Risk-Averse Zero-Order Trajectory Optimization

Anonymous Author(s)
Affiliation
Address
email

Abstract: We introduce a simple but effective method for managing risk in zero-order trajectory optimization that involves probabilistic safety constraints and balancing of optimism in the face of epistemic uncertainty and pessimism in the face of aleatoric uncertainty of an ensemble of stochastic neural networks. Various experiments indicate that the separation of uncertainties is essential to performing well with data-driven MPC approaches in uncertain and safety-critical control environments.

Keywords: zero-order trajectory optimization,CEM, aleatoric, epistemic, safety, uncertainty

1 Introduction

Data-driven approaches to sequential decision-making are becoming increasingly popular [1, 2, 3, 4]. They hold the promise of reducing the number of prior assumptions about the system that are imposed by traditional approaches that are based on nominal models.

Such approaches come in several different flavors [5]. Model-free approaches attempt to extract closed-loop control policies directly from data, while model-based approaches rely on a learned model of the dynamics to either generate novel data to extract a policy or to be used in a model-predictive control fashion (MPC). This study belongs to the latter line of work.

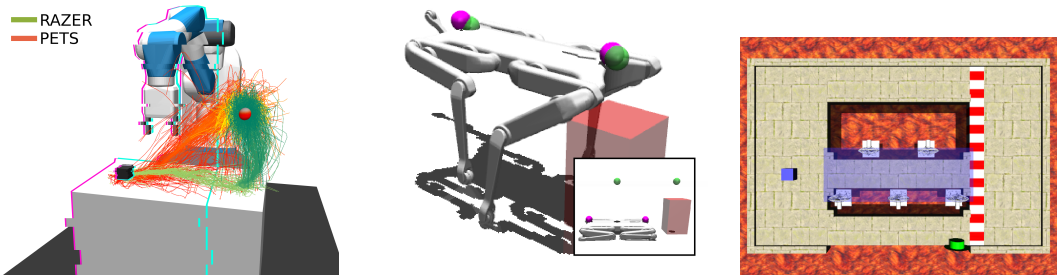
Model-based methods have several advantages over pure model-free approaches. Firstly, humans tend to have a better intuition on how to incorporate prior knowledge into a model rather than into a policy or value function. Secondly, most model-free policies are bounded to a specific task, while models are task-agnostic and can be applied for optimizing arbitrary cost functions, given sufficient exploration.

Nevertheless, learning models for control come with certain caveats. Traditional MPC methods require the model and cost function to permit a closed-form solution which restricts the function class prohibitively. Alternatively, gradient-based iterative optimization can be employed, which allows for a larger class of functions but typically fails to yield satisfactory solutions for complicated function approximators such as deep neural network models. In addition, calculating first-order or even second-order information for trajectory optimization tends to be computationally costly, which makes it hard to meet the time constraints of real-world settings. This motivates the usage of zero-order, i.e gradient-free or sample-based methods, such as the Cross-entropy Method (CEM) that do not rely on gradient information but are efficiently parallelizable.

Many methods relying on a learned model and zero-order trajectory optimizers have been proposed [6, 7, 8], but all share the same problem: compounding of errors through auto-regressive model prediction. This naturally brings us to the question of how can we effectively manage model errors and uncertainty to be more data-efficient and safe. Arguably, this is one of the main obstacles to applying data-driven model-based methods to the real world, e.g. to robotics settings.

In this work, we introduce a risk-averse zero-order trajectory optimization method (RAZER) for managing errors and uncertainty in zero-order MPC and test it on challenging scenarios (Fig. 1). We argue that it is essential to differentiate between the two types of uncertainty in the model-predictive setting: the aleatoric uncertainty arising from inherent noise in the system and epistemic uncertainty arising from the lack of knowledge [9, 10]. We measure these uncertainties by making use of probabilistic ensembles with trajectory sampling [6] (PETS). Our contributions can be summarized

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.



(a) Noisy-FetchPickAndPlace (b) Solo8-LeanOverObject (c) BridgeMaze

Figure 1: Environments considered for uncertainty-aware planning.

as follows: (i) method for separation of uncertainties in probabilistic ensembles (termed PETSUS); (ii) efficient use of aleatoric and epistemic uncertainty in model-based zero-order trajectory optimizers; (iii) an simple but practical approach to probabilistic safety constraints in zero-order MPC.

2 Related Work

Uncertainty Estimation. In the typical model-based reinforcement learning (MBRL) setting, the true transition dynamics function is modeled through an approximator. Impressive results have been achieved by both parametric models [11, 12, 13, 14], such as neural networks, and nonparametric models [15, 16, 17, 18], such as Gaussian Processes (GP). The latter inspired seminal work on the incorporation of the dynamics model's uncertainty for long-term planning [18, 19]. However, their usability is limited to low-data, low-dimensional regimes with smooth dynamics [20, 21], which is not ideal for robotics applications. Alternative parametric approaches include ensembling of deep neural networks, used both in the MBRL community [6, 22], and outside [23, 24]. In particular, ensembles of *probabilistic* neural networks established state-of-the-art results in the MBRL community [6], but focus mainly on estimating the expected cost and disregard the underlying uncertainties. In comparison, we propose a treatment of the resulting uncertainties of the ensemble model.

