IEEE Control Systems Letters paper presented at
2023 American Control Conference (ACC)
San Diego, CA, USA. May 31 - June 2, 2023

Geometry of Radial Basis Neural Networks for Safety Biased Approximation of Unsafe Regions

Ahmad Abuaish[†], Mohit Srinivasan[‡], Patricio A. Vela[†]

Abstract— **Barrier function-based inequality constraints are a means to enforce safety specifications for control systems. When used in conjunction with a convex optimization program, they provide a computationally efficient method to enforce safety for the general class of control-affine systems. One of the main assumptions when taking this approach is the *a priori* knowledge of the barrier function itself, i.e., knowledge of the safe set. In the context of navigation through unknown environments where the locally safe set evolves with time, such knowledge does not exist. This manuscript focuses on the synthesis of a zeroing barrier function characterizing the safe set based on safe and unsafe sample measurements, e.g., from perception data in navigation applications. Prior work formulated a supervised machine learning algorithm whose solution guaranteed the construction of a zeroing barrier function with specific level-set properties. However, it did not explore the geometry of the neural network design used for the synthesis process. This manuscript describes the specific geometry of the neural network used for zeroing barrier function synthesis, and shows how the network provides the necessary representation for splitting the state space into *safe* and *unsafe* regions.**

I. Introduction

Invariant set-based safe control synthesis [1] has become a favorable technique to enforce safety, as it provides theoretical guarantees when used to augment base controllers [2]. For closed dynamical systems, the invariant set is represented by the zero and positive level-sets of a continuously differentiable implicit function, typically termed *zeroing barrier function* (ZBF). Subsequently, for controlled systems, *control barrier functions* (CBFs) are introduced to represent the invariant set. In safety-critical applications, CBF-based controllers are designed to render safe regions in the state space a positively invariant set. Prevailing use of CBFs is in a point-wise optimization program solved via quadratic programming (QP) [3]. CBF-based QPs have been used in a wide range of applications in robotics such as collision avoidance [4], [5], multi-robot coordination and task satisfaction [6], [7], [8], and automotive applications [3].

Usually, a CBF is handcrafted based on *a priori* knowledge of the safe regions in the state space. However, there are applications where the safe regions evolve with time, with navigation being one. In these applications, it is critical to synthesize the CBF online using sensor measurements. Unfortunately, determining the true invariant region defined by a CBF is generally an NP-hard problem, but some techniques exist that can estimate the invariant region [9]. Since the existence of a zeroing barrier function is needed to formally confirm the existence of a CBF, this manuscript solely focuses on zeroing barrier function synthesis to separate state space into *safe* and *unsafe* regions. The ZBF is constructed from a two-layer kernel machine network trained from a labeled dataset of *safe* and *unsafe* samples. Further, the geometry of the kernel functions in the network is explored to efficiently partition the space. The approach is motivated by the Kolmogorov–Arnold representation theorem, which implies that two-layer neural networks may be capable of approximating continuous functions [10].

Recently, several data-driven approaches for constructing a CBF were proposed to account for uncertainties in either the system dynamics, unsafe regions, or both. One approach category for learning CBF's involves supervised offline learning. Instances include imitation learning where training data is generated by an expert actor or optimal control simulations [11], [12]. The offline nature lacks the ability to accommodate real-time changes in the environment. In contrast, *self-supervised* approaches permit online learning. Initial work on self-supervised Bayesian learning system of uncertain dynamics [13] with known barrier functions was merged with [14] to learn the system dynamics and an implicit function representation of the unsafe region [15]. In [14], a signed distance function representing obstacles is modeled as a deep neural network trained from range sensor data via stochastic gradient descent (SGD) with replay memory. Similarly, [16] presented the construction of a probabilistic occupancy map from a kernel-based logistic regression model trained from range sensor data via SGD. However, there were no hard constraints on misclassifying unsafe data points in the underlying SGD optimization process, which nullifies any possible theoretical safety guarantee.

Our previous work focused on creating a ZBF for navigation applications based on data collected from LiDAR sensors [17]. A two-layer network with Gaussian radial basis functions (GRBFs) was synthesized from this data. The first layer used sparsely distributed (over the domain) GRBF centers, while the second GRBF layer was learnt during the kernel support vector machine (kSVM) optimization process. The kSVM optimization specifications provided formal guarantees regarding the partitioning of the domain into *safe* and *unsafe* regions. However, the work did not discuss the geometry and associated properties of the GRBF network. This work analyzes the geometry of the two-layer network and the structure of the optimization problem to prove the existence of a ZBF with known partitioning properties.

This work was supported in part by the National Science Foundation under Award S&AS #1849333, by DARPA PAI, and by KACST Fellowship.

[†] School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA aabuaish@gatech.edu, pvela@gatech.edu

[‡] Ford Motor Company, Dearborn, USA mohit.s@ieee.org

The manuscript organization is as follows: Section II discusses the geometry of Gaussian kernel functions with respect to the first layer. Section III covers the second layer construction, geometry, and optimization specifications. Section IV discusses the qualification of the two-layer kernel machine network to be a zeroing barrier function along with a kernel basis selection strategy. Sections V and VI present case studies and concluding remarks, respectively.

II. Geometry of the Kernel Hilbert Space

GRBF Neural Networks (GRBF-NNs) have several properties ideal for zeroing barrier function creation. First they are universal approximators [18], second they exhibit locality [19], and third they partition space. Since the last property is less frequently mentioned, but often used, this section devotes attention to space partitioning, as it is essential for data-driven safe set generation.

The focus of this section is on GRBF-NNs as kernel machines whose kernel functions are radial basis functions (RBFs), which generally take the following form,

$$k(x_i, x_j) = \varphi(||x_i - x_j||\,; \sigma), \quad \text{for} \quad \varphi : \mathbb{R}^+ \to \mathbb{R}^+, \quad (1)$$

where $x_i, x_j \in \mathbb{R}^{n_d}$, $||\cdot||$ is a norm and σ is the scalar bandwidth, which influences the sensitivity of the basis function φ. A kernel machine based on radial basis functions generates a mapping through the use of multiple kernel function mappings with fixed argument elements c_j in a center set $\mathcal{C}$. For $c_j \in \mathcal{C}$ the kernel mapping is

$$\vec{k} : \mathbb{R}^{n_d} \to \mathcal{H}^{n_c}, \quad x \mapsto [k(x, c_1) \ \cdots \ k(x, c_{n_c})]^T. \quad (2)$$

where $n_c = |\mathcal{C}|$ and $\mathcal{H}^{\cdot}$ indicates that the output space is a Hilbert space. For this kernel machine to define a scalar function requires specifying $\alpha \in \mathbb{R}^{n_c}$ such that

$$f_{\text{KM}}(x) = \left\langle \alpha \, , \, \vec{k}(x) \right\rangle = \alpha^T \vec{k}(x). \quad (3)$$

