Safe Reinforcement Learning for Autonomous Navigation of a Driveable Vertical Mast Lift

Walter Brescia * **Antonio Maci** ** **Saverio Mascolo** *
Luca De Cicco *

* *Politecnico di Bari, Bari, Italy (e-mail: name.surname@poliba.it).*
** *Politecnico di Bari, Bari, Italy (e-mail: a.maci1@studenti.poliba.it).*

Abstract: In this work, we consider the issue of controlling a Driveable Vertical Mast Lift (DVML) to autonomously navigate an environment while ensuring required safety constraints. DMVL are industrial vehicles used in several applications, f.i. in logistics and smart agriculture, to allow operators placed in a basket accessing elevated worksites. When driving such machines, operators are exposed to hazards that could lead to potentially serious accidents. Reinforcement Learning (RL) is a data-driven approach that is increasingly being used to control complex systems. This work investigates the advancements in the field of Safe RL from a practical perspective, employing several state-of-art algorithms to equip a DVML with autonomous driving capabilities. We highlight how benchmark environments, while satisfactorily affirming Safe RL methodologies as proof-of-concepts, can widen the gap that prevents such methodologies from both being applied in real scenarios and becoming much more popular in industrial use-cases.

Keywords: Deep Reinforcement learning in control; Machine learning in modelling, prediction, control and automation; Smart Agriculture; Industry 4.0; Safe Reinforcement Learning; Safe Control of Driveable Vertical Mast Lift.

1. INTRODUCTION

The fourth industrial revolution (Industry 4.0) is changing the way products are manufactured, logistic is executed, and services are delivered to customers. The deep integration of *Cyber-Physical Systems* (CPS) in business operations is boosting innovation in this area, leading to significant improvements in efficiency and productivity. Autonomous mobile robots represent a key asset for the implementation of the Industry 4.0 paradigm. In fact, mobile robots are being increasingly adopted in a wide range of applications ranging from more classical ones, i.e., logistics, to more complex operations requiring robots to cooperate with both other robots and humans, typically in unstructured environments. In such domains, safety is a relevant issue to address: damage to the equipment may result in loss of profit, while damage to humans may cost lives. Such domains, also known as *Safety Critical* CPS, are associated with a number of *safety standards* that aim at preventing the occurrence of those accidents.

In order to make mobile robots autonomous, tools should be designed to allow perception of the environment with sufficient details to feed navigation control algorithms. To this purpose, robotic systems can benefit from Artificial Intelligence (AI) techniques that are showing very promising results in diverse engineering fields. In particular, *Reinforcement Learning* (RL) is an AI data-driven approach which is being increasingly adopted in industrial applications to address complex control tasks. The behavior of a RL control policy, i.e. how the agent acts based on the observations of the environment E in discrete time steps t, is similar to the operation of a controller in a classical feedback control system. Classical RL approaches typically assume that the task can be modelled as a *Markov Decision Process* (MDP), described by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, r)$, where $\mathcal{S}$ is the *Observation Space*; $\mathcal{A}$ is the *Action Space* $\mathcal{T}$ is the transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ that describes the probability of observing state s_t, taking action a_t and yielding a new state s_{t+1}, while r is the reward, defined at each timestep t as $r_t = R(s_t, a_t)$. In such configuration, the policy $\pi : \mathcal{S} \to P(\mathcal{A})$, where $P(\mathcal{A})$ is the probability distribution over the action space $\mathcal{A}$, must learn the actions that optimize the reward directly from its experience, which is collected through a trial and error process, namely *training*.

In order to ensure safety constraints, especially in scenarios regulated by safety standards and laws, a recent RL approach is proposed, namely *Safe Reinforcement Learning* (Safe RL). Safe RL embeds safety constraints into the RL problem formulation to ensure safety during both the phase of learning and deployment (see García et al. (2015)). Such techniques adopt several strategies for embedding safety in the whole process. A popular approach extends the reward function with signals related to the constraint violations, but cannot grant the strict compliance of the policy to the constraints, especially during the learning phase. Another popular approach leverages an extended version of the MDP framework that includes constraints in the formulation, namely *Constrained* MDP (CMDP). This formulation adds a new element $\mathcal{C}$ which represents the set of costs $\{c_i : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, i = 1, 2..., |\mathcal{C}|\}$ associated to $|\mathcal{C}|$ constraints. Such a framework ideally restricts the

optimization to the *safe space* that should respect the constraints. In practice, however, such a formulation tends to produce either poor policy performance or constraint violations.

