Online Detection and Modeling of Safety Boundaries for Aerospace Applications using Active Learning and Bayesian Statistics

Yuning He
UARC, NASA AMES Research Center, Moffett Field, CA
Email: yuning.he@nasa.gov

***Abstract*—The behavior of complex aerospace systems is governed by numerous parameters. For safety analysis it is important to understand how the system behaves with respect to these parameter values. In particular, understanding the boundaries between safe and unsafe regions is of major importance. In this paper, we describe a hierarchical Bayesian statistical modeling approach for the online detection and characterization of such boundaries. Our method for classification with active learning uses a particle filter-based model and a boundary-aware metric for best performance. From a library of candidate shapes incorporated with domain expert knowledge, the location and parameters of the boundaries are estimated using advanced Bayesian modeling techniques. The results of our boundary analysis are then provided in a form understandable by the domain expert. We illustrate our approach using a simulation model of a NASA neuro-adaptive flight control system, as well as a system for the detection of separation violations in the terminal airspace.**

I. Introduction

Safety and controllability of an aircraft is a paramount requirement. Within a given flight envelope, the aircraft must, under all circumstances, perform safely and efficiently. Since the behavior of a complex aerospace system is governed by a multitude of parameters, it is extremely important to obtain knowledge on how the aircraft behaves with respect to these parameter values. In particular, the boundaries between safe and unsafe behavior are of major importance. A typical example is the stall speed v_{stall} of an aircraft. If the current airspeed V_{as} becomes smaller than v_{stall}, the aircraft stalls and and is in severe danger of crashing. However, the stall speed is not a fixed number. It depends on a number of variables, most notably, weight, lift-over-drag, altitude, turning/climbing, or environmental effects like icing [1].

Figure 1 shows the safety boundaries for a commercial transport (taken from the final report of the ill-fated Air France flight AF447 [6]). When considering the safe and unsafe regions in Figure 1 with respect to aircraft speed and altitude, the stall speed clearly indicates boundaries, which separate safe areas where controlled flight is possible, from unsafe areas where the aircraft stalls. The non-linearity of the boundaries are due to physical laws and the specific aircraft design.

For most complex systems, like aircraft, their behavior can only be obtained by system simulation. A typical example might be an adaptive flight control system, where the behavior of the aircraft control dynamically adapts towards counteracting aircraft damage or unexpected environmental

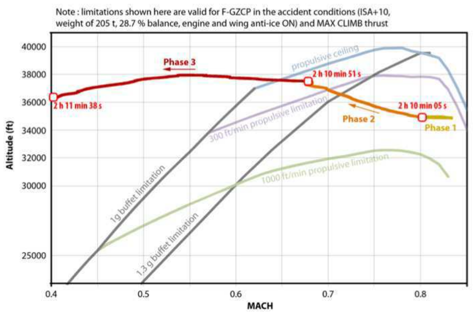


Fig. 1: Safety boundaries (grey) for aircraft over speed (Mach) and altitude different scenarios [6].

conditions. Also many computerized systems for air traffic control, e.g., for detection of safety violations, exhibit a highly complex behavior due to advanced and complex algorithms and their hybrid nature. Such systems often have to be treated as “black boxes”, i.e., there is no information available about their internal structure or underlying design.

Safety boundaries for such systems can be determined by brute-force experiments: for each possible combination of input values or parameter values, a simulation is started and its outcome is labeled “safe” or “unsafe” accordingly. In practice, such an approach usually fails because a large number of parameters (and parameter values) prohibit the exhaustive exploration of the parameter space. Obviously care must be taken that no safety-boundary is missed, because “holes” in an operational envelope can have disastrous consequences and must therefore be recognized properly. For example, a software problem caused multiple computer crashes on-board a group of six F-22 Raptors when they crossed the 180th meridian of longitude (the international date line) [7].

A safety-boundary, once located, must be somehow characterized and described such that it can be understood by the domain expert. Techniques using universal function approximators, like neural networks can be used to learn a boundary of any shape, but the results are hard to interpret by the domain expert. In practice, domain experts often have a good idea on how the boundary should look like. Based upon background knowledge and experience, they might provide information like ”this boundary should have a sphere-like structure”, but can give no indication on center and size of the sphere. For example, the stability boundary of an adaptive flight control system can be expressed as an ϵ environment in the dimensions of the parameters $e_1, \ldots, e_n$ [2]. Then, the boundary shape could be expressed as $\sum_i \lambda_i e_i^2 = \epsilon^2$ for unknown $\lambda > 0$ and $\epsilon > 0$. We believe that such background information substantially helps for locating multiple boundaries and determining their true shapes.

In this paper, we describe an advanced hierarchical statistical approach toward boundary detection and shape estimation. By using active learning techniques as well as efficient data structures and algorithms, we develop a framework that can detect boundaries with a low number of required simulation experiments. Shape and location parameters of boundaries are estimated using an advanced Bayesian approach, which is capable of providing high-quality feedback on the domain expert's input in the presence of noise and system dynamic uncertainties.

The remainder of this paper is structured as follows: Section II gives an overview of our active learning and shape detection approach. In Section III, we focus on boundary finding and present a metric for active learning that is aware of the boundary location. Section IV presents our Bayesian approach to modeling and analyzing the boundary shapes. Experiments with artificial data sets, a simulation of a NASA neuro-adaptive flight control, and a detection system for aircraft separation violations are discussed in Section V. Section VI concludes.