Kernel machines permit more general function classes than RBFs in $\mathbb{R}^{n_d}$ (subsequent sections will use polynomial kernel functions). That said, RBFs have geometric properties that are implicitly exploited in kernel machine learning applications. Specializing to the case of the Gaussian radial basis function (or Gaussian kernel), let the kernel function be

$$k_G(x, c) = \exp\left(-\frac{||x - c||^2}{\sigma^2}\right). \quad (4)$$

Theorem 1. *The kernel mapping, with n_c Gaussian kernels, maps the input domain $\mathcal{D} \subset \mathbb{R}^{n_d}$ into a surface in the Hilbert space $\mathcal{H}^{n_c} \subset \mathbb{R}^{n_c}$ when $n_c \geq n_d + 1$ and there are $n_d + 1$ centers capable of defining a coordinate system in n_d.*

Proof. Showing that the kernel mapping is 1-1 establishes this property. The pre-image, $k^{-1}(\cdot, c_i)$, of each coordinate's kernel mapping is a sphere in the input space. The intersection of all spheres for all coordinate mappings establishes the pre-image point, which is unique only if the intersection is unique. Finite solutions for the intersections has been proven in the context of rigid body geometry for $n_c = n_d$ [20], with $n_c \geq n_d + 1$ necessary for a single valid solution. The requirement for the centers is that they lead to a basis of n_d vectors when using one of the points as the origin and using n_d other points to obtain the basis vectors relative to that origin. Each pre-image imposes a constraint on the degrees of freedom of the input point $x \in \mathbb{R}^{n_d}$, such that the $n_d + 1$ intersecting pre-image spheres do so at a single point. When $n_c > n_d + 1$, the additional pre-image constraints are redundant and effectively impose no constraints. The rank of the mapping is $n_d < n_c$, hence it maps to a surface of dimension n_d. □

Theorem 1 applies to any RBF with infinite support; an RBF with finite support will have similar properties but will include an ϵ-covering constraint. Basis functions generated from other norms also have similar properties but may require more centers to induce a 1-1 mapping. If the kernel is differentiable, then the 1-1 mapping is an embedding. Safe set generation with an appropriate kernel mapping will involve defining the concept of a *kernel embedding*[1] and its associated *kernel embedding inducing* data set.

Definition 1. *The kernel mapping $\vec{k} : \mathbb{R}^{n_d} \to \mathcal{H}^{n_c}$ is a* kernel embedding *if the center set $\mathcal{C} \subset \mathcal{D}$ generating the kernel mapping are such that it is 1-1 and the kernel function $k : \mathbb{R}^{n_d} \times \mathbb{R}^{n_d} \to \mathbb{R}$ is differentiable. The set of centers is called the* kernel embedding inducing set *(KEI set).*

Corollary 1. *Consider a finite set of points $\mathcal{X}_p \subset \mathbb{R}^2$ with an associated triangulation. Under a kernel embedding, each triangular region maps to a surface in $\mathcal{H}^{n_c}$ homeomorphic to a 2-simplex.*

Corollary 2. *Consider a finite set of points $\mathcal{X}_p \subset \mathbb{R}^3$ with an associated tetrahedralization. Under a kernel embedding, each tetrahedron maps to a surface in $\mathcal{H}^{n_c}$ homeomorphic to a 3-simplex.*

If the domain $\mathcal{D} \subset \mathbb{R}^{n_d}$ is a set of disconnected regions excluding points at infinity, then a kernel embedding will map to a set of disconnected surfaces. Similarly, a collection of non-intersecting triangulations/tetrahedralizations maps to a set of disconnected surfaces under a kernel embedding.

1) Geometry of a Kernel Embedding: The Gaussian kernel mapping outputs lie in the unit cube of $\mathcal{H}^{n_c} \subset \mathbb{R}^{n_c}$. Each center c_i in the set $\mathcal{C}$ maximizes its associated coordinate (evaluates to 1), which means that there is a neighborhood of c_i in $\mathcal{D}$ for which this same coordinate is also maximal for all points in the neighborhood. Points tending to infinity map to the origin in $\mathcal{H}^{n_c}$ since the Gaussian radial basis function tends to zero as the input radius tends to infinity.

Corollary 3. *For a kernel embedding defined using the Gaussian kernel, a compact domain $\mathcal{D}$ maps to a compact surface whose points lie outside of a ball centered at the origin. Furthermore, $\vec{k}(\mathbb{R}^{n_d}) \cup \{\vec{0}\}$ is a compact surface.*

[1]Not to be confused with the *kernel mean embedding* which is for probability distributions [21].

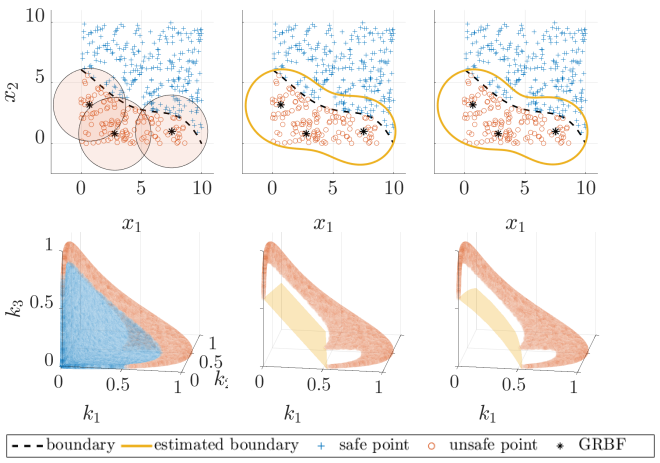

Fig. 1. Depiction of the input domain and the associated Hilbert space along with the GRBF centers and the separating boundaries and surfaces.

With the surface geometry of the kernel embedding $\vec{k}$ for a given KEI set $\mathcal{C}$ established, the next step is to consider partitions of the surface in $\mathcal{H}^{n_c}$. Given that the unbounded input space maps to a compact surface, defining a partition of the space is equivalent to defining a cutting surface transverse to the compact surface that divides space into positive and negative regions. The intersection of the compact surface with the cutting surface in the Hilbert space maps back to separating boundaries in the original space. The objective will be to define cutting surfaces by specific level-sets of an implicit function in the Hilbert space. In short, safe region generation in the input space involves construction of an implicit function in the Hilbert space based on sampled points with *safe* / *unsafe* labels.

