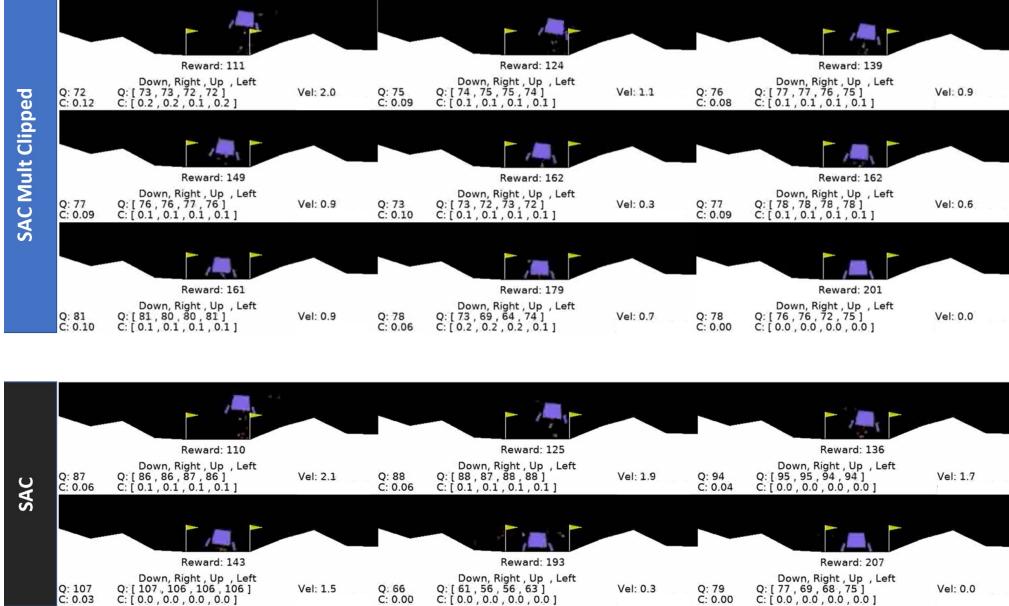


Fig. 11: Qualitative comparison of SAC vs SAC Mult Clipped on seed six after 300k steps on Lunar Lander Safe. We depict every fifth frame of a one-episode video evaluation. The images have to be read from left to right, top to bottom. In each frame, the first column with Q and C denotes the estimated value and constraint violation probability of the next suggested action. The second column shows the action values and the constraint violation probabilities for basic actions like going up, left, right or down. The last column depicts the current downwards velocity of the lander.