Zero-order MPC. The learned model can be used for policy search like in PILCO [25, 18, 19, 26] or for online model-predictive control (MPC) [27, 28, 6]. In this work, we do planning in an MPC fashion and employ a zero-order method as a trajectory optimizer, since less sensitive to hyperparameter tuning and less likely to get stuck in local minima of complex objective functions. Specifically, we consider a sample-efficient implementation of the Cross-Entropy method [29, 30] introduced in [31].

Safe MPC. Separating the sources of uncertainty is of particular importance for AI applications directly affecting humans' safety, as self-driving cars, elderly care systems, or in general any application that involves a physical interaction between the AI system and humans. Disentangling epistemic from aleatoric uncertainty allows for separate optimization of the two, as they represent semantically different objectives: efficient exploration and risk-awareness. Extensive research on uncertainty decomposition has been done in the Bayesian setting and the context of safe policy search [32, 33, 34, 35], MPC planning [36, 37, 38], and distributional RL [39, 40]. On the other side, a state-of-the-art baseline for ensemble learning like PETS [6], despite estimating both uncertainties, only optimizes for the *expected* cost during action evaluation. Our work aims at filling this gap by explicitly integrating the propagated uncertainty information in the zero-order MPC planner.

3 Method

Our approach concerns itself with the efficient usage of uncertainties in zero-order trajectory optimization and is therefore generally applicable to such optimizers. We are interested in modeling noisy system dynamics $x_{t+1} = f(x_t, u_t, w(x_t, u_t))$ where f is a nonlinear function, x_t the observation vector, u_t applied control input and $w(x_t, u_t)$ a noise term sampled from an arbitrary distribution.

Consequently, in the absence of prior knowledge about the function f, the system needs to be modeled by a complex function approximator such as a neural network. Furthermore, we are interested in managing uncertainties based on our fitted model, which is erroneous. To this end, we use stochastic ensembles of size K, where the output of each model $\vartheta^k(x_t, u_t)$ are parameters of

a normal distribution depending on input observation x_t and control u_t. As a by-product, our auto-regressive model prediction based on controls $\boldsymbol{u}$ becomes a predictive distribution over trajectories τ; $\psi^\tau(x_t, \boldsymbol{u}) \coloneqq p(\tau|x_t, \boldsymbol{u}; \theta)$ where θ denotes the parameters of the ensemble. For convenience, from this point onward we will differentiate between multiple usages of ψ^τ. We denote with $\psi^x_{\Delta t}$ the distribution $p(x_{t+\Delta t}|x_t, \boldsymbol{u}_{t:t+h}; \theta)$ over states at time step $t + \Delta t$ and $\psi^{\boldsymbol{\vartheta}}_{\Delta t}$ the distribution over the Gaussian parameter outputs $p(\boldsymbol{\vartheta}_{t+\Delta t}|x_t, \boldsymbol{u}_{t:t+h}; \theta)$ at time step $t + \Delta t$ of the planner.

3.1 Planning and Control

To validate our hypothesis that accounting for uncertainty in the environment and model prediction is essential to develop risk-averse policies, we use the Cross-Entropy Method (CEM) with improvements suggested in Pinneri et al. [31]. Accordingly, at each time step t we sample a finite number of control sequences $\boldsymbol{u}$ for a finite horizon H from an isotropic Gaussian prior distribution which we evaluate from the state x_t using an auto-regressive forward-model and the cost function. The sampling distribution is refitted in multiple rounds based on good-performing (elite) trajectories. After this optimization step, the first action of the mean of the fitted Gaussian distribution is executed. Since this approach utilizes a predictive model for a finite horizon at each time step, it naturally falls into the category of Model Predictive Control (MPC) methods.

Although we use CEM, our approach of managing uncertainty can generically be applied to other zero-order trajectory optimizers such as MPPI [28], by a modification of the trajectory cost function.

3.2 The Problem of Uncertainty Estimation

Since we have a stochastic model of the dynamics, at the model prediction time step t we observe a distribution over potential outcomes. Indeed, since our model outputs are parameters of a Gaussian distribution, with auto-regressive predictions we end up with a distribution over possible Gaussians for a certain time step t.

Given a sampled action sequence $\boldsymbol{u}$ and the initial state x_t we observe a distribution over trajectories ψ_τ. To efficiently sample from the trajectory distribution ψ_τ we use the technique introduced by Chua et al. [6] (PETS) which involves prediction particles that are sampled from the probabilistic models and randomly mixed between ensemble members at each prediction step. In this way, the sampled trajectories are used to perform a Monte Carlo estimate of the expected trajectory cost $\mathbb{E}_{\tau\sim\psi^\tau}[c(\tau)]$. However, this does not take the properties of ψ^τ into account, which might be a high-entropy distribution and may lead to very risky and unsafe behavior. In this work, we alleviate this by looking at the properties of ψ^τ, i.e. different kinds of uncertainties arising from the predictive distribution.

3.3 Learned Dynamics Model

We learn a dynamics model f_θ that approximates the true system dynamics $x_{t+1} = f(x_t, u_t, w(x_t, u_t))$. As a model class, we use an ensemble of neural networks with stochastic outputs as in Chua et al. [6]. Each model k, parameterizes a multivariate Gaussian distribution with diagonal covariance, $f^k_\theta(x_t, u_t) = \mathcal{N}(x_{t+1}; x_t + \mu^k_\theta(x_t, u_t), \Sigma^k_\theta(x_t, u_t))$ where $\mu^k_\theta(\cdot,\cdot)$ and $\Sigma^k_\theta(\cdot,\cdot)$ are model functions outputting the respective parameters.