The observation leading to Corollary 3 hints that the geometry induced by the KEI set plays a role in establishing the surface image in the Hilbert space. Here, we connect the KEI set to coverings of the input domain. For the sets and spaces defined here, define an ϵ-covering as follows:

Definition 1. *For $\mathbb{R}^{n_d}$ with the Euclidean norm (2-norm), let $\mathcal{D} \subset \mathbb{R}^{n_d}$. For $\epsilon > 0$, the set $\mathcal{C} \subset \mathcal{D}$ is an ϵ-cover for $\mathcal{D}$ if for every $x \in \mathcal{D}$ there exists a $c_i \in \mathcal{C}$ such that $||x - c_i|| \leq \epsilon$. Equivalently, $\mathcal{D} \subset \bigcup_i \mathcal{B}_\epsilon(c_i)$.*

In the context of a Gaussian radial basis neural network or Gaussian kernel machines, ϵ-covers partition the original space in a predictable manner in the kernel mapped Hilbert space. An equivalent ϵ-cover specification is the following ∞-norm inequality applied to the kernel mapping output:

$$\mathrm{C} = \left\{ x \in \mathbb{R}^{n_d} \mid \left|\left| \vec{k}(x) \right|\right|_\infty \geq \varphi(\epsilon;\sigma) = e^{-\epsilon^2/\sigma^2} \right\}. \quad (5)$$

If the KEI set is defined based on an ϵ-cover of $\mathcal{D} \subset \mathbb{R}^{n_d}$, then the kernel map will map points in $\mathcal{D}$ to points in $\mathcal{H}^{n_c}$ on a surface some minimal distance from the origin. Points in $\mathbb{R}^{n_d}$ receding from $\mathcal{D}$ will tend towards the origin in $\mathcal{H}^{n_c}$.

The top row in Figure 1 depicts a 2D example consisting of a collection of safe (blue +) and unsafe (orange ∘) points generated from a nonlinear separating boundary (dashed black curve), as well as three GRBFs (black *) centered in the unsafe region, $\mathcal{X}^u \subset \mathcal{D}$, that are an ϵ-cover. These three centers create a mapping of the 2D domain into a 3D Hilbert space, depicted in the bottom row where the orange surface is the unsafe region surface $\vec{k}(\mathcal{U}^*)$ and the blue surface is the safe region surface $\vec{k}(\mathcal{S}^*)$. The cutting surface sought is one that separates points "near" to the origin from those in $\vec{k}(\mathcal{U}^*)$. Two such surfaces are depicted in the second and third columns (bottom row) along with their level-set boundaries in the input space (top row). The next section describes how to derive cutting surfaces from solutions to constrained optimization problems.

III. Cutting Surfaces and Partitions

This section exploits the geometry of the kernel mapping to define cutting hyperplanes in its output Hilbert space. These hyperplanes translate to partitions of the input space. Initially, the cutting hyperplanes will be in the original Hilbert space and are defined by a single-layer kernel machine network. Next, a second layer is added to the kernel machine network to create cutting hyperplanes that translate to elliptical volumes in the original Hilbert space. More complex cutting surfaces are then explored based on the inclusion of positive (*unsafe*) and negative (*safe*) samples.

A. Single-Layer Kernel Machine Network Partitions

Constructing an implicit partition function with barrier function properties with a single-layer kernel machine involve identifying a cutting hyperplane for the collection of safe and unsafe data. Consider a kernel embedding built using the *unsafe* data where $\mathcal{C}$ is a KEI set and covering of the unsafe set for some ϵ. One cutting hyperplane results from the following linear program:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^{n_c}} \quad & \vec{1}^T \alpha \\ \text{s.t.} \quad & \alpha^T \vec{k}(x_i) \geq 1, \ \forall i \in \{\, l \mid x_l \in \mathcal{X}^u \,\} \\ & \alpha_j \geq 0, \ \forall j = 1, \ldots, n_c \end{aligned} \quad (6)$$

where $\vec{1} = [1 \cdots 1]^T \in \mathbb{R}^{n_c}$, and $\alpha \in \mathbb{R}^{n_c}$ are the coefficients of the separating hyperplane. The one level-set of $f_{\mathrm{KM}}(x) = \alpha^T \vec{k}(x)$ defines the hyperplane in the positive hyperoctant that is furthest from the origin and places all unsafe points in $\mathcal{X}$ in the positive half-plane. The linear program is guaranteed to have a solution:

Theorem 2. *Given a kernel embedding $\vec{k}$ defined from an ϵ-covering of $\mathcal{X}^u$, there is a hyperplanar splitting of $\mathcal{H}^{n_c}$ described by*

$$\mathcal{S} = \left\{ x \in \mathcal{D} \mid \left\langle \vec{b}\,,\, \vec{k}(x) \right\rangle < 1 \right\} \quad (7)$$

where $\mathcal{S} \subset \mathcal{S}^* \subset \mathcal{D}$ and $\mathcal{X}^u \subset \bar{\mathcal{S}}$.

Proof. Consider the covering set of centers $\mathcal{C}$. Generate a clustering of the unsafe data points $\mathcal{X}^{\mathrm{u}}$ based on the cluster assignment function $x^u \in \mathcal{X}^{\mathrm{u}} \mapsto \arg\max_i k(x^u, c_i)$ for $c_i \in \mathcal{C}$. For each cluster set $\mathcal{X}_i^{\mathrm{u}}$ find the minimum value $y^i = k(x, c_i)$ for $x \in \mathcal{X}_i^{\mathrm{u}}$. Each cluster minimum defines the

i-th coordinate intercept for a cutting plane in the Hilbert space, which defines $\vec{b}$. All *unsafe* points $\mathcal{X}^{\mathrm{u}}$ lie on the non-negative side, possibly with some *safe* points from $\mathcal{X}^{\mathrm{s}}$. Only *safe* points in $\mathcal{X}^{\mathrm{s}}$ lie on the negative side (it is half-plane containing origin). □

Corollary 4. *By virtue of defining a splitting with a kernel embedding, this same operation generates a splitting of the original domain $\mathcal{D}$, and likewise the full space $\mathbb{R}^{n_d}$.*

The second column of Figure 1 depicts the separating hyperplane, generated by the linear program in (6). The bottom plot shows the hyperplane (yellow surface) in the Hilbert space, and the top plot shows the separating boundary (yellow closed contour) in the input domain.