Contribution: in this work we investigate the advancements in the field of Safe RL from a practical perspective, employing several state-of-art algorithms to equip an *Aerial Mobile Lifting Working Platform* (AMLWP) with autonomous driving capabilities, while safeguarding constraints. In particular, we compare the performance of such algorithms on a *Driveable Vertical Mast Lift* (DVML), a particular configuration of AMLWP. Such machines are particularly heavy, with weights in the range of several hundreds of kilograms. The DVML has been carefully modelled to faithfully represent an actually employed robot. We argue that this application allows to highlight critical aspects regarding the gap between the design of such algorithms and their deployment in real use case scenarios, where safety standards are involved. In fact, these algorithms are often tested in simpler environments that work as proof-of-concept and tend to neglect important aspects of real machines and scenarios. A DVML has been selected since it is suitable to represent a safety critical scenario. In fact, we not only consider the performance of the policy in reaching a desired goal position while avoiding collisions, but we also consider the intensity of vibrations during the operations (and possible tip-overs).

2. RELATED WORK

All RL algorithms considered in this work can be considered as variations of three popular RL algorithms: Deep Deterministic Policy Gradient (DDPG) (Lillicrap et al. (2015)), Proximal Policy Optimization (PPO, which can be seen as a natural extension of Trust Region Policy Optimization (TRPO) methods, Schulman et al. (2015)) (Schulman et al. (2017)) and Soft Actor-Critic (SAC) (Haarnoja et al. (2018)). Such algorithms share an Actor-Critic agent's structure, in which the critic's role is to approximate the loss function, while the actor interacts with the environment while optimizing the performance with respect to the task.

Dalal et al. (2018) adopt a Safety Layer in order to compute an action correction in a closed loop form. However, it is not enough to guarantee constraint satisfaction at training. Shao et al. (2020) compute a Forward Reachable Set by leveraging an uncertain dynamical model of the robot. If the action is considered to be unsafe, it is discarded in favour of a previously computed safe action. Thumm and Althoff (2022) employ an RL agent at a lower sampling rate in order to obtain intermediate goals. Safety is tackled through a *safety shield* which evaluates the trajectory and produces corrections to grant constraint satisfaction. The approach is proved on a manipulator application.

Achiam et al. (2017) propose a new RL algorithm based on *Trust Region methods* which approximates the constraints satisfaction (or violation) variation caused by each update, in order to guarantee a constraint satisfaction within a certain threshold, which is an hyperparameter, namely Constrained Policy Optimization (CPO). Ma et al. (2021) tackle the problem of Safe RL by estimating a *safety certificate* of the current state. A safety certificate is a measure of how safe the current state is: higher values are associated with states closer to a constraint violation. A safe control policy is trained to learn to keep the safety certificate to a low value, i.e. granting the constraint satisfaction. Zhao et al. (2021) propose an *Implicit Safety Set Algorithm* (ISSA) which employs an optimization algorithm, namely *Adaptive Momentum Boundary Approximation Algorithm* (AdamBA), in order to guarantee that each action satisfies the constraints. Liu et al. (2022) introduce a Constrained Variational Policy Optimization (CVPO) algorithm, which leverages a distribution of safe trajectories (approximated while learning) and train the agent in an Expectation-Maximization fashion to produce trajectories belonging to said distribution.

