Bayesian Safety Validation for Black-Box Systems

Robert J. Moss* and Mykel J. Kochenderfer†
Stanford University, Stanford, CA, 94305

Maxime Gariel‡ and Arthur Dubois§
Xwing, San Francisco, CA, 94102

Accurately estimating the probability of failure for safety-critical systems is important for certification. Estimation is often challenging due to high-dimensional input spaces, dangerous test scenarios, and computationally expensive simulators; thus, efficient estimation techniques are important to study. This work reframes the problem of black-box safety validation as a Bayesian optimization problem and introduces an algorithm, *Bayesian safety validation*, that iteratively fits a probabilistic surrogate model to efficiently predict failures. The algorithm is designed to search for failures, compute the most-likely failure, and estimate the failure probability over an operating domain using importance sampling. We introduce a set of three acquisition functions that focus on reducing uncertainty by covering the design space, optimizing the analytically derived failure boundaries, and sampling the predicted failure regions. Mainly concerned with systems that only output a binary indication of failure, we show that our method also works well in cases where more output information is available. Results show that Bayesian safety validation achieves a better estimate of the probability of failure using orders of magnitude fewer samples and performs well across various safety validation metrics. We demonstrate the algorithm on three test problems with access to ground truth and on a real-world safety-critical subsystem common in autonomous flight: a neural network-based runway detection system. This work is open sourced[1] and currently being used to supplement the FAA certification process of the machine learning components for an autonomous cargo aircraft.

Nomenclature

f	=	black-box system under test
$\hat{f}$	=	predicted mean of probabilistic surrogate model
$\hat{\sigma}$	=	predicted standard deviation of probabilistic surrogate model
$\hat{g}$	=	surrogate model binary failure classification
p	=	operational likelihood model (target/nominal distribution)
q	=	importance sampling proposal distribution
$\mathbf{x}$	=	input vector from the design space

I. Introduction

CERTIFYING safety-critical autonomous systems is a crucial step for their safe deployment in aviation. Examples of safety-critical systems include those for detect and avoid [1, 2], collision avoidance [3], runway detection [4], and auto-land [5]. One way to provide a quantitative measure of safety is to estimate the probability of system failure. The process of estimating the probability of failure can highlight areas of weakness in the system (by uncovering failures) and can show how well the system performs in their operating environments. The rarity of failures makes it challenging to accurately estimate failure probability especially when using computationally expensive simulators [6]. Therefore, it is important to efficiently sample the design space when searching for failures (using a minimum set of inputs) and

*PhD Student, Department of Computer Science, Stanford, CA, AIAA Student Member, mossr@cs.stanford.edu
†Associate Professor, Department of Aeronautics and Astronautics, Stanford, CA, AIAA Associate Fellow, mykel@stanford.edu
‡Chief Technology Officer, San Francisco, CA, maxime@xwing.com
§Director of Engineering, San Francisco, CA, arthur.dubois@xwing.com
[1] https://github.com/sisl/BayesianSafetyValidation.jl

to maximize a measure of confidence in the resulting failure probability estimate. A standard approach to estimating this rare-event probability involves Monte Carlo (MC) sampling to generate a set of system inputs from a likelihood model of the operating environment. Estimating this rare-event probability through Monte Carlo sampling can be computationally expensive and usually requires a large number of samples to minimize variance of the estimate [6]. A variance-reduction technique to more efficiently estimate the failure probability uses *importance sampling* [7, 8] to instead draw samples from a different distribution, called the proposal, and then re-weight the expectation based on the likelihood ratio between the operational model and the proposal (details left for section III.A). Importance sampling is especially useful in the safety-critical case due to the unbiased failure probability estimate [8].

Bayesian optimization algorithms such as the cross-entropy method (CEM) [9, 10] have been adapted to the problem of rare-event estimation through a multi-level procedure [6, 11], but rely on a real-valued system output with a defined failure threshold to adaptively narrow the search. Arief et al. [12] proposed a deep importance sampling approach for rare-event estimation of black-box systems (Deep-PrAE) but rely on similar assumptions regarding the output of the system. In our problem, the system under test outputs a binary value indicating failure and thus cannot effectively use these methods. Population-based methods, like population Monte Carlo (PMC) [13] and optimized population Monte Carlo (O-PMC) [14], have no assumption on the system outputs and use adaptive importance sampling [15] to iteratively estimate the optimal proposal distribution. The PMC algorithms use self-normalized importance sampling (SNIS) to estimate the probability in question, which can be slightly biased [8]. Population-based approaches often require a large number of system evaluations to adequately converge (see Luengo et al. [16] for a comprehensive survey of Monte Carlo estimation algorithms). Vazquez and Bect [17] and Wang et al. [18] consider the problem of failure probability estimation when dealing with computationally expensive systems. They fit a Gaussian process surrogate model to the underlying real-valued system (i.e., not the system output indication of failure) and then estimate the failure probability over this surrogate; similar to work from Renganathan et al. [19] for the multifidelity case. Those methods may not work on binary-valued systems or scale to complex systems such as image-based neural networks. He and Schumann [20] propose a framework for analyzing safety-critical deep neural networks using Bayesian statistics to iteratively fit a decision boundary from a predefined dictionary of shapes. They use a boundary acquisition function that is based on expected improvement [21], requiring a definition of an ϵ-threshold around the predicted boundaries at $0.5 \pm \epsilon$. Our proposed approach constructs the probabilistic surrogate model so that a failure boundary can be analytically derived.

With the goal of sample efficiency, this work reformulates the safety validation problem [22] as a Bayesian optimization problem [23–25] and introduces a set of acquisition functions each with their own safety validation objective. Applying a Bayesian approach allows us to fit a probabilistic surrogate model to a minimal set of design points evaluated from the true system and then estimate failure probability using importance sampling on the inexpensive surrogate. As a real-world case study, we use the proposed algorithm to estimate the probability of failure for a neural network-based runway detection system where the design space consists of the glide slope angle and the distance to runway. The parametric design space is used to generate an input image of a runway in simulation, conditioned on the knowledge that the aircraft is in an approach, and the output is a binary value of failure (i.e., a misdetection).

The goals of this work are to: (1) estimate the probability of failure for a black-box safety-critical subsystem, (2) focus on sample efficiency using the minimal number of data points, (3) find realistic cases using a model of the environment the system will be operating in, weighting the failures based on their operational likelihood, (4) characterize the entire set of failure regions to identify model weaknesses for further development, and (5) ensure the entire design space is adequately covered. The proposed *Bayesian safety validation* (BSV) algorithm can be applied to general black-box systems to find failures, determine the most-likely failure, and estimate the overall failure probability. An open-source Julia framework* was developed to extend this work to other black-box systems and reproduce the results in this paper.

II. Background

To understand the methods developed in this work, we will provide the necessary background by first introducing the problem of safety validation and then will briefly discuss Gaussian processes and their use in Bayesian optimization.

A. Safety Validation

Safety validation has three primary tasks [22] shown in fig. 1. The first task, *falsification*, is the process of finding any input that results in system failure. The second task, *most-likely failure analysis*, tries to find the failures with maximum likelihood. And the third task, *failure probability estimation*, estimates the probability that a failure will occur.

*`https://github.com/sisl/BayesianSafetyValidation.jl`

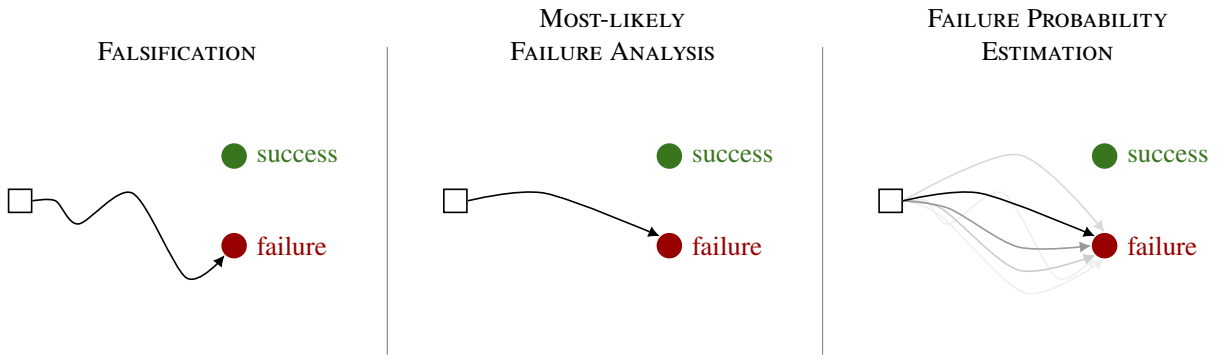


Fig. 1 The three tasks of safety validation.

In focusing on failure probability estimation, we can achieve all three safety validation tasks. This is because when we estimate the probability of failure, we generate a distribution of failures. Thus, we achieve falsification by finding failures in the process of constructing the distribution and can easily compute the most-likely failure by maximizing the input likelihood across the distribution. Motivated to achieve all three safety validation tasks, this work develops an efficient approach to estimate probability of failure for black-box systems. For a survey of existing black-box safety validation algorithms, including falsification and most-likely failure analysis, we refer to Corso et al. [22].

In the case of *black-box safety validation*, we treat the system f as a "black box" and attempt to perform the three tasks described above. The black-box assumption means that the only way to interact with the system is by passing inputs $\mathbf{x}$ and observing outputs $y = f(\mathbf{x})$. This is in contrast to *white-box validation* which requires information about the internals of the system to prove properties of safety [26–28]. In choosing to perform black-box validation, we can apply the developed methods to more general systems, particularly to systems with neural network components. Although recent work has focused on verifying deep neural networks [29, 30], scaling to large networks remains a challenge.

B. Bayesian Optimization and Probabilistic Surrogate Models

The basic optimization problem [24] is to maximize (or minimize) a real-valued function $f : \mathbb{R}^n \to \mathbb{R}$ subject to $\mathbf{x}$ lying in the design space $\mathcal{X} \subseteq \mathbb{R}^n$:

$$\begin{aligned} \underset{\mathbf{x}}{\text{maximize}} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & \mathbf{x} \in \mathcal{X} \end{aligned} \tag{1}$$

Bayesian optimization is a black-box approach to globally optimize the objective f without requiring any information about internals of the function, e.g., no requirement on gradient information [25, 31]. The main idea is to iteratively fit a *probabilistic surrogate model*—such as a Gaussian process [32]—to evaluation points of the true objective function and then propose new design points to evaluate based on the information and uncertainty quantified in the surrogate. Bayesian optimization is especially useful when f is computationally expensive to evaluate and the surrogate $\hat{f}$ is fast to evaluate in comparison [31]. Figure 2 illustrates a Bayesian optimization example where the next sampled design point $\mathbf{x}'$ (shown as a green triangle) maximizes the *upper-confidence bound* (UCB) acquisition function [24]:

$$\mathbf{x}' = \underset{\mathbf{x} \in \mathcal{X}}{\arg\max}\, \hat{f}(\mathbf{x}) + \lambda \hat{\sigma}(\mathbf{x}) \tag{2}$$

where $\hat{f}$ is the mean of the surrogate model, $\hat{\sigma}$ is the standard deviation, and $\lambda \geq 0$ controls the trade-off between exploration (based on the uncertainty) and exploitation (based on the mean). Using a probabilistic approach when fitting the surrogate model allows us to use uncertainty in the underlying objective when acquiring subsequent samples.

