A Learnable Safety Measure

Steve Heim[1,*], Alexander von Rohr[2,3,*], Sebastian Trimpe[2] and Alexander Badri-Spröwitz[1]
[1]: Dynamic Locomotion Group, Max Planck Institute for Intelligent Systems, Germany
[2]: Intelligent Control Systems Group, Max Planck Institute for Intelligent Systems, Germany
[3]: Ingenieurgesellschaft Auto und Verkehr (IAV), Germany

Abstract: Failures are challenging for learning to control physical systems since they risk damage, time-consuming resets, and often provide little gradient information. Adding safety constraints to exploration typically requires a lot of prior knowledge and domain expertise. We present a safety measure which implicitly captures how the system dynamics relate to a set of failure states. Not only can this measure be used as a safety function, but also to directly compute the set of safe state-action pairs. Further, we show a model-free approach to learn this measure by active sampling using Gaussian processes. While safety can only be guaranteed after learning the safety measure, we show that failures can already be greatly reduced by using the estimated measure during learning.

Keywords: viability, safe learning, active sampling

1 Introduction

Learning control directly on hardware has great promise: learning would enable robots to adapt to changing environments, exploit un-modeled dynamics, as well as greatly decrease the engineering effort required to deploy a robot in the field. One of the challenges during the exploration process is that the system might visit failure states, such as a flying robot crashing or a legged robot falling over. These failure states can be costly in terms of time, damage to the robot or environment, and are often uninformative for the learning process. Unfortunately, the learning agent may not know a priori which actions lead to failure states. Furthermore, there may be actions which lead to unviable states: states which have not yet failed, but from which it is inevitable to reach a failure state in the future. Ideally, the learning agent only explores actions which keep the system within the set of viable states, also known as the *viability kernel* [1].

Although algorithms to compute conservative approximations of the viability kernel are available, they are contingent on accurate dynamics models, require substantial engineering effort and do not scale well for many types of systems [2, 3]. Alternatively, it can be useful to have a safety function which indicates how close the system is to leaving the viability kernel. Safety functions can help guide exploration to stay within the viability kernel without having to know its precise bounds. However, designing these functions is non-trivial, and faces the same issues commonly seen in designing reward functions: they are typically only approximate indicators of potential failures, require much handcrafting, and often introduce unwanted designer bias into the exploration.

We propose a model-free approach to learn a safety function, which captures the notion of viability without requiring the viability kernel to be explicitly computed. Our first contribution is a safety measure taken over the set of viable state-action pairs. Intuitively speaking, this measure describes the quantity of actions available that can avoid leaving the viability kernel. It therefore implicitly captures the structure of the systems dynamics, and how this relates to failure states, making it an effective safety function. Our second contribution is an algorithm for model-free learning of probabilistic estimates of both the measure and the viable set, using a Gaussian process (GP). On the one hand, making no model assumptions means we cannot guarantee safety until the measure has converged. We show, nonetheless, that the estimated measure can already be used during learning to reduce the number of visits to failure states significantly. On the other hand, keeping assumptions

[*] Equally contributing.

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

to a minimum allows this approach to be applied more readily to systems with arbitrary dynamics, where accurate models may be difficult to come by. This makes our approach particularly well-suited to systems that are difficult to model and where failures are costly but not critical.

1.1 Background and Related Work

Safe Learning with Viability Kernels and Back-reachable Sets. Two common model-based approaches to find safe sets are the computation of viability kernels [1, 3, 4, 5] and back-reachable sets [2, 6]. For viability, the user first defines a set of failure states; the viability kernel is then the set of states from which there exist actions, such that the system can avoid the failure set for all time. For back-reachability, a target set is defined; the back-reachable set is the set of states from which there exist actions, such that the system can reach the target set in finite time. In practice, these sets often coincide [5, 7, 8] and can be used interchangeably[2]. This depends, however, not only on the system dynamics but also on the specified failure and target sets. For example, the failure set for a walking robot may be defined as all states from which the robot cannot move its center of mass (e.g., it has fallen over and cannot recover), and the target set may be defined as reaching a specific location. If obstacles are blocking the path to the target location, the robot may be outside the back-reachable set of the target set, even though it is inside the viability kernel.

There are several algorithms used to compute back-reachable sets or viability kernels in state space; their effectiveness depends greatly on the assumptions used to model the system. For an overview, we recommend [2, 3]. To circumvent the difficulty of obtaining an accurate model from first-principles, models can also be learned from data. For example, Akametalu et al. [9] and Fisac et al. [10] learn a GP model of the dynamics of the system and disturbances, and use this to compute a conservative reachable set. As the system explores this set, the GP model is refined, and the set can (usually) be expanded. Fisac et al. [10] demonstrate their approach on quadcopter flight. They also point out the strong interdependence of safety and learning the systems true dynamics: safety guarantees are only as good as the models they are based on. In contrast, we do not model or learn the system dynamics, but a safety measure instead. We then estimate the set of viable state-actions directly from our measure, which enables model-free safe learning. Although this loses safety guarantees while learning the safety measure, it can be substantially easier to apply to complex systems.

Recently, the notion of viability has been extended to sets in state-action space. Zaytsev et al. [7] use this to directly link the reachable and viable sets. Heim and Spröwitz [11] use this to quantify the influence of system design on robustness to noisy actions, which is particularly relevant in learning control. We use the same notion of viability in state-action space, but extend the binary notion of viability (a state-action pair either belongs to the set or not) with a measure.

Bayesian Optimization and Reinforcement Learning with Safety Functions. Recently, safe Bayesian optimization (BO) using GPs has been used to apply model-free learning of controller parameters for systems with failure conditions [12, 13, 14]. In addition to modeling the controller performance, a second GP is used to model the safety of the controller parameter space. The safety model is used to restrict active sampling to parameters with a high probability of being safe. Though the controller parameters are applied to dynamical systems, safety is evaluated as purely dependent on parameter space, such that it can be considered as a static bandit problem. Thus, each sample of the parameter-space does not affect the safety of future samples. This approach is challenging to apply to situations that include non-steady-state behavior or where a set of controller parameters may be safe for some states but not others. In contrast to safe BO, we consider the more general case where safety is dependent on the current state. This emphasizes the role of the system dynamics, as they constrain the paths that can be taken through the search space. This type of problem can be modeled as a non-ergodic Markov decision process: that is, where not every state can be reached from every other state.