3. PROBLEM STATEMENT

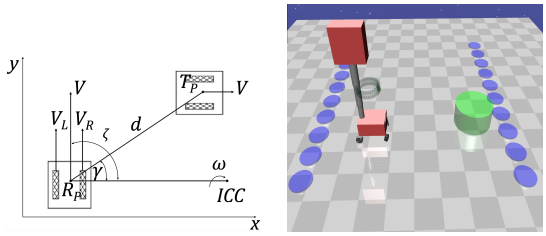
In order to simulate the real robot, an accurate model of a DVML is designed and implemented. The inertial properties of the model are tuned to reproduce the actual properties of a real DVML: the basket motion along the vertical axis is achieved with a number of *actuators* on the boom; the boom's vibrations are simulated through an ad-hoc *mass-spring-damper system* setup; the Differential Drive (DD) locomotion mechanism is implemented using two *PID actuators*, one for each driving wheel.

The simulation environment used in this work is *MuJoCo* [Todorov et al. (2012)], an accurate open source framework for physics simulations thoroughly evaluated in Erez et al. (2015). In particular: a custom actuator has been implemented to reproduce the PID controllers; the boom has been implemented with three components, the first attached to the base and the last one to the basket; the boom components are linked with each other through sliding joints, which allow the movement (upon actuation) along the z-axis. In particular, for each sliding joint we tuned the *dumping* and *stiffness* values to obtain the desired vibration settling time. In addition, a null *springref* value is considered. It is important to notice that the actuation of the boom during navigation is forbidden by the regulations of such machine. For this reason, the height of the basket becomes a fixed parameter for each test and is, therefore, kept out from the set of control inputs. Finally, a LiDAR sensor has been employed onboard of the robot for the obstacle detection.

3.1 Autonomous Navigation of a DVML

As mentioned before, DVML typically employ a DD mechanism which consists of: 1) two independently motorized wheels, mounted on a common axis; 2) two castor wheels, used to ensure the vehicle static stability. By applying different velocities on the wheels, the robot can follow trajectories coherently with its kinematics constraints.

The autonomous navigation problem regards the pursuit of a set of target coordinates $T_P = (x_T, y_T)$ from the current robot's position $R_P = (x_R, y_R)$. In other words, the *controller* must provide a set of control inputs to drive the robot from R_P to T_P. A further requirement is represented by the vibrations of the boom: the controller of a DVML must not only bring the robot to the desired position, but must prevent vibrations (or at least reduce their intensity to a negligible value) throughout the entirety of trajectory.



(a) (b)

Fig. 1. (a) Model notation; (b) Test environment and robot model

Figure 1(a) shows the typical scenario and the involved variables in the problem of controlling the position of a DD mobile robot, where d represents the linear distance between the current position and the target one.

This work focuses on equipping a DVML with autonomous driving capabilities. In particular, we analyze the performance of the controller in logistic and industrial environments, typically composed of two main types of working surroundings: large working areas (which will be represented with areas without walls for simplicity) in which shelves, boxes and humans can operate, and hallways and corridors, with much less obstacles in the way. The environment in which the robot will move is implemented in *OpenAI SafetyGym* (SG) [Alex Ray (2019)], an environment builder which leverages MuJoCo as physics simulator and is compliant to *OpenAI Gym*, one of the most popular frameworks for building RL environments. SG has been chosen due to its flexibility in the creation of new environments, which can be composed of a wide range of physics items, targets, and – more importantly – safety constraints. Moreover, a relevant feature of SG is the possibility of randomizing the environment's layout at the beginning of each episode. This allows to prevent RL agents from learning trivial solutions to a particular setting of the environment and, instead, pushes the agent to learn more general behaviors with respect to the given task.

In each simulation, a green cylinder (see Figure 1(b)) at a fixed position will represent the goal area. Obstacles are represented by *hazards* (blue circles in Figure 1(b)). Driving through obstacles will violate constraints.