B. Two-Layer Neural Network Partitions

The Hilbert space splitting based on a separating hyperplane may not capture the true classification boundary in the Hilbert space. Consequently, a richer decision boundary based on a nonlinear separating surface should create a splitting with fewer misclassified points. Consider the problem of defining a quadratic safety volume in the Hilber space specified by a quadratic boundary

$$\zeta^T A_2 \zeta + b_2^T \zeta + c_2 = 0, \tag{8}$$

where elements below the surface (evaluate to less than 0) are guaranteed to be *safe*, and *unsafe* elements are guaranteed to be above the surface (evaluate to greater than 0). Now, the optimization problem becomes a quadratically constrained linear program (QCLP), which is an NP-hard problem in theory [22], [23]. Although relaxation techniques for solving QCLPs exist, such optimization problems should be avoided when possible. Adding a quadratic polynomial kernel as a second layer to the network will preserve the LP formulation in (6) for the cutting surface optimization problem.

1) A Quadratic Polynomial Layer: Using a quadratic polynomial kernel of the form $p_2(x, y) = (x^T y + \lambda)^2$ leads to quadratic cutting surfaces in $\mathcal{H}^{n_q}$ where n_q is the number of kernels. The mapping is now $\vec{p}_2 \circ \vec{k} : \mathcal{D} \to \mathcal{H}^{n_q}$, where

$$\vec{p}_2(x) = \left[p_2(x, y_1) \ \cdots \ p_2\left(x, y_{n_q}\right) \right]^T. \tag{9}$$

For the quadratic polynomial layer, the parametric degrees of freedom are the vectors y_i such that each kernel coordinate is $p_2(x, y_i; \lambda)$. For simplicity, let $n_q \geq n_c$ such that $y_i = e_i$ for $i \in \{ 1, \ldots, n_c \}$ where e_i is the i^{th} unit coordinate vector. The remaining vectors y_i, if any, for $n_c < i \leq n_q$ can be any basis elements that support the task at hand (typically selected based on the training data). Creating a separating boundary is equivalent to establishing the coefficients α_i for the following constraint equation,

$$z^T \left(\sum_i \alpha_i y_i y_i^T \right) z + 2\lambda \left(\sum_i \alpha_i y_i^T \right) z + \lambda^2 \left(\sum \alpha_i \right) = 0, \tag{10}$$

where $z \in \mathcal{H}^{n_c}$. The structure of (10) matches that of (8). Note that due to the limited number of kernels, and thus limited number of basis vectors y_i, (10) can only represent a subset of solutions generated by (8). However, the trade-off is that the optimization problem for solving the coefficients of kernel machines remains a linear program:

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^{n_q}} \quad & \vec{1}^T \alpha \\ \text{s.t.} \quad & \alpha^T \vec{p}_2 \circ \vec{k}(x_i) \geq 1, \ \forall i \in \{ \, l \mid x_l \in \mathcal{X}^u \, \} \\ & \alpha_j \geq 0, \ \forall j = 1, \ldots, n_q \end{aligned} \tag{11}$$

Theorem 3. *Given a kernel embedding $\vec{k}$ defined from an ϵ-covering of $\mathcal{X}^u$, there is a hyperspherical splitting of $\mathcal{H}^{n_c}$ described by*

$$\mathcal{S} = \left\{ x \in \mathcal{D} \mid \left|\left| \vec{k}(x) + \lambda \vec{1} \right|\right|_2 < \rho_P \text{ for } x \in \mathcal{X}^u \right\} \tag{12}$$

where $\mathcal{S} \subset \mathcal{S}^* \subset \mathcal{D}$, $\lambda \geq 0$, $\rho_P = \min \left(\left|\left| \vec{k}(x) + \lambda \vec{1} \right|\right|_2 \right)$ for $x \in \mathcal{X}^u$, and $\mathcal{X}^u \in \bar{\mathcal{S}}$.

Proof. The theorem asserts the existence of a feasible quadratic surface with hard *unsafe* constraints for splitting the Hilbert space using the 2-norm. The 2-norm operation is equivalent to setting $\lambda = 0$, $\alpha_i = \rho_P^{-2}$ for $i \leq n_c$, and $\alpha_i = 0$ for $n_c < i \leq n_q$ (should such i exist) for the polynomial quadratic kernel problem specified by (11). Thus, the set of feasible solutions to (11) for $\lambda = 0$ has at least one element in it. For $\lambda > 0$, $\rho_P = \min \left(\left|\left| \vec{k}(x) + \lambda \vec{1} \right|\right|_2 \right)$ and gives a hyper-sphere centered at $-\lambda \vec{1}$. □

The theorem establishes the existence of a solution to the linear program defined in (11), thereby showing that the solution space is non-empty when $\lambda \geq 0$. The case of $\lambda < 0$ is possible but trickier to solve for, and may lead to poor solutions when $-1 < \lambda < 0$. Referring again to the example in Figure 1, the third column depicts the separating quadratic surface, solved by the linear program in (11). The bottom plot shows the quadratic surface (yellow surface) in the Hilbert space, and the top plot shows the separating boundary (yellow closed contour) in the input domain. There are slightly fewer misclassified safe data points compared to the linear hyperplane results in the second column.

Higher order polynomial kernels may be used to partition the space. Doing so should provide more space carving degrees of freedom but will require a policy for selecting the kernel basis elements (beyond the unit coordinate vectors of the Hilbert space). Solving for the second layer as a kernel SVM using a sequential minimal optimization (SMO) algorithm [24] will provide a solution that identifies the basis elements and solves for their coefficients, much like in [17]. This section focused on a constructive approach to guarantee a non-empty solution space; it can be augmented with an SMO-like basis expansion solver to obtain $n_q > n_c$.

C. Two-Layer Partitions with Safe and Unsafe Samples

The linear programs for the splitting hyperplane and hyperellipsoid in the lifted Hilbert space $\mathcal{H}^{n_c}$ attempt to find the furthest hyperplane or largest hyperellipsoid. They may be conservative relative to other options for synthesizing such a boundary due to the curvature of the Hilbert space

or the shape of the separating boundary. As a result, misclassification of safe points as unsafe will occur, as seen in Figure 1. Richer surfaces exist that better capture the regions misclassified by the linear and ellipsoidal cutting surfaces. However, the linear program establishing the model coefficients α will require knowledge of what parts of the mapped space should be considered *safe* versus *unsafe*, and require having sufficient samples of both classes. These samples permit recovery of more complex cutting surfaces.