II. Methodology Overview

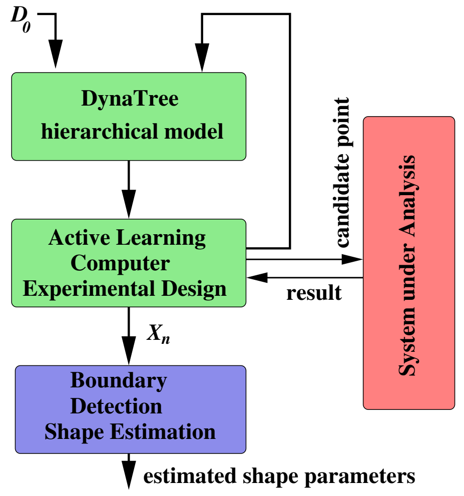
A. Algorithm Overview

We propose a sequential method for the estimation of parameterized boundary shapes in high dimensional spaces. A dictionary of shape classes is provided by the domain expert. Additional constraints on the parameters, e.g., parameter ranges and other prior information can be given. Typical examples for such shape classes include (hyper-)surfaces, polygons, spheres, or ellipses.

We represent our boundary problem as learning the response surface for the function f, where $f(x) = 1 + \epsilon$ if the experiment succeeds and $f(x) = 0 + \epsilon$ otherwise for some small $\epsilon > 0$. In this representation a boundary is determined by points x with $f(x) = 0.5$. This representation allows us to formulate powerful methods to select the next data point, which is explained later.

Our framework is depicted in Figure 2. Given an set of labeled data D_0, an initial classifier is built, which provides an initial partitioning of the space and provides the information to estimate posteriors over given sets of data points. Then, candidate points (i.e., sets of input parameters) are iteratively selected by the algorithm and handed over to the computer experiment, which executes the system under consideration using the given parameters and returns a categorical result (success or failure). Since each run of the simulator requires substantial computational resources, the overall number of new data points should be kept as small as possible. By adding new data points, the classifier will be extended and improved with the main goal of identifying and characterizing the boundaries.

Our algorithm is based upon the sequential classification and regression framework as given by DynaTree [8], [9]. It features dynamic regression trees and a sequential tree model. Particle learning for posterior simulation makes DynaTrees a good candidate for applications, where new data points are processed sequentially. In our architecture, the classifier is represented by a Dynatree at any given point in time. After adding a number of new data points, the current classifier is used to estimate a set of data points, which are close to the current prediction of the boundary. This is a subset of data points from a regular grid or a Latin hyper square, for which their entropy measure is high (classification representation) or the estimated response value is close to 0.5.



D_0
DynaTree hierarchical model
Active Learning Computer Experimental Design
candidate point
result
System under Analysis
X_n
Boundary Detection Shape Estimation
estimated shape parameters

Fig. 2: Architecture overview

This set of data points is then used to estimate the best parameters Θ for each of the the boundary shapes, together with a confidence interval for each of the parameters.

The candidate point selection in this active learning algorithm can use as much information as is available at the current stage, for example, prior information given by the domain expert. It then selects a new point (i.e., set of input parameters), for which the label is obtained by running the system simulator. Next we present the individual steps in detail.

III. Active Learning and Expected Improvement

A. Finding boundaries

Each data point describing one simulation run (experiment) is defined as $\mathbf{x} = \langle P_1, \ldots, P_p \rangle$, where P_i are the input parameter settings and the outcome $o(\mathbf{x}) \in \{success, failure\}$. Thus these data define a classification problem with $C = 2$ classes. Informally, a boundary can be found between regions, where all experiments yield success $p(\mathbf{x} = success) = 1$ and those, where the experiments do not meet the success criterion $p(\mathbf{x} = failure) = 1$. Therefore, we can define a point $\mathbf{x}$ to be on the boundary if $p(\mathbf{x} = success) = p(\mathbf{x} = failure) = 0.5$. Although this condition can easily be generalized to more than two classes, in this work, we will focus on $C = 2$.

A common metric to characterize points on the boundary is based upon the entropy. The entropy $entr = -\sum_{c \in c_1,..,c_C} p(\mathbf{x} = c) \log p(\mathbf{x} = c)$ becomes maximal at the boundary. In cases of more than two classes, [10] uses a BVSB (Best vs. Second Best) strategy. [11] defines a metric *advantage* as essentially $adv(\mathbf{x}) = |p(\mathbf{x} = success) - p(\mathbf{x} = failure)|$. Then [11] considers points with minimal advantage to be close to the boundary. In the general case with more than two dimensions, [11] proposed to use the difference between the two most likely classes.

In general, there are two basic methods: explicitly from knowledge of the classification function, or by treating the classifier as a black box and finding the boundaries numerically. For some classifiers it is possible to find a simple parametric formula that describes the boundaries between groups, for example, LDA or SVM. Most classification functions can output the posterior probability of an observation belonging to a group. Much of the time we do not look at these, and just classify the point to the group with the highest probability.

Points that are uncertain, i.e., have similar classification probabilities for two or more groups, suggest that the points are near the boundary between the two groups. For example, if a point is in Group 1 with probability 0.45, and in Group 2 with probability 0.55, then that point will be close to the boundary between the two groups. We can use this idea to find the boundaries. If we sample points throughout the design space we can then select only those uncertain points near boundaries. The thickness of the boundary can be controlled by changing the value, which determines whether two probabilities are similar to each other or not. Ideally, we would like this to be as small as possible so that our boundaries are accurate. Some classification functions do not generate posterior probabilities. In this case, we can use a k-nearest neighbors approach. Here we look at each point, and if all its neighbors are of the same class, then the point is not on the boundary and can be discarded. The advantage of this method is that it is completely general and can be applied to any classification function. The disadvantage is that it is slow ($O(n^2)$), because it computes distances between all pairs of points to find the nearest neighbors. In general, finding of the boundaries faces the "curse of dimensionality": as the dimensionality of the design space increases, the number of points required to make a perceivable boundary (for fitting or visualization purposes) increases substantially. This problem can be attacked in two ways, by increasing the number of points used to fill the design space (uniform grid or random sample), or by increasing the thickness of the boundary.