Turchetta et al. [15] have extended safe BO to Markov decision processes, and they demonstrate this on a non-ergodic grid-world example, where there exist states which are reachable, but from which the system cannot return. The notion of safety as ergodicity was previously formalized by Moldovan and Abbeel [16] in the general reinforcement learning context, who also point out the counter-intuitive result that more cautious exploration can often lead to faster convergence.

[2] This is the case when the viability kernel is ergodic.

In all of these approaches, it is assumed that a safety function can be sampled whenever visiting a new point in the search space (whether this is the parameter or state space). Safety is then inferred for nearby, unvisited points. The probable safety of these states can then be guaranteed using certain assumptions on the safety function, such as Lipschitz-continuity. However, this safety function is typically user-defined, and only indicative of what might cause failure. For example, Schillinger et al. [14] use the temperature of the engine at steady-state, Berkenkamp et al. [12] use a minimum performance threshold, and Turchetta et al. [15] use the ground inclination a rover needs to negotiate. Just as guarantees for model-based methods depend on the quality of the model, safe BO depends on a well-chosen safety function. In practice, the safety function is often chosen to be more conservative than strictly necessary. In contrast, our safety measure implicitly encodes the structure of the system dynamics and a definition of failure states. Furthermore, we show that the measure does not need to be known a priori, and can be learned in a model-free manner by sampling. With no model assumptions, safety guarantees can only be given once the measure has converged. Prior knowledge can, however, be introduced to reduce failures significantly.

The rest of the paper is structured as follows: In Section 2, we define all the necessary objects and introduce the safety measure. These concepts are illustrated with a toy example. In Section 3, we extend this to a probabilistic setting and present an algorithm to learn the safety measure in a model-free context. In Section 4, we show simulation results using our algorithm and point out key properties. In Section 5 we summarize our contribution and potential future work.

2 A Measure over the Viable Set

We consider systems with deterministic dynamics of the form $s' = T(s, a)$, where $s \in S$ is a state, $a \in A$ is an action, and T is the transition map of the dynamics to a new state s'. The set of failure states $S_F \subset S$ can be defined arbitrarily. For the sake of simplicity, we consider here a set of states from which there are no meaningful transitions and the system would need to be reset or replaced. We will define objects in the state-action space $Q := S \times A$. We use the shorthand $s' = T(q)$ where $q := (s, a) \in Q$. We will illustrate the defined objects on a discrete grid-world, amenable to pen-and-paper computation, and shown in Fig. 1.

Toy Model. Intuitively, the transition map in Fig. 1 can be thought of as representing a hovering spaceship affected by gravity, which is stronger near the ground. The spaceship can apply two levels of thrusters or allow itself to fall. The failure set is $S_F : \{5\}$, when the spaceship crashes.

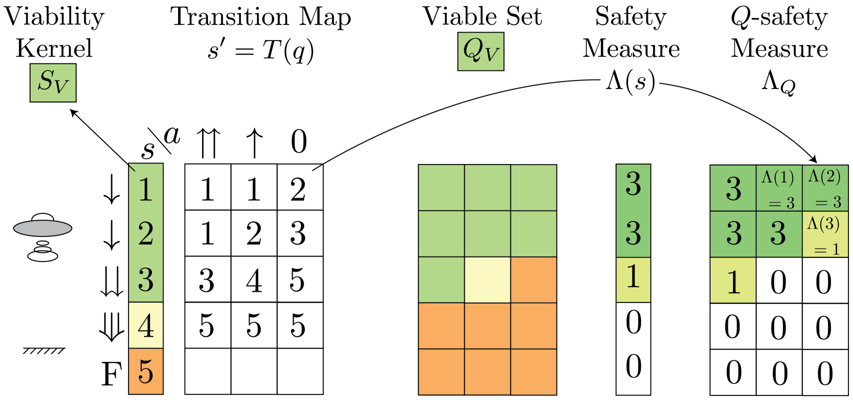


Figure 1: Shown are the transition map of our toy model, as well as each object: the viability kernel S_V, the viable set Q_V, the safety measure Λ and the Q-safety measure Λ_Q. Both S_V and Q_V are highlighted in green. We also highlight state-action pairs which result directly in failure in orange, and those that are unviable in yellow. The arrows and illustration are only to help with intuition.

We next define important mathematical objects for this work and illustrate them with the toy example. First, we define the viability kernel S_V.

Definition 1 (Viability Kernel)**.** *The* viability kernel $S_V \subset S \setminus S_F$ *is the maximal set of all states* $s \in S$*, from which there exists an action that keeps the system inside* S_V *(cf. [1, Chapter 1.1]).*

By its definition, states outside of S_V have either already failed, or will fail within finite time [1]. In the toy model, the viability kernel is $S_V = \{1, 2, 3\}$: for each of these states, there exists at least one action which can keep the spaceship from ever failing. Avoiding failure does not require ergodicity: the state $s = 3$ is viable, but it can no longer reach the other viable states. Note also that $s = 4$ is neither in the viability kernel S_V nor in the set of failure states S_F: it has not yet failed but cannot avoid reaching the failure set eventually. Next, we define the viable set in state-action space, Q_V.

Definition 2 (Viable Set)**.** *The* viable set $Q_V \subset Q$ *is the maximal set of all state-actions* q*, such that* $s' = T(q) \in S_V$*.*

By its definition, the viability kernel S_V is the projection of the viable set Q_V onto state space: for any state in S_V, the agent can sample a state-action from Q_V which maps back into itself [11]. Both S_V and Q_V are highlighted with green in Fig. 1. We can now define the safety measure Λ.

