Autonomous Flipper Control with Safety Constraints

Martin Pecka[1,2], Vojtěch Šalanský[2], Karel Zimmermann[2], and Tomáš Svoboda[1,2]

***Abstract*— Policy Gradient methods require many real-world trials. Some of the trials may endanger the robot system and cause its rapid wear. Therefore, a safe or at least gentle-to-wear exploration is a desired property. We incorporate bounds on the probability of unwanted trials into the recent Contextual Relative Entropy Policy Search method. The proposed algorithm is evaluated on the task of autonomous flipper control for a real Search and Rescue rover platform.**

I. INTRODUCTION

The task of Reinforcement Learning (RL) [1] is to search through the space of policies $\pi : S \to A$, which map agent states S to possible actions A; the actions are then applied either in reality or using a transition model, and the agent reaches a new state (this description is usually known as Markov Decision Process, MDP). RL expects that the agent is rewarded for being in state s with a reward $R(s) \in \mathbb{R}$, and searches for a policy that maximizes the expected reward over all trajectories the agent might execute. It is different from supervised learning, which maximizes the immediate reward, and not the long-term one.

Policy Gradient methods are used when the policy $\pi(s)$ has a parametric form and can be written as $\pi(s|\omega)$, where $\omega)$ is a parameter vector. They stochastically optimize the expected sum of rewards by direct sampling in the policy parameter space. Thus, the learning performance of PG methods does not depend on the complexity of controlled systems, only on the number of policy parameters being optimized [2]. Such property makes them suitable for learning of controllers for robotic systems for which robust real behavior prediction using the first-principle models is difficult (such as closed form equations describing kinematic and/or dynamic behaviors for rover-terrain interaction, or Navier-Stokes aerodynamic laws). Unfortunately, PGs usually require many trials which endanger the real system or cause its excessive wear. Therefore, they are usually not used directly on the real system, but on data-driven models. For example, Kupcsik et al. [3] demonstrate data-driven PG learning of the ball throwing problem with a robotic arm, and Tedrake et al. [2] argues that Policy Gradient learning for aerial maneuvers with an ornithopter may be very efficient in fact. Transeth et al. [4] show that for snake-like robots with significant side-slip, no closed form expression of the snake's motion exists, therefore policy learning must resort to simulation.

Contextual REPS uses a stochastic upper-level policy which generates deterministic lower-level policy samples. The performance of these policies is evaluated by executing them in the real world, and is used to estimate the upper-level policy gradient. The Gaussian Process REPS (GPREPS) method [3] adds a Gaussian Process (GP) in the loop, which learns a representation of the system dynamics. The GP is used for better evaluation of the policies without the need for executing more real-world samples. Constraints have been added to several PG methods. Uchibe and Doya [5] propose constrained policy search for GPOMDPs [6]. However, GPOMDPs belong to early PG algorithms which use the likelihood-ratio trick to compute the gradient of the expected sum of rewards and then update the policy parameters by a user-defined learning rate. Prashanth [7] propose constrained PG method for Stochastic Shortest Path problem with inequality constraints on Conditional Value-at-Risk (CVaR) as a risk measure. This method does not allow to include implicit constraints and cannot be easily extended for general episodic rewards, such as minimum distance of the trajectory from a target position. We propose a combination of GPREPS with the work of Uchibe and Doya – to extend Contextual REPS with constraints. The proposed method is called *Constrained REPS*, (CREPS). It evaluates the generated policies in a simulator and successively constrains the upper-level policy distribution. This (i) reduces the number of needed samples/iterations and consequently speeds-up the learning process of the model, and (ii) provides a safe policy when used with the real system.

Despite the fact that safe exploration becomes a key issue ([8], [9]), it has remained almost neglected in the general reinforcement learning community [10]. Akametalu et al. [11] propose Policy Gradient with safety metrics based on reachability analysis and demonstrate the algorithm on experimental quadrotor application. In contrast to us, they restrict the model to the class of control-affine systems

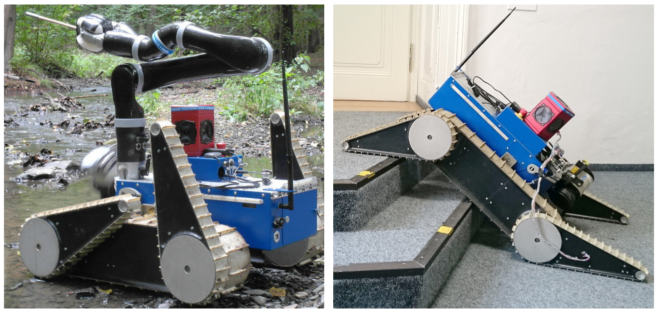


Fig. 1. Search and rescue rover platform. Track flippers visible on the side of the robot. A policy governs active tilting of the flippers assuring a *safe traversal*.