1. Gaussian Processes

One method for constructing a probabilistic surrogate model is to use a *Gaussian process* (GP) [32]. Given true observations from the objective function, a GP is defined as a distribution over possible underlying functions that describe the observations [24] (illustrated in fig. 2 as purple dashed lines showing five functions sampled from the GP).

Given the set of n inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ and n true observations $\mathbf{y} = [y_1, \ldots, y_n]$ where $y_i = f(\mathbf{x}_i)$, a Gaussian process is parameterized by a mean function $\mathbf{m}(\mathbf{X})$, generally set to the zero-mean function $\mathbf{m}(\mathbf{X})_i = m(\mathbf{x}_i) = 0$ if no prior information is given, and a kernel function $\mathbf{K}(\mathbf{X}, \mathbf{X})$ that captures the correlations between data points as covariances. The output of the kernel function is an $n \times n$ matrix where the element $\mathbf{K}(\mathbf{X}, \mathbf{X})_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$. The kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ may be selected based on spatial information about the relationship of neighboring data points in design space (e.g., if the relationship is smooth, then one can use a squared exponential kernel [24]). In this work, we use the isotropic Matérn 1/2 kernel [33] with length scale $\ell = \exp(-1/10)$ and signal standard deviation $s_\sigma = \exp(-1/10)$:

$$k(\mathbf{x}_i, \mathbf{x}_j) = s_\sigma^2 \exp\left(-|\mathbf{x}_i - \mathbf{x}_j|/\ell\right) \tag{3}$$

The choice of kernel and its parameters can be separately optimized depending on the problem (see Williams and Rasmussen [32]), where the kernel was chosen for this work based on the characteristic that the Matérn kernel can capture more variation in neighboring values [32]. Using the mean function and kernel parameterization and conditioning on the true observations $\mathbf{y}$, the GP produces samples for new points $\mathbf{X}'$ of the function it is trying to estimate as

$$\hat{\mathbf{y}} \mid \mathbf{y} \sim \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{X}, \mathbf{X}', \mathbf{y}), \boldsymbol{\Sigma}(\mathbf{X}, \mathbf{X}')\big) \tag{4}$$

where

$$\boldsymbol{\mu}(\mathbf{X}, \mathbf{X}', \mathbf{y}) = \mathbf{m}(\mathbf{X}') + \mathbf{K}(\mathbf{X}', \mathbf{X})\mathbf{K}(\mathbf{X}, \mathbf{X})^{-1}(\mathbf{y} - \mathbf{m}(\mathbf{X})) \tag{5}$$

$$\boldsymbol{\Sigma}(\mathbf{X}, \mathbf{X}') = \mathbf{K}(\mathbf{X}', \mathbf{X}') - \mathbf{K}(\mathbf{X}', \mathbf{X})\mathbf{K}(\mathbf{X}, \mathbf{X})^{-1}\mathbf{K}(\mathbf{X}, \mathbf{X}'). \tag{6}$$

Across the domain $\mathbf{X}'$, these estimate $\hat{\mathbf{y}}$ can now be used as surrogates for the true function $f(\mathbf{x}')$ for $\mathbf{x}' \in \mathbf{X}'$.

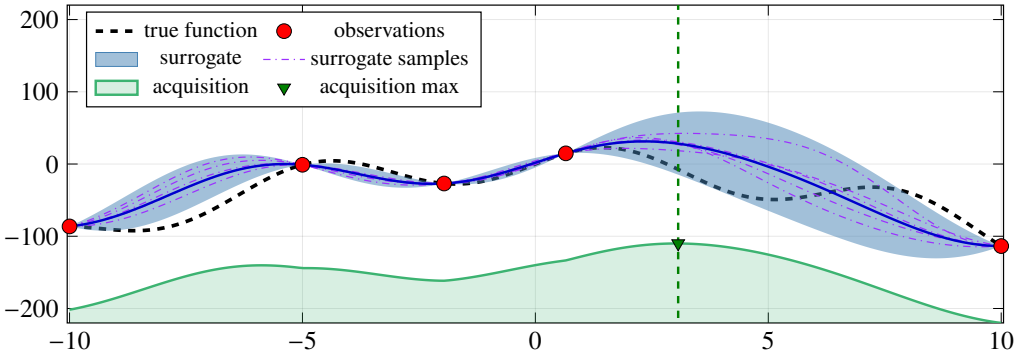


Fig. 2 An example maximization problem using Gaussian process Bayesian optimization with UCB exploration.

Predicting a probability with a Gaussian process. Because our system f returns discrete values in $\{0, 1\}$ and we want to predict a real-valued probability in $[0, 1]$, we consider this a binary classification problem [34, 35]. We construct the GP to predict the logits $\hat{\mathbf{z}}$ (which we naturally define with zero mean to indicate no prior knowledge about failures) and then apply the logistic function (i.e., inverse logit or sigmoid) to get the predictions $\hat{\mathbf{y}}$:

$$\hat{\mathbf{z}} \mid \operatorname{logit}(\mathbf{y}) \sim \mathcal{N}\big(\boldsymbol{\mu}(\mathbf{X}, \mathbf{X}', \operatorname{logit}(\mathbf{y})), \boldsymbol{\Sigma}(\mathbf{X}, \mathbf{X}')\big) \tag{7}$$

$$\operatorname{logit}(y_i) = \log\left(\frac{\phi(y_i)}{1 - \phi(y_i)}\right)/s \tag{8}$$

$$\hat{\mathbf{y}} = \phi^{-1}\Big(\operatorname{logit}^{-1}(\hat{\mathbf{z}})\Big) = \phi^{-1}\left(\frac{1}{1 + \exp(-s\hat{\mathbf{z}})}\right) \tag{9}$$

where $\phi(y_i) = y_i(1 - \epsilon) + (1 - y_i)\epsilon$ and $\phi^{-1}(\hat{y}_i) = (\hat{y}_i - \epsilon)/(1 - 2\epsilon)$ to ensure well defined logits and s controls the steepness of the sigmoid curve. We set $\epsilon = 10^{-5}$ and $s = 10^{-1}$ for our experiments. This construction can still be used even if f already outputs values in $[0, 1]$ instead of binary indicators; the GP will fit directly to the provided failure probability of each point. When the output is binary, applying the logit transformations ensure that the prediction lies in $[0, 1]$ and can be interpreted probabilistically. Other approaches to predict a probability using a Gaussian process explore the case where f is bounded and can be modeled as a Beta distribution [36]. We chose the logit approach which allows us to analytically compute failure boundaries.

III. Problem Formulation

We reframe the black-box safety validation problem as a Bayesian optimization problem and use a Gaussian process surrogate model to predict failures. Bayesian optimization is a natural approach to optimize some function $f : \mathbb{R}^n \to \mathbb{R}$, e.g., a black-box system. But our problem uses a function $f : \mathbb{R}^n \to \mathbb{B}$, where $\mathbb{B}$ represents the Boolean domain (returning `true` for failures and `false` for non-failures, which can also be interpreted as 1 and 0, respectively). Instead of maximizing or minimizing f, we reframe the problem to find failure regions through exploration, refine failure boundaries, and refine likely failure regions through sampling the theoretically optimal failure distribution [8, 37, 38]. We introduce a set of three acquisition functions that accomplish these objectives and call the acquisition procedure *failure search and refinement* (FSAR), shown together in algorithm 1. Although we are primarily interested in the more restrictive case where f outputs a Boolean, we define the procedure to also work when f outputs a probabilistic value of failure (demonstrated in section IV.F). Throughout, we use the fact that the surrogate model provides a probabilistic interpretation of the failure predictions regardless of the type of system outputs, namely Boolean or probabilistic.

Uncertainty exploration. To find failures and provide coverage of the design space $\mathcal{X}$, we want to explore areas with high uncertainty. The first proposed acquisition function is a simple search over the uncertainty provided by the Gaussian process $\hat{\sigma}(\mathbf{x})$ to find the points $\mathbf{x} \in \mathcal{X}$ with maximal uncertainty:

$$\mathbf{x}_1' = \arg\max_{\mathbf{x}\in\mathcal{X}} \hat{\sigma}(\mathbf{x}) \tag{10}$$

This will ensure that the design space $\mathcal{X}$ is fully explored in the limit [32] (noting that in practice the limiting factor is the $O(n^3)$ time for the Gaussian process to fit n data points due to the $n \times n$ matrix inversion [39]).

Boundary refinement. To better characterize the areas of all failure regions, we want to refine the known failure boundaries to tighten them as much as possible. Because our surrogate $\hat{f}(\mathbf{x})$ is modeled as a logistic function (shown in eq. (9)), we can take the derivative and get the analytical form as

$$\mu'(\mathbf{x}) = \hat{f}(\mathbf{x})(1 - \hat{f}(\mathbf{x})) \tag{11}$$

where $\mu'(\mathbf{x})$ is maximal when $\hat{f}(\mathbf{x}) = 0.5$, thus giving us the failure boundary at the peaks. Therefore, the second proposed acquisition function selects the point that maximizes the upper confidence of μ' to refine the failure boundary:

$$\mathbf{x}_2' = \arg\max_{\mathbf{x}\in\mathcal{X}} \big(\mu'(\mathbf{x}) + \lambda\hat{\sigma}(\mathbf{x})\big)\, p(\mathbf{x})^{1/t} \tag{12}$$

where upper confidence provides an over estimation and the factor parameter is set to $\lambda = 0.1$ in our experiments. The operational model $p(\mathbf{x})$ is used to first focus on the failure boundary with high operational likelihood, then decay the emphasis of the likelihood as a function of the current iteration t (here using an inverse decay of $1/t$). This will first acquire likely points along the boundaries, then refine all of the boundaries because as $t \to \infty$ then $p(\mathbf{x})^{1/t} \to 1$.