3.2 Environment Setting

In order to allow a fair comparison among all RL algorithms, the observation space, the action space, the reward function and the cost function are fixed for all experiments. The **Observation Space** is defined as the concatenation of: the linear distance from the goal, defined as the exponential of the negative distance $d = e^{-||T_P - R_P||}$; the LiDAR information that only detects obstacles in the scene (the information is represented as thirty two distances along thirty two different angles scanning the area around the robot); the position of the goal, arranged as a LiDAR reading. The **Action Space** is composed of the velocities of each wheel, each in the range $[-1, 1]$. The **Reward Function** is defined as the sum of two main components: a utility function, represented by the instantaneous variation of the distance Δd from the goal, and a cost function. The cost function is composed of two terms: a penalization for any vibration of the boom ψ, computed as $\psi = ||P_{base} - P_{basket}||$ with P_{base} and P_{basket} being respectively the base position and the basket position; a constraint violation term η, proportional to the distance of the robot from the center of the hazard. The latter is added only when the robot is trespassing the hazard (i.e., when the robot-hazard distance is less than the hazard's size). Finally, a bonus β is added only when the linear distance is below a given threshold (tolerance) ρ: $||T_P - R_P|| < \rho$. Note that both β and ρ are hyperparameters to be tuned. The resulting reward function is reported below:

$$r_t = \begin{cases} \Delta d - \psi - \eta, & \text{if } d \geq \rho \\ \Delta d - \psi - \eta + \beta, & \text{if } d < \rho \end{cases} \tag{1}$$

where

$$\eta = \begin{cases} 0, & \text{if } ||H_P - R_P|| \geq H_s \\ ||H_P - R_P||, & \text{if } ||H_P - R_P|| \leq H_s \end{cases} \tag{2}$$

and H_P being the hazard's position and H_s the hazard's size.

4. SRL FOR DVML AUTONOMOUS NAVIGATION

In this section are highlighted the main aspects of the implemented Safe RL algorithms before proceeding to a comparison. In particular, we focus on the following algorithms:

Lagrangian Methods Lagrangian methods expand the optimization problem, with $f(\theta)$ being the objective, in constrained contexts by adding a penalty coefficient $\lambda g(\theta)$, with λ being the Lagrangian multiplier. The optimization problem to solve becomes max-min optimization problem: $\max_\theta \min_{\lambda \geq 0} \mathbb{L}(\theta, \lambda) \doteq f(\theta) - \lambda g(\theta)$. Alex Ray (2019) implement a Lagrangian version of PPO and TRPO algorithms. Another implementation is proposed by Stooke et al. (2020). In particular, they take a control perspective on the Lagrangian methods and leverage the constraint function derivatives to implement a PID controller. Therefore, the Lagrangian multiplier variation can be written as: $\dot{\lambda} = \alpha g(\theta) + \beta \dot{g}(\theta) + \gamma \ddot{g}(\theta)$, with each term being composed by a coefficient and, respectively, the integral, the proportional and the derivative of the constraint function.

Constrained Policy Optimization (CPO) [Achiam et al. (2017)] is a trust region method for Safe RL which approximately enforces the constraints in every policy update. The authors show that the proposed methodology can train neural network policies with a large number of parameters in computationally complex constrained control tasks. This is possible by solving the CRL optimization problem:

$$\begin{aligned} \theta_{k+1} &= \arg\max_\theta g^T(\theta - \theta_k) \\ \text{s.t.} \quad & c_i + b_i^T(\theta - \theta_k) \leq 0 \quad i = 0, \ldots, m \\ & \frac{1}{2}(\theta - \theta_k)^T H(\theta - \theta_k) \leq \delta \end{aligned}$$

using its dual formulation. Note that g represents the gradient of the objective function, c_i and b_i are respectively the i-th constraint and the gradient of the i-th constraint, H is the Hessian of the KL-divergence $D_{KL}(\pi, \pi_k)$. The authors augment the optimization problem with a term that bounds to a step-size δ the weights updates, in order to limit the error due to linearization around π_k. When

the optimization step leads to an unfeasible optimization problem, a backtracking algorithm which updates the policy just to decrease the constraint value is employed.