B. Active Learning

Computer simulation of a complex system like those discussed above, is frequently used as a cost-effective means to study complex physical and engineering processes. It typically replaces a traditional mathematical model in cases where such models do not exist or cannot be solved analytically.

Active learning, or sequential design of experiments (DOE), in the context of estimating response surfaces (in our case boundaries), is called adaptive sampling. Adaptive sampling starts with a relatively small space-filling input data, and then proceeds by fitting a model, estimating predictive uncertainty, and choosing future samples with the aim of minimizing some measure of uncertainty, or trying to maximize information. We perform active learning with new data until the boundary is characterized with sufficient accuracy and confidence, and the whole space has been sufficiently explored to not miss any boundaries in the space.

Consider an approach which maximizes the information gained about model parameters by selecting the location $\mathbf{x}$, which has the greatest standard deviation in predicted output. This approach has been called ALM for Active Learning-Mackay, and has been shown to approximate maximum expected information designs [12]. An alternative algorithm is to select Σ^2 minimizing the expected reduction in the squared error averaged over the input space [13]. This method is called ALC for Active Learning-Cohn. Rather than focusing on design points which have large predictive variance, ALC selects configurations that would lead to a global reduction in predictive variance.

The ALM/ALC algorithms are suitable for classification but not primarily for boundary detection [14]. These heuristics are in general not suited for the boundary-finding task because they do not take the specifics of the boundaries into account and they tend to also explore sparsely populated regions far away from current boundaries.

C. Boundary Expected Improvement

Finding a boundary between two classes can be considered as finding a contour with $a = 0.5$ in the response surface of the system response. Inspired by [16] and work on contour finding algorithms, we loosely follow [15], and define our heuristics by using an improvement function. In order to use the available resources as efficiently as possible for our contour/boundary finding task, one would ideally select candidate points which lie directly on the boundary, but that is unknown. Therefore, new trial points x are selected, which belong to an ϵ-environment around the current estimated boundary. This means that $0.5 - \epsilon \leq \hat{y}(x) \leq 0.5 + \epsilon$. New data points should maximize the information in the vicinity of the boundary. Following [16] and [15], we define an improvement function for x as

$$I(x) = \epsilon^2(x) - \min\{(y(x) - 0.5)^2, \epsilon^2(x)\}$$

Here, $y(x) \sim N(\hat{y}(x), \sigma^2(x))$, and $\epsilon(x) = \alpha\,\sigma(x)$ for a constant $\alpha \geq 0$. This term defines an ϵ-neighborhood around the boundary as a function of $\sigma(x)$. This formulation makes it possible to have a zero-width neighborhood around existing data points. For boundary sample points, $I(X)$ will be large when the predicted $\sigma(x)$ is largest.

The expected improvement $E[I(x)]$ can be calculated easily following [15] as

$$\begin{aligned}E[I(x)] = &-\int\limits_{0.5-\alpha\sigma(x)}^{0.5+\alpha\sigma(x)} (y - \hat{y}(x))^2 \phi\left(\frac{y - \hat{y}(x)}{\sigma(x)}\right) dy \\ &+2(\hat{y}(x) - 0.5)\sigma^2(x)\left[\phi(z_+(x)) - \phi(z_-(x))\right] \\ &+(\alpha^2\sigma^2(x) - (\hat{y}(x) - 0.5)^2)\left[\Phi(z_+(x)) - \Phi(z_-(x))\right],\end{aligned}$$

where $z_\pm(x) = \frac{0.5-\hat{y}(x)}{\sigma(x)} \pm \alpha$, and ϕ and Φ are the standard normal density and cumulative distribution, respectively. Each of these three terms are instrumental in different areas of the space. The first term summarizes information from regions of high variability within the ϵ-band. The integration is performed over the ϵ-band as $\epsilon(x) = \alpha\,\sigma(x)$. The second term is concerned with areas of high variance farther away from the estimated boundary. Finally, the third term is active close to the estimated boundary. After the expected improvement has been calculated, the candidate point is selected as the point, which maximizes the expected improvement.

IV. Boundary shape

1) Notation: Suppose there are m shape classes $M_1, \ldots, M_m$ with $m \geq 1$, which are parameterized by $\Theta_1, \ldots, \Theta_m$. The task is to fit l shapes $S_1, \ldots, S_l$, $l \geq 1$, where $S_1 = (i_1, \Theta_1), \ldots, S_l = (i_l, \Theta_l)$ and i_j denotes the shape class for the j^{th} shape with $i_j \in \mathcal{M} = \{M_1, \ldots, M_m\}$. Several of the i_j can be the same to accommodate more than one shape belonging to the same class. The Θ_i should be different since we do not want to represent the same boundary shape twice. We also seek to determine the correct number of shapes l that represents the input point set X_n.

For example, we may consider the $m = 2$ shape classes $M_1 = \textit{hyperplane}$ and $M_2 = \textit{sphere}$ in R^d. Hyperplanes are represented as $a_1x_1 + \cdots + a_dx_d + a_{d+1} = 0$ with parameter vector $\Theta_1 = (a_1, \ldots, a_d, a_{d+1}) \in R^{d+1}$. In the same d-dimensional space, a sphere of radius r with center $c = (c_1, \ldots, c_d)$ is described by $(x_1 - c_1)^2 + \cdots + (x_d - c_d)^2 = r^2$ with parameter vector $\Theta_2 = (c, r) \in R^{d+1}$.

2) What is a Good Shape Set $\mathcal{S}$ for an Input Point Set X_n?: There are three conditions that specify when a shape set $\mathcal{S}$ provides a good fit to the data X_n:

(i) *Summary*: each point on a shape $S \in \mathcal{S}$ is close to some classifier boundary point in X_n,

(ii) *Completeness*: each classifier boundary point in X_n is close to some shape point on some shape $S \in \mathcal{S}$, and

(iii) *Minimality*: the shapes in $\mathcal{S}$ are as different from one another as possible.