[1]Czech Technical University in Prague, Czech Institute of Informatics Robotics and Cybernetics

[2]Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics

with locally Lipschitz continuous functions. Birdwell and Livingston [12] suggest using handcrafted policies which assure safe corrections of the evaluated policy and prevent the robot from deviating significantly from the desired behavior and endangering itself.

Bagnell [8] encodes safety into uncertainty of the dynamics model, and assigns negative rewards for leaving an area close to already visited states. Generally, connecting safety and rewards into a single function is popular [13], [14], [10]. However, as it is shown in e.g. [15], [16], separating safety and rewards into independently optimized measures simplifies the learning process and allows for re-using the safety constraints for multiple different tasks. It is also usually unclear how to choose the weight vector to sum up the reward and safety terms.

Moldovan and Abbeel [17] define safe policies as those preserving *ergodicity*, which means from any state there exists a policy that returns the robot to the initial state. This definition is too strict, since it discards a whole class of problems where inverse actions do not exist or returning to the starting state is not possible or even desired.

Since it is difficult to provide any guarantees on data-driven models created from real-world samples without any prior knowledge about the underlying physics [18], [11], we replace the GP model from GPREPRS by a *cautious* physics-based simulator [19] that certifies the safety of policies. The cautious model of a system can never classify an unsafe policy (in the real world) as safe; the other kind of classification error is allowed (marking safe policies as unsafe).

The major contribution of this work is twofold: (i) We propose extending Contextual REPS [3] to a new constrained Policy Gradient method by including implicit constraints which cannot be derived explicitly from first-principle models. (ii) The new algorithm is evaluated on an autonomous flipper control task on real Search&Rescue rover platform (see Figure 1) and the results are compared with two existing methods [3] and [14].

II. CONSTRAINED RELATIVE ENTROPY POLICY SEARCH

A. Preliminaries

Model-free policy search algorithms usually follow these steps: (i) generate trajectories from the real-world system, (ii) compute a policy maximizing the expected sum of rewards on the so-far-generated trajectories, (iii) use the policy to generate a new real-world trajectory, (iv) repeat from (ii). Contextual REPS [3] adds a task-dependent context $\mathbf{s}$ (a changing property of the environment, e.g. the height of an obstacle), from which it extracts a feature vector $\boldsymbol{\phi}(\mathbf{s})$. This feature vector is an input of the stochastic *upper-level* policy $q(\mathbf{s}, \boldsymbol{\omega})$, which generates parameter vectors $\boldsymbol{\omega}$ which, in turn, define the lower-level policy. Samples $\boldsymbol{\omega} \simeq q(\mathbf{s})$ are evaluated on the model (which is called a *rollout*) and the corresponding sums of collected rewards (or a single episodic reward) $\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}$ are recorded. Using this data, Contextual REPS searches for a new upper-level distribution $p(\mathbf{s}, \boldsymbol{\omega})$, which maximizes the expected sums of rewards while staying close to the so-far-generated trajectories. The distance of trajectories is measured by Kullback-Leibler (KL) divergence. Bounding the KL divergence between $p(\mathbf{s}, \boldsymbol{\omega})$ and $q(\mathbf{s}, \boldsymbol{\omega})$ as follows:

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \log \frac{p(\mathbf{s}, \boldsymbol{\omega})}{q(\mathbf{s}, \boldsymbol{\omega})} \leq \epsilon,$$

where ϵ specifies the trade-off between exploration and exploitation, was shown to lead to uniform convergence in the whole parameter space [20]. Another constraint imposed on the upper-level distribution in Contextual REPS is to preserve the average distribution of context features:

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega}) \boldsymbol{\phi}(\mathbf{s}) = \hat{\boldsymbol{\phi}},$$

where $\hat{\boldsymbol{\phi}}$ is the average feature value.

