Synthesis of Control Barrier Functions Using a Supervised Machine Learning Approach

Mohit Srinivasan[1], Amogh Dabholkar[2], Samuel Coogan[3], and Patricio A. Vela[4]

Abstract— **Control barrier functions are mathematical constructs used to guarantee safety for robotic systems. When integrated as constraints in a quadratic programming optimization problem, instantaneous control synthesis with real-time performance demands can be achieved for robotics applications. Prevailing use has assumed full knowledge of the safety barrier functions, however there are cases where the safe regions must be estimated online from sensor measurements. In these cases, the corresponding barrier function must be synthesized online. This paper describes a learning framework for estimating control barrier functions from sensor data. Doing so affords system operation in unknown state space regions without compromising safety. Here, a support vector machine classifier provides the barrier function specification as determined by sets of safe and unsafe states obtained from sensor measurements. Theoretical safety guarantees are provided. Experimental ROS-based simulation results for an omnidirectional robot equipped with LiDAR demonstrate safe operation.**

I. INTRODUCTION

Autonomous vehicles , industrial robots, and multi-robot systems deployed in uncertain domains are often tasked to respect safety-critical constraints while advancing a given task [1]. When operating in unknown and dynamic environments with insufficient advanced information regarding the workspace, controllers which translate sensory information from the environment into *safe* control actions are of paramount importance. Control barrier functions (CBFs) are level-set functions used to provide formal safety guarantees for controlled dynamical systems. Barrier function based real-time controllers in robotics support collision avoidance for multi-robot motion [2], task allocation for robotic swarms [3], and motion planning [4].

A key assumption commonly imposed is that the robotic system has complete knowledge of the unsafe state space regions. Leveraging the knowledge translates to formal safety guarantees arising from its translation to CBFs. In practice, this assumption need not hold and limits more widespread application of barrier functions. As a motivating example, consider an autonomous robot operating in an environment for which it has no knowledge of the obstacle boundaries. If these boundaries are to be as level-sets of smooth functions, the process of finding closed-form barrier functions for these obstacles is not straightforward. Without the functions, one cannot leverage the safety guarantees that CBFs provide. Thus, this paper describes a support vector machine (SVM) approach to CBF synthesis from sensor measurements. In particular, sensory information obtained from the environment defines the set of safe and unsafe samples and is used for training the SVM classifier.

Learning algorithms, or data-driven synthesis methods, for ensuring safety have been explored in several contexts. The most prevalent has been to establish stable state space regions meeting safety specifications by identifying a control Lyapunov function (CLFs) compatible with given CBFs. Techniques for doing so include sum-of-squares (SOS) methods [5] and neural network designs [6], with the aim of identifying the largest possible stable region within the safe region. Investigations more closely aligned with barrier function synthesis using tools from machine learning include the use of kernel machines [7] to synthesize occupancy map functions for navigation and planning purposes [8], [9]. Our aim is to explore how machine learning constructs can be used to synthesize CBFs in a manner that the learned function provides the necessary safety guarantees.

The contributions of this work are as follows: First, we present a SVM approach for the synthesis of a barrier function from a training dataset consisting of *safe* and *unsafe* samples obtained from sensor measurements. We describe offline and online training methods. Second, a formal guarantee on correct classification of unsafe regions is provided for both the methods. We show that in the offline method, the system is rendered safe for an under-approximated (conservative) safe set. A similar guarantee holds locally in the online approach. The proposed framework is implemented in a ROS-based simulator with a LiDAR equipped omnidirectional robot. Evaluation metrics for the trajectories generated by the proposed CBF synthesis framework quantify how well they match the ideal case where the CBF is known. To the best of our knowledge, this is the first paper addressing the problem of CBF synthesis from sensed environmental data.

This paper is organized as follows: Section II reviews control barrier functions, their safety properties, and their use in QP-based control. Section III describes the problem addressed. Section IV covers the main results of the CBF synthesis framework, for both the offline and online versions. Section V covers implementation scenarios from a motion planning perspective along with evaluation metrics for comparing the generated trajectories with ground truth data. Section VI provides concluding remarks.

This work was supported in part by the National Science Foundation under Grant Nos. #1836932, S&AS 1849333, and in part by DARPA PAI

[1]Mohit Srinivasan, [3]Samuel Coogan and [4]Patricio A. Vela are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, USA mohit.srinivasan@gatech.edu; sam.coogan@gatech.edu; pvela@gatech.edu

[2]Amogh Dabholkar is with the Department of Electrical and Electronics Engineering, Birla Institute of Technology and Science (BITS), Pilani - K. K. Birla Goa Campus, India adabholkar6@gatech.edu

[3]Samuel Coogan is also with the School of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta, USA

II. MATHEMATICAL BACKGROUND

This section summarizes the concept of control barrier functions and the formal safety guarantees they provide. To begin, consider an affine control robotic system:

$$\dot{x} = f(x) + g(x)u\,, \quad x \in \mathcal{D} \subset \mathbb{R}^n,\; u \in \mathbb{R}^m, \tag{1}$$

where x is the state of the robot, u is the control input, and $x(0) = x_0$. Both $f : \mathcal{D} \to \mathbb{R}^n$ and $g : \mathcal{D} \to \mathbb{R}^{n\times m}$ are locally Lipschitz continuous vector fields.