Iteratively, while interacting with the environment, we collect a dataset of transitions $\mathcal{D}$ and train each model k in the ensemble by the following negative log-likelihood loss on the Gaussian outputs:

$$\mathcal{L}(\theta, k) = \mathbb{E}_{x_t, u_t, x_{t+1}\sim\mathcal{D}}\Big[-\log\mathcal{N}(x_{t+1}; x_t + \mu^k_\theta(x_t, u_t), \Sigma^k_\theta(x_t, u_t))\Big] \tag{1}$$

In addition, we use several regularization terms to make the model training more stable. We provide more details on this in Suppl. A.

3.4 Separation of Uncertainties

In the realm of parametric estimators, two uncertainties are of particular interest. *Aleatoric* uncertainty is the kind that is irreducible and results from inherent noise of the system, e.g. sensor noises in robots. On the other hand, we have *epistemic* uncertainty resulting from lack of data or knowledge which is reducible. This begs the question, how can we separate these uncertainties given an auto-regressive

dynamics model f_θ? The way that we efficiently sample from ψ^τ is by mixing sampled prediction particles, similarly as in PETS[6]. This process is illustrated by the red lines in Fig. 2.

Simple model prediction disagreement is not a good measure for *aleatoric* uncertainty since it can be entangled with epistemic uncertainty. Given our assumptions about the system dynamics, we measure *aleatoric* uncertainty as the entropy of the predicted normal distributions of the ensemble models. More concretely, given a sampled particle state $\tilde{x}_t$, we define the estimated aleatoric uncertainty for ensemble model associated to particle b at time step t as:

$$\mathfrak{A}_b(x|\tilde{x}_t, u_t) = \mathcal{H}_{x\sim\psi^x_{\Delta t,b}}(x) \tag{2}$$

Where $\psi^x_{\Delta t,b}$ is the output distribution of ensemble model based on inputs $\tilde{x}_t$, u_t. Since in the end we are interested in the aleatoric uncertainty incurred from applying the action sequence $\boldsymbol{u}$ from initial state x_t, the quantity of interest for us is the expected aleatoric uncertainty for time slice t:

$$\mathfrak{A}(x|u_t) = \mathbb{E}_{\tilde{x}_b\sim\psi^x_{\Delta t}}\left[\mathfrak{A}_b(x|\tilde{x}_b, u_t)\right] \tag{3}$$

Intuitively, because we only have access to the ensemble for sampling, we take a time-slice in the sampled trajectories from ψ^τ and compute the output entropies. Moreover, since we assume a Gaussian 1-step predictive distribution this is an expectation over differential Gaussian entropy. An alternative way of computation which we also explore in this work is calculating the expected particle variance for time slice t of the prediction horizon:

$$\mathrm{Var}^{\mathfrak{A}}_{t+1} = \frac{1}{B}\sum_{b=1}^{B}\Sigma^k_\theta(\tilde{x}_{t,b}, u_t) \tag{4}$$

For estimating the *epistemic* uncertainty, one would be tempted to look at the disagreement between ensemble models in parameter space $\mathrm{Var}[\theta]$, but this is not completely satisfying, since neural networks tend to be over-parametrized and variance within the ensemble still may exist albeit the optimum has been reached by all ensemble models. An alternative would be to calculate the Fisher information metric $\mathcal{I} := \mathrm{Var}[\nabla_\theta \log \mathcal{L}(x_{t+1}|x_t, u_t)]$ where $\mathcal{L}$ denotes the likelihood function, but this tends to be expensive to compute.

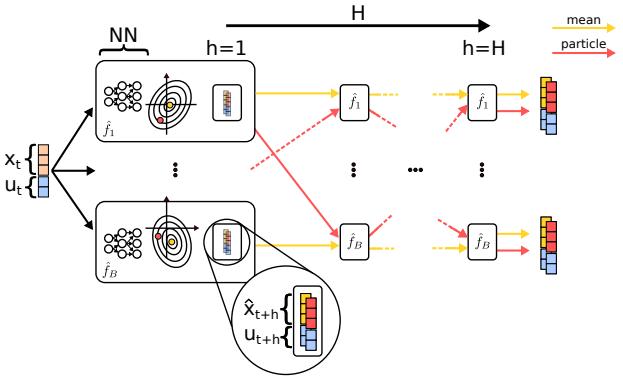


Figure 2: Probabilistic Ensembles with Trajectory Sampling and Uncertainty Separation (PETSUS)

Given the assumption of local Gaussianity, the true epistemic uncertainty for this case is the predictive entropy over the Gaussian parameters $\boldsymbol{\vartheta}$ at time step $t+h$.

$$\mathfrak{E}(x_t, \boldsymbol{u}_{t:t+h}) = \mathcal{H}_{\psi^{\boldsymbol{\vartheta}}_{\Delta t}}(\boldsymbol{\vartheta} \mid x_t, \boldsymbol{u}_{t:t+h}) \tag{5}$$