B. Additional constraints

We extend Contextual REPS with additional constraints. In particular, for systems which are not inherently safe, or which are prone to wear, we use a cautious physics-based simulator (see Section III) to determine a rollout safety $S_{\mathbf{s}\boldsymbol{\omega}}$. The safety equals to 1 if policy $\boldsymbol{\omega}$ generated a safe trajectory for context $\mathbf{s}$, and equals to 0 otherwise. We force the upper-level distribution $p(\mathbf{s}, \boldsymbol{\omega})$ to have expected safety bigger than user-defined threshold δ

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(1 - S_{\mathbf{s}\boldsymbol{\omega}}) \leq \delta \tag{1}$$

Another source of additional constraints is a prior knowledge of physical limits such as the maximal joint angles (which are the control actions in our experiment). Violating such constraint is usually not safety-critical, since reaching an impossible pose is often prevented by some low-level motor drivers. However, evaluating many impossible samples naturally slows down the learning process.

Since all of these constraints have the same form, we compose a vector $\mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}$ as a collection of evaluated quantities (e.g. safety and mechanical constraints) and vector $\boldsymbol{\delta}$ as a collection of corresponding bounds. Such notation yields the following set of inequalities

$$\sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s}, \boldsymbol{\omega})(\mathbf{1} - \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta} \tag{2}$$

where $\mathbf{1}$ denotes a vector with all-ones of a corresponding dimension.

C. Constrained REPS

Constrained REPS searches for an upper level policy distribution $p(\mathbf{s}, \boldsymbol{\omega})$ corresponding to the solution of the following optimization problem:

$$\begin{aligned}
&\max_p \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s},\boldsymbol{\omega})\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}}, \\
&s.t. \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s},\boldsymbol{\omega}) \log \frac{p(\mathbf{s},\boldsymbol{\omega})}{q(\mathbf{s},\boldsymbol{\omega})} \leq \epsilon, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s},\boldsymbol{\omega})(\mathbf{1} - \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}) \leq \boldsymbol{\delta}, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s},\boldsymbol{\omega})\boldsymbol{\phi}(\mathbf{s}) = \hat{\boldsymbol{\phi}}, \\
&\quad \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} p(\mathbf{s},\boldsymbol{\omega}) = 1.
\end{aligned} \tag{3}$$

We follow the same derivation as proposed in [3] and solve the problem by the method of Lagrange multipliers (the detailed derivation is provided in [3] and is not given here due to space constraints). By setting the gradient of the corresponding Lagrangian with respect to $p(\mathbf{s},\boldsymbol{\omega})$ to zero, we obtain the closed form solution

$$p(\mathbf{s},\boldsymbol{\omega}) \propto q(\mathbf{s},\boldsymbol{\omega}) \exp\left(\frac{\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \tag{4}$$

where $\boldsymbol{\gamma}, \boldsymbol{\theta}$ and η are solutions of the following dual problem

$$\begin{aligned}
&\max_{\eta,\boldsymbol{\gamma},\boldsymbol{\theta}} \; g(\eta,\boldsymbol{\gamma},\boldsymbol{\theta}) \\
&s.t. : \boldsymbol{\gamma} > 0, \\
&\qquad \boldsymbol{\eta} > 0,
\end{aligned} \tag{5}$$

where

$$\begin{aligned}
g(\eta,\boldsymbol{\gamma},\boldsymbol{\theta}) =& \eta \log \sum_{\mathbf{s}} \sum_{\boldsymbol{\omega}} q(\mathbf{s},\boldsymbol{\omega}) \exp\left(\frac{\mathcal{R}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \\
&+ \eta\epsilon + \boldsymbol{\theta}^\top \hat{\boldsymbol{\phi}} + \boldsymbol{\gamma}^\top \boldsymbol{\delta}.
\end{aligned} \tag{6}$$

Using a dataset $\mathcal{D} = [\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}, \mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}, \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}]_{i=1,\ldots,N}$ where samples are picked from distribution $q(\mathbf{s},\boldsymbol{\omega})$, we can rewrite the previous equation as

$$\begin{aligned}
g(\eta,\boldsymbol{\gamma},\boldsymbol{\theta};\mathcal{D}) =& \eta \log \left[\frac{1}{N} \sum_{i=1}^{N} \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right)\right] \\
&+ \eta\epsilon + \boldsymbol{\theta}^\top \hat{\boldsymbol{\phi}} + \boldsymbol{\gamma}^\top \boldsymbol{\delta},
\end{aligned} \tag{7}$$

Dual problem (5) is a convex function with lower bound constraints. We achieved the fastest convergence with the interior point algorithm [21] with supplied gradients:

$$\begin{aligned}
\frac{\partial g}{\partial \eta} =& \epsilon + \log \frac{1}{N} \sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}) + \\
&- \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})(\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}})}{\eta \sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})},
\end{aligned} \tag{8}$$

$$\frac{\partial g}{\partial \boldsymbol{\gamma}} = \boldsymbol{\delta} + \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})\mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})} - \mathbf{1}, \tag{9}$$

$$\frac{\partial g}{\partial \boldsymbol{\theta}} = \hat{\boldsymbol{\phi}} - \frac{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})\boldsymbol{\phi}(\mathbf{s}^{[i]})}{\sum_{i=1}^{N} Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]})}, \tag{10}$$

where $Z(\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}) = \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}\mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right)$.