Consider further that the system has a set of safe states $\mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0 \text{ and } h \in C^1(\mathcal{D}; \mathbb{R})\}$ given by the super zero level-set of the function h. The boundary of the safe set is the zero level-set, $\partial\mathcal{C} = \{x \in \mathcal{D} \mid h(x) = 0\}$. During controlled evolution, the system (1) is considered to be *safe* if for all $t \geq 0$, $x(t) \in \mathcal{C}$ when $x(0) \in \mathcal{C}$. As detailed in [10], zeroing control barrier functions (ZCBFs) can be used to guarantee safety of the system. To define ZCBFs, we first define an extended class $\mathcal{K}$ function $\alpha : \mathbb{R} \to \mathbb{R}$ as a function that is strictly increasing and $\alpha(0) = 0$.

Definition 1. The function $h \in C^1(\mathcal{D}; \mathbb{R})$ is a Zeroing Control Barrier Function (ZCBF) if there exists a locally Lipschitz extended class $\mathcal{K}$ function α such that for all $x \in \mathcal{D}$

$$\sup_{u\in\mathbb{R}^m} \left\{ L_f h(x) + L_g h(x)u(x) + \alpha(h(x)) \right\} \geq 0\,,$$

for the Lie derivatives $L_f h(x) = \frac{\partial h(x)}{\partial x} f(x)$ and $L_g h(x) = \frac{\partial h(x)}{\partial x} g(x)$ of h in the direction of the vector fields f and g.

Define the state-dependent set of control inputs $\mathcal{U}(x)$,

$$\mathcal{U}(x) \equiv \left\{ u \in \mathbb{R}^m \mid L_f h(x) + L_g h(x)u(x) + \alpha(h(x)) \geq 0 \right\}. \tag{2}$$

Safety of the system can then be guaranteed under the action of a suitable control input $u(x) \in \mathcal{U}(x)$ for all $x \in \mathcal{D}$, formalized by the following theorem:

Theorem 1. *[10] Let there be a safe set $\mathcal{C} = \{x \in \mathcal{D} \mid h(x) \geq 0$ and $h \in C^1(\mathcal{D}; \mathbb{R})\}$ specified for the affine control system* (1). *If h is a ZCBF, then any control input $u \in C(\mathcal{D}; \mathbb{R}^m)$ where $u(x) \in \mathcal{U}(x)$ for all $x \in \mathcal{D}$ renders the set $\mathcal{C}$ forward invariant. That is, $x(t) \in \mathcal{C}$ for all $t \geq 0$.*

The constraint (2) arising from a ZCBF h is affine in the control input u. and hence can be encoded as a quadratic program (QP) constraint in u. For fixed $x \in \mathcal{D}$, the requirement $u \in \mathcal{U}(x)$ becomes a linear constraint for the following point-wise in time, minimum norm QP:

$$\begin{aligned} \underset{u\in\mathbb{R}^m}{\text{minimize}} \quad & ||u - k(x)||_2^2 \\ \text{s.t} \quad & u \in \mathcal{U}(x)\;, \end{aligned} \tag{3}$$

where $k : \mathcal{D} \to \mathbb{R}$ is a user-defined nominal control policy. This QP (a) results in a control input for following the nominal policy while simultaneously guaranteeing safety, and (b) is amenable to efficient online computation.

III. PROBLEM STATEMENT

Consider an affine control robotic system as in (1) evolving in $\mathcal{D} \subset \mathbb{R}^2$ and equipped with LiDAR sensors that provide depth information. By virtue of the depth measurement vector $z_t \in \mathbb{R}^N_{>0}$ at time t, where N is the total number of samples, the robot can detect unsafe state space regions. Regarding the LiDAR sensor, denote by θ_{res} the angular resolution (increment angle) of the measurements. We make the following assumption in order to account for spatial variations in the nature of the workspace: assume that the resolution of the LiDAR sensor is high enough to capture the spatial profile of the environment from a given offset distance, i.e., the LiDAR has a sufficiently small increment angle θ_{res}. Sensors such as the ones from Velodyne [11] with increment angles as small as $0.08°$ are capable of satisfying the above assumption. Hence, it is reasonable to assume such sensor resolution capabilities.

Let $k \in C(\mathcal{D}; \mathbb{R}^m)$ be a user-defined nominal feedback control policy to be followed by the robot. Examples of such policies include proportional (go-to-goal) control or MPC based policies [12]. The state space is assumed to contain unknown unsafe regions. That is, there exist p unsafe sets in the state space defined as $\mathcal{C}_i = \{x \in \mathcal{D} \mid h_i(x) \leq 0\, h_i \in C(\mathcal{D}; \mathbb{R})\}$ for all $i \in \{1, 2, \ldots, p\}$, such that h_i are unknown ZCBFs. The safe region is $\mathcal{D} \setminus \cup_{i=1}^{p} \mathcal{C}_i$.