Failure region sampling. It has been shown [8, 37, 38] that the optimal importance sampling distribution is

$$q_{\text{opt}} \propto f(\mathbf{x})p(\mathbf{x}), \tag{13}$$

which, intuitively, is the distribution of failures (when $f(\mathbf{x}) = 1$) over the likely region (weighed by $p(\mathbf{x})$). Yet this is exactly what we are trying to estimate and sampling this distribution would require a prohibitive number of evaluations of f, which we would like to avoid. Therefore, the third acquisition function we propose uses the surrogate to get the upper confidence of the failure prediction

$$\hat{h}(\mathbf{x}) = \hat{f}(\mathbf{x}) + \lambda\hat{\sigma}(\mathbf{x}) \tag{14}$$

$$\hat{g}(\mathbf{x}) = \mathbb{1}\big\{\hat{h}(\mathbf{x}) \geq 0.5\big\} \tag{15}$$

and then using the estimated failure region $\hat{g}$ we draw a sample from the distribution

$$\mathbf{x}_3' \sim \hat{g}(\mathbf{x})p(\mathbf{x}). \tag{16}$$

Here we use the indicator function $\mathbb{1}\{\cdot\}$ that returns 1 when the input is `true` and 0 otherwise. Sampling from the approximate failure distribution defined by the surrogate helps to refine likely failure regions to ensure a better estimate

of the probability of failure. But if the system under test f outputs a probability of failure value in $[0, 1]$ instead of a binary failure indication, then we can use this information and sample from the following distribution that weights towards those failures that have higher confidence:

$$\mathbf{x}_3' \sim \hat{g}(\mathbf{x})\hat{h}(\mathbf{x})p(\mathbf{x}) \tag{17}$$

The proposed acquisition functions work under a more restrictive binary system $f : \mathbb{R}^n \to \{0, 1\}$ and a system $f : \mathbb{R}^n \to [0, 1]$ that outputs a probabilistic value of failure (which can be interpreted as confidence or stochasticity). We define *failure region sampling* using the more general distribution in eq. (17) because it works for both types of system outputs. When a granular measure of system failure is available, it can be used to make a more informative surrogate model. One use of such a surrogate would be to analyze the severity of failure based on the regions with the highest failure confidence. When using this type of surrogate during system development, developers could focus on addressing or mitigating those failures with both high confidence and high operational likelihood as a form of triage. If only binary failure information is available, developers could simply focus on those failures with high operational likelihood. Applying to binary-valued systems is more general and thus the primary focus of this work, but we demonstrate on a probability-valued case in section IV.F. Algorithm 1 describes the full *failure search and refinement* procedure to compute the subsequent points from the three proposed acquisition functions and fig. 3 provides an illustrative example.

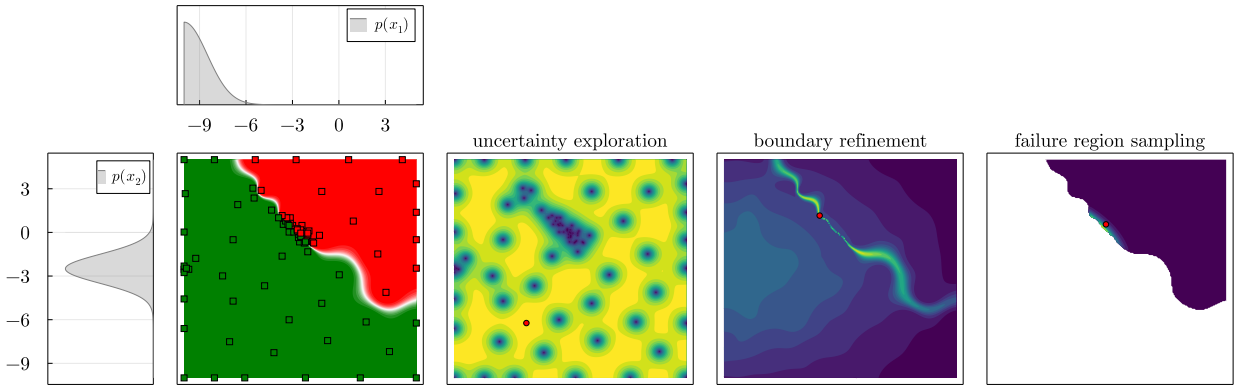


Fig. 3 The probabilistic surrogate model (predicted failures shown in red) and the *failure search and refinement* acquisition functions after $T = 30$ iterations ($N = 90$ data points, showing true observations as red/green squares). Lighter colors indicate maximums and red circles indicate the next selected point. The operational likelihood model $p(\mathbf{x})$ is shown as marginal distribution subplots. Notice the low uncertainty around the likely failure region and the influence of $p(\mathbf{x})$ on the boundary refinement; the algorithm first refines the likely boundary, then refines the entire boundary in the limit. The system under test is an example system shown in fig. 5a.

Algorithm 1 Failure search and refinement acquisition functions.

function FAILURESEARCHANDREFINEMENT($\mathcal{GP}, p, t$)
  $\hat{f} \leftarrow$ MEANFUNCTION($\mathcal{GP}$)
  $\hat{\sigma} \leftarrow$ STANDARDDEVIATIONFUNCTION($\mathcal{GP}$)
  `# 1) uncertainty exploration`
  $\mathbf{x}_1' \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} \hat{\sigma}(\mathbf{x})$
  `# 2) boundary refinement`
  $\mu'(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x})(1 - \hat{f}(\mathbf{x}))$ ▷ compute failure boundaries
  $\mathbf{x}_2' \leftarrow \arg\max_{\mathbf{x}\in\mathcal{X}} \big(\mu'(\mathbf{x}) + \lambda\hat{\sigma}(\mathbf{x})\big)p(\mathbf{x})^{1/t}$
  `# 3) failure region sampling`
  $\hat{h}(\mathbf{x}) \leftarrow \hat{f}(\mathbf{x}) + \lambda\hat{\sigma}(x)$ ▷ compute upper confidence bound
  $\hat{g}(\mathbf{x}) \leftarrow \mathbb{1}\{\hat{h}(\mathbf{x}) \geq 0.5\}$ ▷ compute failure regions
  $\mathbf{x}_3' \sim \hat{g}(\mathbf{x})\hat{h}(\mathbf{x})p(\mathbf{x})$
  return $\{\mathbf{x}_1', \mathbf{x}_2', \mathbf{x}_3'\}$
end

A. Importance Sampling Estimate of Failure Probability

To compute an efficient and unbiased estimate of the probability of failure, we use *importance sampling* [8]. Probability estimation can be defined as computing the expectation of the Boolean-valued function f over the *target/nominal distribution* p (what we call the *operational likelihood model* in this work) as

$$\mathbb{P}[f(\mathbf{x})] = \mathop{\mathbb{E}}_{\mathbf{x}\sim p}[f(\mathbf{x})] = \int_{\mathcal{X}} p(\mathbf{x}) f(\mathbf{x})\, dx. \tag{18}$$

In general, the expectation of the indicator function of an event A, denoted $\mathbb{1}\{A\}$, is equal to the probability of that event occurring $\mathbb{E}[\mathbb{1}\{A\}] = \mathbb{P}[A]$. In our problem, we define $f : \mathbb{R}^n \to \mathbb{B}$ as a Boolean-valued function for convenience. Nevertheless, the following work could easily be extended to a real-valued function $v : \mathbb{R}^n \to \mathbb{R}$ where failures are defined by violating some safety threshold c, i.e., $f(\mathbf{x}) = \mathbb{1}\{v(\mathbf{x}) \geq c\}$. Now to approximate the expectation—and therefore the probability of failure—we can use n samples from p:

$$\mathop{\mathbb{E}}_{\mathbf{x}\sim p}[f(\mathbf{x})] \approx \frac{1}{n}\sum_{i=1}^{n} f(\mathbf{x}_i) \tag{19}$$

If failures are rare under the distribution p (i.e., $f(\mathbf{x}_i)$ is rarely equal to 1 when $\mathbf{x}_i \sim p$), then we may need an extremely large number of samples from p to get an accurate estimate. But this would require prohibitively many system evaluations of f. Instead, importance sampling states that we can sample from some other distribution q, called the *proposal distribution*, and re-weight the outputs of f based on the *likelihood ratio* [8]. To see this, consider the following [40]:

$$\begin{aligned}
\mathop{\mathbb{E}}_{\mathbf{x}\sim p}[f(\mathbf{x})] &= \int_{\mathcal{X}} p(\mathbf{x}) f(\mathbf{x})\, dx && \text{(expected value of } f(\mathbf{x}), \text{ same as eq. (18))} \\
&= \int_{\mathcal{X}} p(\mathbf{x}) \frac{q(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\, dx && \text{(introduce proposal } \tfrac{q(\mathbf{x})}{q(\mathbf{x})} = 1) \\
&= \int_{\mathcal{X}} q(\mathbf{x}) \frac{p(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\, dx && \text{(reorder to isolate likelihood ratio } \tfrac{p(\mathbf{x})}{q(\mathbf{x})}) \\
&= \mathop{\mathbb{E}}_{\mathbf{x}\sim q}\left[\frac{p(\mathbf{x})}{q(\mathbf{x})} f(\mathbf{x})\right] && \text{(definition of expectation over } q) \\
&\approx \frac{1}{n}\sum_{i=1}^{n} \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)} f(\mathbf{x}_i) && \text{(importance sampling estimate using samples } \mathbf{x}_i \sim q)
\end{aligned}$$

Now we can use samples from q to approximate the probability over p. But selecting the right q-proposal distribution is a challenge and an important topic of research in itself (see Bugallo et al. [15]). To address this challenge, we could use a uniform proposal over the design space $q = \mathcal{U}_{\mathcal{X}}$ and replace the expensive function calls to f with inexpensive evaluations of the surrogate $\hat{f}$ using orders of magnitude more samples. Thus our problem gets simplified to estimate

$$\hat{P}_{\text{fail}} = \mathop{\mathbb{E}}_{\mathbf{x}\sim p}[\hat{f}(\mathbf{x})] = \mathop{\mathbb{E}}_{\mathbf{x}\sim q}\left[\frac{p(\mathbf{x})}{q(\mathbf{x})}\hat{f}(\mathbf{x})\right] \approx \frac{1}{n}\sum_{i=1}^{n} \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)}\hat{f}(\mathbf{x}_i). \tag{20}$$

Yet using a uniform distribution can induce variance in the estimate [41]. Therefore, an even further simplification is to use a discretized set of n points $\bar{\mathcal{X}}$ over the range $\mathcal{X}$ as our proposal. We assigned equal likelihood to each point $\mathbf{x}_i \in \bar{\mathcal{X}}$, namely $q(\mathbf{x}_i) = 1/n \sum_{j=1}^{n} p(\mathbf{x}_j)$. Then eq. (20) becomes

$$\hat{P}_{\text{fail}} \approx \frac{1}{n}\sum_{i=1}^{n} \frac{p(\mathbf{x}_i)}{1/n \sum_{i=1}^{n} p(\mathbf{x}_i)}\hat{f}(\mathbf{x}_i) = \frac{\sum_{i=1}^{n} p(\mathbf{x}_i)\hat{f}(\mathbf{x}_i)}{\sum_{i=1}^{n} p(\mathbf{x}_i)} = \frac{\mathbf{w}^\top \hat{\mathbf{y}}}{\sum_{i=1}^{n} w_i} \tag{21}$$

where $\mathbf{w} = [p(\mathbf{x}_1), \ldots, p(\mathbf{x}_n)]$ and $\hat{\mathbf{y}} = [\hat{f}(\mathbf{x}_1), \ldots, \hat{f}(\mathbf{x}_n)]$ for $\mathbf{x}_i \in \bar{\mathcal{X}}$. Here we are using *likelihood weighting* which is a special case of importance sampling [38, 41]. Using a discretized set of points as the proposal distribution has lower variance than sampling the uniform space, but may not scale well to higher dimensions. For this paper, we use a simplified 500×500 discrete grid as the proposal for two-dimensional systems. To incorporate better proposal distributions when scaling to higher dimensions, see Bugallo et al. [15] for recent adaptive importance sampling work. Importantly, we show our discrete choice works well but a better proposal would only benefit the following approach.