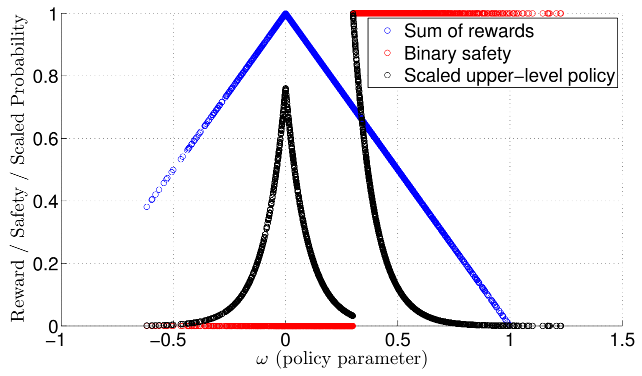


Fig. 2. Toy example demonstrating solution of problem (3). Resulting upper-level probability distribution (equation (11)) is in black. Please notice the maximum reward is located in an unsafe area. Therefore, CREPS computes a distribution that prefers safe, though suboptimal choices.

Probabilities $p^{[i]}$ of the new upper-level distribution are estimated from the optimal dual variables $\boldsymbol{\theta}, \boldsymbol{\gamma}, \eta$:

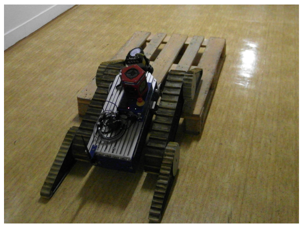
$$p^{[i]} \propto \exp\left(\frac{\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}} - \boldsymbol{\theta}^\top \boldsymbol{\phi}(\mathbf{s}^{[i]}) - \boldsymbol{\gamma}^\top \mathbf{1} + \boldsymbol{\gamma}^\top \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}}{\eta}\right) \tag{11}$$

To generate samples from this distribution, we either use weighted maximum likelihood to fit a normal distribution into samples $(\boldsymbol{\omega}^{[i]}, \mathbf{s}^{[i]})$ weighted by probabilities $p^{[i]}$ as suggested in [3], or we use importance sampling to generate samples from the non-parametric distribution. Constrained REPS is described in Algorithm 1.

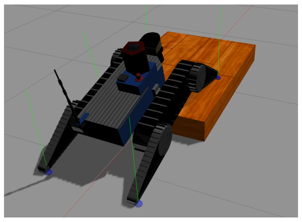
```
Input: maximal information loss $\epsilon$, vector of
constraints $\boldsymbol{\delta}$, context distribution $\mu(\mathbf{s})$, initial
upper-level policy $\pi(\boldsymbol{\omega}|\mathbf{s})$, number of policy updates $K$,
number of samples $N$.
for $k = 1, \ldots, K$ do
    for $i = 1, \ldots, N$ do
        Observe $\mathbf{s}^{[i]}$ from $\mu(\mathbf{s})$.
        Generate parameters $\boldsymbol{\omega}^{[i]}$ from $\pi(\boldsymbol{\omega}|\mathbf{s}^{[i]})$.
        Using $\boldsymbol{\omega}^{[i]}$ execute lower-level policy on the
        model and collect $\mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}$, $\mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}$.
    end
    Fill in dataset
        $\mathcal{D} = [\mathbf{s}^{[i]}, \boldsymbol{\omega}^{[i]}, \mathcal{R}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}, \mathbf{C}^{[i]}_{\mathbf{s}\boldsymbol{\omega}}]_{i=1,\ldots,N}$.
    Optimize dual function
        $[\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}] = \text{argmin}_{\eta',\boldsymbol{\gamma}',\boldsymbol{\theta}'} \, g(\eta, \boldsymbol{\gamma}, \boldsymbol{\theta}; \mathcal{D})$.
    Compute weights $p^{[i]}$ for all samples in $\mathcal{D}$, Eq. (11).
    Update upper-level policy $\pi(\boldsymbol{\omega}|\mathbf{s})$.
end
```