It is easy to verify that this quantity is 0 given perfect predictions of the model. Note that, because of auto-regressive predictions of a nonlinear model, this is a very difficult object to handle. Nevertheless, since our predictive distribution $p(x \mid x_t, u_t; \boldsymbol{\vartheta})$ is parametrized by model outputs, we may utilize disagreement in $\boldsymbol{\vartheta}_t$ to approximate $\mathfrak{E}$. To get correct estimations, we need to propagate mean predictions $\bar{x}$ in addition to the particles as illustrated as the yellow lines in Fig. 2. We quantify epistemic uncertainty as ensemble disagreement at time step t:

$$\mathrm{Var}^{\mathfrak{E}}(x_{t+1}) = \mathrm{Var}^e[\mu^k_\theta(\bar{x}_t, u_t)] + \mathrm{Var}^e[\Sigma^k_\theta(\bar{x}_t, u_t)] \tag{6}$$

where Var^e is the empirical variance over the $k = 1 \ldots K$ ensembles.

3.5 Probabilistic Safety Constraints

When applying data-driven control algorithms to real systems, safety is of utmost importance. In the realm of zero-order optimization, safety constraints can be easily introduced by putting an infinite cost on constraint-violating trajectories. Nevertheless, we are dealing with erroneous stochastic nonlinear models which lead to nontrivial predictive distributions of future states, based on the control

sequence $\boldsymbol{u}$. For this reason, we want to control the risk of violating the safety constraints that we, as practitioners, are willing to tolerate. If we denote the observation space as $\mathbb{X}$, given a violation set $\mathbb{C} \subset \mathbb{X}$, we define the probability of the control sequence $\boldsymbol{u}$ to enter the violation set at time $t + \Delta t$ as $p(x \in \mathbb{C} \mid x_t, \boldsymbol{u}) = \int_{x \in \mathbb{C}} \psi^x_{\Delta t}(x \mid x_t, \boldsymbol{u})$. In practice, it is hard to compute this integral efficiently, since our distribution $\psi^x_{\Delta t}$ is nontrivial as a result of nonlinear propagation of uncertainty. Furthermore, the violation set $\mathbb{C}$ might not have the structure necessary to allow an efficient solution to the integral, in which case one needs to resort to Monte Carlo estimation.

To simplify computation and gain speed, we consider box violation sets resulting in each dimension of x being constrained to be outside of $[a, b] \in \{a, b \,|\, a, b \in \mathbb{R}^2,\ a < b\}$. By performing moment matching by a Gaussian in each time-slice $\psi^x_{\Delta t}$, the probability of ending up in state x at time step $t + \Delta t$ is given by integrating $\mathcal{N}(x; \mu_{t+\Delta t}, \Sigma_{t+\Delta t})$, where μ and Σ are estimated by Monte Carlo sampling. If we further assume a diagonal covariance Σ, this integral can be deconstructed into d univariate Gaussian integrals, which can be computed fast and in closed form (error function). Hence, the probability of a constraint violation happening at time step t is defined by:

$$p(x \in \mathbb{C} \mid x_t, \boldsymbol{u}) = \prod_{i=0}^{d} \int_{x \in \mathbb{C}} \mathcal{N}(x^i; \mu^i_{t+\Delta t}, \sigma^i_{t+\Delta t}) \tag{7}$$

3.6 Implementing Risk-Averse ZERo-Order Trajectory Optimization (RAZER)

We assume the task definition is provided by the cost $c(x_t, \boldsymbol{u})$. For trajectory optimization, we start from a state x_t and predict with an action sequence $\boldsymbol{u}$ the future development of the trajectory τ. Along this trajectory, we want to compute a single cost term which is conveniently defined as the expected cost of all particles $\tilde{x}$ summed over the planning horizon H:

$$c(x_t, \boldsymbol{u}) = \sum_{\Delta t=1}^{H} \frac{1}{B} \sum_{b=1}^{B} c(\tilde{x}^b_{t+\Delta t}, u_{t+\Delta t}). \tag{8}$$

The optimizer, in our case CEM, will optimize the action sequence $\boldsymbol{u}$ to minimize the cost in a probabilistic sense, i.e. $p(\boldsymbol{u} \mid x) \propto \exp(-\beta\, c(x, \boldsymbol{u}))$ where β reflects the strength of the optimizer (the higher the more likely it finds the global optimum). To make the planner uncertainty-aware, we need to make sure it avoids unpredictable parts of the state space by making them less likely. Using the aleatoric uncertainty provided by PETSUS Eq. 4, we define the aleatoric penalty as

$$c_{\mathfrak{A}}(x_t, \boldsymbol{u}) = w_{\mathfrak{A}} \cdot \sum_{\Delta t=1}^{H} \sqrt{\mathrm{Var}^{\mathfrak{A}}_{t+\Delta t}}, \tag{9}$$

where $w_{\mathfrak{A}} > 0$ is a weighting constant. The larger the aleatoric uncertainty, the higher the cost.