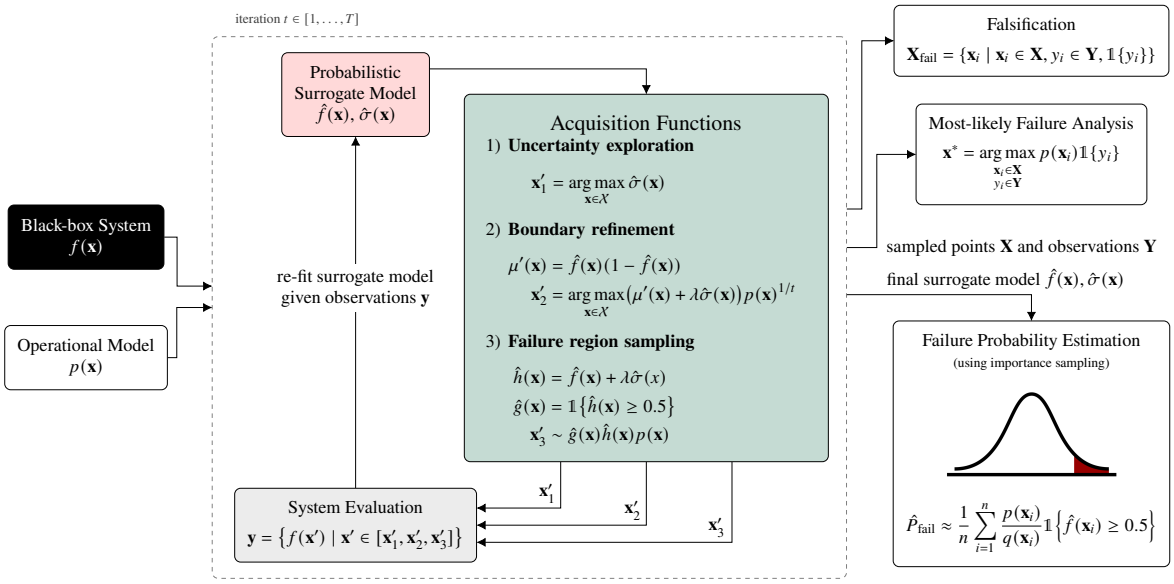


Fig. 4 The proposed *Bayesian safety validation* algorithm used for all three safety validation tasks.

B. Bayesian Safety Validation

The proposed algorithm, *Bayesian safety validation* (BSV), takes as input the black-box system $f : \mathbb{R}^n \to \mathbb{B}$, an operational likelihood model $p : \mathbb{R}^n \to \mathbb{R}$, and a proposal distribution $q : \mathbb{R}^n \to \mathbb{R}$, each taking an input $\mathbf{x} \in \mathbb{R}^n$ of some parametric space, and iteratively refits a probabilistic surrogate model given selected points from three acquisition functions using *failure search and refinement* described in section III. The first acquisition function, *uncertainty exploration*, explores areas with high uncertainty to provide coverage and search for failure regions. The next acquisition function, *boundary refinement*, selects operationally likely points that refine the failure boundaries to better characterize likely failure regions (and includes a decaying weighted operational likelihood to refine all failure boundaries in the limit). The final acquisition function, *failure region sampling*, is based on the theoretically optimal importance sampling q-proposal distribution [37] and will sample from the likely failure regions to ensure a better estimate of the probability of failure. After the algorithm runs for T iterations, a total of $3T$ sampled points were used to fit the surrogate model. The three safety validation tasks are then computed (lines 10–12). Falsification and most-likely failure analysis use only the true observations $y_i \in \mathbf{Y}$ and actual inputs $\mathbf{x}_i \in \mathbf{X}$ to find those inputs that led to failures and the most-likely failure, respectively. The final surrogate model is then used to efficiently compute an importance sampling estimate of the probability of failure. Algorithm 2 describes the Bayesian safety validation algorithm and fig. 4 illustrates the process.

Algorithm 2 Bayesian safety validation algorithm.

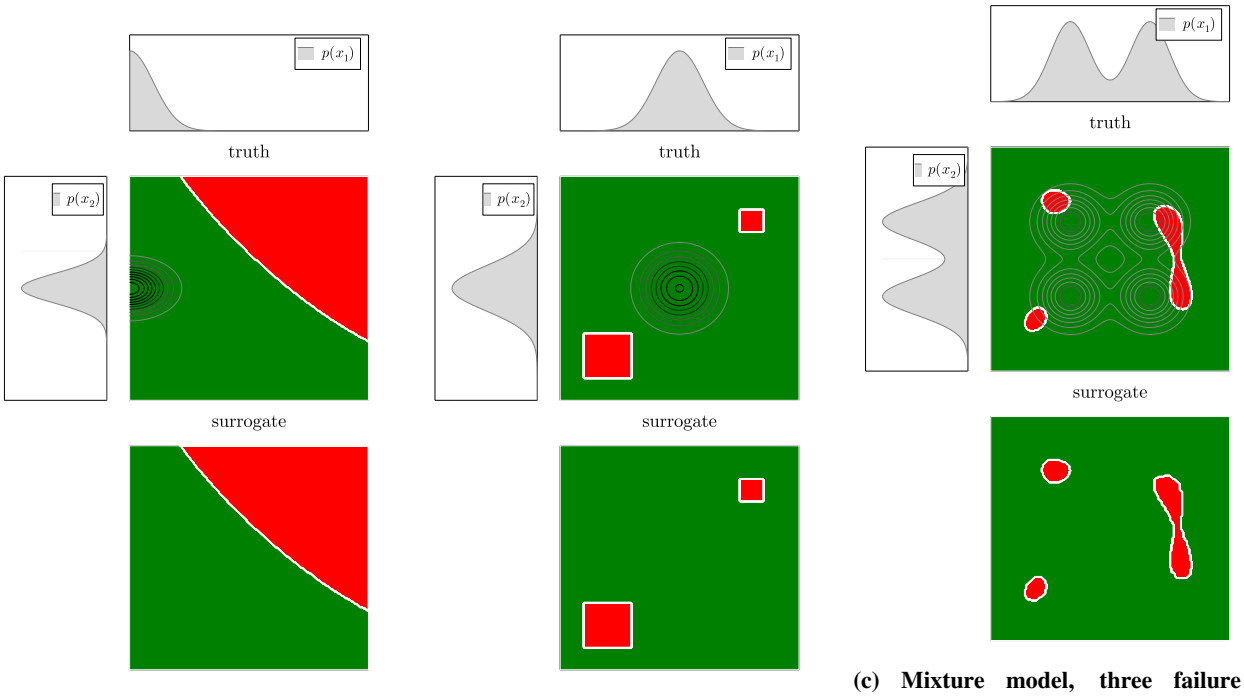
1: **function** BayesianSafetyValidation(f, p, q, T)
2: $\mathcal{GP} \leftarrow$ InitializeGaussianProcess(m, k)
3: $\mathbf{X}, \mathbf{Y} \leftarrow \emptyset, \emptyset$
4: **for** $t \leftarrow 1$ **to** T
5: $\mathbf{X}' \leftarrow$ FailureSearchAndRefinement($\mathcal{GP}, p, t$) ▷ select new design points (defined in algorithm 1)
6: $\mathbf{Y}' \leftarrow \{f(\mathbf{x}') \mid \mathbf{x}' \in \mathbf{X}'\}$ ▷ evaluate true system f across design points
7: $\mathbf{X}, \mathbf{Y} \leftarrow \mathbf{X} \cup \mathbf{X}', \mathbf{Y} \cup \mathbf{Y}'$ ▷ append to input and output sets
8: $\mathcal{GP} \leftarrow$ Fit($\mathcal{GP}, \mathbf{X}, \mathbf{Y}$) ▷ refit surrogate model over all points conditioned on observations
9: **end**
10: $\mathbf{X}_{\text{fail}} \leftarrow \{\mathbf{x}_i \mid \mathbf{x}_i \in \mathbf{X}, y_i \in \mathbf{Y}, \mathbb{1}\{y_i\}\}$ ▷ (1) falsification (set of all true failures)
11: $\mathbf{x}^* \leftarrow \arg\max_{\mathbf{x}_i \in \mathbf{X}, y_i \in \mathbf{Y}} p(\mathbf{x}_i)\mathbb{1}\{y_i\}$ ▷ (2) most-likely failure analysis
12: $\hat{P}_{\text{fail}} \leftarrow \frac{1}{n}\sum_{i=1}^{n} \frac{p(\mathbf{x}_i)}{q(\mathbf{x}_i)} \mathbb{1}\left\{\hat{f}(\mathbf{x}_i) \geq 0.5\right\}$ ▷ (3) failure probability estimation (using importance sampling)
13: **return** $\mathbf{X}_{\text{fail}}, \mathbf{x}^*, \hat{P}_{\text{fail}}$ ▷ return all three safety validation tasks
14: **end**

IV. Experiments and Results

To test the effectiveness of the Bayesian safety validation algorithm, we ran experiments across several different example systems and a real-world case study using a prototype neural network-based runway detection system. We split the experiments into two sections: 1) comparison against existing methods for rare-event simulation (this tests the full Bayesian safety validation algorithm), and 2) comparison of the Gaussian process-based approach with different sampling/selection methods (this tests the failure search and refinement acquisition functions). We ran an ablation study to empirically show the influence of each acquisition function on the performance of the safety validation tasks. We demonstrate the algorithm on a system that outputs a probabilistic value of failure (instead of strictly binary) to show the algorithms general applicability to a less restrictive problem. Lastly, we report results on the runway detection system as a real-world case study.