Algorithm 1: Constrained REPS

Figure 2 shows a toy example. We generate 1000 samples from one-dimensional normal distribution $q(\boldsymbol{\omega})$ with both mean and variance equal to 0.3. We have intentionally chosen the position of rewards maximum into $\boldsymbol{\omega} = 0$, safety equal to one for $\boldsymbol{\omega} > 0.5$ and mean of q into 0.3 to make all constraints active. We set the upper bound on KL-divergence $\epsilon = 0.1$ and the lower bound on safety $\delta = 0.6$. We verify



(a) Real robot executing a safe policy



(b) Visualization of the simulated robot

Fig. 3. The task is to learn how to safely traverse a pallet without any prior knowledge about the correct policy. There are more substantially different policies satisfying our constraints on safety and forward speed.

the average safety of samples generated from the distribution given by (11) is 0.6064, which is indeed above the required safety bound.

III. SAFETY FUNCTION

One of the additional constraints uses the term $S_{s,\omega}$, which denotes the safety of the rollout (1 is safe, 0 unsafe), and is computed by a *cautious* physics-based simulator. In simple cases, it can be an equation (even implicit), that checks some constraints of arbitrary order. When modeling a complex system, standard software physics and dynamics simulators can be used. The only requirement is that the simulation has to fulfill the cautiousness requirement. From the implementation point of view, the *cautiousness* can be a core part of the simulator design, or it is achieved by adding noise to the inputs and outputs, and testing more possible values of uncertain parameters (such as track-soil interaction). This way, it should be possible to create cautious simulations of most of the real-world systems (given the simulator can simulate all the important interactions and influences).

Since the simulator is only approximate (and contrary to GPs, it cannot be easily updated by new samples), real-world verification of its reward estimates has to be performed. Therefore, after finding an optimal policy in the simulator, a few more policy search iterations are performed on the real robot. Safety is still a concern, so every sampled lower-level policy is first tested for safety in the simulator, and if it is safe, it is executed on the real robot, and the real-world reward is collected (instead of the one reported by the simulator).

IV. EXPERIMENTS

A. Safe Traversal Task Description

The robot has to learn a *flipper control policy* that would allow it to traverse an obstacle without any prior knowledge about the correct traversal strategy (see Figure 3).

This task has been chosen because it very well separates good and bad policies (as well as safe and unsafe). A bad policy is not even able to get the robot on top of the obstacle, and therefore the robot gets stuck in front of it and travels only a short distance (receiving low reward). This task also allows for a wide variety of unsafe policies.

Some results of this experiment are compared to a similar task called *Adaptive Traversability* (AT) presented in [14] and improved in [22]. The goal of that task is to find a policy to control the flippers so that the robot maximizes a weighted sum of rewards (and minimizes penalties). The training process is a combination of supervised and reinforcement learning and requires a large set of manually annotated data. Since the forward speed is constant in the task, we compare the policies using the penalties for high pitch angle and for high acceleration.

We use a similar environment for the *Safe Traversal* (ST) task. The tracked robot starts in front of a standard wooden EUR 1 pallet. The robot is automatically driven forward by a constant speed, and the experiment ends after 30 seconds. For an illustration, see Figure 3.

a) States: States of the ST task are: (i) robot body pitch, and (ii) height of the terrain approximately 20 cm in front of the robot body (read from an octomap built online from laser scans).

b) Actions: ST policy controls independently the pairs of front and rear flippers using positional control. Therefore, the action space is continuous and 2-dimensional.

c) Rewards and Safety: In the AT task, safety is not modeled separately, and some of the safety features are part of the reward. The reward for the AT task is a weighted sum of (i) manually assigned safety penalty, (ii) high pitch/roll angle penalty, (iii) penalty for excessive flipper motion, (iv) robot forward speed reward, and (v) motion roughness penalty measured by accelerometers [14].

In the ST task, the reward is simply the distance traveled in 30 seconds over the pallet (the choice of policy influences e.g. track slippage and motor stress, which lower the speed). Safety is modeled explicitly by the cautious simulator, which marks as unsafe all rollouts in which the robot tops over, hits hard on the ground or obstacle (measured as deceleration), or hits objects with delicate parts of its body (e.g. sensors).

d) Policy: We use a policy that is linear in the states, and that controls front and rear flippers separately. The state vector is 2-dimensional, which yields 3 parameters per action, summing up to 6 policy parameters, $\omega = (\omega_1, \ldots, \omega_6)$, to be learned.

e) Context: In this experiment, we did not make use of the context—it was always set to zeros. This helped to keep the experiment simple, and was also needed to keep the possibility of comparing ST and AT results. The theory, however, supports using the context, and our future plans include designing an experiment that would make use of the context.