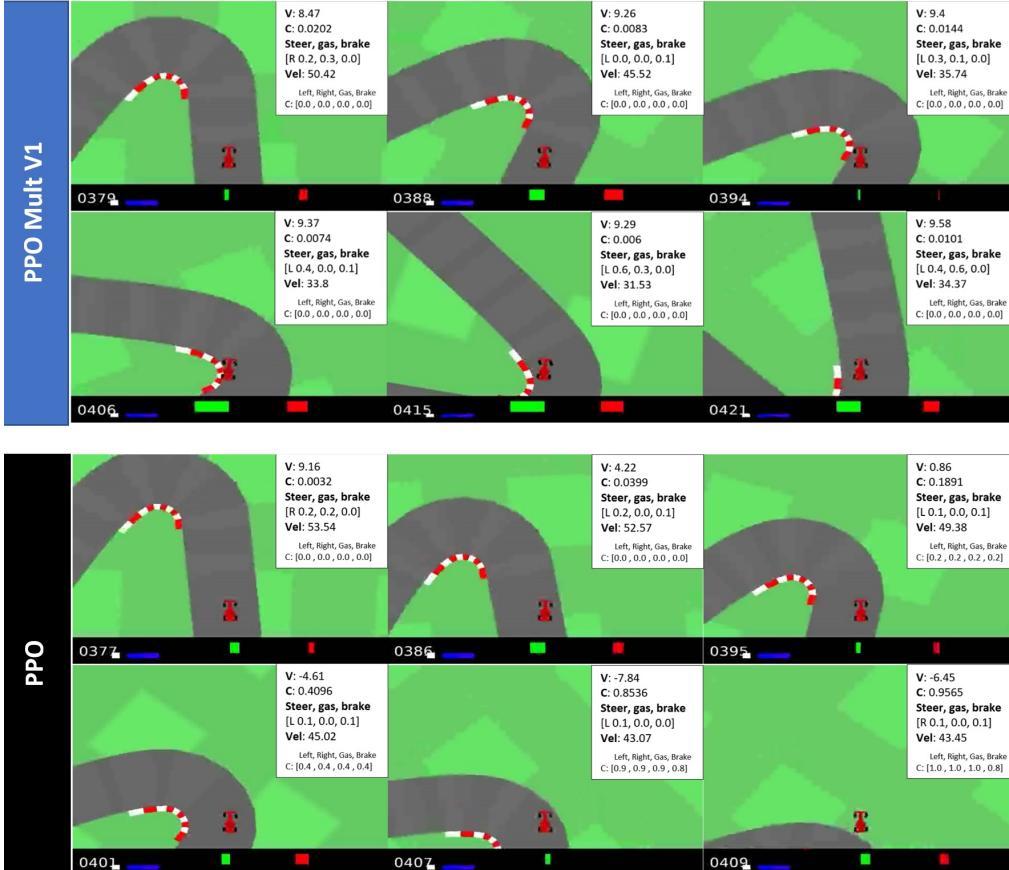


Fig. 12: Qualitative comparison of PPO base vs PPO Mult V1 on seed six after 1M steps. We depict every fifth frame of a one-episode video evaluation. The images have to be read from left to right, top to bottom. In each frame, the first two rows with V and C denote the estimated value and constraint violation probability at the current state with the next suggested action. The third and fourth row depict the next suggested action. The last row contains the constraint violation probabilities for basic actions like steering left, right, accelerating and braking.

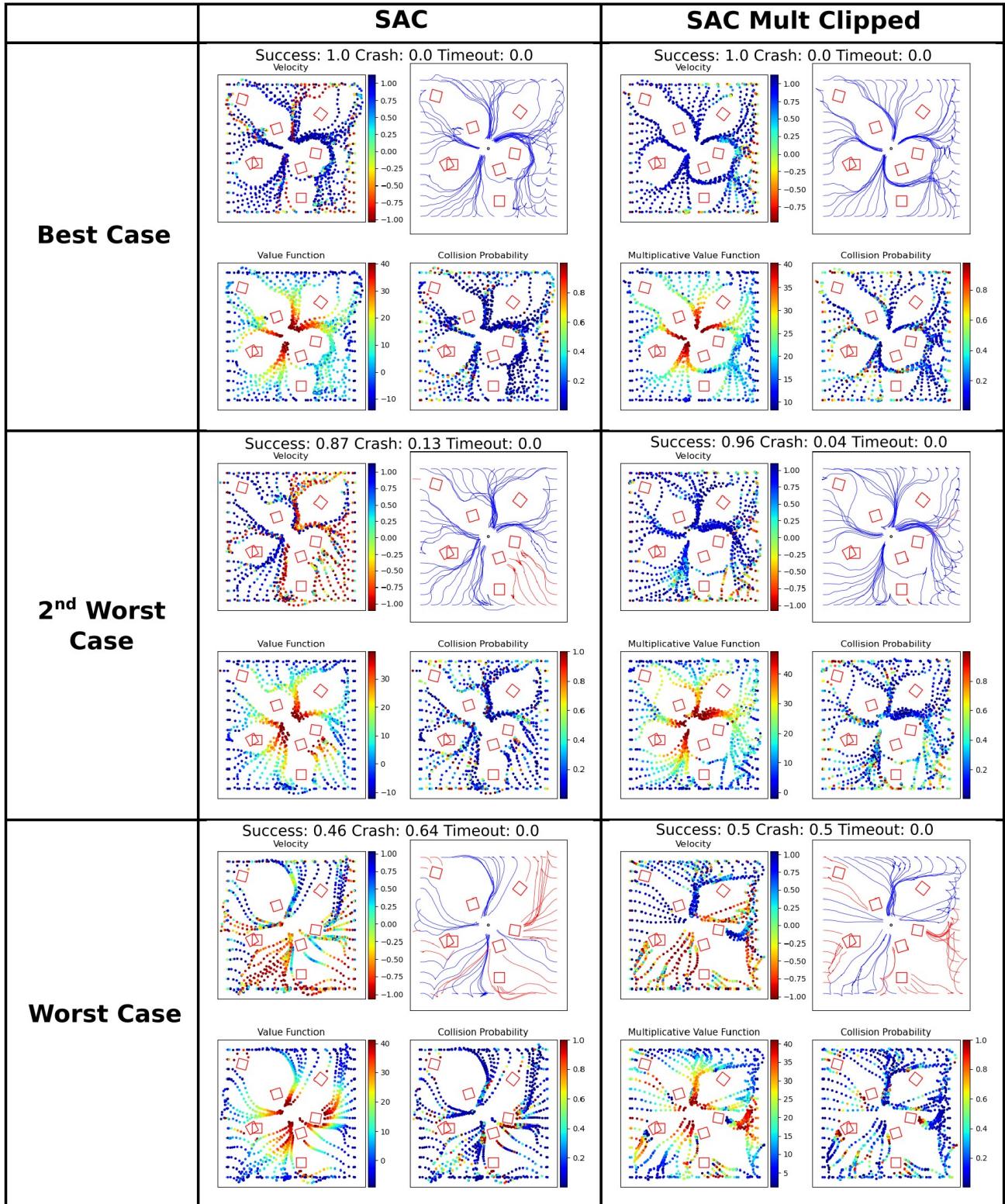


Fig. 13: Qualitative results on Gazebo Gym with SAC after 200k steps. The Velocity, Value Function, and Collision Probability plots show the corresponding metric at each third trajectory point. The best and worst case distinction are with respect to the best and worst success rates out of ten seeds.