Definition 3 (Safety Measure)**.** *The* safety measure Λ *is the* n*-dimensional volume of the viable set* Q_V*. When applied to a point* $s \in S$*,* $\Lambda(s) \in \mathbb{R}_{\geq 0}$ *is the measure of the corresponding slice of* Q_V*.*

We use the Lebesgue measure for continuous spaces (assuming the sets are Lebesgue-measurable), and the counting measure for discrete spaces. Intuitively, a higher value $\Lambda(s)$ indicates that at state s, more viability-maintaining actions are available. A low value indicates that the agent should be very precise and deliberate since very few actions allow the system to avoid failure. In our toy model, for example, $\Lambda(3) = 1$ means that there is only one action which allows the system to avoid failure, this step and in the future. We can now map Λ into state-action space with the transition matrix.

Definition 4 (Q-Safety Measure)**.** *The* Q-Safety Measure Λ_Q *is defined as* $\Lambda(s')$ *where* $s' = T(q)$*. We use the shorthand* $\Lambda_Q(q) = \Lambda(s')$*.*

Next, we define safe level sets as the sets with measure $\Lambda_Q > \lambda$, where the minimum safety level λ is a non-negative scalar.

Definition 5 (Safe Level Sets)**.** *A safe level set* S_λ *is a set of states where* $\Lambda(s) > \lambda$*. A safe level set* Q_λ *is a set of state-action pairs which map into* S_λ*, such that* $\Lambda_Q(q) > \lambda$*.*

In other words, sampling from Q_λ will map the system to a state from which there is at least one action which maintains a safety level λ. Thus, the system can continue to choose actions which maintain a minimum safety level of λ indefinitely. In Fig. 1, the safe level-sets $Q_{\lambda=0}$ and $Q_{\lambda=2}$ are highlighted in different shades of green. This can be useful when certain types of disturbances are expected. We can recover the viable set from Λ_Q with $Q_V = Q_{\lambda=0}$, since viability implies $\Lambda(s) > 0$. Thus, if the safety measure Λ_Q is known, both the viable set Q_V and Λ can be obtained directly. If only Λ is known, Λ_Q can be computed directly using the transition map T. In the next section, we will use these facts to learn Λ_Q in a model-free fashion by sampling the dynamics.

3 Learning the Measure by Sampling

Given a system with an unknown transition map T and an unknown failure set S_F, our main objective is to estimate the viable set $\hat{Q}_V$ as a large, conservative approximation of the true viable set, $\hat{Q}_V \subseteq Q_V$. Since we assume an accurate dynamics model is unavailable, we directly sample the transition map T from a given state s by choosing an action a. Specifically, we begin sampling sequences from an initial state s. We then receive the tuple $(s', \textit{\textbf{failed}})$, where $s' = T(q)$ is the new state, and ***failed*** is a boolean indicating if $s' \in S_F$. The estimate $\hat{\Lambda}_Q$ can only be updated with the estimate $\hat{\Lambda}$, except when sampling a failure. The goal is to choose actions a that are informative for learning the safety measure while avoiding the failure set S_F.

To achieve this goal, we model Λ_Q, from which we compute $\hat{Q}_V$ and $\hat{\Lambda}$. The estimate $\hat{\Lambda}_Q$ can already be used during learning to avoid actions with a low estimated probability of being safe.

3.1 Convergence Properties

To examine the requirements for $\hat{\Lambda}_Q$ to converge to the true measure Λ_Q, we separately consider the viable set Q_V and its complement Q_V^c, the set of unviable and failed state-action pairs.

Theorem 1. *Under the assumption of infinite random sampling over Q, the measure $\hat{\Lambda}_Q$ converges to the correct value 0 for all $q \in Q_V^c$.*

A proof for discrete state-action spaces can be found in appendix A. Once the measure $\hat{\Lambda}_Q$ inside Q_V^c has converged, the estimate for the viable set $\hat{Q}_V$ is tightly bounded from above. Therefore, the estimated safety measure is also tightly bounded to be $\hat{\Lambda} \leq \Lambda$. We can then ensure $\hat{\Lambda}_Q$ converges to the true measure by initializing optimistically, such that initially $\hat{\Lambda} \geq \Lambda$. These two conditions of infinite sampling and optimistic starts are typical for model-free learning [17, Chapter 2.6] but are also impractical. In particular, optimistic starts encourage visits to the failure set. We will now extend Λ_Q to a probabilistic setting, and use confidence bounds over the measure to estimate $\hat{Q}_V$. Although this loses the guarantee of converging to the true Λ_Q, we show that in practice it allows us to converge to conservative subsets while reducing failures effectively.

3.2 Probabilistic Estimates: Modeling Λ_Q with Gaussian processes

A probabilistic estimate allows us to (i) include prior knowledge without an explicit model of the dynamics, and (ii) estimate the uncertainty of the safety measure for a given state-action pair q, which we will use for active sampling. When modeling q as a random variable, the distribution should only allow for positive values and also have non-zero probability mass on the point zero, to model the probability of a point being unviable. We use a normal distribution as a practical approximation, where the probability mass below zero is treated as the discrete probability for the point zero. Specifically, we use a GP [18] to model the probabilistic estimate of Λ_Q. The posterior estimate of the measure at any point in Q is normally distributed, and it includes the prior assumptions on the estimate as well as the samples $\mathcal{D} = (q_i, \hat{\Lambda}_i(s_i'))$,

$$\hat{\Lambda}_Q(q)|\mathcal{D} \sim \mathcal{N}(\mu(q), \sigma^2(q))$$

where $\hat{\Lambda}_Q(q)|\mathcal{D}$ means the estimate is conditioned on the samples, $\mathcal{N}$ is the normal distribution, μ is the posterior mean function and σ^2 the posterior variance, given by the covariance function. The prior mean and covariance function can be used to encode the prior knowledge of the measure function, such as smoothness or known safe sets.