The coordinate vectors, y_i, for the polynomial kernels in the multi-order polynomial layer should at least include n_c basis unit vectors for each polynomial order (i.e., the minimum LP problem size grows linearly with the maximum polynomial order). These can be complemented with a specialized or task-specific basis expansion criteria. The polynomial kernel mapping is the following,

$$\begin{aligned}\vec{p}(z) = [&(y_1^T z + \lambda_1) \ \cdots \ (y_{n_c}^T z + \lambda_1) \ \cdots \\ &(y_1^T z + \lambda_{p_i})^{p_i} \ \cdots \ (y_{n_c}^T z + \lambda_{p_i})^{p_i} \ \cdots]^T,\end{aligned} \tag{13}$$

where $p_i \in \{1, \ldots, n_p\}$, n_p is the maximal polynomial order used, and N_p is the total number of polynomial kernels used across all orders. The minimum basis elements requirement means that $N_p \geq n_p n_c$. The linear-in-α constraint equation for a multi-order polynomial surface cut is

$$\alpha^T p(z) = \sum_i \alpha_i \left(y_i^T z + \lambda_{p_i}\right)^{p_i} = 0. \tag{14}$$

Per [17], the inclusion of positive and negative samples should involve hard constraints on the *unsafe* points and soft constraints for the *safe* points

$$\begin{aligned}\min_{\alpha \in \mathbb{R}^{N_p}, \xi \in \mathbb{R}^{n_s}} \quad & \vec{1}^T \xi \\ \text{s.t.} \quad & \alpha^T \vec{p} \circ \vec{k}(x_i) \geq 1, \ \forall i \in \{\, l \mid x_l \in \mathcal{X}^u \,\} \\ & \alpha^T \vec{p} \circ \vec{k}(x_j) \leq -1 + \xi_j, \\ & \xi_j \geq 0 \qquad\qquad \forall j \in \{\, l \mid x_l \in \mathcal{X}^s \,\}\end{aligned} \tag{15}$$

Theorem 4. *Defining the two layer neural network $f = \vec{p} \circ \vec{k}$ with $n_p \geq 2$ and performing a hyper-planar partitioning of the space in the output space provides an equivalent or better splitting of the space, based on safe misclassification counts.*

Proof. The solution space for (15) contains the linear and quadratic cases by setting the appropriate coefficients β_i to zero. The optimization problem will either return one of these options or it will find a better one. By virtue of having a slack minimizing cost, better solutions will involve either the same slack or smaller. If more slack variables evaluate to zero, there will be less *safe* point classification errors. □

The original two-layer formulation in [17] uses a 2-layer GRBF network created using a kernel-SVM process for the second layer. The Gaussian kernel in the second layer defines an alternative nonlinear space for generating the cutting surface, which may provide a richer cutting surface solution space. The theorems and corollaries here provide a more formal analysis of the partitioning properties

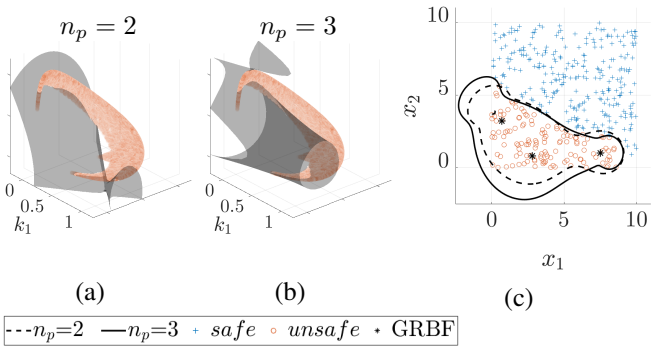

Fig. 2. Depiction of cutting surfaces and separating curves for two-layer, multi-order polynomial partitions using safe and unsafe data points.

of the 2-layer kernel machine, which is an instance of a shallow layer kernel machine (SKM), and the specification of learning problems on the output layer that are linear program formulations with a non-empty solution space. Similar results apply for a Gaussian kernel second layer.

a) Discussion: The provision of negative (*safe*) samples changes the structure of the problem by adding a point set complementary to the unsafe set, deemed to be safe. Incorporating this data into the optimization problem adds information about what regions of the Hilbert space $\mathcal{H}^{n_c}$ should lie on the negative side of the boundary, thereby pushing the boundary outwards when there are negative samples on the other side of the hyperplanar or hyperellipsoidal splitting (based on available degrees of freedom in the polynomial kernel layer). Solutions should improve the accuracy of the estimated bound and reduce the quantity of *safe* misclassification errors, per Theorem 4.

Figure 2 depicts the cutting surfaces for the case of $n_p = 2$ (left plot) and $n_p = 3$ (middle plot), and their input domain boundaries (right plot) for the same example problem data of Figure 1. Both surfaces were generated using positive and negative samples per (15). There are less misclassified *safe* data points, with the $n_p = 3$ case achieving zero slack cost.

b) Maximally Flexible Shallow Network Design: The constructions to date are based on a covering of the positive set (*unsafe* set). For potentially more accurate results, additional basis centers can be chosen from the negative (*safe*) set. Extremizing coordinates associated to these centers indicate movement towards *safe* regions and away from *unsafe*. Their inclusion changes the structure of the optimization since these are points in the Hilbert space surface to avoid. The optimization problem specification should seek to generate a cutting surface that carves out such regions. Because of the flipped nature, a bias term will be needed. The bias term can be easily appended to the multi-order polynomial layer mapping, updating $\vec{p}$ from (13) to

$$\begin{aligned}\vec{p}(z) = [&(y_1^T z + \lambda_1) \ \cdots \ (y_{n_c}^T z + \lambda_1) \ \cdots \\ &(y_1^T z + \lambda_{p_i})^{p_i} \ \cdots \ (y_{n_c}^T z + \lambda_{p_i})^{p_i} \ \cdots \ 1]^T,\end{aligned} \tag{16}$$

where, $p_i \in \{1, \ldots, n_p\}$. The optimization formulation (15) still applies. Repeating the earlier analysis will show that

(15) has at least one point in the solution space for separating hyperplanes (and one for separating hyperellipsoids). Consequently, (15) has a non-empty solution space.