Condition (i) encourages each shape $S \in \mathcal{S}$ to be a good *summary* of one of the parts of the boundary of classifier P_n. That is, the points of a shape should lie along high entropy areas of P_n.

Condition (iii) encourages that shape set $\mathcal{S}$ to be *minimal*; i.e., $\mathcal{S}$ will not use any extra shapes to form a complete summary of the boundaries of classifier P_n. A complete summary $\mathcal{S}$ (i.e., one satisfying (i) and (ii)) remains a complete summary if one of its shapes $S \in \mathcal{S}$ is added to $\mathcal{S}$ either exactly or after a small perturbation. In fact, adding a small perturbation $\widehat{S}$ of S may actually improve completeness slightly since $\widehat{S}$ can be even closer to some high entropy points than S. And if S were a good summary, then so too would $\widehat{S}$. We need the minimality condition (iii) to be able to obtain the simplest (i.e., smallest) shape set that is a complete summary of the classifier boundaries.

3) Statistical Modeling: The shape set posterior is

$$P(\mathcal{S}|X_n) = \frac{P(X_n|\mathcal{S})P(\mathcal{S})}{P(X_n)} \propto P(X_n|\mathcal{S})P(\mathcal{S}).$$

We build the posterior model $P(\mathcal{S}|X_n)$ by modeling the likelihood $P(X_n|\mathcal{S})$ and the shape set prior $P(\mathcal{S})$. In the posterior $P(\mathcal{S}|X_n) \propto P(X_n|\mathcal{S})P(\mathcal{S})$, we will model the likelihood $P(X_n|\mathcal{S})$ to encourage *completeness* and the prior $P(\mathcal{S})$ to encourage distance between shapes and therefore *minimality*. It makes sense that the data likelihood accounts for completeness because completeness requires observed points to be close to a shape and the observed points arise from the ground truth shapes with the addition of noise. We will encourage good *summary* using a Bayesian loss function that grows with increasing distance of the shapes to the point set. Finally, we determine the number of shapes l by minimizing the expected posterior loss.

a) Likelihood: Our likelihood will encourage completeness. For the completeness condition (ii), we are interested in making the average squared distance $\overline{D}^2_{X_n,\mathcal{S}}$ of boundary points in $X_n = \{x_1, \ldots, x_n\}$ to shapes in $\mathcal{S}$ small:

$$\overline{D}^2_{X_n,\mathcal{S}} = \frac{\sum_{x \in X_n} d^2_{X_n,\mathcal{S}}(x)}{|X_n|} = \frac{\sum_{j=1}^n d^2_{X_n,\mathcal{S}}(x_j)}{|X_n|}, \quad (1)$$

where

$$d^2_{X_n,\mathcal{S}}(x) = \min_{s \in \mathcal{S}} ||x - s||_2^2 \quad (2)$$

is the minimum squared distance of a high entropy point x to a point on any shape in the collection $\mathcal{S} = (S_1, \ldots, S_l)$.

An observed point $x_j \in X_n$ is assumed to have been generated from a shape S_{z_j}, where z_j gives the shape number that explains x_j. Given z_j, we model the likelihood of x_j as a decreasing function of the minimum distance from x_j to S_{z_j}. The closer x_j is to shape S_{z_j}, the higher the likelihood of x_j. The observations x_j are assumed to be independent and modeled as

$$x_j = s_j + \varepsilon_j = s_j + r_j n_j, \quad r_j \sim N(0, \sigma_r^2),$$

where n_j is a unit normal to S_{z_j} at s_j and $r_j = (x_j - s_j) \cdot n_j$. Here the noise vector $\varepsilon_j = r_j n_j$ is along a unit normal n_j to the shape S_{z_j} at the closest shape point s_j to x_j. The scalar residual r_j is the signed distance along n_j from the shape S_{z_j} to x_j. We model the observation error ε_j by modeling the signed residual as a $N(0, \sigma_r^2)$ random variable.

Note that the squared residual r_j^2 is just the minimum distance squared from x_j to the closest point s_j on shape S_{z_j}:

$$r_j^2 = \min_{s \in S_{z_j}} ||x_j - s||_2^2,$$

where the minimum occurs at $s = s_j$. Let $Z = (z_1, \ldots, z_n)$. Assuming independence of points and that x_j depends only on shape S_{z_j}, then $P(X_n|Z, \mathcal{S}) = \prod_{j=1}^n P(x_j|z_j, S_{z_j}) = \prod_{j=1}^n N(r_j|0, \sigma_r^2)$. Since $r_j \sim N(0, \sigma_r^2)$, it follows that

$$P(X_n|Z, \mathcal{S}) = K\sigma_r^{-n} \exp\left(-\frac{1}{2\sigma_r^2} \sum_{j=1}^n \min_{s_j \in S_{z_j}} ||x_j - s_j||_2^2\right), \quad (3)$$

for a constant K. Note that if the observed point set X_n is close to the shapes in $\mathcal{S}$, then $P(X_n|Z, \mathcal{S})$ is high. This statement assumes, of course, that the correct shape S_{z_j} explaining each point x_j has also been identified.

We can obtain the likelihood $P(X_n|\mathcal{S})$ by modeling $Z|\mathcal{S}$ and integrating out Z as in $P(X_n|\mathcal{S}) = \int_Z P(X_n|Z, \mathcal{S})P(Z|\mathcal{S})dZ$. We could, for example, model $Z|\mathcal{S}$ by modeling a count vector $C = (c_1, \ldots, c_l)$ which holds the number of observations c_i explained by shape S_i. Here $c_i = \sum_{j=1}^n 1_{z_j = i}$. We can encourage good summary by modeling $C \sim \text{multinomial}(n, (1/l, 1/l, \ldots, 1/l))$ where each of the l shapes in $\mathcal{S}$ has the same probability $1/l$ of generating an observed point. This would make shape sets with any shapes that are from the data quite unlikely because we would expect to see points around each shape according to the given multinomial distribution.