Given $\hat{\Lambda}_Q$, the probability that a state-action pair belongs to the safe level set Q_λ can be calculated using the cumulative distribution function of the normal distribution $F_{\hat{\Lambda}_Q}$ as

$$\mathbb{P}\,[\hat{\Lambda}_Q|\mathcal{D} > \lambda] \approx 1 - F_{\hat{\Lambda}_Q}\,[\lambda]\,.$$

3.3 A Learning Algorithm for Λ_Q

We provide an approach for learning $\hat{\Lambda}_Q$ and the derived $\hat{Q}_V$ and $\hat{\Lambda}$, described in Algorithm 1. As discussed in Section 3.1, convergence requires an *optimistic* estimate of Q_V, such that the intitial estimate $\hat{\Lambda} \geq \Lambda$. Otherwise, a viable state-action pair may be incorrectly assigned the value 0. At the same time, the algorithm should use a *cautious* estimate for active sampling to reduce the probability of failing. To deal with this challenge, we use an optimistic set $\hat{Q}_{\text{opt}}$ to compute $\hat{\Lambda}$. A separate, cautious set $\hat{Q}_{\text{caut}}$ is used for active sampling. We obtain these as

$$\hat{Q}_{\text{opt}}(\gamma_{\text{opt}}) = \begin{cases} 1 & \text{if } \mathbb{P}\,[\hat{\Lambda}_Q|\mathcal{D} > 0] > \gamma_{\text{opt}} \\ 0 & \text{otherwise,} \end{cases}$$

$$\hat{Q}_{\text{caut}}(\gamma_{\text{caut}}, \lambda_{\text{caut}}) = \begin{cases} 1 & \text{if } \mathbb{P}\,[\hat{\Lambda}_Q|\mathcal{D} > \lambda_{\text{caut}}] > \gamma_{\text{caut}} \\ 0 & \text{otherwise} \end{cases}$$

by thresholding the probability with a minimum confidence $\gamma \in [0, 1]$. The algorithm has three tuning parameters: γ_{opt} governs the level of optimism in $\hat{Q}_{\text{opt}}$, and λ_{caut} and γ_{caut} govern the level of caution for active sampling. Choosing $\gamma_{\text{caut}} \geq \gamma_{\text{opt}}$ ensures that $\hat{Q}_{\text{caut}} \subseteq \hat{Q}_{\text{opt}}$, so we never purposefully explore outside the current estimate of the viability set. The algorithm samples the action from the cautious set $\hat{Q}_{\text{caut}}$ with highest variance. By actively reducing variance, the confidence in the measure is increased. Choosing actions with high variance also encourages exploration of the state space.

Algorithm 1 Learning the safety measure

```
Input: initial measure estimate $\hat{\Lambda}_Q$; thresholds $\gamma_{\text{caut}}$, $\gamma_{\text{opt}}$, and $\lambda_{\text{caut}}$; initial state $s_0$; maximum number of samples $n$.
while $i < n$ do
    Compute $\hat{\Lambda}$, $\hat{Q}_{\text{opt}}$ and $\hat{Q}_{\text{caut}}$ from $\hat{\Lambda}_Q$.
    $A_{caut} \leftarrow \forall a$ s.t. $(s_i, a) \in \hat{Q}_{\text{caut}}$.                ▷ Determine safe actions.
    if $A_{caut}$ is empty then
        $a_i \leftarrow \underset{(s_i,a)\in A}{\operatorname{argmax}} \mathbb{P}[a \in \hat{Q}_{\text{caut}}]$                ▷ Take safest action.
    else
        $a_i \leftarrow \underset{a\in A_{caut}}{\operatorname{argmax}} \sigma^2(s_i, a)$.                ▷ Explore based on variance of the GP model.
    $(s_{i+1}, \textit{failed}) \leftarrow T(s_i, a_i)$.                ▷ Sample the dynamics.
    if failed then
        Update $\mathcal{D}$ with $((s_i, a_i), 0)$ and recompute $\hat{\Lambda}_Q$
        $s_{i+1} \leftarrow$ random state from $\hat{Q}_{\text{caut}}$.                ▷ Reset if failed.
    else
        Compute $\hat{\Lambda}(s_{i+1})$ from $\hat{Q}_{\text{opt}}$
        Update $\mathcal{D}$ with $((s_i, a_i), \hat{\Lambda}(s_{i+1}))$ and recompute $\hat{\Lambda}_Q$                ▷ Update measure estimate.
```

4 Results

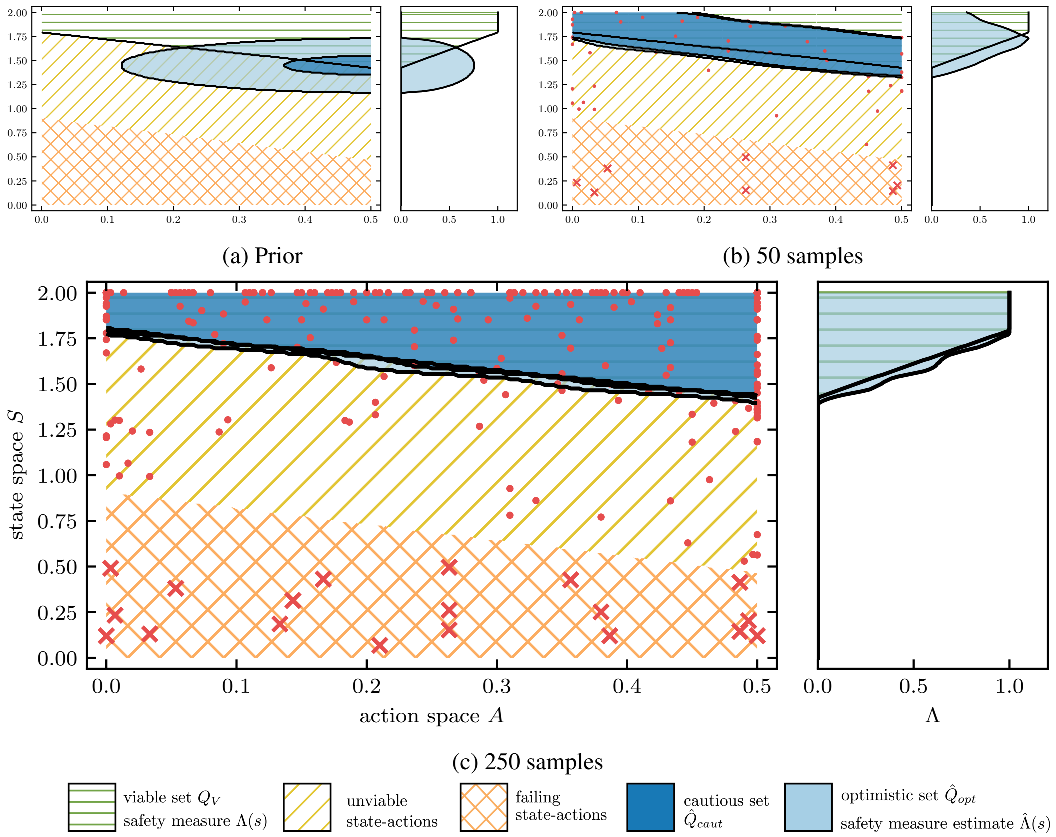
We have tested our algorithm in simulation, and provide a Python implementation using the SciPy [19] and GPy packages [20]. The code can is available in the supplementary material and online at github.com/sheim/vibly, and includes example code to reproduce the results shown here, some additional examples, and a template to implement dynamics of other systems.