IV. Construction of the Barrier Function

The previous sections described methods to synthesize level-set functions for approximating the boundary of the safe set from labeled, sampled data. The level-set functions have the necessary properties to serve as zeroing barrier functions after a constant shift and rescaling. We call the barrier function designed from concatenated kernel machines a *shallow kernel machine-zeroing barrier function* (*SKM-ZBF*). An additional extended class $\mathcal{K}$ outer function $\psi \in \mathcal{K}_e$ may be needed to adjust the sensitivity based on the application, e.g., $h(x) = \psi_0 \circ \bar{h}(x)$, where $\bar{h}(\cdot)$ is the *SKM-ZBF*. This section connects the cutting surface layer with ZBF properties and concludes with some practical considerations for implementing the *SKM-ZBF*.

A. Suitability as Zeroing Barrier Functions

To serve for safety-critical control applications, there are conditions on the level-set function to be a valid ZBF.

a) Monotonicity: ZBFs must be monotonic when evaluated across the boundary. The specification of the boundary as a cutting surface constraint means that the domain surface and the constraint surface are transverse to each other. The intersection of the two surfaces occur at the zero level-set. Movement on the (Hilbert space) domain surface away from the boundary necessarily has monotonic behavior, locally. The hard constraint in (15) for the unsafe samples ensures local monotonicity, at least from the -1 to +1 level-sets (0 to +2 for the equivalent SKM-ZBF) and for some non-trivial band away from their boundaries. Figure 3 depicts SKM-ZBF as a color map for various architectures, all of which exhibit monotonicity across the boundary.

b) Continuously Differentiable: The canonical control barrier function constraint used for safety depends on the gradient of the barrier function [2]. The function needs to be continuously differentiable to avoid undesirable behavior in the constraint. The Gaussian and polynomial layers consist of smooth basis functions, thus their composition and linear combination is also smooth, thereby resulting in smooth gradients of the ZBF. The gradient has a closed form solution based on equations (2) and (16) which can be used in the CBF constraint [2] to enforce safety.

c) Dead Gradients: Based on the CBF constraint discussed in [2], vanishing gradients for $h(x)$ for $x \in \mathcal{D}$ may lead to loss of control [25], which is undesirable in safety critical applications. Monotonicity in the vicinity of the zero level-set guarantees no vanishing gradients in that region. Due to the nature of the first layer and the mapping of $\mathbb{R}^{n_d}$ to a compact surface in $\mathcal{H}^{n_c}$, the function has known limits, which give local extrema. One such will be the output of $z = 0$ (points at infinity in the original input space). These points will tend to the same ZBF value (visible as nearly constant regions in the level-set plots of Fig. 3). At the other extreme will be the center locations; they locally maximize distance from $z = 0$ in $\mathcal{H}^{n_c}$, and may have similar extremizing properties relative to the transverse cutting surface. These points may be local extrema of the ZBF, in which case the gradients will vanish there (visible as local minima in the level-set plots of Fig. 3). Since the focus of this SKM-ZBF construction is on spatially meaningful or location-based ZBFs related to navigation, the existence of the aforementioned extrema on sets of measure zero do not have a strong impact as for other use cases of ZBFs. Due to the possible topology of *unsafe* space, attempts to define one signed distance function for all disconnected regions will result in an extrema set of zero measure (related to junctions in the Voronoi partition).

d) Guarantees on the Safe Set: The ability to capture the interface between *safe* and *unsafe* regions is a function of the measurement density. There is an operating assumption of an ϵ-covering for the *unsafe* regions where the ϵ value is related to the bandwidth of the Gaussian kernels, σ. If the sensor resolution is too coarse to capture boundary variation, then the *unsafe* set may not lie within the resulting covering and the asserted guarantees cannot hold. It is impossible to guarantee safety when the sensors cannot measure or sample the space as needed (unless it is known how much the local structure can change for small changes in sample location).

The optimization problem guarantees that $h(\mathcal{X}^{\mathrm{u}}) \subset \mathbb{R}^{-}$, which implies the existence of a point-dependent closed covering $\bar{\mathrm{C}}(\mathcal{X}^{\mathrm{u}})$ of the unsafe points for which $h(\bar{\mathrm{C}}(\mathcal{X}^{\mathrm{u}})) \subset \mathbb{R}^{-}$. Assuming that the sensor can indeed measure and capture the local structure, and the GRBF layer reflects it, then $\mathcal{U}^* \subset \bar{\mathrm{C}}(\mathcal{X}^{\mathrm{u}}) \subset \mathcal{D}$ and $\mathcal{S} \subset \mathcal{S}^* \subset \mathcal{D}$.

B. Practical Method for Barrier Function Synthesis

The standard kernel machine mapping consists of a single spatial scale, which places a limit on the boundary that can be captured for a given ϵ-cover. This limit is related to the Fourier spectral properties of the functions generated by the kernel machine [26]. Multi-scale implementations improve this limitation [27], with each Gaussian center of the first layer having a bandwidth parameter chosen from a finite set.

In effect, each c_i has a bandwidth $\sigma_i \in \{\beta_1, \ldots, \beta_{N_b}\}$ where N_b is the number of unique bandwidths used. For multiple Gaussian bandwidths the ϵ-cover concept changes to be a *Gaussian kernel cover* extending the single-bandwidth version. The same equation (5) is used but the equivalent balls $\mathcal{B}_{\epsilon_i}(x)$ in the original input space will have differing radii ϵ_i based on the kernel function bandwidth β_i for the i^{th} kernel coordinate mapping, similar to how p_i varies for a multi-order polynomial layer. These (radius variable) balls should cover the space of interest. The cover C from (5) satisfies

$$\mathcal{D} \subset \mathrm{C} = \bigcup_i \mathcal{B}_{\epsilon_i}(c_i) \quad \text{for } \epsilon_i \in \{\varepsilon_1, \ldots, \varepsilon_{N_b}\}, \tag{17}$$

and each ε_j depends on β_j. The same linear programs for hyperplanar, hyperellipsoidal, and multi-order polynomial cutting surfaces apply, as do the existence of at least one solution in the feasible space for each LP specification.