B. Simulator Setup

To run the simulation on a computer, we use the Open Dynamics Engine (ODE) which aims at fast approximate simulation. In ODE, we use a simple, yet reasonably plausible way to simulate the mobile robot with non-deformable tracks – a method typically used for simulating conveyor belts.

The collision model used in the simulator consists of simple-shape collision links (boxes, cylinders) approximating the CAD model of the robot. Weights, centers of mass, inertias, friction coefficients and other dynamics coefficients are estimated manually. It is important to estimate these parameters precisely enough, so that the simulator can be assumed cautious (which can be easily done using the CAD model). Or, in case of very influential unknown parameters, the simulator has to be run with multiple possible values and the worst-case outcome treated as the simulation result.

C. Experiment Setup

The experiment follows the pipeline described in Section III. First, 10 to 30 iterations of the CREPS algorithm are done using only the simulator (until the policy converges). It returns an upper-level policy, from which we can draw lower-level policies that are safe and lead to high expected rewards. In each iteration, we test approximately 150 simulated rollouts to estimate the sum in the dual function (7).

If improvement by real-world samples is intended, the experiment continues with 10 lower-level policies sampled from the optimal upper-level policy and checked in the simulator for safety. If they are safe, they are executed on the real robot. Otherwise, different policies are sampled until the desired number of safe policies is reached.

These real samples are further used to update the upper-level policy in the CREPS algorithm. This way we can correct the policy if the simulator estimated the rewards incorrectly.

D. Results

We tested the algorithm with several settings to show it consistently converges to high rewards in safe regions. The settings differed in e.g. the initial policy, upper-level policy representation (either a multivariate Gaussian distribution or the Importance Sampling mechanism as described in Section II-C), and the expected safety lower bound (generally 0.8, but in one experiment it was set to 0.0 to simulate unconstrained REPS).

The probability distribution of safe/unsafe policies showed to be very complex during the experiments, and it is far from being Gaussian or uniform. As can be seen in Figure 4, most of the time, the mean safety of rollouts is below the desired threshold of 0.8. However, it still tries to reach the threshold. In this case, the Importance Sampling method yields better results, as it better represents the complex distribution.

In the two experiments with unconstrained REPS, comparing Figure 4 and Figure 5, it is clear the algorithm strived for the highest rewards possible and safety quickly dropped to almost zero (which means the traversal was faster, but the robot hit ground too hard during the rollout). Interestingly, one of the unconstrained experiments reach a level of expected rewards not seen in any of the safety-constrained cases, which suggests the best policies are unsafe and CREPS correctly avoids these maxima.

Figures 4 and 5 show that the CREPS algorithm maximizes the rewards in cca the first 10 iterations, and then it

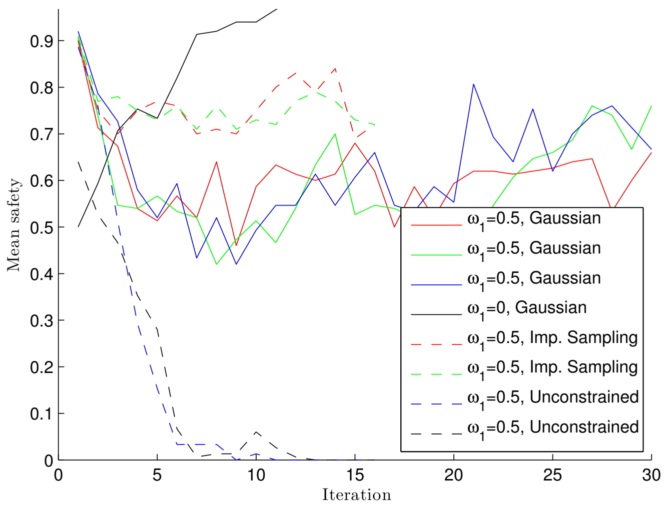


Fig. 4. Mean safety during rollouts. ω_1 is the initial value of the first element of the policy parameter vector, and it showed to have large influence on safety. The solid black curve is from a policy initialized close to a local maximum of the safety function. Compare with Figure 5 to see that the reward increased very slowly with expected safety higher than the desired threshold of 0.8. The last two policies were not constrained by safety at all (behaving like Contextual REPS), and they quickly found the best rewards lie in the unsafe space. Also note the Importance Sampling experiments tend to achieve higher expected safety.