We report the results of two examples, which each highlight a specific challenge: dealing with unviable state-action pairs, and dealing with complex dynamics. We also use the second example to suggest guidelines for choosing the algorithm parameters, though this will typically be system-specific. Both examples are low-dimensional, and the ground-truth is computed by brute force. This allows us to easily choose reasonable parameters for the GP model, which is otherwise a separate challenge for using Gaussian processes. In practice, choosing these parameters is highly system-dependent [21]. We use a covariance function from the Matérn family [18, Chapter 4], which has two parameters: the length scales for each input dimension and the signal variance. The length scales describe how fast the measure changes when moving away from a known state-action pair. The second parameter is the signal variance, which relates to the total variation of the measure estimate $\hat{\Lambda}_Q$. Details for the models are in the appendix.

Unviable state-action pairs. Our first example is based on the hovership spaceship from Section 2. The model has been modified with a continuous state-action space, and the dynamics have been adjusted to increase the portion of the state-action space which is unviable. The GP prior mean is purposefully initialized poorly, such that most of the initial estimate $\hat{Q}_{\text{opt}}$ lies outside the true Q_V. This example shows that the algorithm can cope with unviable states, even though the ground truth is only sampled at failure. The confidence thresholds γ_{opt}, γ_{caut} and λ are initialized to encourage rapid initial exploration, then gradually increased to speed up convergence to a safe subset of Q_V. After 250 samples, it has nearly converged to the ground-truth, with an 8% failure rate (see Fig. 2c). In both examples shown in this paper, the confidence thresholds are increased linearly with each iteration as a heuristic that helps speed up convergence and reduces failures.

Complex, unmodeled dynamics. Our second example is a simulation of the spring-loaded inverted pendulum (SLIP) model, a low-dimensional idealized model commonly used to design controllers for running robots [6, 22, 23]. Control, and therefore learning, is applied once per step-cycle, at the apex of the flight phase. The system dynamics are therefore treated as a nonlinear, discrete map with a 2-dimensional state-action space; this nonlinear map is obtained by numerically simulating the full dynamics between two apex events. The set of failures, which includes falling and reversing direction, is evaluated on the full state space of the continuous dynamics. For this system, the measure Λ_Q has a non-smooth edge on the lower part of the state space, due to an infeasibility[3] constraint (see Fig. 3). Attempting to sample infeasible state-action pairs returns a failure. At this

[3] Infeasible state-action pairs have no physical meaning and cannot exist, such as starting underground.



(a) Prior

(b) 50 samples

(c) 250 samples

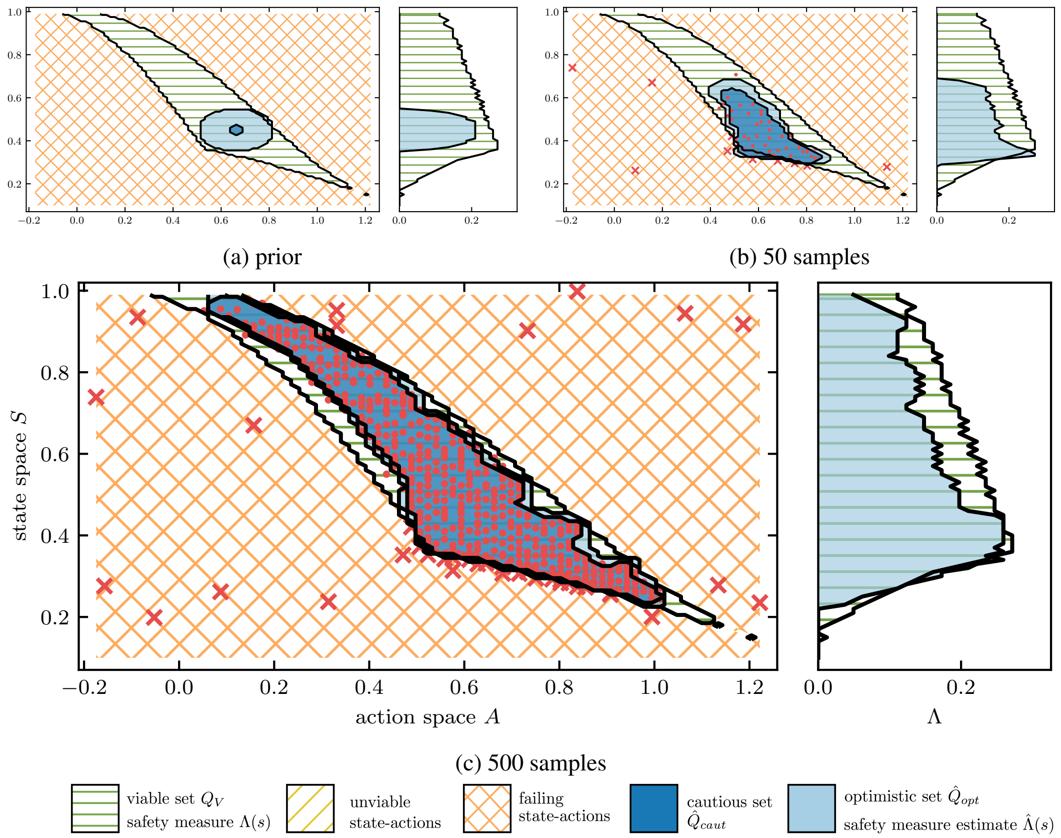
Figure 2: The progress of learning the $\hat{\Lambda}_Q$ for the spaceship model. Starting from an inaccurate prior 2a, after 50 samples 2b and after 250 samples 2c. Red dots indicate samples, red crosses samples that result in failures. After 250 samples, the optimistic set (light blue) and the measure Λ (green) are close to the ground truth. The cautious set $\hat{Q}_{\text{caut}}$ (dark blue) begins with a substantial region outside the viable set (2c), but quickly converges to the ground truth.