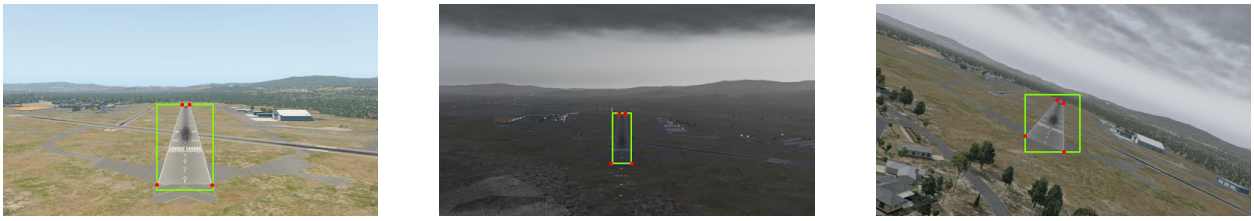
A. Simplified Test Problems

Three example toy problems with access to the true value of P_{fail} were used for testing. The first problem (called REPRESENTATIVE) was chosen based on the observed shape of the failure region of the runway detection system, which is our primary case study and a system which we do not have access to the true failure probability. The representative toy problem is modeled using Booth's function [24] $f(\mathbf{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \leq 200$, thresholded to make this a binary function. We define the operational parameters to be over the range $[-10, 5]$ for both x_1 and x_2 and set the operational likelihood model as $x_1 \sim \mathcal{N}_{\text{trunc}}(-10, 1.5; [-10, 5])$ and $x_2 \sim \mathcal{N}(-2.5, 1)$ where $\mathcal{N}_{\text{trunc}}(\mu, \sigma; [a, b])$ is the normal distribution truncated between $[a, b]$. The second toy problem (called SQUARES) has two, square, disjoint failure regions to test the exploration of BSV and refinement of rigid and disjoint failure boundaries. The operational parameters are over the range $[0, 10]$ for both x_1 and x_2, each with the operational likelihood model of $\mathcal{N}(5, 1)$. The third toy problem (called MIXTURE) has three, smooth, disjoint failure regions and is designed to test the failure region refinement characteristic of BSV using a multimodal operational model. The operational range is over $[-6, 6]$ with identical Gaussian mixture models that have two equal components of $\mathcal{N}_{\text{trunc}}(2, 1; [-6, 6])$ and $\mathcal{N}_{\text{trunc}}(-2, 1; [-6, 6])$. Similar to the representative example, we define this last problem as the thresholded Himmelblau function [42]: $f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 \leq 15$. The test problems and their operational models are shown in fig. 5.



(a) Representative problem (Booth's). **(b) Two square failure modes.** **(c) Mixture model, three failure modes.**

Fig. 5 Failure regions (in red) for the test problems. Operational models are illustrated as subplots/contours. The true system is shown above the surrogate model failure classification which is fit with 999 samples using BSV.



(a) Nominal runway conditions. (b) Low-likely weather conditions. (c) Potentially dangerous to fly.

Fig. 6 Applying the neural network runway detector to simulated runway conditions in X-Plane.

B. Neural Network-based Runway Detection System

As a real-world case of a safety-critical subsystem, we chose a common application in autonomous flight: runway detection (RWD) using neural networks. Synthetic images were generated using the flight simulator X-Plane [43], sampling over different parameters of the final approach to land (e.g., glide slope angle and distance to runway). We search over this parametric space instead of dealing directly in pixel or image space. We use an operational model that centers around likely glide slope angles with a small standard deviation, namely $\alpha \sim \mathcal{N}(3, 0.5)$ and a model that increases the likelihood of requiring a detection as the distance to the runway decreases, namely $d \sim \mathcal{N}_{\text{trunc}}(0, 1; [0, 4])$. The parametric space is continuous in glide slope angle $\alpha \in [1, 7]$ degrees and distance to runway $d \in [0.1, 4]$ nmi. These models can be learned from historical flight data for more accurate estimates of the failure probability.

Treated as a black box, the runway detector is a convolutional neural network that processes runway images from a front-facing RGB camera. The network predicts the runway corners and the runway bounding box and is intended to be used as a subsystem to provide position estimates during autonomous landing [44]. Figure 6 illustrates several example images with detected runway corners and bounding boxes. A failure is defined as a misdetection (i.e., a false negative). Since the system is designed to be active only during the landing phase, we condition on the aircraft being on the approach. The use of a simulator means the runway detection system can be stressed outside the normal flight envelope to better characterize the full range of system failures, with a potentially dangerous-to-fly example shown in fig. 6c.

C. Safety Validation Metrics

In this section, we define several metrics to measure how well BSV and the baselines perform across the three safety validation tasks (see section II.A).

Falsification metrics. The total number of failure cases, or more generally, the proportion of all system evaluations that resulted in failures is the primary metric used to assess falsification (sometimes called the failure rate):

$$R_{\text{fail}} = \frac{\text{number of failures}}{\text{total number of evaluations}} = \frac{|\mathbf{X}_{\text{fail}}|}{|\mathbf{X}|} \tag{22}$$

Most-likely failure analysis metrics. The goal of most-likely failure analysis, as the name suggests, is to determine the failure with maximum operational likelihood. A natural way to assess the relative performance of this task against baselines is to compare the likelihood of the determined most-likely failure:

$$\mathcal{L}^* = \max_{\mathbf{x}_i \in \mathbf{X}, y_i \in \mathbf{Y}} p(\mathbf{x}_i) \mathbb{1}\{y_i\} \tag{23}$$

Failure probability estimation metrics. Because probability of failure estimation is the primary objective of this work—capturing all three safety validation tasks—we look at the performance across several different metrics. When we have access to the true P_{fail} (e.g., in the toy examples), then we can measure the relative error in the estimated $\hat{P}_{\text{fail}}$:

$$\hat{\Delta}_{\text{fail}} = \frac{|P_{\text{fail}} - \hat{P}_{\text{fail}}|}{P_{\text{fail}}} \tag{24}$$

Measured as a proportion, relative error can be interpreted as the percent difference in the estimate and makes it easier to compare performance across problems. We are also interested in analyzing the failure likelihood distribution $\{\log p(\mathbf{x}_i)\}_{\mathbf{x}_i \in \mathbf{X}_{\text{fail}}}$. Distributions with higher concentration in operationally likely regions are preferred as they cover more relevant example failures.

Coverage of design space. To measure the coverage of the design space, an average dispersion coverage metric has been used in the context of safety validation to estimate how well the sampled points cover the input space [22, 45]:

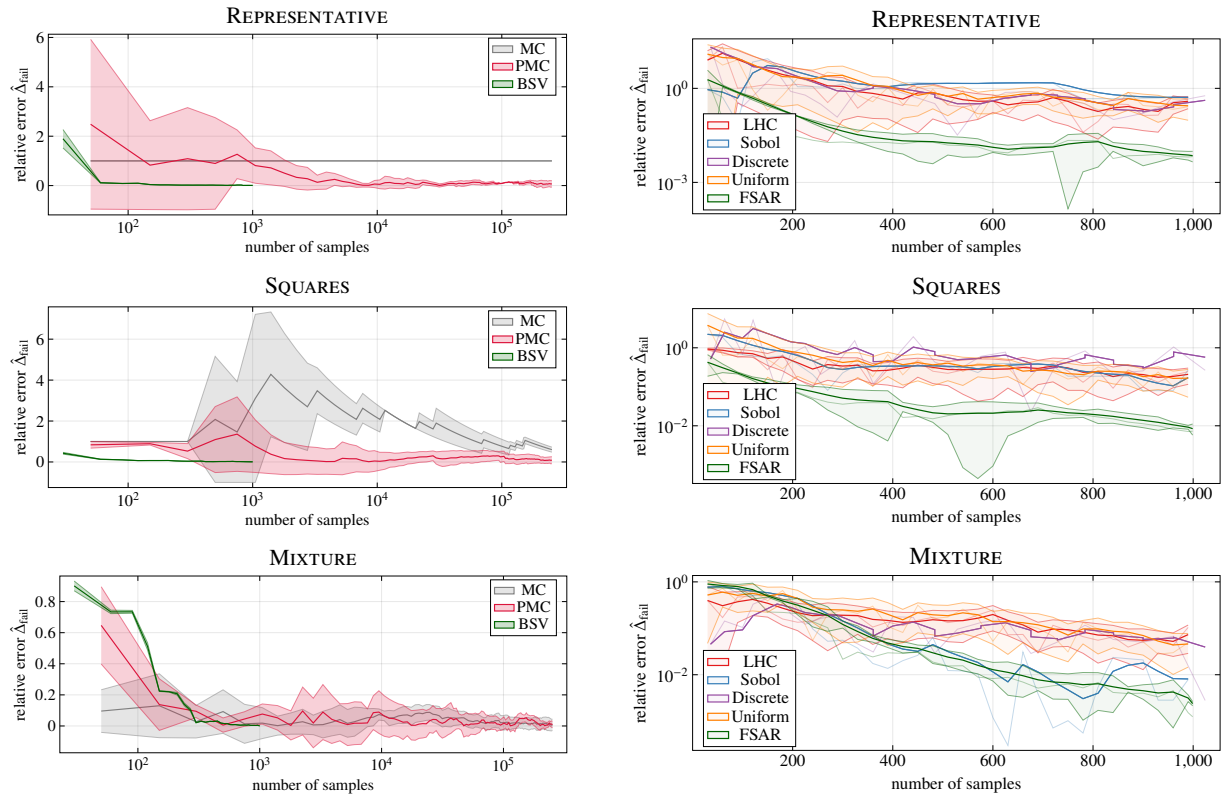
$$C_{\text{input}}(\mathbf{X}) = 1 - \frac{1}{\delta}\sum_{j=1}^{n}\frac{\min(d_j(\mathbf{X}), \delta)}{n} \tag{25}$$

where $C_{\text{input}}(\mathbf{X}) \in [0, 1]$ and the metric is defined over a grid of n points, separated by δ. The distance function $d_j(\mathbf{X})$ is defined as the minimum distance between the jth point in the grid to a point in $\mathbf{X}$ [22, 45].

When ground truth is available, we are also interested in the characterization of the failure and non-failure regions over the entire domain as predicted by the surrogate model. We define $C_{\text{output}} \in [0, 1]$ as the proportion of the output space that the surrogate and the true system agree upon. This can be interpreted as the surrogate classification accuracy.

D. Baseline Methods

To compare against existing rare-event estimation algorithms, we test the proposed BayesianSafetyValidation algorithm (algorithm 2) compared to standard Monte Carlo (MC) sampling and population Monte Carlo (PMC) [13] which uses self-normalized importance sampling [8]. PMC requires an initial adaptive proposal q_{PMC} which we set to be equal to the operational likelihood model for each of the example problems. The experiments were run for $T_{\max} = 100$ iterations using $N_q = 50$ samples per iteration and ran across 3 RNG seeds. This results in $N_q T_{\max}(T_{\max}+1)/2 = 252{,}500$ total number of samples per seed. Because sampling-based methods like MC and PMC tend to require many samples to adequately estimate the rare-event [6], we only test these methods on the example toy problems as scaling to the runway detection system would be prohibitively expensive. Motivated by sample efficiency, we focus our comparison on the relative error in the estimated probability of failure as a function of the number of samples, defined in eq. (24).



(a) Comparison to other estimators. Note that MC fails to estimate anything in the allotted number of samples in the representative example. The horizontal axis is on a log scale.

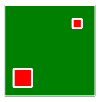
(b) The same initial GP is used and then fit using different sampling schemes. Showing the same FSAR results from (a), now with the vertical axis on a log scale.