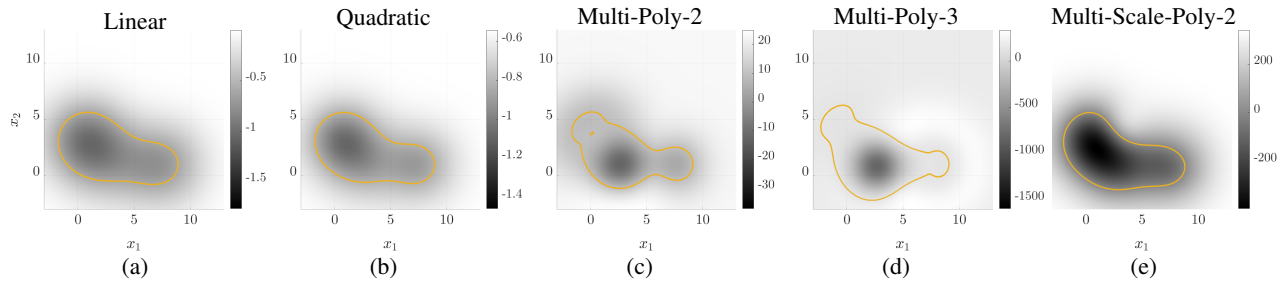

Fig. 3. Depiction of SKM-ZBF values as a color map along with its zero level-set (yellow contour) for different network constructions. (a) and (b) are replica of Figure 1(b,c) (top row); (c) and (d) are replica of Figure 2(c); (e) is the same as (c) but with multi-scale Gaussian kernels.

Earlier work [28] established a greedy selection policy for recovering a domain covering center set $\mathcal{C}$ for optimized function approximation. A similar policy applies to the case of multi-scale, first layer synthesis from data, modified with finer bandwidth values based on residual error minimization of the coarser single layer basis functions. Likewise, regression or function recovery from Gaussian kernel machines operate best when capturing the deviation from some known parametric model [29], leading to a semi-parametric first layer (fixed + data-adaptive elements).

V. Examples and Implementation

A. Curved Lane Modeling

This case study shows how the proposed method generates a suitable SKM-ZBF that models curved lanes, as shown in the low-speed driving scenario of Figure 4(a). Closed-form expressions for curved roads can be challenging to determine, and thus, using control barrier function policy synthesis frameworks for such applications can be difficult. To that end, we show SKM-ZBF synthesis alleviates this issue. The safe and unsafe data points used for network training are shown in Figure 4(b) and are generated synthetically. In practice, such points would be generated from visual sensing and processing, such as from semantic segmentation of data from a forward-facing camera on the car.

A simple strategy for choosing the GRBFs centers is to uniformly place them along the center line of the lane. Seven GRBFs with a bandwidth of $\sigma = 2w_r/\sqrt{\log(2)}$ are used in this case study, as shown in Figure 4(b) by the black (*); where w_r is the width of the road. The synthesized SKM-ZBF boundary is shown in Figure 4(c) as the yellow dashed line and as a surface in Figure 4(d). The SKM-ZBF compute time was 10 msec (MATLAB, Ubuntu 20.04, Intel i7-8750H CPU), which supports a 100 Hz update rate.

B. Planar Mobile Robot Navigation in ROS STDR Simulator

This study involves a differential drive robot equipped with a LiDAR sensor navigating a 2D environment with obstacles, see Figure 5. The robot tracked a predetermined, dynamically feasible, collision-free trajectory. Ideal localization was assumed to be available for the robot. The procedure to generate the dataset of safe and unsafe LiDAR samples was done similar to [17]. Global and local SKM-ZBF synthesis was performed. Global synthesis aggregated all of the sensor data and applied either a multi-polynomial second layer, Figure 5(a), or a Gaussian second layer via kernel SVM optimization [17], Figure 5(b). The latter generates a richer solution space and will be more accurate. The first layer used a fixed 5×8 uniform grid. Local SKM-ZBF synthesis used a fixed robot-centered 4×4 grid for three locations, shown as gray dots in the global maps. The zero level-set of the locally and globally synthesized SKM-ZBFs are shown as yellow curves. Notice that the local SKM-ZBF zero level-set cuts into some sides of obstacles that do not have LiDAR measurements. However, once measurements are taken on those sides the SKM-ZBF will include them.

To further improve the accuracy of the SKM-ZBF in capturing the domain, multi-scale kernels can be added to the first layer in addition to the fixed grid kernels. The centers and bandwidths of the multi-scale kernels are turned according to the adaptive strategy described in section IV.B.

The global SKM-ZBF compute time was 3.9 sec for the multi-polynomial SKM-ZBF and 4.3 sec for the Gaussian SKM-ZBF when solved using the SMO solver in the LIBSVM package (dual QP formation takes 18 sec). The local SKM-ZBF compute time averaged 18 msec, which supports a 50 Hz update rate. The training time of the local SKM-ZBF using the proposed LP formulation is significantly shorter than the training time of neural network-based ZBFs. For example, the training time of a Softplus-based deep neural network via gradient descent for local ZBF synthesis was 98 msec on GPU and 407 msec on CPU [14].

VI. Conclusion

This paper presented a zeroing barrier function synthesis method based on a two-layer shallow kernel machine network architecture. The first layer is constructed from Gaussian kernels whose geometry in the associated Hilbert space was analyzed while considering positive samples (unsafe points) only. The analysis provided theoretical guarantees on the existence of a separating hyperplane, synthesized via a linear program (LP), that splits the Hilbert space into safe and unsafe regions via its level-sets. A second layer was added to the network that uses polynomial kernels and negative samples (safe points) to reduce misclassifications, while also maintaining the LP formulation. The cutting surface geometry of this second layer also applies to the previous work [17]. Two case studies demonstrate the efficacy of the SKM-ZBF architecture in generating valid zeroing barrier functions for motion planning applications. Future work will

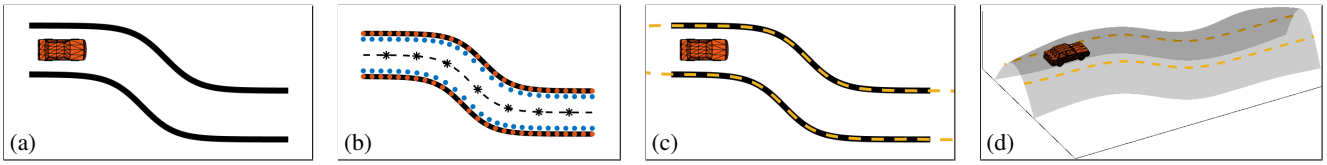

Fig. 4. Curved lane modeling using SKM-ZBF. (a) curved road; (b) road with labels for synthesized safe (blue) and unsafe (red) points along with GRBF center locations (black *); (c) synthesized zero level-set of SKM-ZBF (dashed yellow lines); (d) synthesized *SKM-ZBF* as a 3D surface plus zero level-set.