discontinuity, the smoothness assumptions encoded in the GP are violated. Therefore, more failures are sampled to learn the border of the viable set (see Fig. 3a). At the other borders of the viable set, where the smoothness assumption holds, the estimated sets approach the border despite sampling very few to no failures. When getting closer to the border of the viable set, the measure shrinks and the border of the set can be inferred without sampling unviable state-action pairs.

We also use this example to illustrate best practices for a realistic scenario, and the influence of different choices for the tuning parameters γ_{opt}, γ_{caut} and λ. The prior covariance function is obtained from simulations of an incorrect model, in which spring constant of the SLIP model is 20% lower. Since the covariance function encodes qualitative properties, it is reasonable to use a low fidelity simulation to obtain these GP parameters. The GP prior mean is typically more sensitive to simulation inaccuracies. It is therefore chosen around a known operating point, which we assume can be determined with conventional means without the need for a full model. Ideal operating points will feature a stable equilibrium-point or slow divergence from the operating point. Thus, the learning system can drive down variance locally before exploring more distant states, and the confidence bound γ_{opt} and γ_{caut} can be initialized more aggressively. We initialize the operating point of the SLIP model near a known limit-cycle of the running model. Although the prior is very conservative (see Fig. 3), the algorithm converges to a conservative yet nearly maximal approximation of the viable set Q_V, with a failure rate of 8% after 500 samples.

5 Discussion and Outlook

The first contribution of this paper is a safety measure taken over the set of viable state-action pairs. While this measure is useful in itself, computing viable sets relies on accurate models and is



(a) prior

(b) 50 samples

(c) 500 samples

Figure 3: Learning the viable set and the corresponding measure $\hat{\Lambda}$ for a SLIP model. Starting from a conservative prior (3a), after 50 samples (3b) and after 500 samples (3c). The color coding is as in Fig. 2. A non-smooth infeasibility constraint bounds the bottom edge of the viable set. This violates the smoothness assumption of the GP, and requires many samples to learn accurately. The other edges are learned fairly accurately without many failures sampled. Actions close to the left edge are avoided, as they bring the system to states with low safety measure.

often intractable for systems with complex, high-dimensional dynamics. Our second contribution is a probabilistic, model-free approach to learn this measure and a safe set of state-action pairs using GPs. On the one hand, this makes it applicable to a variety of systems. On the other hand, making almost no assumptions means there are no hard guarantees for avoiding failure, even with a reasonable prior. This approach is therefore appropriate for systems which are difficult to model and where failures are costly but not critical, such as robots with soft or compliant components [24, 25] and small to mid-sized legged robots [21, 26, 27].

An issue with our current algorithm is that old samples contain old, potentially incorrect estimates of the measure, which can interfere with newer samples. A principled approach to only keep informative samples would improve the estimate and reduce computation costs. As with most other learning approaches, scaling to higher dimensions is a key challenge. An exploration strategy with an information-theoretic approach, especially with a heteroscedastic model (with state-dependent uncertainty), should improve both accuracy as well as sample-efficiency. In practice, it may be desirable to balance information gain of the safety measure and performance. How to balance this in a principled manner is an open question. We believe that leveraging a dynamics model will be key to scaling. How to map assumptions of the dynamics to the safety model requires further investigation. In addition, sample-efficiency might be greatly improved by updating all state-action pairs that are close in a dynamical sense instead of a Euclidean sense. How to obtain such a metric of closeness in state-action space is a problem we find is both challenging and has significant potential.

Acknowledgments

We thank Dominik Baumann, Matthias Neumann-Brosig and Friedrich Solowjow for insightful discussions during the project and while preparing the manuscript, as well as the anonymous reviewers for constructive and thorough feedback. We thank IMPRS-IS for the academic development of Steve Heim and Alexander von Rohr. This work was partly funded through the Cyber Valley Initiative and the Max Planck Society.

References

[1] J.-P. Aubin, A. M. Bayen, and P. Saint-Pierre. *Viability theory: new directions*. Springer Science & Business Media, 2011.

[2] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin. Hamilton-jacobi reachability: A brief overview and recent advances. In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, pages 2242–2253, Dec 2017. doi:10.1109/CDC.2017.8263977.

[3] A. Liniger and J. Lygeros. Real-time control for autonomous racing based on viability theory. *IEEE Transactions on Control Systems Technology*, 27(2):464–478, March 2019. doi:10.1109/TCST.2017.2772903.

[4] P. Wieber. Viability and predictive control for safe locomotion. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1103–1108, Sep. 2008. doi:10.1109/IROS.2008.4651022.

[5] A. M. Bayen, E. Crück, and C. J. Tomlin. Guaranteed overapproximations of unsafe sets for continuous and hybrid systems: Solving the hamilton-jacobi equation using viability techniques. In *Hybrid Systems: Computation and Control*, pages 90–104. Springer Berlin Heidelberg, 2002.