Since there is no *a priori* knowledge of the unsafe sets, data obtained from the LiDAR sensor must be used to synthesize the unknown barrier functions $h_i : \mathcal{D} \to \mathbb{R}$, $i \in \{1, 2, \ldots, p\}$, to render the system safe while minimally deviating from the nominal feedback policy k. In conjunction with the robot's state, the measurements obtained from the on-board depth sensors provide the location of points on the boundary of the unsafe sets, and hence are points $x \in \mathcal{D}$ for which $h(x) = 0$. To learn the unsafe regions and follow the nominal policy safely, a framework for the synthesis of barrier functions is required with guarantees on safety of the system, as formalized by the problem statement:

Problem Statement 1. *Consider the affine control robotic system in* (1) *and the unsafe sets $\mathcal{C}_i \subset \mathcal{D}$, $i \in \{1, 2, \ldots, p\}$. Given the nominal feedback control policy $k : \mathcal{D} \to \mathbb{R}$ and LiDAR measurements z_t obtained at any time instant $t \geq 0$, formulate a barrier function synthesis framework which either (a) Learns the unsafe region $\bigcup_{i=1}^{p} \mathcal{C}_i$ offline given a dataset of safe and unsafe samples from the domain, or (b) Learns the unsafe region online using instantaneous measurements z_t, as the system traverses the domain.*

IV. CONTROL BARRIER FUNCTION SYNTHESIS FRAMEWORK

This section describes the methodology for obtaining the training dataset, the control barrier function synthesis framework, and two QP based approaches which utilize the synthesized barrier function to guarantee safety.

A. Support Vector Machines

The learning approach to be used for barrier function specification via-a-vis the unsafe regions will be support vector machine (SVMs), namely kernel SVMs [7]. Suppose a dataset $\mathcal{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ is provided where $x_i \in \mathbb{R}^n$ is an n dimensional vector and $y \in \mathcal{Y} = \{-1, 1\}$ is a label associated with the vector x_i for all $i \in \{1, 2, \ldots, N\}$. Using the dataset, the linear SVM algorithm determines an affine decision boundary function $\widehat{f}(w^T x + b)$, where $x \in \mathbb{R}^n$ is a training sample, $w \in \mathbb{R}^n$ are coefficients and $b \in \mathbb{R}$ is a bias term, which translates the sample x into a corresponding label $y \in \mathcal{Y}$ that belongs to one of the two classes i.e. $+1$ or -1. When the data is not separable by a hyperplane in the native space, a non-linear mapping transforming the data into a higher dimensional space with better separability properties may be used. This paper makes use of such a mapping, via a kernel function, to facilitate separation of unsafe obstacle regions from safe regions.

Since the domain $\mathcal{D}$ consists of states which are either safe or unsafe, their separation can be cast as a binary SVM classification problem. However, it is imperative that unsafe states be classified as unsafe, whereas all the safe states need not strictly be classified as safe. To that end, we consider the non-linear, biased-penalty SVM optimization problem [13]:

$$\begin{aligned} \underset{w}{\text{minimize}} \quad & \frac{1}{2}||w||_2^2 + C^+ \sum_{i|y_i=+1}^{N} \xi_i + C^- \sum_{j|y_j=-1}^{N} \xi_j \\ \text{s.t} \quad & y_i \cdot (w\phi(x_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0, \text{ for all } i \in \{1, 2, \ldots, N\} \ , \end{aligned} \tag{4}$$

where $C^+, C^- > 0$ are constants penalizing misclassification of the positive and negative samples, and $\phi : \mathbb{R}^n \to \mathbb{R}^d$ is a non-linear mapping into a higher dimensional space. In practice, the dual of the above optimization problem is solved by using a kernel function k_ϕ to bypass the need to explicitly define ϕ [7]. We use the Gaussian kernel,

$$k_\phi(x_i, x_j) = \exp\left(- \frac{\|x_i - x_j\|^2}{\sigma^2} \right), \tag{5}$$

where $\sigma > 0$ is the bandwidth of the kernel (and is a hyper parameter).

Observe that in (4) there are two separate costs for the positive and negative classes. Unequal costs permit a greater bias towards correctly classifying one class over the other. In particular, having $C^- = \infty$ and $\infty > C^+ > 1$ induces a hard margin classification for the unsafe states and allows for some misclassification for the safe states. This outcome is captured by the so called *cost matrix* (M) of the form

True \ **Estimated**	Safe	Unsafe
Safe	0	C^+
Unsafe	C^-	0

Each entry $[M]_{ij}$ of the matrix represents the cost of classifying a sample as label j when it truly belongs to label i. The penalty for classifying a truly safe (or unsafe) state as safe (or unsafe) is zero. It is undesirable to classify a truly unsafe state as safe, motivating a high penalty for C^-. Since safe states being classified as unsafe do not compromise safety, the penalty C^+ may be smaller. The optimization problem (4) provides compliance (in favor of safety) to measurement errors and noise in the sensor data which can affect the generated decision boundary. The mixed hard/soft margin classification is what supports the theoretical safety guarantees of the system as discussed in the following subsections. *When designed properly, the margin classifier function for an SVM locally varies in the vicinity of the safe boundary similarly to a signed distance function* (i.e., it provides the distance to the margin, or boundary, in the feature space). In contrast, an occupancy map function behaves more like a step function and does not have the correct variational behavior in the vicinity of boundaries.

B. Training Dataset Generation

This section details the training data generation process suited to binary SVM classification per (4). Below we provide a detailed explanation for generating the dataset.