It is difficult, however, to optimize over shape sets with the hidden random variables Z in our models. Instead, we make a simple but accurate and effective approximation in our models and assume that the shape S_{z_j} that explains observation x_j is the shape in $\mathcal{S}$ which is closest to x_j. Thus we replace the minimization in equation (3) over $s_j \in S_{z_j}$ with a minimization $s_j \in \mathcal{S}$ over the entire shape set to obtain the approximation

$$P(X_n|\mathcal{S}) = K\sigma_r^{-n} \exp\left(-\frac{1}{2\sigma_r^2}\sum_{j=1}^{n}\min_{s_j\in\mathcal{S}}||x_j - s_j||_2^2\right). \quad (4)$$

From equations (1),(2), we can see that the inner sum in equation (4) is just a scaled version $|X_n|\overline{D}^2_{X_n,\mathcal{S}}$ of our completeness measure. We can easily write our likelihood in terms of the completeness measure $\overline{D}^2_{X_n,\mathcal{S}}$. To do so cleanly, define $\sigma^2_{\text{complete}} = \sigma_r^2/|X_n|$. Then

$$P(X_n|\mathcal{S}) = K\sigma^{-n}_{\text{complete}} \exp\left(-\frac{1}{2\sigma^2_{\text{complete}}}\overline{D}^2_{X_n,\mathcal{S}}\right),$$

where another constant factor has been absorbed into K.

b) Shape Set Prior: We build the shape set prior $P(\mathcal{S})$ based on the distances of points on each shape S_i to the rest of the shape set $\mathcal{S}_{-i} = \mathcal{S}\backslash\{S_i\}$. To keep shapes apart from one another, we want a large average squared distance from points on each shape to the rest of the shapes. Let $d^2_{S_i,S_j}(s_i)$ be the minimum squared distance of a point $s_i \in S_i$ to another shape S_j:

$$d^2_{S_i,S_j}(s_i) = \min_{s_j\in S_j}||s_i - s_j||_2^2.$$

Then the squared distance of $s_i \in S_i$ to the shape set $\mathcal{S}_{-i}$ is

$$d^2_{S_i,\mathcal{S}_{-i}}(s_i) = \min_{S_j\in\mathcal{S}_{-i}} d^2_{S_i,S_j}(s_i),$$

which finds the closest point in the rest of the shapes $\mathcal{S}_{-i}$ to $s_i \in S_i$. Finally we average the inter-shape squared distances over all points on all shapes to get

$$\overline{D}^2_{\mathcal{S}} = \frac{\sum_{S_i\in\mathcal{S}}\sum_{s_i\in S_i} d^2_{S_i,\mathcal{S}_{-i}}(s_i)}{\sum_{S_i\in\mathcal{S}}|S_i|}$$

To keep the shapes apart a priori, we want $\overline{D}^2_{\mathcal{S}}$ to be large, indicating that on average the inter-shape distance is large. Equivalently, $1/\overline{D}_{\mathcal{S}}$ should be small. Therefore we model the prior for $\mathcal{S}$ using the normal distribution

$$\mathcal{S} \sim N(\overline{D}^{-1}_{\mathcal{S}}; 0, \sigma^2_{\text{shapesim}}).$$

c) Bayesian Loss: Next we define a Bayesian loss function that encourages good summary. We can think of the summary condition (i) as requiring a small distance from each shape $S \in \mathcal{S}$ to the set of classifier boundary points X_n. Let $d^2_{S,X_n}(s)$ denote the squared distance from a shape point $s \in S$ to the point set X_n:

$$d^2_{S,X_n}(s) = \min_{x\in X_n}||s - x||_2^2.$$

We capture the average squared distance $\overline{D}^2_{\mathcal{S},X_n}$ from the shape set $\mathcal{S}$ to the input points X_n by averaging over all points on all shapes in $\mathcal{S} = (S_1, \ldots, S_l)$:

$$\overline{D}^2_{\mathcal{S},X_n} = \frac{\sum_{a=1}^{l}\sum_{s\in S_a} d^2_{S_a,X_n}(s)}{\sum_{a=1}^{l}|S_a|}.$$

We define our Bayesian loss function as

$$\text{loss}(\mathcal{S}, X_n) = \lambda_{\text{summary}}\overline{D}^2_{\mathcal{S},X_n}$$

The smaller the distance from each shape in $\mathcal{S}$ to the point set X_n, the smaller the loss. Thus minimizing the loss will encourage good summary.

4) Shape Fitting Method: Our shape fitting method has two main steps:

Step 1 Minimize the expected posterior loss

$$g(l) = E[\text{loss}(\mathcal{S}, X_n)], \quad |\mathcal{S}| = l$$

over l to obtain the number of shapes l^*

Step 2 Compute the MAP shape set $\mathcal{S}^{*,l^*}$ for sets of size l^*

a) Determining the Number of Shapes: We assume that we can apriori limit the number of shapes l to some set $\mathcal{L}$. For example, if we know that there will not be more than five boundaries then we can set $\mathcal{L} = \{1, 2, 3, 4, 5\}$.