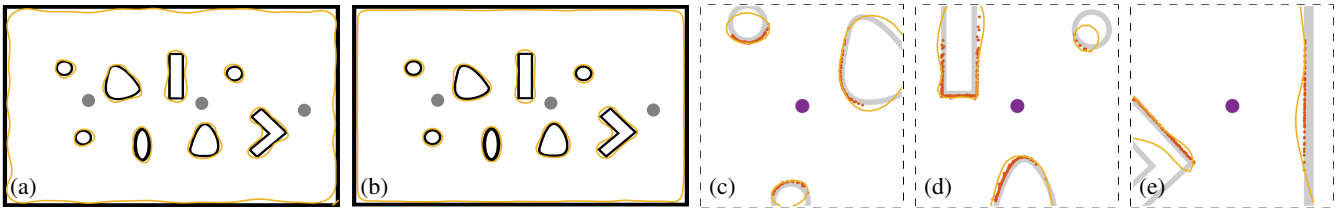

Fig. 5. Mobile robot planar navigation problem. Synthesis of a global SKM-ZBF using (a) multi-polynomial second layer and (b) Gaussian kernel second layer, to define unsafe region boundaries (yellow contours). Local navigation SKM-ZBF generated at three marked locations (gray dots) in the global maps, are depicted in (c), (d), and (e). Orange dots are LiDAR points, the purple circle is the robot's location, and the yellow contour is the SKM-ZBF boundary.

explore optimized construction of SKM-ZBFs for boundary estimation accuracy in navigation contexts.

REFERENCES

[1] F. Blanchini, "Set invariance in control," *Automatica*, vol. 35, no. 11, pp. 1747–1767, 1999.

[2] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference*, pp. 3420–3431, 2019.

[3] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Trans. on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[4] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Trans. on Robotics*, vol. 33, no. 3, pp. 661–674, 2017.

[5] L. Wang, A. D. Ames, and M. Egerstedt, "Safe certificate-based maneuvers for teams of quadrotors using differential flatness," in *IEEE International Conference on Robotics and Automation*, pp. 3293–3298, 2017.

[6] P. Pierpaoli, A. Li, M. Srinivasan, X. Cai, S. Coogan, and M. Egerstedt, "A sequential composition framework for coordinating multirobot behaviors," *IEEE Trans. on Robotics*, vol. 37, no. 3, pp. 864–876, 2021.

[7] M. Srinivasan and S. Coogan, "Control of mobile robots using barrier functions under temporal logic specifications," *IEEE Trans. on Robotics*, vol. 37, no. 2, pp. 363–374, 2021.

[8] L. Lindemann and D. V. Dimarogonas, "Decentralized control barrier functions for coupled multi-agent systems under signal temporal logic tasks," in *European Control Conference*, pp. 89–94, June 2019.

[9] Y. Chen, M. Jankovic, M. Santillo, and A. D. Ames, "Backup control barrier functions: Formulation and comparative study," in *IEEE Conference on Decision and Control*, pp. 6835–6841, 2021.

[10] J. Schmidt-Hieber, "The kolmogorov–arnold representation theorem revisited," *Neural Networks*, vol. 137, pp. 119–126, 2021.

[11] A. Robey, H. Hu, L. Lindemann, H. Zhang, D. V. Dimarogonas, S. Tu, and N. Matni, "Learning control barrier functions from expert demonstrations," in *Conference on Decision and Control*, pp. 3717–3724, 2020.

[12] Y. Meng, Z. Qin, and C. Fan, "Reactive and safe road user simulations using neural barrier certificates," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6299–6306, 2021.

[13] V. Dhiman, M. J. Khojasteh, M. Franceschetti, and N. Atanasov, "Control barriers in bayesian learning of system dynamics," *IEEE Trans. on Automatic Control*, pp. 1–1, 2021.

[14] K. Long, C. Qian, J. Cortés, and N. Atanasov, "Learning barrier functions with memory for robust safe navigation," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4931–4938, 2021.

[15] K. Long, V. Dhiman, M. Leok, J. Cortés, and N. Atanasov, "Safe control synthesis with uncertain dynamics and constraints," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 7295–7302, 2022.

[16] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *The International Journal of Robotics Research*, vol. 35, no. 14, pp. 1717–1730, 2016.

[17] M. Srinivasan, A. Dabholkar, S. Coogan, and P. A. Vela, "Synthesis of control barrier functions using a supervised machine learning approach," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 7139–7145, 2020.

[18] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 1991.

[19] D. B. McDonald, W. J. Grantham, W. L. Tabor, and M. J. Murphy, "Global and local optimization using radial basis function response surface models," *Applied Mathematical Modelling*, vol. 31, no. 10, pp. 2095–2110, 2007.

[20] K. M. Lynch and F. C. Park, *Modern Robotics: Mechanics, Planning, and Control*. USA: Cambridge University Press, 1st ed., 2017.

[21] K. Muandet, K. Fukumizu, B. Sriperumbudur, and B. Schölkopf, "Kernel mean embedding of distributions: A review and beyond," *Foundations and Trends in Machine Learning*, vol. 10, no. 1-2, pp. 1–141, 2017.

[22] J. Linderoth, "A simplicial branch-and-bound algorithm for solving quadratically constrained quadratic programs," *Mathematical Programming*, vol. 103, pp. 251–282, Jun 2005.

[23] K. Chatterjee, H. Fu, A. K. Goharshady, and E. K. Goharshady, "Polynomial invariant generation for non-deterministic recursive programs," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, p. 672–687, 2020.

[24] J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines," Tech. Rep. MSR-TR-98-14, Microsoft, April 1998.

[25] M. Srinivasan, N.-s. P. Hyun, and S. Coogan, "Weighted polar finite time control barrier functions with applications to multi-robot systems," in *IEEE Conference on Decision and Control*, pp. 7031–7036, 2019.

[26] F. Girosi, M. Jones, and T. Poggio, "Regularization theory and neural networks architectures," *Neural Computation*, vol. 7, no. 2, pp. 219–269, 1995.

[27] H. Wendland, *Scattered Data Approximation*. Cambridge Monographs on Applied and Computational Mathematics, Cambridge University Press, 2004.

[28] H. Kingravi, P. Vela, and A. Gray, "Reduced set KPCA for improving the training and execution speed of kernel machines," in *SIAM International Conference on Data Mining*, pp. 441–449, 2013.

[29] A. Chang, C. Hubicki, J. Aguilar, D. Goldman, A. Ames, and P. Vela, "Learning terrain dynamics: A Gaussian process modeling and optimal control adaptation framework applied to robot jumping," *IEEE Trans. on Control Systems Technology*, vol. 29, no. 4, pp. 1581–1596, 2020.