Generating meaningful data for the kernel SVM from the LiDAR sensor requires converting the scalar variables into world Cartesian coordinates by means of a laser scan transform $g : \mathbb{R} \times \mathcal{D} \to \mathbb{R}^2$, whose main input is the laser scan measurements in polar coordinates and the current robot state (for mapping from the robot frame to the world frame). Assume that if the sensor detects an unsafe region, then the output from the sensor is a finite depth reading, else it is infinite. In particular, given a measurement vector $z_t = \begin{bmatrix} z_t^1 & z_t^2 & \ldots & z_t^N \end{bmatrix}^T \in \mathbb{R}^N$ at time t with N samples, define $\mathcal{F} \subset \mathcal{I} = \{1, \ldots, N\}$ to be the index set of the finite scan measurements. Define $\mathcal{O}^- = \bigcup_{i \in \mathcal{F}} \{g(z_t^i; x_t)\}$ as the set of unsafe samples. $\mathcal{O}^-$ represents points on the boundary of the unsafe set detected by the sensor which is used to populate a dataset of negative labeled samples $\mathcal{T}^- = \bigcup_{i \in \mathcal{F}} \{(g(z_t^i; x_t), -1)\}$. To obtain the positive samples from the environment, each $g(z_t^i; x_t) \in \mathcal{O}^-$ is projected radially backwards along the line segment joining the state of the robot $x(t)$ and the point $g(z_t^i; x_t)$, by a finite distance $d \in \mathbb{R}_{>0}$. Define

$$\widehat{z}_t^i = g(z_t^i - d; x_t) \in \mathbb{R}^2 \tag{6}$$

for all $i \in \{1, 2, \ldots, N\}$ where $d > 0$ is the finite offset distance. Define the set of positive samples as $\mathcal{O}^+ = \bigcup_{i \in \mathcal{F}} \{\widehat{z}_t^i\}$, with the dataset for positive labeled samples constructed as $\mathcal{T}^+ = \bigcup_{i \in \mathcal{F}} \{(\widehat{z}_t^i, +1)\}$. Collecting the set of positive and negative labeled samples generates the training dataset $\mathcal{T} = \mathcal{T}^+ \cup \mathcal{T}^-$. The training dataset $\mathcal{T}$ contains all unsafe samples and corresponding safe samples for training the SVM classifier.

C. Barrier Function Synthesis with Kernel-SVMs

To improve the ability to capture unsafe region boundaries, the point data is transformed by a fixed set of Gaussian kernels of the form (5) using a sparse set of grid points over the domain $\mathcal{D}$. This provides a first kernel machine layer that behaves like an approximate Hilbert space occupancy map [8] and roughly captures the different safe and unsafe regions of the state space. Passing the vector output of this Hilbert space to the kernel SVM generates a second layer that can

Algorithm 1: Training Dataset Generator

Input: Laser Scan Measurement z_t and Robot State x_t
Output: Training Dataset $\mathcal{T}$
function TrainingDataGenerator(z_t)
  Identify $\mathcal{F} \subset \{1, \ldots, N\}$
  $\mathcal{T}^- = \bigcup_{i \in \mathcal{F}} \{(g(z_t^i; x_t), -1)\}$
  $\widehat{z}_t^i = g(z_t^i - d; x_t)$, $\forall i \in \mathcal{F}$
  $\mathcal{T}^+ = \bigcup_{i \in \mathcal{F}} \{(\widehat{z}_t^i, +1)\}$
  $\mathcal{T} \leftarrow \mathcal{T}^- \cup \mathcal{T}^+$
  return $\mathcal{T}$

Algorithm 2: Kernel-SVM based Barrier Function Synthesis

Input: Training Dataset $\mathcal{T}$
Output: Estimated Barrier Function $\widehat{h}$
function BarrierEstimator($\mathcal{T}$)
  $\mathcal{T}_{HS} \leftarrow$ Map samples in $\mathcal{T}$ to Hilbert space
  $Cl \leftarrow$ Train kernel SVM classifier (4) using $\mathcal{T}_{HS}$
  $\widehat{h} \leftarrow$ Recover signed distance function from Cl and first Gaussian kernel layer mapping
  return $\widehat{h}$

Algorithm 3: Offline SVM-based QP controller

Input: Nominal controller k
$\mathcal{T} \leftarrow \emptyset$
forall $t \in [0, T]$ **do**
  $z_t \leftarrow$ `LaserScanMeasurement`
  $\mathcal{T}_t \leftarrow$ `TrainingDataGenerator`(z_t, x_t)
  $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_t$
$\hat{h} \leftarrow$ `BarrierEstimator` $(\mathcal{T})$
while *`Goal is not reached`* **do**
  Solve the QP:

$$u^*(x) = \underset{u \in \mathbb{R}^m}{\text{argmin}} \quad ||u - k(x)||_2^2$$
$$\text{s.t} \quad L_f\widehat{h}(x) + L_g\widehat{h}(x)u(x) \geq -\alpha(\widehat{h}(x))$$

  $u \leftarrow u^*(x)$
  Solve (1), update state $x(t)$

refine the boundary to better separate the safe and unsafe regions. The solution to the hard/soft margin kernel SVM in (4) defines the parameters for a non-linear decision boundary separating the training data (the output layer of the full classification network). Evaluating the two-layer classifier model for $x \in \mathcal{D}$ outputs a posterior probability describing the likelihood that the sample $x \in \mathcal{D}$ belongs to a particular class i.e., safe or unsafe. The posterior probabilities obtained from the model are then converted into margin scores which define a signed level-set function and provide the barrier function we seek. The barrier function approximator is thus a two hidden layer Gaussian kernel neural network. This entire procedure is summarized in Algorithm 2. By virtue of the methodology used to generate the training data , and the biased-penalty hard margin SVM optimization problem (4), the synthesized barrier function correctly classifies the unsafe samples. This is formalized in Proposition 1.