For each $l \in \mathcal{L}$, we compute the expected posterior loss

$$g(l) = E[\text{loss}(\mathcal{S}, X)] = \int_{\{\mathcal{S}:|\mathcal{S}|=l\}} \text{loss}(\mathcal{S}, X_n)\widehat{P}(\mathcal{S}|X_n)d\mathcal{S}.$$

Here we denote the shape set posterior probability distribution for shape sets with a fixed number of shapes as $\widehat{P}(\mathcal{S}|X_n)$. Then we choose the number of shapes to minimize the expected posterior loss:

$$l^* = \arg\min_{l\in\mathcal{L}} g(l).$$

5) Our Shape Set Posterior Sampling Method: For a fixed shape set size $|\mathcal{S}| = l$, we will draw samples from the posterior $P(\mathcal{S}|X_n) \propto P(X_n|\mathcal{S})P(\mathcal{S})$ using an iterative procedure. Shape set samples $\mathcal{S}$ with a small value for

$$-\log(P(X_n|\mathcal{S})P(\mathcal{S})) = -\log(P(X_n|\mathcal{S})) - \log(P(\mathcal{S}))$$

should be more likely to occur.

V. Experiments and Results

A. Evaluation on Artificial Data Sets

1) Active Learning: We illustrate the behavior of our approach using a 2D artificial data set and a quadratic boundary function normalized to a unit square. Starting with a low number of $N_{init} = 126$ randomly selected initial data points, the active learning procedure selects $N = 500$ new data points according to different candidate selection rules (random, ALC, ALM, EI, and boundary EI). N has been selected this large for visualization purposes. Figure 3 shows, how the different selection algorithms behave. Our goal is to find many data points near the threshold curve in order to enable accurate representation and to facilitate subsequent shape estimation.

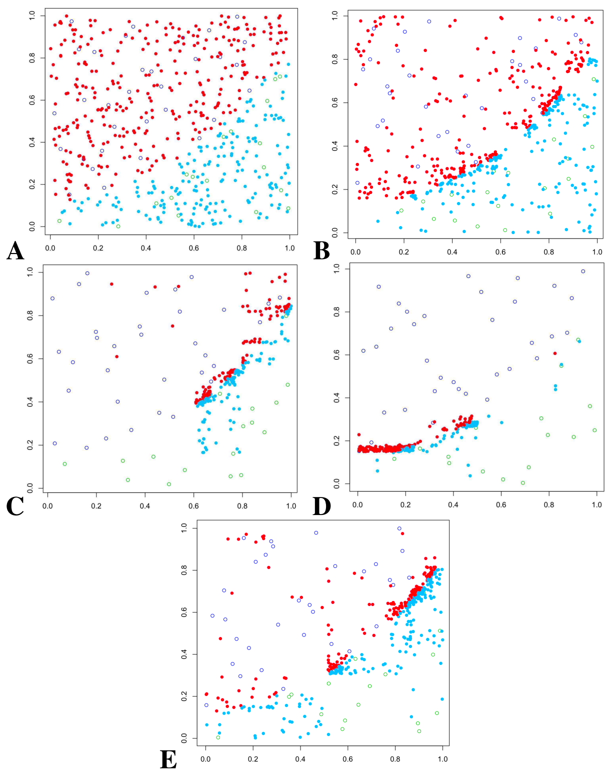


Fig. 3: Candidate points during active learning: (A) random selection, (B) ALC, (C) ALM, (D) EI, and (E) boundary-EI. Circles: initial data points. Solid: points added during active learning (colored according to experiment outcome).

On the other hand, the entire area should be considered as well in order not to miss any other boundary.

The random Monte-Carlo style selection (Figure 3A) needs a prohibitively large N for reasonable results. The classical ALC [13] (Figure 3B) finds many points near the boundary, but still too many data points are away from the curve, demanding large N. Other algorithms are too localized and do not even explore the entire threshold curve (Figure 3C, D). Our approach (Figure 3E) tries to find a suitable balance between both requirements.

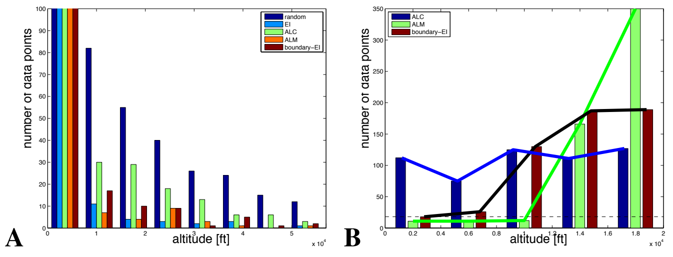


Fig. 4: A: Histogram for distances d of candidate points from boundary. Leftmost bars cropped for better visibility. B: Histograms of boundary coverage.

There is a trade-space between closeness of selected points to the safety boundary and a good curve coverage. For example, a greedy algorithm might always select the same point near the threshold (high closeness) but extremely low coverage. Figure 4 shows this trade space. For each of the newly added points, we calculate d as the minimal distance of that point to the boundary. Obviously, small values should be preferred, as such points close to the threshold help to accurately estimate its shape. Figure 4A shows a histogram of distances d for various update rules. Whereas random and ALC have large numbers of points that are far away from the threshold surface, ALM seems to perform best for this metric. However, Figure 4B reveals that ALM only covers a very small portion of the threshold surface. Random selection provides the best coverage here. With our analysis goal in mind, our boundary-aware EI metric features a good overall coverage and a high density of points close to actual threshold surface.

Our boundary metric is parameterized by the parameter α (see Section III-C). This parameter influences the width of the "band" around the threshold surface that is considered for the selection of the candidate point. Figure 5 shows runs with several values of α. It seems that values around $\alpha = 0.8$ produce the best results; values of α that are too small or too large tend to lead to a situation, where the new points are located too far from the threshold surface.