To guide the exploration to states where the model has epistemic uncertainty Eq. 6 (due to lack of data), we use an epistemic bonus:

$$c_{\mathfrak{E}}(x_t, \boldsymbol{u}) = -w_{\mathfrak{E}} \cdot \sum_{\Delta t=1}^{H} \sqrt{\mathrm{Var}^{\mathfrak{E}}_{t+\Delta t}}, \tag{10}$$

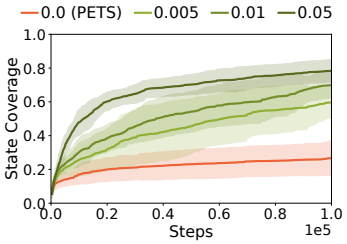
where $w_{\mathfrak{E}} > 0$ is a weighting constant. To be able to operate on a real system, the most important part is to adhere to safety constraints. As formulated in Eq. 7, the predicted safety violations need to be uncertainty aware, independent of the source of uncertainty. We integrate this into the planning method by adding:

$$c_{\mathfrak{S}}(x_t, \boldsymbol{u}) = w_{\mathfrak{S}} \cdot \sum_{\Delta t=1}^{H} [\![p(\hat{x}_{t+\Delta t} \in \mathbb{C}) > \delta]\!] \tag{11}$$

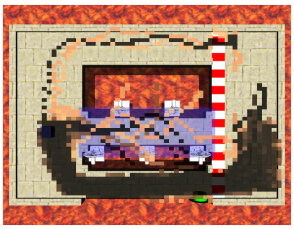
where $[\![\cdot]\!]$ is Iverson bracket and $w_{\mathfrak{S}}$ is either a large penalty $c_{\max}$ or 0 to disable safety. An alternative for implementing safety constraints into CEM is by changing the ranking function [41]. The overall algorithm used in a model-predictive control fashion is outlined in Suppl. B.

4 Experiments

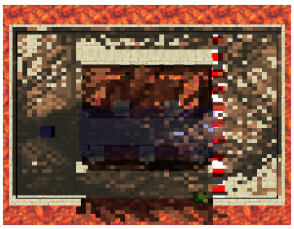
We study our uncertainty-aware planner in 4 continuous state-/action-space environments and compare results with PETS [6]. PETS does not use the model's uncertainties explicitly, instead, they enter only by taking the mean over the particle costs. We first describe the environments.



(a) State space exploration over time depending on epistemic bonus ($w_{\mathfrak{E}}$).



(b) State-space coverage of PETS.



(c) State-space coverage of RAZER with $w_{\mathfrak{E}} = 0.05$.

Figure 3: Active learning setting: The epistemic bonus let RAZER seek states for which no or only little training data exists (a,c) (5 runs each). PETS overfits to a particular solution (b). In (b) and (c), visited states as dots being black for early times and fading into pale orange if visited later.

BridgeMaze This toy environment (see Fig. 1c), we specifically designed to study the different aspects of uncertainty independently. The agent (blue cube) starts on the left platform and has to reach the goal platform on the right. To reach the goal platform, the agent has to move over one of three bridges without falling into the lava. The upper bridge is safeguarded by walls; hence, it is the safest path to the goal but also the longest. The lower bridge has no walls and therefore is more dangerous for an unskilled agent to cross but the path is shorter. The middle bridge is the shortest path to the goal. However, randomly appearing strong winds perpendicular to the bridge might cause the agent to fall off the bridge with some probability, rendering this bridge dangerous.

Noisy-HalfCheetah This environment is based on *HalfCheetah-v3* from the OpenAI Gym toolkit. We introduce aleatoric uncertainty to the system by adding Gaussian noise $\xi \sim \mathcal{N}(\mu, \sigma^2)$ to the actions when the velocity is above 6. The action noise translates into a non-Gaussian and potentially very complicated state-space noise distribution that makes the control problem very challenging.

Noisy-FetchPickAndPlace Based on the *FetchPickAndPlace-v1* gym environment. Additive action noise is applied to the gripper so that its grip on the box might become tighter or looser. The noise is applied for x-positions < 0.8 which is illustrated in Fig. 1a by a blue line causing the agent to drop the box with high probability if it tries to lift the box too early.

Solo8-LeanOverObject In this robotic environment, the task of a quadrupedal robot [42] is to stand up and lean forward to reach a target position (purple markers need to reach green dots in Fig. 1b) without hitting an object visualized by the red cube representing the unsafe zone. The robot starts in a laying position as shown in the inset of Fig. 1b. As in the *Noisy-HalfCheetah* environment, Gaussian action noise is applied to mimic real-world perturbances.

4.1 Algorithmic Choices and Training Details

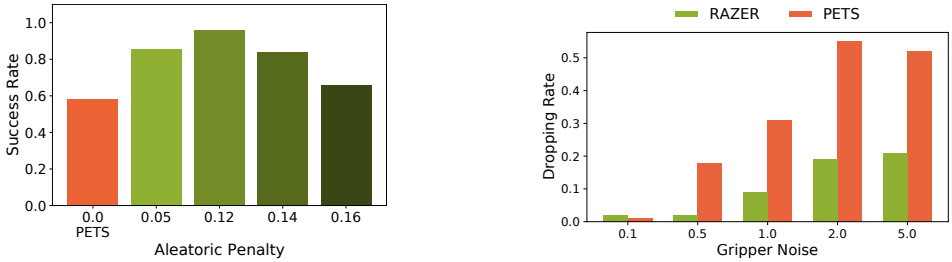
For the control part, we use the CEM implementation from [31], both for RAZER and PETS. See Suppl. A.2 for further details on the parameters we used in each experiment. For planning, we use the same architecture for the ensemble of probabilistic models, both in RAZER and in PETS. The only difference is that in RAZER we also forward propagate the mean state predictions in addition to the sampled state predictions. Further details can be found in Suppl. A.1.