Proposition 1. *Given a training dataset $\mathcal{T}$ generated as per Algorithm 1, if Algorithm 2 is used to synthesize the barrier function $\widehat{h}$, then the unsafe samples $\mathcal{O}^-$ are such that $\widehat{h}(x) < 0$ for all $x \in \mathcal{O}^-$.*

Proof. By the method presented in Algorithm 1 to generate the training dataset $\mathcal{T}$, we have that the set $\mathcal{O}^-$ consists of points on the boundary of the unsafe set. From the kernel-SVM approach used in Algorithm 2, a function $\widehat{h}$ is generated which classifies the *safe* and *unsafe* samples. Since the optimization problem (4) is a hard margin SVM for the unsafe samples and RBF kernels have universal function approximation capabilities (Theorem 2, [14]), we can guarantee that $\widehat{h}(x) < 0$ for all $x \in \mathcal{O}^-$ and thus the proposition follows. ∎

D. *Offline Barrier Function Synthesis & Control*

Here, we discuss the *offline* approach to CBF synthesis using Algorithm 2. Per the problem setup in Section III, we consider the workspace consisting of p unsafe regions characterized by ZCBFs h_i, $i \in \{1, 2, \ldots, p\}$. We assume that there exists an *oracle* which provides a set of unsafe samples corresponding to the boundary of each unsafe set $i \in \{1, 2, \ldots, p\}$ in the state space by means of a LiDAR sensor and dense enough to cover the true obstacle boundaries.

Once the requisite training data is generated using the *oracle*, executing Algorithm 2 leads to a ZBF estimate. Note that a single ZCBF $\widehat{h} \in C^1(\mathcal{D}; \mathbb{R})$, whose zero level-set captures the boundaries between safe and unsafe regions, is obtained as opposed to p different ZCBFs characterizing the unsafe sets. With the synthesized barrier function $\widehat{h}$, we then implement a QP controller with (2) as the constraint. Capturing all the unsafe sets with a single function means that the QP involves only one constraint which reduces the computational complexity involved in computing the control input. The QP is solved, and the control is applied, until the system completes the specified task associated to the nominal controller. The entire offline barrier function synthesis and control methodology is formalized in Algorithm 3. In the algorithm, the initial loop from $t = 0$ to $t = T$ where $T < \infty$, indicates the time period when the training data is gathered for generating the barrier function.

Recall that the increment angle of the LiDAR sensor is given by θ_{res}. Intuitively, as $\theta_{\text{res}} \rightarrow 0$, the LiDAR sensor captures the true nature of the boundary of the unsafe region. Hence, using Proposition 1, we can guarantee that Algorithm 2 synthesizes a barrier function whose level-sets

Algorithm 4: Online SVM-based QP controller

Input: Aggregate Flag δ, Nominal controller k

$\mathcal{T} \leftarrow \emptyset$
while *Goal is not reached* **do**
  $z_t \leftarrow$ `LaserScanMeasurement`
  $\mathcal{T}_t \leftarrow$ `TrainingDataGenerator`(z_t)
  if $\delta = 1$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \mathcal{T}_t$
  else
    $\mathcal{T} \leftarrow \mathcal{T}_t$
  $\hat{h} \leftarrow$ `BarrierEstimator` $(\mathcal{T})$
  Solve the QP:

$$u^*(x) = \underset{u \in \mathbb{R}^m}{\text{argmin}} \quad ||u - k(x)||_2^2$$
$$\text{s.t} \quad L_f \widehat{h}(x) + L_g \widehat{h}(x) u(x) \geq -\alpha(\widehat{h}(x))$$

  $u \leftarrow u^*(x)$
  Solve (1), update state $x(t)$

are over-approximations of the true unsafe regions. That is, denote $\widehat{S} = \{x \in \mathcal{D} \mid \widehat{h}(x) \leq 0\}$ where $\widehat{S} : \mathcal{D} \to \mathbb{R}$ as the unsafe region estimated by Algorithm 2. Then, we have that $\mathcal{S} \subset \widehat{\mathcal{S}}$, where $\mathcal{S} = \bigcup_{i=1}^{p} \{x \in \mathcal{D} \mid h_i(x) \leq 0\}$ is the true unsafe region characterized by the unknown barrier functions h_i for all $i \in \{1, 2, \ldots, p\}$. In practice, this statement holds true for high resolution LiDAR sensors. Next, we provide a formal guarantee that Algorithm 3 ensure safety.

Theorem 2. *Suppose $\mathcal{S} \subset \widehat{S}$ and the controller from Algorithm 3 is used. Then given any $x(0) \in \widehat{\mathcal{S}}^c$ where $\widehat{\mathcal{S}}^c = \{x \in \mathcal{D} \mid \widehat{h}(x) \geq 0\}$, the robot trajectory is such that $x(t) \in \widehat{\mathcal{S}}^c$ for all $t \geq 0$.*

Proof. From Algorithm 3, the QP enforces the barrier function constraint (2) with $\widehat{h}$ as the ZCBF. Since the cost function of the QP is quasi-convex in u, the constraints are quasi-convex in u and the nominal policy k is continuous, from Proposition 8 in [15] we have that the generated control u is continuous. Hence from Theorem 1 and by assumption $\mathcal{S} \subset \widehat{\mathcal{S}}$, we have that the set $\widehat{\mathcal{S}}^c = \{x \in \mathcal{D} \mid \widehat{h}(x) \geq 0\}$ is rendered forward invariant. That is, we have that $x(t) \in \widehat{\mathcal{S}}^c$ for all $t \geq 0$. ∎

E. Online Barrier Function Synthesis & Control

When access to the full set of unsafe samples from the environment is not available, a real-time barrier function synthesis method is preferable. Here, we describe an online approach to synthesizing barrier functions, based on Algorithm 4. For online ZCBF synthesis, the set of unsafe samples covering the boundary of all the unsafe regions is not known a priori. Hence, at time $t = 0$, the system is initialized with no information regarding the state space, except the nominal feedback control policy. At each time instant t, the system obtains the depth measurement z_t and generates the training dataset $\mathcal{T}$. Then, Algorithm 2 synthesizes a local barrier function. Implementing the QP controller generates the control input at time instant t. In the next time instant, the same procedure repeats and a new barrier function is synthesized based on the updated sensor measurements.