[6] G. Piovan and K. Byl. Reachability-based control for the active slip model. *The International Journal of Robotics Research*, 34(3):270–287, 2015. doi:10.1177/0278364914552112.

[7] P. Zaytsev, W. Wolfslag, and A. Ruina. The boundaries of walking stability: Viability and controllability of simple models. *IEEE Transactions on Robotics*, 34(2):336–352, April 2018. doi:10.1109/TRO.2017.2782818.

[8] S. Kaynama, J. Maidens, M. Oishi, I. M. Mitchell, and G. A. Dumont. Computing the viability kernel using maximal reachable sets. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '12, pages 55–64, New York, NY, USA, 2012. ACM. doi:10.1145/2185632.2185644.

[9] 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. In *53rd IEEE Conference on Decision and Control*, pages 1424–1431, Dec 2014. doi:10.1109/CDC.2014.7039601.

[10] J. F. Fisac, A. K. Akametalu, M. N. Zeilinger, S. Kaynama, J. Gillula, and C. J. Tomlin. A general safety framework for learning-based control in uncertain robotic systems. *IEEE Transactions on Automatic Control*, 64(7):2737–2752, July 2019. doi:10.1109/TAC.2018.2876389.

[11] S. Heim and A. Spröwitz. Beyond basins of attraction: Quantifying robustness of natural dynamics. *IEEE Transactions on Robotics*, 35(4):939–952, Aug 2019. doi:10.1109/TRO.2019.2910739.

[12] F. Berkenkamp, A. P. Schoellig, and A. Krause. Safe controller optimization for quadrotors with gaussian processes. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 491–496, May 2016. doi:10.1109/ICRA.2016.7487170.

[13] J. Schreiter, D. Nguyen-Tuong, M. Eberts, B. Bischoff, H. Markert, and M. Toussaint. Safe exploration for active learning with gaussian processes. In *Machine Learning and Knowledge Discovery in Databases*, pages 133–149. Springer International Publishing, 2015.

[14] M. Schillinger, B. Ortelt, B. Hartmann, J. Schreiter, M. Meister, D. Nguyen-Tuong, and O. Nelles. Safe active learning of a high pressure fuel supply system. In *Proceedings of The 9th EUROSIM Congress on Modelling and Simulation*, pages 286–292, 2016.

[15] M. Turchetta, F. Berkenkamp, and A. Krause. Safe exploration in finite markov decision processes with gaussian processes. In *Advances in Neural Information Processing Systems 29*, pages 4312–4320. 2016.

[16] T. M. Moldovan and P. Abbeel. Safe exploration in markov decision processes. In J. Langford and J. Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ICML '12, pages 1711–1718, July 2012.

[17] R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction, 2nd Edition*, volume 1. MIT press Cambridge, 2019. URL `http://incompleteideas.net/book/the-book-2nd.html`.

[18] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[19] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. URL `http://www.scipy.org/`. [Online; accessed 23.09.2019].

[20] GPy. GPy: A gaussian process framework in python. `http://github.com/SheffieldML/GPy`, since 2012.

[21] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson. Bayesian optimization using domain knowledge on the atrias biped. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1771–1778, May 2018. doi:10.1109/ICRA.2018.8461237.

[22] P. M. Wensing and D. E. Orin. High-speed humanoid running through control with a 3d-slip model. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5134–5140, Nov 2013. doi:10.1109/IROS.2013.6697099.

[23] C. Hubicki, J. Grimes, M. Jones, D. Renjewski, A. Spröwitz, A. Abate, and J. Hurst. Atrias: Design and validation of a tether-free 3d-capable spring-mass bipedal robot. *The International Journal of Robotics Research (IJRR)*, 35(12):1497–1521, 2016. doi:10.1177/0278364916648388.

[24] D. Surovik, K. Wang, M. Vespignani, J. Bruce, and K. E. Bekris. Adaptive tensegrity locomotion: Controlling a compliant icosahedron with symmetry-reduced reinforcement learning. *International Journal of Robotics Research (IJRR)*, 2019.

[25] D. Büchler, R. Calandra, B. Schölkopf, and J. Peters. Control of musculoskeletal systems using learned dynamics models. *IEEE Robotics and Automation Letters*, 3(4):3161–3168, Oct 2018. doi:10.1109/LRA.2018.2849601.

[26] Z. Xie, G. Berseth, P. Clary, J. Hurst, and M. van de Panne. Feedback control for cassie with deep reinforcement learning. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1241–1246, Oct 2018. doi:10.1109/IROS.2018.8593722.

[27] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26), 2019. doi:10.1126/scirobotics.aau5872.

A Convergence of the measure estimate for unviable state-action pairs

We will show here the convergence result for systems with discrete time and finite, discrete state-action spaces. Specifically we show that, under the assumption of inifinite sampling, the estimated measure $\hat{\Lambda}_Q$ for the unviable state-action pairs $q \in Q_V^c$ converges to the true value of 0 when following the updates

$$\hat{\Lambda}_Q(q) = \begin{cases} 0 & \text{if } \textbf{\textit{failed}} \\ \hat{\Lambda}(s') & \text{else.} \end{cases} \tag{1}$$

A direct consequence is that the estimated measure $\hat{\Lambda}$ for all unviable states also converges to the true value of 0. This provides an upper bound for the measure. We proceed to the theorem.

Theorem 1. *Under the assumption of infinite random sampling over Q, the measure $\hat{\Lambda}_Q$ converges to the correct value 0 for all $q \in Q_V^c$.*

We define $S_U = (S_V \cup S_F)^c$ as the set of unviable states, and $Q_U \coloneqq S_U \times A$. We also define the operator $\mathrm{len}(s)$, which returns the integer length of the longest trajectory starting from the state s. We begin by showing that this theorem holds for all $q \in Q_U$, which ensures that the estimated measure converges to 0 for all $s \in S_U$. Consider all possible trajectories starting from any state $s \in S_U$. By the definition of viability, they are all *acyclic*, meaning no state is ever visited more than once. They therefore all end in S_F within finite time.