For training the predictive model, we alternate between two phases: data collection and model fitting. In the BridgeMaze environment, we collect 5 rollouts of length 80 each time and append them to the previous rollouts and fit the model for 25 epochs. For Noisy-HalfCheetah we collect 1 rollout every time and fit for 50 epochs.

Next, we will present RAZER's exploration and safety in the *BridgeMaze* environment. Afterward, we are going to discuss planning with external safety constraints in the *Solo8* environment. We complete this section with results on *Noisy-HalfCheetah* and *Noisy-FetchPickAndPlace*.

4.2 Active Learning for Model Improvement

If model uncertainties are used for risk-averse planning, they are only meaningful if the model has the right training data. Only from good data can the parameters of the approximate noise model be learned correctly. In case of too little data, the agent might avoid parts of the state space due to an



(a) BridgeMaze success depending on $w_{\mathfrak{A}}$ for 50 runs. (b) Dropping rate in FetchPickAndPlace for 5 runs.

Figure 4: Risk-averse planning in the face of aleatoric uncertainty yields higher success rates in noisy environments. For (b) we use ground truth models and a fixed aleatoric penalty weight $w_{\mathfrak{A}}$.

overestimation of the model uncertainties. On the other hand, the agent might enter unsafe regions for which the uncertainties are underestimated. By adding the epistemic bonus to our domain-specific cost, the planner can actively seek states with high epistemic uncertainty, i.e. for which no or only little training data exists.

Figure 4a shows this active data gathering process for the BridgeMaze environment. PETS finds one particular solution to the problem of reaching the goal platform. It chooses the path over the safer, lower bridge over the dangerous middle path and the longer path via the upper bridge (Fig. 3b). Once, one solution is found, the model overfits to it without exploring any other parts of the state space. This is also reflected in the plateauing of the red curve in Fig. 3a.

In comparison, RAZER actively explores larger and larger parts of the state space with an increasing weight of the epistemic bonus (Fig. 3a). RAZER not only finds the easy solution found by PETS but also extensively explores other parts of the state space (Fig. 3c). To not get stuck at the middle bridge during exploration due to the inherent noise in the associated states, it is important to separate between epistemic and aleatoric uncertainties. Only the former should be used for exploration. With enough data, our model can correctly capture the uncertainties of these states resulting in the epistemic uncertainty approaching zero.

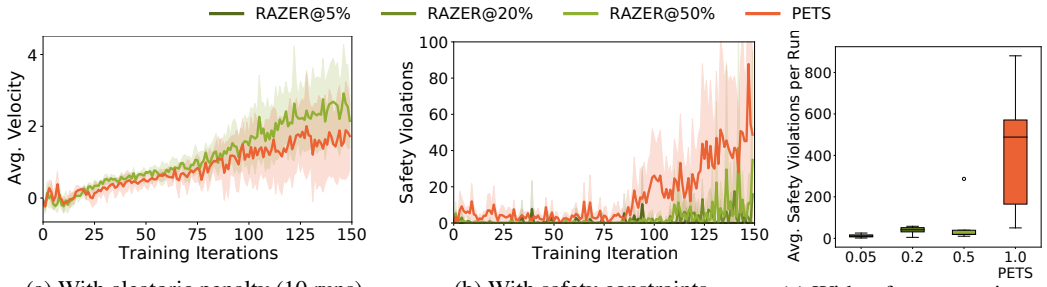
4.3 Risk-Averse Planning

Once a good model is learned, the estimated noise model can be actively used for safe planning. What differentiates RAZER from PETS is that our method makes full usage of the information from the noise model while in PETS the uncertainties only enter planning by taking the mean over the particle costs. First, the PETS approach does not differentiate between aleatoric and epistemic uncertainty, and second, the sample-based approach suffers from the typically low (~ 5) sample size.

BridgeMaze:. Figure 4a shows the success rate of PETS and RAZER in the BridgeMaze. In both cases, we use the same model that was trained from data collected during a training run with $w_{\mathfrak{E}} = 0.05$. Hence, the model saw enough training data from all parts of the state space. The noise in the env is tuned such that there is a chance to cross the bridge without falling. While in Fig. 3b PETS avoided this path because of an overestimation of the state due to a lack of enough training data now it sometimes sees a chance to cross the bridge. However, these attempts are very likely to fail because of stronger winds that occur randomly, resulting in a success rate of only 58%. RAZER does not rely on sampling for the aleatoric part and can thus avoid risk. With a higher penalty constant the success rate increase up to 96% but only as long as the agent is willing to take a risk at all. For large values of $w_{\mathfrak{A}}$ the agent becomes so afraid that it only moves slowly (decreasing reward in Fig. 4a).

Noisy-HalfCheetah. How does RAZER perform on the Noisy-HalfCheetah environment when learning models from scratch? Without aleatoric penalty, the planner is optimistic. Risky situations are only detected if a failing particle is sampled. Thus, the noise is mostly neglected and the robot increases its velocity, gets destabilized, and ends up slower than with the aleatoric penalty (Fig. 5a).