Fig. 7 Test problem baseline results. Shaded regions show standard error in (a) and standard deviation in (b).

Figure 7a shows the error curves over the number of samples, which is equivalent to the number of system evaluations. It is clear that BSV outperforms both MC and PMC in reducing the error in the probability of failure estimate using several orders of magnitude fewer samples; converging before 1000 samples in each case and closer to 100 samples in the first two problems. Results also indicate that BSV has lower variance compared to MC and PMC, which can partially be explained by the fact that two of the three acquisition functions take deterministic maximums and only one, the failure region sampling acquisition, samples predicted failure points stochastically.

To test the proposed FAILURESEARCHANDREFINEMENT procedure (algorithm 1), we use the same GP fitting technique as in algorithm 2 but replace the selection process in line 5 with several baseline methods. The baselines we use are Latin hypercube sampling (LHS) [46], Sobol sequence selection [47], discrete grid selection, and uniform sampling. Each technique is defined over the entire operational domain. Importantly, we note that all methods fit the selected points to the same initial Gaussian process and use the same importance sampling procedure defined in algorithm 2. The failure search and refinement (FSAR) approach is the only method that uses incremental information to optimize the subsequent points. Figure 7b illustrates the relative error in the estimate when running BSV for $T = 333$ iterations ($N = 999$ system evaluations), run over 3 RNG seeds with shaded regions reporting standard deviation (noting that Sobol and discrete do not use stochasticity). Exponential smoothing is applied to the curves with the raw values as thin lines of the same color. Using FSAR for acquiring subsequent points outperforms the baselines by orders of magnitude in the first two problems, and is comparable to Sobol sequence selection in the third problem but with more stability in the estimate. Table 1 reports the quantitative results from the baseline experiments. FSAR achieves the best performance across the various safety validation metrics and comparable input coverage relative to the baselines.

Table 1 Comparison against other sampling/selection methods.

Example Problem	Selection Method	$R_{\text{fail}} \uparrow$	$\mathcal{L}^* \uparrow$	$\hat{\Delta}_{\text{fail}} \downarrow$	$C_{\text{input}} \uparrow$	$C_{\text{output}} \uparrow$
REPRESENTATIVE 	Latin hypercube sampling	0.332	2.49×10^{-8}	0.44777	0.682	0.9922
	Sobol sequence sampling	0.335	4.98×10^{-8}	0.51179	0.719	0.9912
	Discrete grid selection	0.336	5.62×10^{-8}	0.58225	**0.774**	0.9933
	Uniform sampling	0.342	4.02×10^{-8}	0.25978	0.673	0.9919
	Failure search and refinement	**0.585**	$\mathbf{5.78 \times 10^{-8}}$	**0.00667**	0.638	**0.9998**
SQUARES 	Latin hypercube sampling	0.0525	0.00192	0.24486	0.682	0.9907
	Sobol sequence sampling	0.0525	0.00142	0.27050	0.719	0.9935
	Discrete grid selection	0.0439	0.00196	0.26473	**0.774**	0.9894
	Uniform sampling	0.0532	0.00086	0.17017	0.673	0.9909
	Failure search and refinement	**0.4800**	**0.00253**	**0.00727**	0.643	**1.0**
MIXTURE 	Latin hypercube sampling	0.0441	0.0349	0.10469	0.682	0.9864
	Sobol sequence sampling	0.0505	0.0369	0.00787	0.719	0.9881
	Discrete grid selection	0.0479	0.0345	0.00279	**0.774**	0.9900
	Uniform sampling	0.0576	0.0360	0.04919	0.673	0.9871
	Failure search and refinement	**0.4640**	**0.0383**	**0.00124**	0.663	**0.9984**

E. Ablation Study

To empirically test the importance of all three acquisition functions, we perform an ablation study on the disjoint squares problem to determine the effect of the combinations of acquisition functions across the safety validation metrics. The SQUARES example problem was chosen based on having two disjoint failure regions with precise boundaries; one region which is less likely than the other. Thus this problem requires careful balance between boundary refinement and deliberate exploration for multiple potential failure regions. Each ablation was run with 90 samples over 5 RNG seeds for a fair comparison (i.e., when using all three acquisitions, each one gets a third of the budget). Results in table 2 indicate that the individual acquisitions perform well on the single metric they were designed for (i.e., *exploration* covers the input space, *failure sampling* has the highest failure rate, yet *boundary refinement* requires exploration in order to avoid exploiting a single failure mode). Using all three acquisition functions balances between the safety validation metrics and achieves the smallest error in the probability of failure estimate $\hat{\Delta}_{\text{fail}}$ while also finding a failure with the highest relative likelihood. In table 2, arrows indicate whether the given metric is better to be high ($\uparrow$) or low ($\downarrow$).

Table 2 Ablation study of the effect of the three *failure search and refinement* acquisition functions.

Acquisition(s)	$R_{\text{fail}} \uparrow$	$\mathcal{L}^* \uparrow$	$\hat{\Delta}_{\text{fail}} \downarrow$	$C_{\text{input}} \uparrow$	$C_{\text{output}} \uparrow$
[1] exploration	0.044	0.00029	0.04382	**0.233**	0.9795
[2] boundary refinement	0.411	0.00025	0.93670	0.056	0.9598
[3] failure sampling	**0.707**	0.00186	0.21682	0.066	0.9604
[1, 2] exploration + boundary refinement	0.189	0.00205	0.18483	0.159	**0.9801**
[2, 3] boundary refinement + failure sampling	0.436	0.00197	0.24817	0.087	0.9647
[1, 3] exploration + failure sampling	0.318	0.00171	0.10057	0.163	0.9761
[1, 2, 3] exploration + boundary refinement + failure sampling	0.298	**0.00243**	**0.03553**	0.138	0.9777

F. Test on Probabilistic-Valued System

As mentioned in section III, the GP construction and proposed acquisition functions were designed to estimate failure probability over binary-valued systems that indicate failure, but the same techniques are applicable when the system outputs a probabilistic value of failure (that can be interpreted as confidence in the output, distance to failure boundary, or stochasticity of the system—which has been addressed by similar approaches from Gong and Pan [48]). Using the same Himmelblau function [42] defined for the MIXTURE problem in section IV.A, we change the system to output a measure of failure (where $f(\mathbf{x}) \geq 0.5$ means failure): $f(\mathbf{x}) = \text{logit}^{-1}\left(c - \left((x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2\right)\right)$ for the threshold $c = 15$ (same as the previous problem) and passing the output through a sigmoid to interpret it as a probability with a steepness of $s = 1/c$. Figure 8 illustrates this example with a final relative error of $\hat{\Delta}_{\text{fail}} = 0.012$, a falsification rate of 53.6% of samples, an input coverage of $C_{\text{input}} = 0.653$, and output coverage of $C_{\text{output}} = 0.9998$.

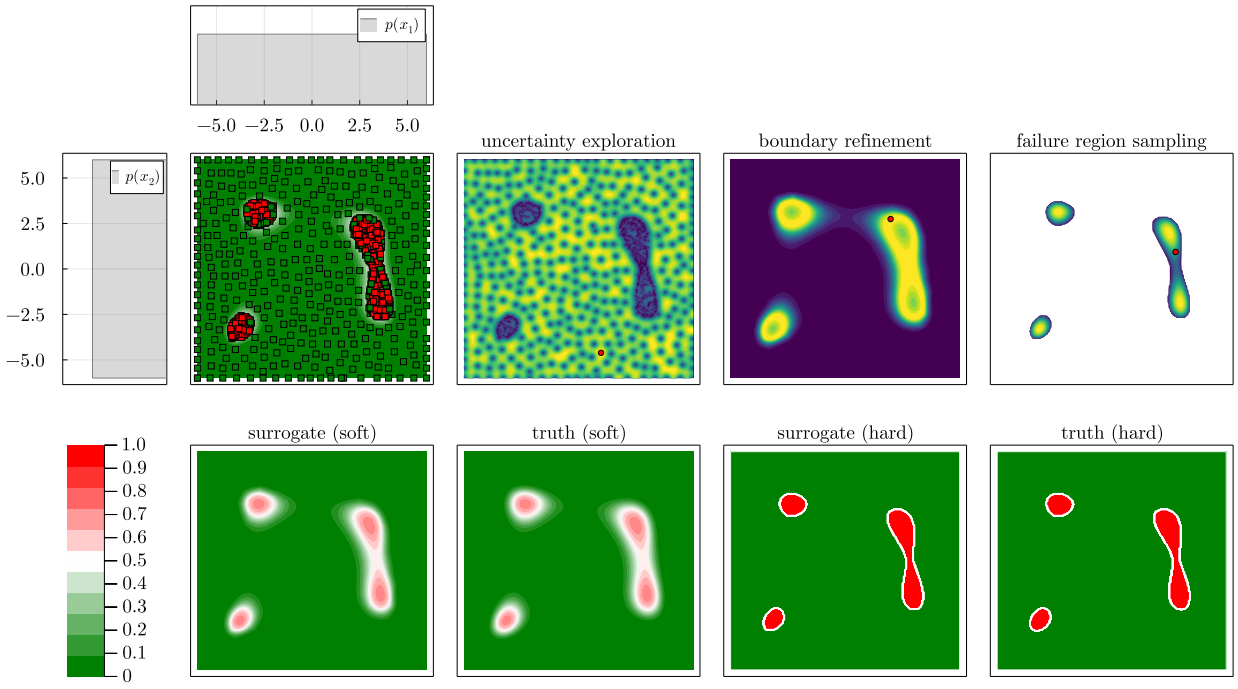


Fig. 8 Test on the MIXTURE (Himmelblau) problem, where the output is a probability value indicating distance to the failure boundary at $f(\mathbf{x}) = 0.5$. Top row illustrates BSV and the FSAR acquisition functions after $N = 999$ true observations (shown as red/green squares), with a uniform operational model p shown as subplots. The uniform model helps highlight that the failure region sampling is now more influenced by those failures that are farther away from the failure threshold c, shown as yellow peaks. The bottom row shows the surrogate model and ground truth, where "soft" is the probabilistic output and "hard" is the binary failure classification.