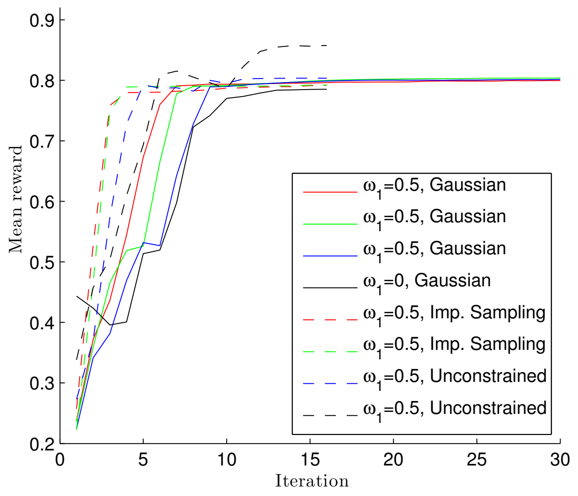


Fig. 5. Mean reward during rollouts. The last policy not constrained by safety converged to an optimum with the highest expected reward.

holds the good rewards and tries to satisfy the expected safety constraint. As we discussed earlier in this section, since the safe policy distribution is difficult to represent, the expected safety constraint is often broken. It does not, however, mean, that the robot could be damaged because of this imperfection. It only means we probably need to sample more lower-level policies until we find a safe one (which is always tested in the simulator before real execution).

We compared the high-pitch and high-acceleration penalties gathered by both an optimal policy for the ST task and also for the AT task. We performed 10 rollouts with each of the methods, and compared the penalties; results are shown in Figure 6. The pitch histogram was essentially the

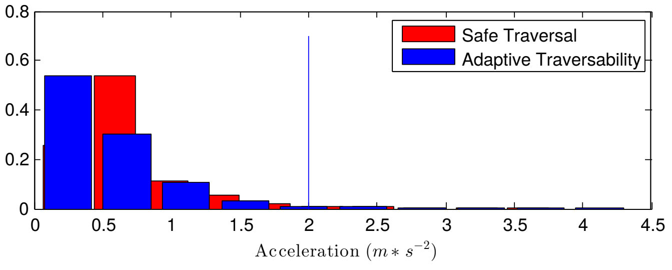


Fig. 6. Comparison of acceleration during 10 rollouts with AT and with ST (shown as relative histograms). Values over 2 (right to the blue line) are penalized in the AT task.

TABLE I

EXECUTION IN THE REAL WORLD

Iteration	Kind	Policy Converged	Mean Reward
29	Simulated	yes (in sim.)	0.80 ± 0.01
30	Real	no	0.73 ± 0.10
31	Real	yes (in real)	0.85 ± 0.05

The converged simulated policy performance was lower when used in the real world, but after only 2 real-world iterations, the CREPS algorithm converged to the real-world optimum (which is different from the simulated optimum, since the simulator is only approximate). The reader should notice that the simulated optimum had to be close to the real-world optimum, since CREPS doesn't allow large changes of the policy.

same for both tasks, so only the acceleration histograms are shown in Figure 6. The figure illustrates that the CREPS policy doesn't generate more dangerous trajectories than the AT policy (which was however trained with a large set of manually annotated data).

Last, we closed the loop improving one of the best policies found in the simulator by real-world reward samples. We executed two CREPS iterations, each with 10 samples. Safety was always checked in the simulator, then the sampled policy was executed, and the real-world reward collected. After two gradient search steps, the expected reward is higher than the best reward achieved in the simulator, as is shown in Table I. It is important to note that the policy search now cannot reuse samples from the simulator, since the reward estimate may be biased by imperfection of the simulator.

V. CONCLUSION AND FUTURE WORK

In a small number of iterations (and with about 2000 simulated trajectories), the robot learned how to safely traverse a previously unknown obstacle, and even the learning process itself was safe—thanks to the cautious simulator. We show that the cautious simulator can be designed in such a way that it does not constrain the possible actions too much and the robot is still able to reach the safe optimal rewards.

In our future work, we would like to address some of the open problems – first, constructing an experiment in which the context would be utilized (which it is not in this work, although the theory fully supports utilizing it). Second, examining different classes of the safety function and deducing e.g. convergence properties based on these classes.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union under grant agreement FP7-ICT-609763 TRADR; from the Czech Science Foundation under Project GA14-13876S, and by the Grant Agency of the CTU Prague under Project SGS16/161/OHK3/2T/13.