Implicit Safe Set Algorithm (ISSA) (Zhao et al. (2021)) exploits the Safe Set Algorithms (SSA, Liu and Tomizuka (2014)) in order to guarantee the safety of the system when a given learning algorithm is employed. The algorithm employs a rule to synthesize proper safety indexes that always grant the existence of a safe control input over the whole set of states. When employing the policy, they use a black box approach to project the nominal action a_t^r on the set of safe control inputs $\mathcal{U}_s(x)$, by solving the following optimization problem:

$$\min_{a_t \in \mathcal{A}} ||a_t - a_t^r||^2$$
$$\text{s.t.} \quad \phi(f(s_t, a_t)) \leq \max\{\phi(s_t) - \eta, 0\}$$

where ϕ is the safety index function. In order to solve such problem, authors propose an algorithm called *Adaptive Momentum Boundary Approximation Algorithm* (AdamBA). Starting from the nominal control action, AdamBa performs a linear search in order to find the boundaries of the safe set. Unit gradient vectors that sample the control input set are exponentially increased until the boundaries are reached. The gradient vectors that do not reach such boundaries are discarded. An exponential decay is, then, performed to find the boundary points of the safe control space. The final output action is chosen with respect to the minimum deviation from the nominal action. Note that authors don't make any assumptions on such method and are, therefore, able to leverage such technique on a wide variety of RL algorithms.

Constrained Variational Policy Optimization (CVPO) (Liu et al. (2022)) leverages an Expectation-Maximization approach to include safety during training. In particular, they introduce an optimality variable $\mathcal{O}$ to represent the event of a trajectory τ of maximizing the expected reward. Denoting with $p_\pi(\tau)$ the probability of following the trajectory τ under the policy π, then, it is shown that the log-likelihood of optimality under policy π, defined as $\log p_\pi(\mathcal{O} = 1)$, is bound:

$$\log p_\pi(\mathcal{O} = 1) \geq$$
$$\mathbb{E}_{\tau\ q}[\sum_{t=0}^{\infty} \gamma^t r_t] - \alpha D_{KL}(q(\tau)||p_\pi(\tau)) = \mathcal{J}(q, \pi)$$

with $\mathbb{E}_{\tau\ q}[\sum_{t=0}^{\infty} \gamma^t r_t]$ being the expected infinite discounted reward, $D_{KL}(q(\tau)||p_\pi(\tau))$ being the Kullback-Leibler distance between an auxiliary trajectory distribution $q(\tau)$ and α being a temperature parameter. To guarantee the satisfaction of the constraints, $q(\tau)$ is chosen from the feasible distribution set:

$$\Pi_{\mathcal{Q}}^{\epsilon_1} := \{q(a \mid s) : \mathbb{E}_{\tau \sim q}[\sum_{t=0}^{\infty} \gamma^t c_t] < \epsilon_1, a \in \mathcal{A}, s \in \mathcal{S}\}$$

with c_t being the cost and ϵ_1 being a threshold. q is, then, updated during training in order to maximize $\mathcal{J}(q, \pi)$ during the expectation step. Then, during the Maximization step, the policy parameters θ are updated with respect to the objective:

$$\bar{\mathcal{J}}(\theta) = \mathbb{E}_{\rho_q}\left[\alpha \mathbb{E}_{q_i^*(\cdot|s)}\left[\log \pi_\theta(a \mid s)\right]\right] + \log p(\theta) \qquad (3)$$

where ρ_q is the stationary state distribution induced by q. This algorithm can be applied in an off-policy setting by approximating ρ_q through samples collected in the replay buffer. Authors show good performance with an improved sample efficiency with respect to other state of art algorithms.

5. RESULTS

In order to obtain a good representation of the state of art performance we consider the following algorithms:

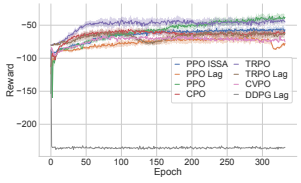
(1) TRPO, and its Lagrangian version, called TRPO Lag;
(2) PPO, and its Lagrangian version, called PPO Lag;
(3) PPO with ISSA, called PPO ISSA;
(4) Lagrangian version of DDPG, called DDPG Lag;
(5) CPO;
(6) CVPO.