G. Real-World Case Study Results

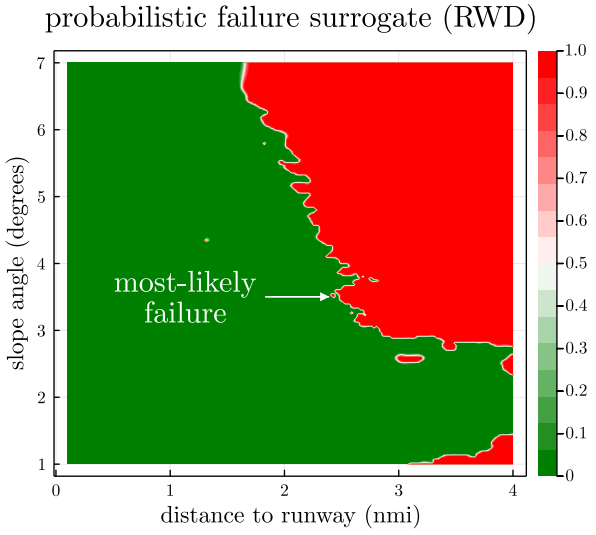
After empirically validating the BSV algorithm on the example problems with access to the ground truth, we now report the performance on a real-world example: a runway detection system. Figure 9a shows the final surrogate after running BSV for $T = 333$ iterations (resulting in 999 sampled points). The algorithm focused the search budget on the highly likely regions of the design space and found several disjoint failure modes. We can efficiently determine the most-likely failure, which is indicated in fig. 9a, and found 571 failures out of 999 evaluations, shown in table 3 as the failure rate $R_{\text{fail}} = 57.2\%$. The primary goal, estimating failure probability, is shown to quickly converge in fig. 9c after just over 400 system evaluations. The final estimated failure probability was $\hat{P}_{\text{fail}} = 5.8 \times 10^{-3}$. If we instead used Monte Carlo sampling of p, we would expect to find only about 6 failures in the 999 system evaluations.

One way to characterize the spread of failures is to plot the log-likelihood of the observed failures under the operational model. Shown in fig. 9b, the right skewed peak of the distribution indicates that the failures that were found have high likelihood and thus are more useful failures to fix first before system deployment. Five different iterations of the BSV algorithm and acquisition functions for the RWD system are illustrated in fig. 10 and table 3 reports the safety validation metrics, noting that $\hat{\Delta}_{\text{fail}}$ and C_{output} are not reported since we do not have access to the true failure boundaries of the system.

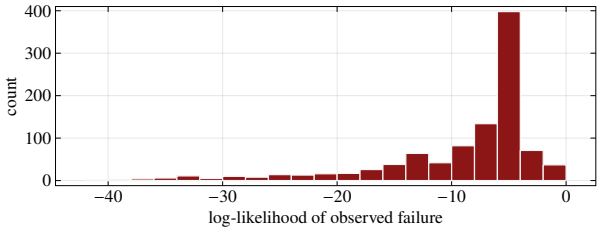
Results show that we can characterize failure regions, generate a set of likely failures, compute the most-likely failure, and use the surrogate model to estimate the probability of failure of the runway detector in a small number of samples; only use 999 samples in our experiments. Post-analysis could even further characterize the failure boundary by focusing on the likely region centered around the glide slope angle of 3 degrees. We demonstrate the BSV algorithm on a two-dimension case but this work could be scaled to higher-dimensional problems that incorporate additional environmental parameter models such as roll angle, time-of-day, weather, and across different airport runways. Binois and Wycoff [49] provide a survey on methods, challenges, and guidelines when modeling high-dimensional problems using Gaussian processes for Bayesian optimization.

Table 3 Runway detection system: Safety validation metrics when using BSV.

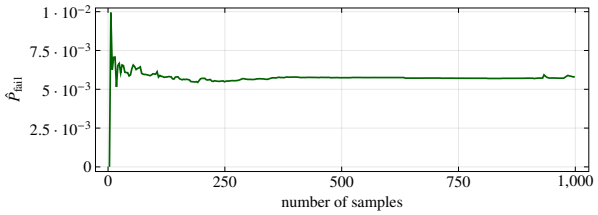
R_{fail}	$\mathcal{L}^*$	$\hat{P}_{\text{fail}}$	C_{input}
0.572	0.02	5.8×10^{-3}	0.681



(a) The final surrogate for RWD using 999 data points.



(b) Distribution of $\log p(\mathbf{x})$ for observed failures.



(c) Convergence of the probability of failure estimate.

Fig. 9 Results using Bayesian safety validation on the runway detection system.

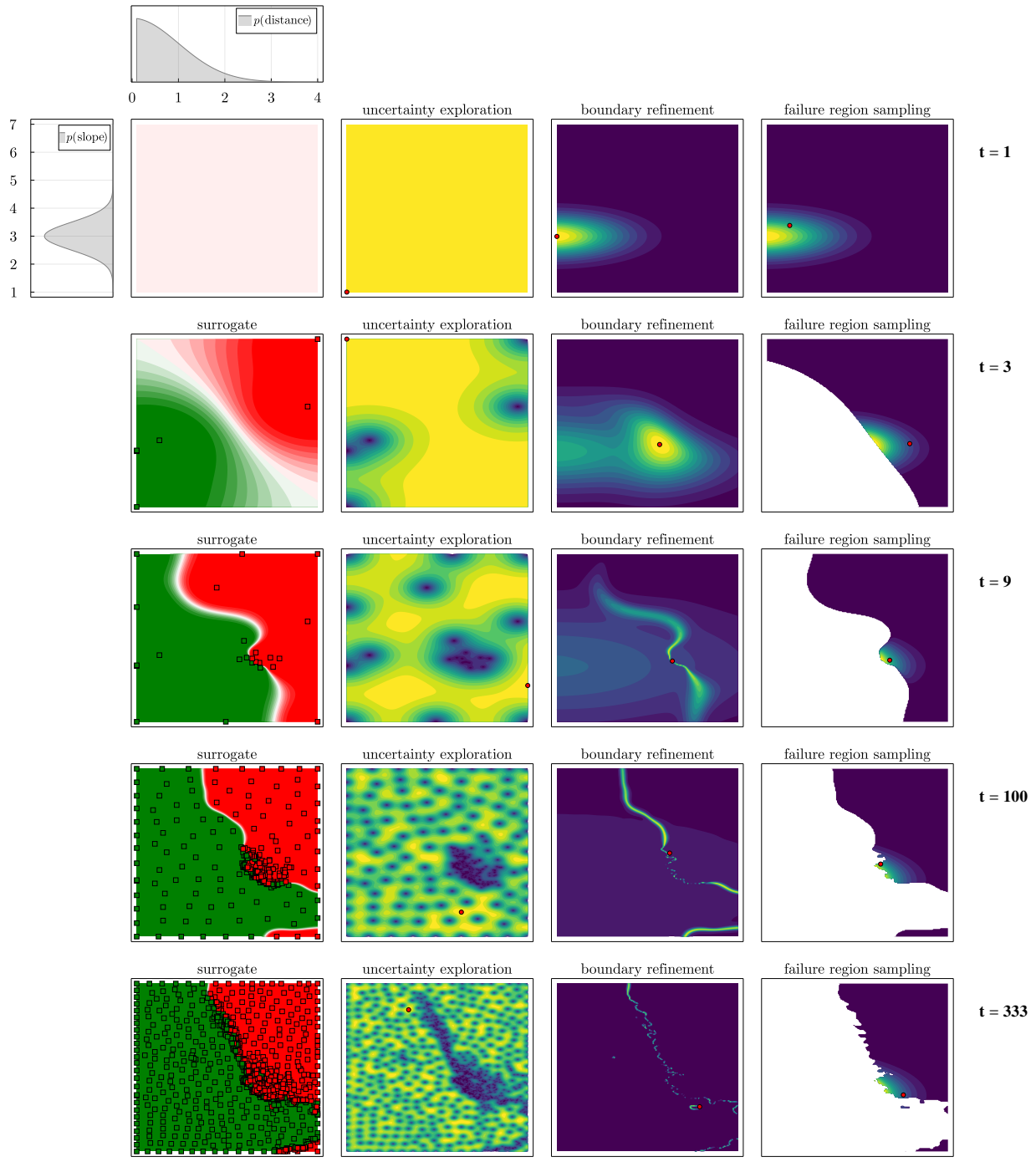


Fig. 10 Five different iterations of *Bayesian safety validation* showing the probabilistic surrogate model (red for failure) and acquisition functions (lighter colors indicate maximums) applied to the runway detector. The red and green squares overlaid on the surrogate are the true system evaluations and the red circles in the acquisition functions show the next selected point. Notice the low uncertainty in the concentration of points around the likely glide slope region (the operational models for glide slope and distance to runway are shown as subplots). The likelihood decay in the boundary refinement acquisition is illustrated as the spread of the likelihood influence as it dissipates over time. Finally, the refined failure region using $N = 999$ samples represents the predicted distribution of failures.

V. Conclusion

In this paper, we reframe the black-box safety validation problem as a Bayesian optimization problem and introduce an iterative algorithm, *Bayesian safety validation* (BSV), to build a probabilistic surrogate model that predicts system failures and uses importance sampling to efficiently estimate the failure probability. In the process, we propose a set of acquisition functions, called *failure search and refinement* (FSAR), that each help achieve the safety validation tasks; primarily by covering the design space to search for failures and refining likely failure boundaries and regions. The Gaussian process construction allows us to analytically derive the predicted failure boundaries and we show that the combination of acquisition functions is important to find more failures, find more likely failures, and minimize error in the failure probability estimate. Primarily interested in cases where the black-box system only outputs a binary indication of failure, we also show that our method works well in the less restrictive case where the system outputs a real-valued measure of failure confidence, severity, or distance. As a real-world example, this technique was applied to validate an image-based neural network runway detection system in simulation. Alongside traditional DO-178C procedures [50], this work is currently being used to supplement the FAA certification process of an autonomous cargo aircraft [44].

Extensions of this work include investigating efficient proposal distributions when scaling to higher input dimensions. The use of a simulator allows us to quickly assess the performance of systems, yet validating that the simulator correctly captures reality is an important research challenge being address by the sim-to-real community [51]. For the runway detection case, one way to perform this validation would be to run BSV and select a representative subset of safe-to-fly points to flight test, then compare their outputs. We emphasize that the exact values of the most-likely failure likelihood and the estimated probability of failure are largely dependent on the choice of operational model and learning these models from collected flight data would provide a more realistic understanding of the true probability of failure. This work is open sourced and available at `https://github.com/sisl/BayesianSafetyValidation.jl`.

Appendix

A. Open-Source Interface

This work has been open sourced[†] as a Julia package[‡] to be applied to other of black-box systems and is intended to fit into the suite of safety validation tools when considering autonomous aircraft certification. To extend to another system, a user can implement the following interface:

```
abstract type SystemParameters end
function reset() end
function initialize() end
function generate_input(sample::Vector)::Input end
function evaluate(input::Input)::Vector{Bool} end
```

Below is an example of setting up a system with a two-dimensional operational model, running the Bayesian safety validation algorithm to learn the surrogate, and then computing the three safety validation tasks.

```
using BayesianSafetyValidation

system_params = RunwayDetectionSystemParameters() # defined by user as <: SystemParameters
px1 = OperationalParameters("distance", [0.1, 4], TruncatedNormal(0, 1.0, 0, 4))
px2 = OperationalParameters("slope", [1, 7], Normal(3, 0.5))
model = [px1, px2]

surrogate  = bayesian_safety_validation(system_params, model; T=100)
X_failures = falsification(surrogate.x, surrogate.y)
ml_failure = most_likely_failure(surrogate.x, surrogate.y, model)
p_failure  = p_estimate(surrogate, model)
```

[†]The package and experiment code are available at `https://github.com/sisl/BayesianSafetyValidation.jl`.