Lemma 1. *The longest trajectory starting from any state $s \in S_U$ has length $\mathrm{len}(s) \leq n$, where n is the number of states in S_U.*

This can be proven by contradiction. Let us assume that the longest trajectory starting in S_U has length $n_{longest} > n$. We take a sub-trajectory of length n; due to the acyclicity condition, this trajectory has has visited n unique states, and therefore has visited all states in S_U. It therefore cannot be lengthened without breaking the acyclicity condition, contradicting our assumption.

Lemma 2. *For every $i = 1, \dots, n_{longest}$, there exists at least one state $s \in S_U$, for which the longest trajectory beginning from that state has length i.*

Again, this can be shown by contradiction. Let us assume there are no states with $\mathrm{len}(s) = 1$. Then take the longest trajectory starting from any $s \in S_U$, and proceed to the last state in the trajectory. By our assumption, $\mathrm{len}(s) > 1$, which implies that there is at least one action available from this state which would avoid failure and therefore increase the length of the trajectory by at least 1, contradicting the our previous statement. This reasoning can be extended to all other i up to $n_{longest}$ by inserting shorter states in the failure set S_F and repeating this process.

Proof. Now it is clear that sampling any q from a state with $\mathrm{len}(s) = 1$ will immediately transition to S_F, and therefore will be updated with the ground truth as per 1. Once each of such q has been sampled once, the measure estimate will be 0 for any state with $\mathrm{len}(s) = 1$. At this point, sampling any q from a state with $\mathrm{len}(s) = 2$ will be updated with 0, and so on until all $s \in S_U$ have converged to 0.

We can now turn our attention to the remaining $q \in Q_V^c$. By definition, these are state-action pairs that transition to S_U in a single step. Therefore, as soon as the estimated measure for all $s \in S_U$ has converged to 0, these will also be updated correctly with $\hat{\Lambda}(s) = 0$. □

B Dynamics of Simulated Examples

We include here the details of the dynamics for the systems used in Section 4. The implementation in Python is available in the supplementary material. We also include in the supplementary material code to compute the true viable sets, although this is only computationally tractable for small systems.

B.1 Hovering Spaceship

This example is a hovering spaceship loosely based on the toy example in Section 2, with a continuous state-action space. The spaceship has a single state, a vertical position, and is affected by nonlinear gravity. The non-constant gravity has been adjusted to accentuate the issue of nonviable which have not yet failed. The dynamics are:

$$\dot{s} = g_0 + \tanh(0.75\, s) g - a \tag{2}$$

where s is the height, g_0 is a baseline gravitational acceleration, and g is a coefficient for the gravity which increases with state. The spaceship can counteract gravity with the action $0 \leq a \leq a_{max}$. The failure set is defined as $s \geq s_{max}$. We also model a control frequency ω, such that the spaceship can only choose a new thrust a once every $\frac{1}{\omega}$ seconds. This control delay further accentuates the unviable states. The reader is encouraged to adjust these parameters in the code to see how this affects both the viable sets, and the learning of the safety measure.
The parameters used to generate the graph in the paper are:

base gravitational acceleration	g_0 :	0.1
gravitational acceleration	g :	1
max thrust	a_{max} :	0.5
ground height	s_{max}:	2
control frequency	ω :	1

B.2 Spring Loaded Inverted Pendulum

The spring-loaded inverted pendulum is a common model for understanding running dynamics, both in biomechanics and robotics. The body is represented by a point-mass, and a massless spring represents the leg. It has hybrid dynamics, meaning the governing equations of motion switch between different phases, and cyclic orbits. We begin each cycle at the apex, the highest point during flight phase, when vertical velocity is zero. Each step cycle has 3 phases: A flight phase terminating with a touchdown event, a stance phase terminating with a liftoff event, and a second flight phase terminating with an apex event.
Between two apex events, we integrate the full state $[x, y, \dot{x}, \dot{y}]^\intercal$, where x and y are the horizontal and vertical positions of the point-mass. During each flight phase, the dynamics are

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \end{bmatrix},$$

where g is the gravitational acceleration. During stance phase, the dynamics are

$$\begin{aligned} \begin{bmatrix} \ddot{x} \\ \ddot{y} \end{bmatrix} &= \frac{k\,(l_0 - l)}{m} \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} - \begin{bmatrix} 0 \\ g \end{bmatrix} \\ \theta &= \arctan 2\left(\frac{y}{x}\right) - \frac{\pi}{2} \\ l &= \sqrt{(x^2 + y^2)}, \end{aligned}$$

where k is the spring stiffness, l_0 is the spring resting length, and m is the mass. For concise notation, the reference frame is centered on the foot during stance. In the implementation, the foot position is also tracked and accounted for. The events are detected with

$$\begin{aligned} \text{touchdown: } & l = l_0 \\ \text{liftoff: } & \theta = \arctan 2\left(\frac{y}{x}\right) - \frac{\pi}{2} \\ \text{apex: } & \dot{y} = 0. \end{aligned}$$

For simplicity, we can examine the state once per step cycle, on the so-called Poincaré section, at the apex of flight. At apex, all the potential energy at apex is contained in the height of the point-mass, and the kinetic energy in the forward velocity (the vertical velocity is zero by the definition of apex). Since the system is energy-conservative, we can use the potential energy normalized by total energy as our single state. We thus use as state $s = \frac{E_{\text{pot}}}{E_{\text{pot}}+E_{\text{kin}}} = \frac{gy}{\frac{\dot{x}^2}{2}+gy}$, where E_{pot} and E_{kin} are the potential and kinetic energy, respectively. For our simulations, we also define a single action: the landing angle of attack of the leg, $a = \alpha$, which is instantly reset to the desired angle at each apex.

For the figures in the paper, we used the following parameters, which are similar to human averages:

gravitational acceleration	$g:$	9.81	$[m/s^2]$
body mass	$m:$	80	$[kg]$
spring stiffness	$k:$	8200	$[N/m]$
spring resting length	l_0:	1	$[m]$