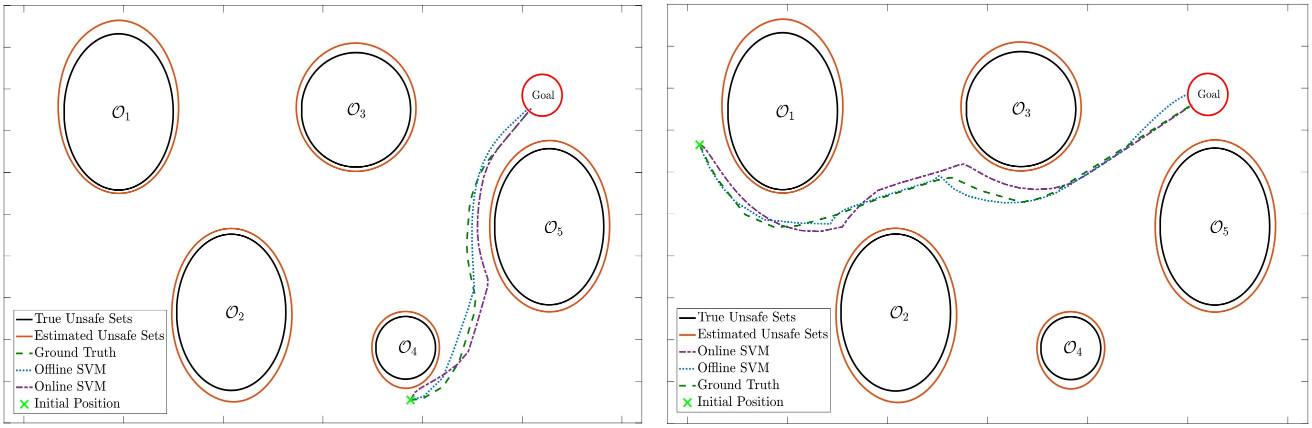
Two variations of the online barrier function synthesis method can be implemented. In the first method, the depth sensor data for all previous time instances is deleted, and the QP is solved with only the immediately sensed measurements. The barrier function approximates the true safe region only locally i.e., in a neighborhood around the state x_t of the robot. In the second method, samples from the previous time instant are aggregated with the samples from the current time instant, with Algorithm 2 implemented with the incremented set. The two cases synthesize different barrier function at each time instant. For the data-aggregation case, the estimate of the barrier improves as the number of samples characterizing the unsafe regions increases. Advantages and drawbacks exist for both approaches. In the data aggregation case, one needs to continuously update the dataset with new measurements and this exhaustive data collection process can become computationally expensive unless one resorts to efficient ways to store data [16]. For the non data aggregation case, computation is faster but the estimate of the barrier function does not improve iteratively as the robot traverses the domain. Both procedures are described in Algorithm 4.

Define the sensing range of the sensor as $\mathcal{B}_r(x) = \{\overline{x} \in \mathcal{D} \mid \|x - \overline{x}\| \leq r\}$, where $r \in \mathbb{R}_{>0}$ is the sensing range of the robot. Similar to the discussion in the previous subsection, it can be guaranteed that if $\theta_{\text{res}} \to 0$, then locally, Algorithm 2 synthesizes a barrier function whose level-set over approximates the true unsafe region. That is, denote $\widehat{\mathcal{S}_r}(x) = \{x \in \mathcal{B}_r(x) \mid \widehat{h}(x) \leq 0\}$ where $\widehat{h} : \mathcal{D} \to \mathbb{R}$ is the estimated ZCBF from Algorithm 2. Then, as $\theta_{\text{res}} \to 0$, we have that $\mathcal{S}_r(x) \subset \widehat{\mathcal{S}}_r(x)$ for all $x \in \mathcal{D}$ locally within the ball $\mathcal{B}_r(x)$, where $\mathcal{S}_r(x) = \bigcup_{i=1}^{p} \{\overline{x} \in \mathcal{B}_r(x) \mid h_i(\overline{x}) \leq 0\}$ is the true unsafe region. In the online case, a statement similar to Theorem 2 cannot be made since the robot does not have access to the full set of samples that characterize the entire boundary of the unsafe set and hence, there is no guarantee that globally in the domain the generated level-sets are over-approximations of the true unsafe regions. However, since the robot dynamics are locally Lipschitz continuous, safety holds locally as seen in Fig 1.