[‡]We make use of the `GaussianProcesses.jl` package [52].

Acknowledgments

We would like to thank Jean-Guillaume Durand and Anthony Corso for their thoughtful and continuous insights into this research. We thank Rob Timpe and Alexander Bridi for the development of the runway detection neural network. We also thank Harrison Delecki for inspiration of fig. 1. This work was supported by funding from Xwing, Inc.

References

[1] RTCA, "Minimum Operational Performance Standards (MOPS) for Detect and Avoid (DAA) Systems," DO-365, 2017.

[2] Owen, M. P., Panken, A., Moss, R., Alvarez, L., and Leeper, C., "ACAS Xu: Integrated Collision Avoidance and Detect and Avoid Capability for UAS," *IEEE/AIAA Digital Avionics Systems Conference (DASC)*, 2019, pp. 1–10.

[3] Kochenderfer, M. J., Holland, J. E., and Chryssanthacopoulos, J. P., "Next-Generation Airborne Collision Avoidance System," *Lincoln Laboratory Journal*, Vol. 19, No. 1, 2012.

[4] Abu-Jbara, K., Alheadary, W., Sundaramorthi, G., and Claudel, C., "A Robust Vision-based Runway Detection and Tracking Algorithm for Automatic UAV Landing," *International Conference on Unmanned Aircraft Systems*, 2015, pp. 1148–1157.

[5] Balduzzi, G., Ferrari Bravo, M., Chernova, A., Cruceru, C., van Dijk, L., de Lange, P., Jerez, J., Koehler, N., Koerner, M., Perret-Gentil, C., Pillio, Z., Polak, R., Silva, H., Valentin, R., Whittington, I., and Yakushev, G., "Neural Network Based Runway Landing Guidance for General Aviation Autoland," Tech. rep., United States. Department of Transportation. Federal Aviation Administration, 2021.

[6] De Boer, P.-T., Kroese, D. P., Mannor, S., and Rubinstein, R. Y., "A Tutorial on the Cross-Entropy Method," *Annals of Operations Research*, Vol. 134, 2005, pp. 19–67.

[7] Robert, C. P., and Casella, G., *Monte Carlo Statistical Methods*, Vol. 2, Springer, 1999.

[8] Owen, A. B., *Monte Carlo Theory, Methods and Examples*, Stanford University, 2013.

[9] Rubinstein, R. Y., and Kroese, D. P., *The Cross-Entropy Method: A Unified Approach to Combinatorial Optimization, Monte-Carlo Simulation, and Machine Learning*, Vol. 133, Springer, 2004.

[10] Moss, R. J., "Cross-Entropy Method Variants for Optimization," *arXiv:2009.09043*, 2020.

[11] Miller, C., Corcoran, J. N., and Schneider, M. D., "Rare Events via Cross-Entropy Population Monte Carlo," *IEEE Signal Processing Letters*, Vol. 29, 2021, pp. 439–443.

[12] Arief, M., Huang, Z., Kumar, G. K. S., Bai, Y., He, S., Ding, W., Lam, H., and Zhao, D., "Deep Probabilistic Accelerated Evaluation: A Robust Certifiable Rare-Event Simulation Methodology for Black-Box Safety-Critical Systems," *International Conference on Artificial Intelligence and Statistics*, PMLR, 2021, pp. 595–603.

[13] Cappé, O., Guillin, A., Marin, J.-M., and Robert, C. P., "Population Monte Carlo," *Journal of Computational and Graphical Statistics*, Vol. 13, No. 4, 2004, pp. 907–929.

[14] Elvira, V., and Chouzenoux, E., "Optimized Population Monte Carlo," *IEEE Transactions on Signal Processing*, Vol. 70, 2022, pp. 2489–2501.

[15] Bugallo, M. F., Elvira, V., Martino, L., Luengo, D., Miguez, J., and Djuric, P. M., "Adaptive Importance Sampling: The Past, the Present, and the Future," *IEEE Signal Processing Magazine*, Vol. 34, No. 4, 2017, pp. 60–79.

[16] Luengo, D., Martino, L., Bugallo, M., Elvira, V., and Särkkä, S., "A survey of Monte Carlo methods for parameter estimation," *EURASIP Journal on Advances in Signal Processing*, Vol. 2020, No. 1, 2020, pp. 1–62.

[17] Vazquez, E., and Bect, J., "A sequential Bayesian algorithm to estimate a probability of failure," *IFAC Symposium on System Identification*, Vol. 42, No. 10, 2009, pp. 546–550.

[18] Wang, H., Lin, G., and Li, J., "Gaussian process surrogates for failure detection: A Bayesian experimental design approach," *Journal of Computational Physics*, Vol. 313, 2016, pp. 247–259.

[19] Renganathan, A., Rao, V., and Navon, I., "Multifidelity Gaussian processes for failure boundary and probability estimation," *AIAA SCITECH 2022 Forum*, 2022, pp. 1–18.

[20] He, Y., and Schumann, J., "A Framework for the Analysis of Deep Neural Networks in Aerospace Applications using Bayesian Statistics," *International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020, pp. 1–9.

[21] Ranjan, P., Bingham, D., and Michailidis, G., "Sequential Experiment Design for Contour Estimation from Complex Computer Codes," *Technometrics*, Vol. 50, No. 4, 2008, pp. 527–541.

[22] Corso, A., Moss, R. J., Koren, M., Lee, R., and Kochenderfer, M. J., "A Survey of Algorithms for Black-Box Safety Validation of Cyber-Physical Systems," *Journal of Artificial Intelligence Research*, Vol. 72, 2021, pp. 377–428.

[23] Mockus, J., and Mockus, J., "Global Optimization and the Bayesian Approach," *Bayesian Approach to Global Optimization. Mathematics and Its Applications*, Vol. 37, 1989.

[24] Kochenderfer, M. J., and Wheeler, T. A., *Algorithms for Optimization*, MIT Press, 2019.

[25] Garnett, R., *Bayesian Optimization*, Cambridge University Press, 2023.

[26] Schumann, J. M., *Automated Theorem Proving in Software Engineering*, Springer Science & Business Media, 2001.

[27] Fitting, M., *First-Order Logic and Automated Theorem Proving*, Springer Science & Business Media, 2012.

[28] Clarke, E. M., Henzinger, T. A., Veith, H., and Bloem, R., *Handbook of Model Checking*, Springer, 2018.

[29] Liu, C., Arnon, T., Lazarus, C., Strong, C., Barrett, C., and Kochenderfer, M. J., "Algorithms for Verifying Deep Neural Networks," *Foundations and Trends® in Optimization*, Vol. 4, No. 3-4, 2021, pp. 244–404.

[30] Sidrane, C., Maleki, A., Irfan, A., and Kochenderfer, M. J., "OVERT: An Algorithm for Safety Verification of Neural Network Control Policies for Nonlinear Systems," *Journal of Machine Learning Research*, Vol. 23, No. 117, 2022, pp. 1–45.

[31] Frazier, P. I., "A Tutorial on Bayesian Optimization," *arXiv:1807.02811*, 2018.

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

[33] Genton, M. G., "Classes of Kernels for Machine Learning: A Statistics Perspective," *Journal of Machine Learning Research*, Vol. 2, 2002, p. 299–312.

[34] Williams, C. K., and Barber, D., "Bayesian Classification with Gaussian Processes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 12, 1998, pp. 1342–1351.

[35] Nickisch, H., and Rasmussen, C. E., "Approximations for Binary Gaussian Process Classification," *Journal of Machine Learning Research*, Vol. 9, No. Oct, 2008, pp. 2035–2078.

[36] Jensen, B. S., Nielsen, J. B., and Larsen, J., "Bounded Gaussian Process Regression," *IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, IEEE, 2013, pp. 1–6.

[37] Kahn, H., and Marshall, A. W., "Methods of Reducing Sample Size in Monte Carlo Computations," *Journal of the Operations Research Society of America*, Vol. 1, No. 5, 1953, pp. 263–278.

[38] Murphy, K. P., *Machine Learning: A Probabilistic Perspective*, MIT Press, 2012.

[39] Quinonero-Candela, J., and Rasmussen, C. E., "A Unifying View of Sparse Approximate Gaussian Process Regression," *Journal of Machine Learning Research*, Vol. 6, 2005, pp. 1939–1959.

[40] Kochenderfer, M. J., Wheeler, T. A., and Wray, K. H., *Algorithms for Decision Making*, MIT Press, 2022.

[41] Bishop, C. M., and Nasrabadi, N. M., *Pattern Recognition and Machine Learning*, Vol. 4, Springer, 2006.

[42] Himmelblau, D. M., *Applied Nonlinear Programming*, McGraw Hill, 1972.

[43] Laminar Research, "X-Plane Flight Simulator," `https://www.x-plane.com/`, 2023.

[44] Durand, J.-G., Dubois, A., and Moss, R. J., "Formal and Practical Elements for the Certification of Machine Learning Systems," *AIAA/IEEE Digital Avionics Systems Conference (DASC)*, 2023, pp. 1–10.

[45] Esposito, J. M., Kim, J., and Kumar, V., "Adaptive RRTs for Validating Hybrid Robotic Control Systems," *Algorithmic Foundations of Robotics VI*, Vol. 17, 2005, pp. 107–121.

[46] McKay, M. D., Beckman, R. J., and Conover, W. J., "A Comparison of Three Methods for Selecting Values of Input Variables in the Analysis of Output from a Computer Code," *Technometrics*, Vol. 21, No. 2, 1979, pp. 239–245.

[47] Sobol', I. M., "On the Distribution of Points in a Cube and the Approximate Evaluation of Integrals," *Zhurnal Vychislitel'noi Matematiki i Matematicheskoi Fiziki*, Vol. 7, No. 4, 1967, pp. 784–802.

[48] Gong, X., and Pan, Y., "Sequential Bayesian experimental design for estimation of extreme-event probability in stochastic input-to-response systems," *Computer Methods in Applied Mechanics and Engineering*, Vol. 395, 2022, p. 114979.

[49] Binois, M., and Wycoff, N., "A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization," *ACM Transactions on Evolutionary Learning and Optimization*, Vol. 2, No. 2, 2022, pp. 1–26.

[50] RTCA, "Software Considerations in Airborne Systems and Equipment Certification," DO-178C, 2011.

[51] Zhao, W., Queralta, J. P., and Westerlund, T., "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey," *IEEE Symposium Series on Computational Intelligence (SSCI)*, IEEE, 2020, pp. 737–744.

[52] Fairbrother, J., Nemeth, C., Rischard, M., Brea, J., and Pinder, T., "GaussianProcesses.jl: A Nonparametric Bayes Package for the Julia Language," *Journal of Statistical Software*, Vol. 102, 2022, pp. 1–36.