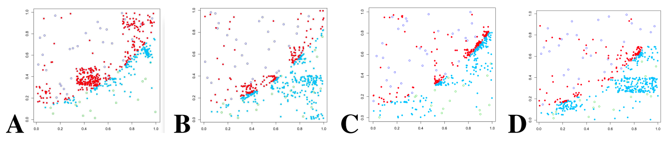


Fig. 5: Boundary-EI for $\alpha = 0.2, 0.5, 0.8, 1$

The performance of the active learning method and the shape estimation can also be assessed by analyzing the algorithm convergence, i.e., how many new data points are necessary, before the estimated shape parameters are close to the ground truth that has been used to generate the artificial data set. For example, for a single hyperplane boundary, we obtain $C = \sqrt{|\hat{\theta}| - |\theta|}$, where θ and $\hat{\theta}$ are the ground truth and the estimated parameters, respectively. Figure 6 compares the convergence of random selection, ALC, and our method over 10 runs and shows a superior performance of our method.

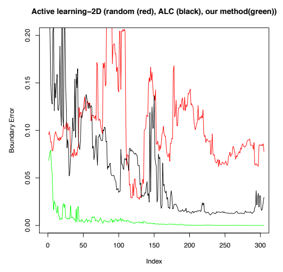


Fig. 6: Convergence C for random (red), ALC (black), and boundary-EI (green) over learning iterations.

2) Shape Selection: To assess the performance of our shape selection and estimation method, we carried out experiments with boundaries in the shape of hyperspheres. Table IA shows the results in the two-dimensional case. With a total 126+700 data points, sphere S_1 was correctly recognized in all 25 runs; S_2 was only correctly estimated in 5 of the runs. The table shows the ground-truth values, means and variances for the successful runs. 74 data points were selected for shape estimation.

TABLE I: Parameters for 2D (A) and 5D spheres (B).

	true value	$\hat{\mu}(\hat{\sigma}^2)$
x_0	0.3	0.295(1.4e-5)
y_0	0.3	0.289(3e-5)
r_0	0.2	0.20(6.6e-6)
x_1	0.7	0.715(8.5e-5)
y_1	0.7	0.72(8.6e-5)
r_0	0.2	0.20(5.2e-5)

	true value	$\hat{\mu}(\hat{\sigma}^2)$
c_1	0.3	0.29(7e-3)
c_2	0.3	0.26(5e-3)
c_3	0.3	0.32(8e-3)
c_4	0.3	0.31(7e-3)
c_5	0.3	0.27(9e-3)
r	0.3	0.29(8e-4)

Table IB shows the situation in a 5D space. With the centers located at $\vec{c}_1 = (0.3, 0.3, 0.3, 0.3, 0.3)^T$, and $\vec{c}_1 = (0.7, 0.7, 0.7, 0.7, 0.7)^T$, respectively and radius $r = 0.3$. Active learning selected 1000 data points, 155 of which were selected for shape estimation. Here, the results are much worse. E.g., the second sphere was not recognized in any of the 10 runs, indicating that for a 5D space, the number of data points must be considerable larger. Future work will address this issue.

B. IFCS Data Set

The Intelligent Flight Control System (IFCS) is a damage-adaptive Neural Networks (NN) based flight control system developed by NASA and test-flown on a manned F-15 aircraft [2]. An on-line trained NN provides control augmentation to dynamically counteract damages to the aircraft. For our experiments, we considered this system as a black box, controlled by numerous parameters (e.g., NN weights, controller gains, or learning rate). A simulation run was considered to be successful, if, after an injected damage, the aircraft remained stable for at least 20 seconds. After an initial parameter sensitivity analysis, we selected the parameters w_p, w_q, w_r, K_{lat}, and ζ for further analysis, where the w_i are proportional gains of the controllers, K_{lat} the lateral stick gain, and ζ a damping coefficient. We generated a combinatorial data set of 32,768 data points, out of which 7,992 runs were successful.

A boundary over these parameters exist in a shape of a hypersphere. This spherical shape is a consequence of the IFCS design, and the shape can be described by $(\frac{w_p - x_0}{\phi_1})^2 + (\frac{w_q - \phi_2 - y_0}{\phi_3})^2 = (\frac{w_r - \phi_3 - z_0}{\phi_4})^2 = \zeta \times \phi_5 - K_{lat}$. This stability boundary is is parameterized by unknown ϕ_i. x_0, y_0, z_0 are design-time constants.

Figure 7A shows the actual and estimated boundary in a projection into w_p, w_q, and K_{lat}. For our shape fitting and estimating experiment, we used 1000+5000 data points. The shape parameters for the boundary in Figure 7B were estimated based upon 485 points near the boundary within an ϵ-band of width 0.2.

C. Experiments with TTSAFE

The Terminal Tactical Separation Assured Flight Environment (TTSAFE) is a software tool to predict violations of separation rules between aircraft near an airport [17]. TTSAFE uses novel algorithms for predicting the trajectories and encodes all complex terminal separation rules. Thus, we decided to analyze TTSAFE as a black box. The inputs to

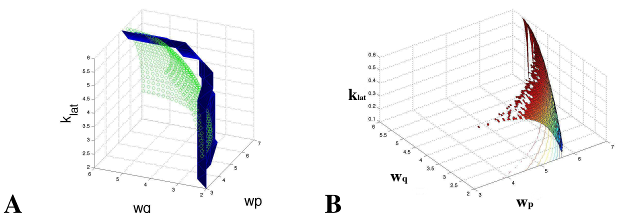


Fig. 7: A: Actual boundary (blue) and estimated boundary (green) over w_p, w_q, K_{lat}. B: Estimated hypersphere shape.

TTSAFE are the aircraft trajectories as well as a number of configuration parameters. During each run, the number of detected losses of separation ("numlos") as well as the time to loss of separation ("ttlos") is reported. For a given scenario, low numbers of numlos and late detection (small ttlos) mean unsafe situations. We analyzed the behavior of the TTSAFE system with respect to different parameter values and altitude biases. Here we focus on the altitude biases, which can be caused by noisy or faulty radar measurements.