FetchPickAndPlace. In this environment, a 7-DoF robot arm should bring the box to a target position – starting and target positions are at the opposite sides of the table. The shortest path is to lift the box and move in a straight line to the target. However, with noise applied to the gripper action, there is a certain probability to drop the box along the way. When penalizing aleatoric uncertainty, this is avoided and also fewer trajectory samples are "wasted" in high-entropic regions, as presented



(a) With aleatoric penalty (10 runs). (b) With safety constraints. (c) With safety constraints.

Figure 5: Halfcheetah environment (task lengths 300 steps) with learning models from scratch. At 150 iterations we have seen then only 45k points. (a) Performance under noisy actions. By applying the aleatoric penalty, RAZER can navigate the uncertainties better – leading to higher returns faster. (b) Safety violations above a certain body height (simulating a low ceiling) for different values of δ. With increasing δ, RAZER is much more seldom violating a constraint in stark contrast to PETS. In (c) the number of violations is averaged over the last 50 iterations (summed over 10 rollouts).

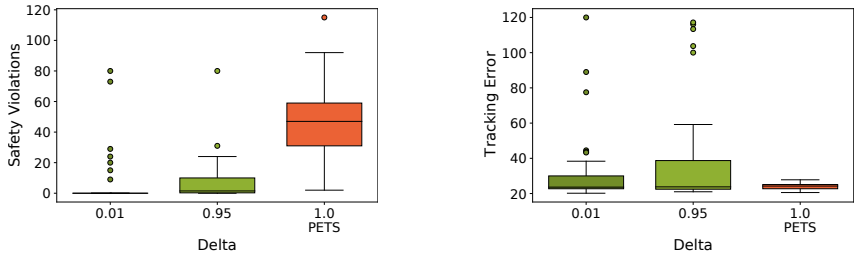


Figure 6: Safe planning vs. task-oriented planning in the *Solo8-LeanOverObject* environment with noisy actions. Left: number of safety violations for different values of δ Eq. 11. Right: tracking error.

in Fig. 1a. Figure 4b shows the number of times the box is dropped on the table depending on the aleatoric penalty. RAZER adopts a cautious behavior, preferring to slide the box on the table and lifting it only in the area without action noise, keeping below 20%, even when considerable noise is applied.

4.4 Planning with External Safety Constraints

Noisy-HalfCheetah:. We consider a safety constraint on the height of the body above ground simulating a narrow passage. Figure 5b shows the number of safety violations. Note that PETS has the same penalty cost for hard violations.

Solo8-LeanOverObject:. In this experiment, the robot has to move to two target points with its front and rear of the trunk while avoiding entering a specified rectangular area (fragile object). The front feet are fixed. To track the points, the robot has to lean forward, such that it can lose balance due to noisy actions. RAZER successfully manages to satisfy the safety constraints almost always, which is in contrast to PETS, as shown in Fig. 6. However, satisfying the safety constraint comes with the cost of reduced tracking accuracy.

5 Conclusion

In this work, we have provided a methodology to separate uncertainties in stochastic ensemble models (PETSUS) which can be used as a tool to build risk-averse model-based planners that are also data-efficient and enforce safety through probabilistic safety constraints (RAZER). This type of risk-averseness can be achieved by a simple modification of the cost function in form of uncertainty penalties in zero-order trajectory optimizers.

Furthermore, the separation of uncertainties allows us to do proper exploration via epistemic bonus which benefits generalization of the model and therefore makes it applicable to more settings. As future work, it would be of interest to see this approach applied to a proper transfer learning setting from simulations to real systems, where risk-averseness combined with exploratory behavior is crucial for efficient learning and safe operation.

References

[1] Y. Yang, K. Caluwaerts, A. Iscen, T. Zhang, J. Tan, and V. Sindhwani. Data Efficient Reinforcement Learning for Legged Robots. *Conference on Robot Learning (CoRL)*, 2019.

[2] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[3] A. S. Polydoros and L. Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[4] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[5] J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

[6] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Advances in Neural Information Processing Systems*, volume 31 of *NeurIPS*. Curran Associates, Inc., 2018.

[7] T. Wang and J. Ba. Exploring model-based planning with policy networks. In *International Conference on Learning Representations (ICLR)*, 2020.

[8] G. Williams, A. Aldrich, and E. Theodorou. Model predictive path integral control using covariance variable importance sampling. *ArXiv*, abs/1509.01149, 2015.

[9] S. C. Hora. Aleatory and epistemic uncertainty in probability elicitation with an example from hazardous waste management. *Reliability Engineering & System Safety*, 54(2):217–223, 1996. ISSN 0951-8320. doi:https://doi.org/10.1016/S0951-8320(96)00077-4. URL https://www.sciencedirect.com/science/article/pii/S0951832096000774.

[10] A. D. Kiureghian and O. Ditlevsen. Aleatory or epistemic? does it matter? *Structural Safety*, 31(2):105–112, 2009. ISSN 0167-4730. doi:https://doi.org/10.1016/j.strusafe.2008.06.020. URL https://www.sciencedirect.com/science/article/pii/S0167473008000556.

[11] I. Lenz, R. A. Knepper, and A. Saxena. Deepmpc: Learning deep latent features for model predictive control. In *Robotics: Science and Systems*. Rome, Italy, 2015.