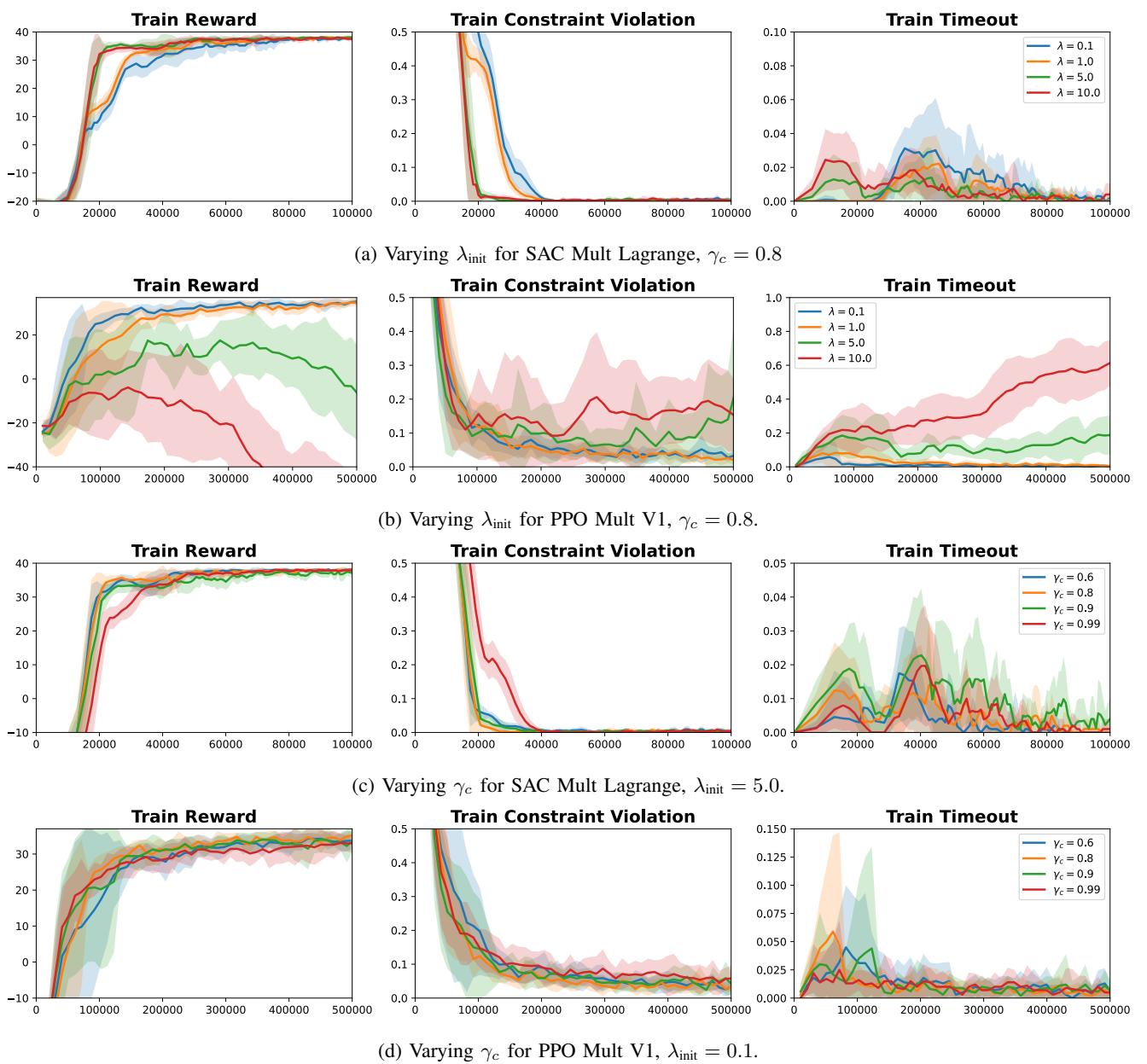


Fig. 14: Ablation Experiments in Point Robot Navigation