REFERENCES

[1] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement Learning: A Survey," *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, 1996.

[2] R. Tedrake, Z. Jackowski, R. Cory, J. W. Roberts, and W. Hoburg, "Learning to fly like a bird," *Under review*, vol. 8, 2009.

[3] A. Kupcsik, M. P. Deisenroth, J. Peters, L. A. Poh, P. Vadakkepat, and G. Neumann, "Model-based contextual policy search for data-efficient generalization of robot skills," *Artificial Intelligence*, 2014.

[4] A. Transeth, K. Pettersen, and P. Liljebäck, "A survey on snake robot modeling and locomotion," *Robotica*, 2009.

[5] "Constrained reinforcement learning from intrinsic and extrinsic rewards," in *IEEE 6th International Conference on Development and Learning*, 2007, pp. 163–168.

[6] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research (JAIR)*, vol. 15, pp. 319–350, 2001.

[7] P. L. A., "Policy gradients for cvar-constrained mdps," in *International Conference Algorithmic Learning Theory*, 2014, pp. 155–169.

[8] J. Bagnell, "Learning decisions: Robustness, uncertainty, and approximation," Ph.D. dissertation, Carnegie Mellon University, PA, USA, 2004.

[9] M. P. Deisenroth, G. Neumann, and J. Peters, "A Survey on Policy Search for Robotics," *Foundations and Trends in Robotics*, vol. 2, pp. 1–142, 2011.

[10] J. Kober and J. Peters, "Reinforcement learning in robotics: a survey," *International Journal of Robotics Research*, 2013.

[11] A. K. Akametalu, J. F. Fisac, J. H. Gillula, S. Kaynama, M. N. Zeilinger, and C. J. Tomlin, "Reachability-Based Safe Learning with Gaussian Processes," *CDC*, 2014.

[12] N. Birdwell and S. Livingston, "Reinforcement learning in sensor-guided AIBO robots." Tech. Rep., 2007.

[13] B. Fernandez-Gauna, M. Graña, J. M. Lopez-Guede, I. Etxeberria-Agiriano, and I. Ansoategui, "Reinforcement Learning endowed with safe veto policies to learn the control of Linked-Multicomponent Robotic Systems," *Information Sciences*, no. April, 2015.

[14] K. Zimmermann, P. Zuzanek, M. Reinstein, and V. Hlavac, "Adaptive Traversability of Unknown Complex Terrain with Obstacles for Mobile Robots," in *IEEE International Conference on Robotics and Automation*, 2014, pp. 5177—-5182.

[15] P. Ertle, M. Tokic, R. Cubek, H. Voos, and D. Soffker, "Towards learning of safety knowledge from human demonstrations," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, oct 2012, pp. 5394–5399.

[16] M. Pecka and T. Svoboda, "Safe Exploration Techniques for Reinforcement Learning An Overview," in *Modelling and Simulation for Autonomous Systems*, ser. Lecture Notes in Computer Science, no. 8906. Springer, 2014.

[17] T. M. Moldovan and P. Abbeel, "Safe Exploration in Markov Decision Processes," in *Proceedings of the 29th International Conference on Machine Learning*, may 2012.

[18] S. Ross and J. Bagnell, "Agnostic system identification for model-based reinforcement learning," in *Proceedings of the 29th International Conference on Machine Learning*, Edinburgh, Scotland, UK, 2012.

[19] M. Pecka, K. Zimmermann, and T. Svoboda, "Safe Exploration for Reinforcement Learning in Real Unstructured Environments," in *20th Computer Vision Winter Workshop*, P. Wohlhart and V. Lepetit, Eds. Graz, Austria: TU Graz, 2015.

[20] C. Daniel, G. Neumann, and J. Peters, "Hierarchical relative entropy policy search," 2012, pp. 1–9.

[21] R. H. Byrd, M. E. Hribar, and J. Nocedal, "An interior point algorithm for large-scale nonlinear programming," *SIAM Journal on Optimization*, vol. 9, no. 4, pp. 877–900, 1999.

[22] K. Zimmermann, P. Zuzánek, M. Reinstein, T. Petříček, and V. Hlaváč, "Adaptive Traversability of Partially Occluded Obstacles," in *IEEE International Conference on Robotics and Automation*, 2015.