V. EXPERIMENTAL RESULTS

This section describes and discusses simulation results from a path planning perspective conducted on the "Simple Two Dimensional Robot (STDR) simulator"[1]. Two environments were created for use in STDR. The first environment contains five ellipsoidal obstacles scattered throughout a 3.2 x 2 workspace domain. The second environment of the same size contains more general obstacles whose shape cannot be characterized easily by level-sets of closed-form polynomials. In both cases, the robot has no a priori knowledge of

[1] `http://wiki.ros.org/stdr_simulator`



(a) (b)

Fig. 1: Trajectories generated for the robot in a five obstacle scenario. The robot must reach a goal region (red circle) which is known a priori. Three different trajectories are shown- the ground truth trajectory (dashed green), the offline kernel-SVM based controller trajectory (dotted blue), and the online kernel-SVM based controller trajectory (dash-dotted purple). Observe that estimated unsafe set is an over-approximation of the true unsafe sets, and hence Algorithm 3 guarantees collision free trajectories in the offline case, as per Theorem 2.

the environment and follows a nominal controller that drives it towards a goal point. More formally, we consider a robot with dynamics $\dot{x} = u$, where $x \in \mathcal{D} \subset \mathbb{R}^2$ is the position of the robot and $u \in \mathbb{R}^2$ is the control input. The nominal feedback control policy for all $x \in \mathcal{D}$ is given by $k(x) = \delta \cdot \frac{(x - x_{\text{goal}})}{\|x - x_{\text{goal}}\|}$, where $\delta \in \mathbb{R}_{>0}$, and $x_{\text{goal}} \in \mathcal{D}$ is a desired final goal position for the robot. Informally, the robot must follow $k(x)$ as close as possible while avoiding the unknown obstacles in the workspace. The robot must reach a goal region which is defined as $\mathcal{G} = \{x \in \mathcal{D} \mid \|x - x_{\text{goal}}\| \leq 0.1\}$. For the first scenario, depicted in Fig. 1, we obtain ground truth data using a grid-based solution, which is a common approach to compute the true signed distance to the obstacles. The signed distance function corresponds to the true barrier function characterizing the obstacles.

A. Evaluation Metrics

Comparison of the trajectory outcomes for the different implementations involves two evaluation metrics. The combination of both these metrics provides a means to evaluate the outcomes of the proposed algorithms.

1) Correlation Coefficient: Informally, the correlation coefficient between two trajectories captures the change in one trajectory with respect to the other. That is, one can obtain information regarding the flow of one trajectory with respect to the other. Typically, two trajectories are said to be highly correlated if they have a correlation coefficient greater than 0.7 [17]. We make use of the correlation coefficient to develop an intuition regarding the nature of the trajectories generated by the offline and online kernel-SVM based approaches compared with the ground truth data.

2) Fréchet Distance: Informally, the Fréchet distance provides a measure of the Euclidean distance mismatch between two trajectories. While the correlation coefficient provides information regarding the flow of two trajectories, the Fréchet distance provides an explicit degree of mismatch between the two. A lower Fréchet distance indicates less mismatch between the two trajectories. In particular, $F = 0$ implies that the two trajectories are identical.

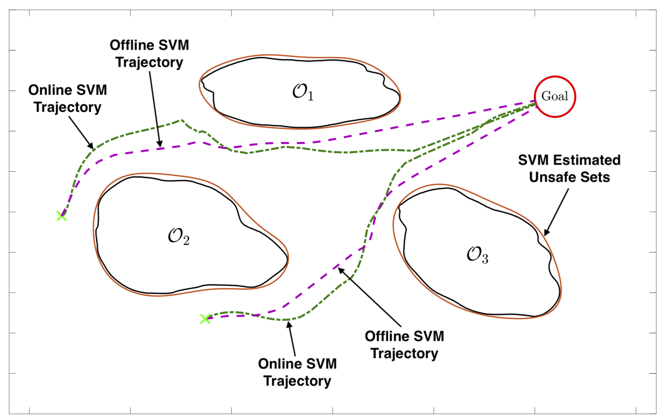


Fig. 2: An implementation in the STDR simulator where the robot has to navigate the unknown environment to reach a goal region (red circle). Offline kernel-SVM based controller and online kernel-SVM based controller trajectories for two different initial conditions (green crosses) are shown. The obstacles $\mathcal{O}_1$, $\mathcal{O}_2$ and $\mathcal{O}_3$ are such that they cannot be easily characterized by closed form polynomials, and hence, using the traditional CBF formulation is difficult. However, using Algorithm 3 and Algorithm 4, we can generate trajectories such that the robot remains safe.

B. Implementation Results

We first consider the five obstacle scenario shown in Fig. 1. Two different initial conditions for the robot are considered. The green dashed trajectory indicates the ground truth trajectory obtained when the barrier function for each obstacle is known a priori. A QP of the form (3) is solved to generate this trajectory. The blue, dotted trajectory is generated from

TABLE I: Correlation Coefficients and Fréchet Distances for Five Obstacle Scenario

Case	Offline SVM vs Ground Truth	Online SVM vs Ground Truth	Offline SVM vs Online SVM
1	0.99/0.04	0.97/0.08	0.97/0.08
2	0.96/0.06	0.80/0.08	0.89/0.13
3	0.99/0.02	0.97/0.04	0.96/0.04
4	0.98/0.05	0.94/0.14	0.91/0.12
5	0.99/0.06	0.94/0.15	0.94/0.13
6	0.99/0.03	0.98/0.03	0.98/0.02
7	0.98/0.11	0.98/0.12	0.98/0.04
8	0.96/0.04	0.76/0.07	0.58/0.06
9	0.98/0.05	0.87/0.13	0.86/0.11
10	0.99/0.03	0.98/0.03	0.98/0.03
Average	**0.98/0.05**	**0.92/0.09**	**0.91/0.08**

the offline kernel-SVM based barrier estimation approach as discussed in Algorithm 3. The purple, dash-dotted trajectory is generated using Algorithm 4 which is the online kernel-SVM based barrier function estimation method. Observe that in both the cases, the robots avoid the obstacle and follow the nominal control policy as close as possible. In the second scenario, we consider a situation where the obstacle shapes are such that finding the closed form expressions for the barrier functions is not straightforward. This setting is as shown in Fig. 2. The pink, dashed trajectories are generated using the offline kernel-SVM based barrier function approach as discussed in Algorithm 3, whereas the green, dash-dotted trajectories are generated using the online kernel-SVM based barrier function method described in Algorithm 4. A video of the simulations results is also provided[2].