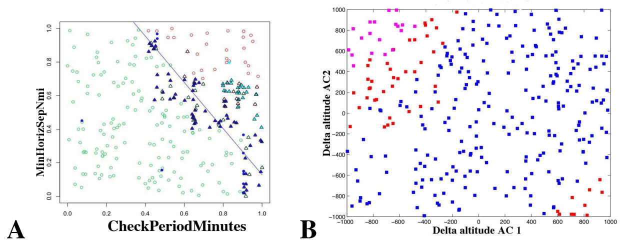


Fig. 8: A: Active learning of boundaries. B: Different values of numlos over different altitude biases for aircraft 1 and 2.

Figure 8A shows a boundary with respect to the TTSAFE parameters "MinHorizSepNmi" (minimal horizontal separation in Nautical miles) and how often the detection algorithm is executed "CheckPeriodMinutes" in a normalized space. Red and green circles denote the initial data points; triangles the points added by active learning. The blue line visualizes the estimated (hyperplane) boundary. A bias in the altitudes of the two converging aircraft can confuse TTSAFE: the aircraft might appear closer to each other than they actually are, causing a false alarm. Alternatively, they might appear farther apart than in reality. This is a dangerous situation as TTSAFE would not detect the loss of separation. In Figure 8B we show the space spanned by the two bias parameters. Blue dots are in a safe region (higher numbers of numlos); red and magenta dots represent unsafe regions.

Figure 9A shows results of the above experiment with respect to time to loss (ttlos). It turned out that this boundary is rather small and hidden in the entire parameter space (Figure 9B).

VI. Conclusion and Discussion

In this paper, we described a statistical modeling approach for the online detection and characterization of safety boundaries. Given a library of candidate shapes, location and parameters of the boundaries are estimated using an advanced Bayesian modeling approach. The results of our boundary analysis is provided in a form understandable by the domain

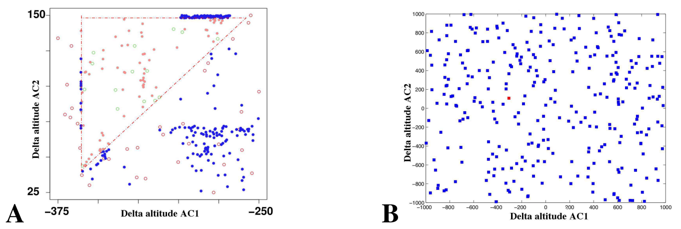


Fig. 9: A: Boundary of ttlos over altitude biases. New data points added by active learning with ttlos > 0.5 (blue) and ttlos < 0.5 (red). B: Boundary shown in A (red) in otherwise insensitive parameter space.

expert. Our active learning procedure uses a boundary-aware metric to quickly and effectively find new data points near the boundary. Experiments show that existing boundaries could be reliably found and their shape parameters estimated with confidence interval. Our boundary-EI substantially improved the selection of new data points.

In our future work, we will investigate whether a tighter coupling of the selection metric and the posteriors of shape detection can improve the active learning performance.

References

[1] B. P. Anderson, "Equations for stall speed," 2007. [Online]. Available: http://www.electraforge.com/brooke/flightsims/aces_high/stallSpeedMath/stallSpeedMath.html

[2] R. Rysdyk and A. Calise, "Fault tolerant flight control via adaptive neural network augmentation," *AIAA American Institute of Aeronautics and Astronautics*, vol. AIAA-98-4483, pp. 1722–1728, 1998.

[3] K. Gundy-Burlet, J. Schumann, T. Menzies, and T. Barrett, "Parametric analysis of Antares re-entry guidance algorithms using advanced test generation and data analysis," in *Proc. iSAIRAS 2008*, 2008.

[4] A. Calise and R. Rysdyk, "Nonlinear adaptive flight control using neural networks," *IEEE Control Sys. Mag*, vol. 21, no. 6, pp. 14–26, 1998.

[5] Department of Defense, "Interface standard: Flying qualities of piloted aircraft," Department of Defense, Tech. Rep., 1997.

[6] BEA, "Final Report on the Accident on 1st June 2009 on the Airbus A330-203," Bureau d'Enquêtes et d'Analyses pour la sécurité de l'aviation civile, Tech. Rep., 2012.

[7] F-22, "F-22 Raptor Stealthfighter," http://www.f-22raptor.com/index_airframe.php1992, 1992. [Online].

[8] M. A. Taddy, R. B. Gramacy, and N. G. Polson, "Dynamic trees for learning and design," *Journal of the American Statistical Association*, vol. 106, no. 493, pp. 109–123, 2011.

[9] R. B. Gramacy, "TGP: An R package for Bayesian nonstationary, semiparametric nonlinear regression and design by treed Gaussian process models," *J Stat Software*, vol. 19, no. 9, pp. 1–46, 2007.

[10] R. Gramacy and N. Polson, "Particle learning of Gaussian process models for sequential design and optimization," *Journal of Computational and Graphical Statistics*, vol. 20, no. 1, pp. 467–478, 2011.

[11] H. Wickham, "Practical tools for exploring data and models," Ph.D. dissertation, Iowa State, 2008.

[12] D. J. C. MacKay, "Information–based objective functions for active data selection," *Neural Computation*, vol. 4, no. 4, pp. 589–603, 1992.

[13] D. A. Cohn, "Neural network exploration using optimal experimental design," *Advances in Neural Information Processing Systems*, vol. 6, no. 9, pp. 679–686, 1996.

[14] R. B. Gramacy, "Bayesian treed gaussian process models," Ph.D. dissertation, University of California at Santa Cruz, Dec. 2005.

[15] P. Ranjan, D. Bingham, and G. Michailidis, "Sequential experiment design for contour estimation from complex computer codes," *Technometrics*, vol. 50, no. 4, pp. 527–541, 2008.

[16] D. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black box functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998.

[17] H. Tang, J. Robinson, and D. Denery, "Tactical conflict detection in terminal airspace," in *10th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, 2010.