[12] J. Fu, S. Levine, and P. Abbeel. One-shot learning of manipulation skills with online dynamics adaptation and neural network priors. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4019–4026. IEEE, 2016.

[13] Y. Gal, R. McAllister, and C. E. Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, volume 4, page 25, 2016.

[14] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[15] J. Kocijan, R. Murray-Smith, C. E. Rasmussen, and A. Girard. Gaussian process model based predictive control. In *Proceedings of the 2004 American control conference*, volume 3, pages 2214–2219. IEEE, 2004.

[16] D. Nguyen-Tuong, J. Peters, and M. Seeger. Local gaussian process regression for real time online model learning and control. In *Proceedings of the 21st International Conference on Neural Information Processing Systems*, pages 1193–1200, 2008.

[17] A. Grancharova, J. Kocijan, and T. A. Johansen. Explicit stochastic predictive control of combustion plants based on gaussian process models. *Automatica*, 44(6):1621–1631, 2008.

[18] M. P. Deisenroth, D. Fox, and C. E. Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2): 408–423, 2013.

[19] S. Kamthe and M. Deisenroth. Data-efficient reinforcement learning with probabilistic model predictive control. In *International Conference on Artificial Intelligence and Statistics*, pages 1701–1710. PMLR, 2018.

[20] C. Rasmussen and M. Kuss. Gaussian processes in reinforcement learning. In *NIPS*, 2003.

[21] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, Jan. 2006.

[22] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. Model-ensemble trust-region policy optimization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SJJinbWRZ.

[23] I. Osband, C. Blundell, A. Pritzel, and B. V. Roy. Deep exploration via bootstrapped dqn. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4033–4041, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

[24] B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6405–6416, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

[25] M. P. Deisenroth and C. E. Rasmussen. PILCO: A model-based and data-efficient approach to policy search. In *International Conference on Machine Learning*, ICML, 2011.

[26] S. Curi, F. Berkenkamp, and A. Krause. Efficient model-based reinforcement learning through optimistic policy search and planning. In *Proc. Neural Information Processing Systems (NeurIPS)*, December 2020.

[27] M. Morari and J. H. Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

[28] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou. Information theoretic mpc for model-based reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1714–1721. IEEE, 2017.

[29] R. Rubinstein and W. Davidson. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1999.

[30] Z. Botev, D. Kroese, R. Rubinstein, and P. L'Ecuyer. *Chapter 3. The Cross-Entropy Method for Optimization*, volume 31, pages 35–59. Elsevier, 2013.

[31] C. Pinneri, S. Sawant, S. Blaes, J. Achterhold, J. Stueckler, M. Rolinek, and G. Martius. Sample-efficient cross-entropy method for real-time planning. In *Conference on Robot Learning 2020*, 2020. URL https://corlconf.github.io/corl2020/paper_217/.

[32] O. Mihatsch and R. Neuneier. Risk-sensitive reinforcement learning. *Machine learning*, 49(2): 267–290, 2002.

[33] J. Garcıa and F. Fernández. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[34] S. Depeweg, J. M. Hernández-Lobato, F. Doshi-Velez, and S. Udluft. Learning and policy search in stochastic dynamical systems with bayesian neural networks. In *International Conference on Learning Representations (ICLR)*, 2017.

[35] S. Depeweg, J.-M. Hernandez-Lobato, F. Doshi-Velez, and S. Udluft. Decomposition of uncertainty in bayesian deep learning for efficient and risk-sensitive learning. In *International Conference on Machine Learning*, pages 1184–1193. PMLR, 2018.

[36] E. Arruda, M. J. Mathew, M. Kopicki, M. Mistry, M. Azad, and J. Wyatt. Uncertainty averse pushing with model predictive path integral control. *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pages 497–502, 2017.

[37] K. Lee, G. N. An, V. Zakharov, and E. A. Theodorou. Perceptual attention-based predictive control. In L. P. Kaelbling, D. Kragic, and K. Sugiura, editors, *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 220–232. PMLR, 30 Oct–01 Nov 2020. URL http://proceedings.mlr.press/v100/lee20b.html.

[38] I. Abraham, A. Handa, N. Ratliff, K. Lowrey, T. D. Murphey, and D. Fox. Model-based generalization under parameter uncertainty using path integral control. *IEEE Robotics and Automation Letters*, 5(2):2864 – 2871, April 2020.

[39] W. Clements, B.-M. Robaglia, B. V. Delft, R. B. Slaoui, and S. Toth. Estimating risk and uncertainty in deep reinforcement learning. *Workshop on Uncertainty & Robustness in Deep Learning Workshop at the International Conference on Machine Learning (ICML UDL)*, 2020.

[40] J. Zhang and P. Weng. Safe distributional reinforcement learning. *ArXiv*, abs/2102.13446, 2021.

[41] M. Wen and U. Topcu. Constrained cross-entropy method for safe reinforcement learning. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/34ffeb359a192eb8174b6854643cc046-Paper.pdf.

[42] F. Grimminger, A. Meduri, M. Khadiv, J. Viereck, M. Wüthrich, M. Naveau, V. Berenz, S. Heim, F. Widmaier, T. Flayols, et al. An open torque-controlled modular robot architecture for legged locomotion research. *IEEE Robotics and Automation Letters*, 5(2):3650–3657, 2020.