Each algorithm has been trained on an environment in which, at each episode, three obstacles and the goal are positioned randomly. We limit the number of obstacles in order to study the generalization capabilities during tests. For each algorithm, several configurations have been tested. For brevity, only the best ones are shown. During training, the simulation is not interrupted neither if a violation happens, nor if the agent reaches the goal. In the latter case, a new goal is generated.

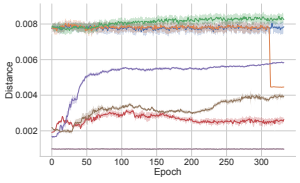
In order to carry out a fair comparison, all algorithms are trained for the same number of epochs and employ the same neural network structure (two hidden layers, 256 neurons each). The number of epochs is fixed to 333 for all algorithms, however, the number of interaction steps is set to 10^7 for all algorithms but the CVPO and DDPG Lag, whose number of interaction steps is limited to $333 \cdot 10^3$. This difference is due the different implementations adopted. In fact, each algorithm implementation is optimized with respect to such numbers of interaction steps and tests with different values have lead to worse performance. For each algorithm, multiple hyper-parameters configurations have been tested. For brevity, only the configurations associated with best performance are retained. To test generalization capabilities of the agents, tests will be carried out in two different settings: 1) a first environment with twenty hazards to simulate a hallway, typical of industrial/logistics scenarios 1(b); 2) a second setting adopts different DVML physical parameters, in particular higher basket's weight and extended boom.

5.1 Overall results

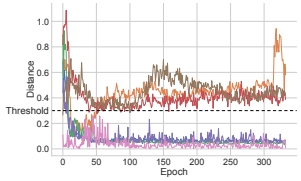
In Figure 2, the overall performance during training are reported. In particular, observing Figure 2(a), all agents show a good progression in terms of reward function. In fact, it would seem that PPO and TRPO are the best performing algorithms. Given the particular application, a key aspect to be taken into account is the intensity of oscillations. During training all algorithms produce very distinct behaviours (see Figure 2(b)): while PPO and PPO ISSA seem to ignore such signal (they produced oscillations in the same range even after lots of updates), TRPO and TRPO Lagrangian worsen their performance as training goes on. On the other hand, CPO and PPO Lagrangian show an increase in vibrations' intensity during the first few epochs of training, but stabilize to much lower values towards the end. This is the behaviour to be expected: initial movements will be



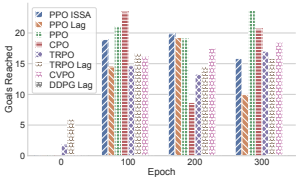
(a) Learning Curves



(b) Oscillations Cost (Basket to base distance)



(c) Average Min Distance from Obstacles



(d) Goals Reached

Fig. 2. Overall Training Performance

ineffective, producing neglectable oscillations; as the agent learns to drive the robot oscillations will increase due to suboptimal accelerations; later on, the agent will be able to optimize its driving abilities with respect to oscillations. A peculiar trend (or absence of) is shown by CVPO, which seems to produce very low values of vibrations throughout the entirety of its training.

In Figure 2(c) and Figure 2(d) are presented respectively the minimum distance from obstacles and the number of goals reached as the policy is updated. In particular, observing Figure 2(d), almost all algorithms seem to learn to reach the goal (note that DDPG Lag never reached the goal and therefore its value is zero), however, this is not enough: as shown in Figure 2(c), only CPO, PPO Lag and TRPO Lag are able to satisfy the constraints. In particular, CPO is the algorithm that manages to optimize both constraint satisfaction and navigation to the goal. Note that, while the minimum distance from obstacles' signal numerically dominates the oscillation cost one, the former is only applied when the constraints are violated, without clouding the oscillation signal in the remaining time. An overall analysis suggests that CPO represents the best solution, since it is able to drive to the objective while minimizing the oscillations and satisfying constraints, at least in the final stages of training. In fact, not even CPO is able to guarantee constraint satisfaction throughout the training.