C. Discussion

Table I compares the correlation coefficient and Freéchet distance for both the online and offline approaches with each other, and against the ground truth trajectory in the first scenario. On average, we obtain correlation coefficient values > 0.90, which shows a high similarity between the ground truth trajectory and the barrier estimated trajectory. In particular, note that the average correlation between the offline kernel-SVM approach and the ground truth trajectory is greater then 0.98. We then provide Fréchet distances which measures the degree of mismatch in terms of the Euclidean distance between two 2D trajectories. The smaller the Fréchet distance, the smaller the mismatch between the two trajectories. Observe that on average, we obtain distances < 0.10 for each case, which shows that the Euclidean distance mismatch between the trajectories is small. A key inference from the above data is that R_{offline} is very high and F_{offline} is very small, which shows that the offline kernel-SVM estimated barrier function closely replicates the true barrier function.

VI. Concluding Remarks

This paper presented a supervised machine learning based approach to automated synthesis of control barrier functions. A kernel-SVM based method classifies the set of safe and unsafe samples, and generates the desired barrier (level-set) function. A formal guarantee on zero misclassification of unsafe samples is provided along with guarantees on safety of the robot. Experimental simulations were conducted on an omnidirectional robot in a ROS-based simulator using synthetic LiDAR data. Some ideas for future work include understanding how to generate barrier functions that respect the constraint in Definition 1, and understanding the effect of computational lag in the safety guarantees.

[2] `https://youtu.be/-XiaR7QchtQ`

Acknowledgement

The authors thank Alex Chang for discussions regarding SVMs and for helping with the initial code base.

References

[1] J. C. Knight, "Safety critical systems: Challenges and directions," in *Proceedings of the 24th International Conference on Software Engineering*, ICSE '02, (New York, NY, USA), p. 547–550, Association for Computing Machinery, 2002.

[2] L. Wang, A. D. Ames, and M. Egerstedt, "Safety barrier certificates for collisions-free multirobot systems," *IEEE Transactions on Robotics*, vol. 33, pp. 661–674, June 2017.

[3] G. Notomista, S. Mayya, S. Hutchinson, and M. Egerstedt, "An optimal task allocation strategy for heterogeneous multi-robot systems," in *2019 18th European Control Conference (ECC)*, pp. 2071–2076, June 2019.

[4] M. Srinivasan, S. Coogan, and M. Egerstedt, "Control of multi-agent systems with finite time control barrier certificates and temporal logic," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 1991–1996, Dec 2018.

[5] L. Wang, D. Han, and M. Egerstedt, "Permissive barrier certificates for safe stabilization using sum-of-squares," in *2018 Annual American Control Conference (ACC)*, pp. 585–590, June 2018.

[6] S. M. Richards, F. Berkenkamp, and A. Krause, "The lyapunov neural network: Adaptive stability certification for safe learning of dynamical systems," in *Proceedings of the 2nd Conference on Robot Learning (CoRL)*, vol. 87, pp. 466–476, PMLR, 2018.

[7] N. Cristianini, J. Shawe-Taylor, *et al.*, *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press, 2000.

[8] 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.

[9] G. Francis, L. Ott, and F. Ramos, "Fast stochastic functional path planning in occupancy maps," in *International Conference on Robotics and Automation*, pp. 929–935, 2019.

[10] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European Control Conference (ECC)*, pp. 3420–3431, June 2019.

[11] *LiDAR Comparison Chart, https://autonomoustuff.com/lidar-chart/*.

[12] T. Gurriet, M. Mote, A. D. Ames, and E. Feron, "An online approach to active set invariance," in *2018 IEEE Conference on Decision and Control (CDC)*, pp. 3592–3599, Dec 2018.

[13] K. Veropoulos, C. Campbell, N. Cristianini, *et al.*, "Controlling the sensitivity of support vector machines," in *Proceedings of the international joint conference on AI*, vol. 55, p. 60, 1999.

[14] B. Hammer and K. Gersmann, "A note on the universal approximation capability of support vector machines," *neural processing letters*, vol. 17, no. 1, pp. 43–53, 2003.

[15] R. Konda, A. D. Ames, and S. Coogan, "Characterizing Safety: Minimal Barrier Functions from Scalar Comparison Systems," Aug 2019. Under review. `https://arxiv.org/abs/1908.09323`.

[16] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635, 2011.

[17] Y. J. Kim, S. W. Park, H. G. Yeom, M. S. Bang, J. S. Kim, C. K. Chung, and S. Kim, "A study on a robot arm driven by three-dimensional trajectories predicted from non-invasive neural signals," *Biomedical engineering online*, vol. 14, no. 1, p. 81, 2015.