5.2 Detailed analysis

Figure 3 aims at providing an analysis of the severity of the constraint violations that occurred while testing each trained policy in an hallway-like environment. In particular, the number of constraint violations and the average velocity are compared: the best algorithm should be able to avoid constraint violations. If violations happen, it is preferable to have very low velocities. Therefore, the best algorithms will be found in the bottom left corner of such figure, whereas the top right area corresponds to algorithms with the poorest performance. By observing this figure, the best performing algorithm is CPO. In particular, it manages to satisfy the constraints in most cases and,

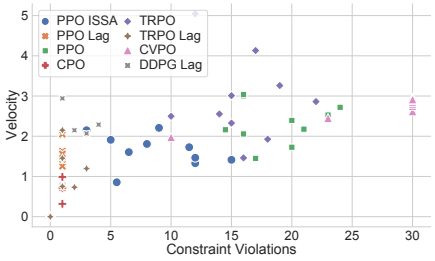
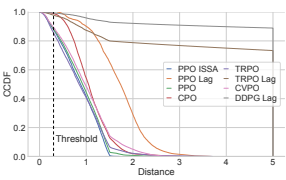
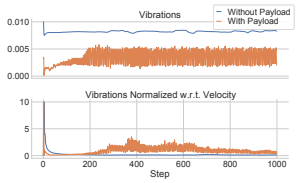


Fig. 3. Average velocity when constraint violations happen, during Tests



(a) CCDF of Minimum Distance from Obstacles during Tests



(b) Vibration evolution of CPO agent

Fig. 4. Tests Evaluation

when violations occur, it is driving the vehicle at a very low speed. Concerning TRPO Lagrangian and PPO Lagrangian, while they are able to reach the goal a sufficient number of times, they also lead the robot towards numerous constraint violations, some of them at high speeds, too. Note that Figure 3 is limited to 30 violations for improved visibility, however, without such limitation, one could observe that CVPO reaches hundreds of violations during test at roughly $3\frac{m}{s}$, which is unacceptable.

To further analyze constraint satisfaction, in Figure 4(a) is reported the cumulative distribution function of the minimum distance of the robot from the obstacles during tests. It is also reported the threshold under which a constraint violation is detected. This figure mainly shows two aspects: 1) none of the agents are able to always satisfy the safety constraints; 2) CPO, PPO Lag, TRPO and TRPO Lag show the best performance. However only CPO and PPO Lag stay actually close to the goal: the remaining agents drive, in the majority of cases, the robot very far from the obstacles (and, hence, also from the goal) exhibiting a very dangerous and unacceptable behaviour.

In Figure 4(b) is considered the CPO agent, which is the one associated with best performance. In order to further investigate the generalization capabilities of the agent with respect to the physical system parameters, we test the policies by increasing the height of the boom of 3 meters and adding 100kg to the basket's payload, in order to simulate the weight of a human operator and other tools. Figure 4(b) shows that the vibrations' intensity with and without payload are comparable with each other (and in both cases neglectable). Since trajectories produced in different conditions will differ, we also evaluate in Figure 3 the vibrations normalized with respect to the current velocity. In fact, Since the intensity is comparable but a higher payload will make the robot move slower, the normalized vibrations will have a higher value. Nonetheless, values are still comparable with each other, highlighting

how such model is able to generalize its driving skills to robots with different physical parameters.

5.3 Discussion

The results show that the best performance are obtained, as expected, by policies trained with Safe RL algorithms. In fact, even if some of the RL algorithms manage to reach several times the goal position, they are unable to satisfy the constraints. The best performing algorithm has been tested on different settings of the physical system and has shown a good robustness to the variation. However, not even the best algorithm is able to guarantee the constraint satisfaction during training. The best performance trade-off between controlling the robot's position and the satisfying the safety constraints, in terms of both vibrations minimization and of constraint satisfaction, are obtained by the CPO algorithm. The experiments carried out clearly exhibit a gap between Safe RL methodologies proposed by the state of art and their employment on real industrial vehicles. In fact, while in proof-of-concept environments such algorithms show good performance, on real industrial use cases, such as the one proposed in this work, none of the considered methodologies are capable of guaranteeing constraints satisfaction neither throughout training nor during tests. Moreover, such algorithms can exhibit an unacceptable dynamic behavior, i.e. can impact hazards at high speeds. In order to allow Safe RL to actually be employed on real use cases, there is the need to prove such methodologies on real scenarios that can highlight criticalities.

6. CONCLUSION

The rapid growth of Industry 4.0 is leading towards industrial, logistic, and smart agriculture scenarios in which CPS are employed to carry out complex tasks in changing environments in which humans can navigate too. In these scenarios, safety not only is a concern, but a hard constraint which must be satisfied at all time. This work considered the task of equipping a DVML with autonomous driving capabilities with a Safe RL approach. We adopt the main state of art Safe RL algorithms and train such agents on a real industrial scenario in which DMVLs are required to navigate in a complex environment. The best performing algorithm is tested on very different physical parameters with respect to the values observed during training and shows good generalization capabilities, yet it is still not able to satisfy the safety constraints in every case. While Safe RL brings the great advantage of synthesizing a control policy that allows neglecting the underlying model and requires hand-made hyperparameters fine-tuning, validating these methodologies in simple scenarios can lead to approaches which hardly generalize to complex scenarios, hence creating a gap with research and industrial applications, keeping such methodologies from being actually employed.

ACKNOWLEDGEMENTS

This work has been partially supported through the MOBIROB project funded by Quavlive s.r.l..

REFERENCES

Achiam, J., Held, D., Tamar, A., and Abbeel, P. (2017). Constrained policy optimization. In *Proceedings of the 34th ICML*, volume 70, 22–31. PMLR.

Alex Ray, Joshua Achiam, D.A. (2019). Benchmarking safe exploration in deep reinforcement learning.

Dalal, G., Dvijotham, K., Vecerik, M., Hester, T., Paduraru, C., and Tassa, Y. (2018). Safe exploration in continuous action spaces. doi:10.48550/ARXIV.1801.08757.

Erez, T., Tassa, Y., and Todorov, E. (2015). Simulation tools for model-based robotics: Comparison of bullet, havok, mujoco, ode and physx. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 4397–4404. doi:10.1109/ICRA.2015.7139807.

García, J., Fern, and o Fernández (2015). A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42), 1437–1480.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In J. Dy and A. Krause (eds.), *Proceedings of the 35th ICML*, volume 80 of *Proceedings of Machine Learning Research*, 1861–1870. PMLR.

Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2015). Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*.

Liu, C. and Tomizuka, M. (2014). Control in a Safe Set: Addressing Safety in Human-Robot Interactions. Dynamic Systems and Control Conference.

Liu, Z., Cen, Z., Isenbaev, V., Liu, W., Wu, Z.S., Li, B., and Zhao, D. (2022). Constrained variational policy optimization for safe reinforcement learning. doi:10.48550/ARXIV.2201.11927.

Ma, H., Liu, C., Li, S.E., Zheng, S., and Chen, J. (2021). Joint synthesis of safety certificate and safe control policy using constrained reinforcement learning. *arXiv preprint arXiv:2111.07695*.

Schulman, J., Levine, S., Moritz, P., Jordan, M.I., and Abbeel, P. (2015). Trust region policy optimization. doi:10.48550/ARXIV.1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Shao, Y.S., Chen, C., Kousik, S., and Vasudevan, R. (2020). Reachability-based trajectory safeguard (rts): A safe and fast reinforcement learning safety layer for continuous control. doi:10.48550/ARXIV.2011.08421.

Stooke, A., Achiam, J., and Abbeel, P. (2020). Responsive safety in reinforcement learning by pid lagrangian methods. doi:10.48550/ARXIV.2007.03964.

Thumm, J. and Althoff, M. (2022). Provably safe deep reinforcement learning for robotic manipulation in human environments. doi:10.48550/ARXIV.2205.06311.

Todorov, E., Erez, T., and Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 5026–5033. doi:10.1109/IROS.2012.6386109.

Zhao, W., He, T., and Liu, C. (2021). Model-free safe control for zero-violation reinforcement learning. In *5th Annual Conference on Robot Learning*.