Autonomous Navigation in Unknown Environments With Sparse Bayesian Kernel-Based Occupancy Mapping

Thai Duong, *Student Member, IEEE*, Michael Yip, *Senior Member, IEEE*, and Nikolay Atanasov, *Member, IEEE*

***Abstract*—This article focuses on online occupancy mapping and real-time collision checking onboard an autonomous robot navigating in a large unknown environment. Commonly used voxel and octree map representations can be easily maintained in a small environment but have increasing memory requirements as the environment grows. We propose a fundamentally different approach for occupancy mapping, in which the boundary between occupied and free space is viewed as the decision boundary of a machine learning classifier. This work generalizes a kernel perceptron model which maintains a very sparse set of support vectors to represent the environment boundaries efficiently. We develop a probabilistic formulation based on relevance vector machines, handling measurement noise, and probabilistic occupancy classification, supporting autonomous navigation. We provide an online training algorithm, updating the sparse Bayesian map incrementally from streaming range data, and an efficient collision-checking method for general curves, representing potential robot trajectories. The effectiveness of our mapping and collision checking algorithms is evaluated in tasks requiring autonomous robot navigation and active mapping in unknown environments.**

***Index Terms*—Autonomous navigation, collision avoidance, kernel-based occupancy mapping, relevance vector machine (RVM), sparse Bayesian classification.**

I. Introduction

AUTONOMOUS navigation in robotics involves online localization, mapping, motion planning, and control in partially known environments perceived through streaming data from onboard sensors [1], [2]. This article focuses on the occupancy mapping problem and, specifically, on enabling large-scale, yet compact, representations and efficient collision checking to support autonomous navigation. Occupancy mapping is a well-established and widely studied problem in robotics, and a variety of explicit and implicit map representations have been proposed. Explicit maps model the obstacle surfaces Directly, e.g., via surfels [3]–[7], geometric primitives [8]–[12], or polygonal meshes [13]–[15]. Implicit maps model the obstacle surfaces as the level set of an occupancy [16]–[21] or signed distance [22]–[26] or spatial function encoded via voxels [25]–[27] or octrees [19]–[21]. The goal of this work is to generate *sparse probabilistic* maps online, enabling large-scale environment modeling, map uncertainty quantification, and efficient collision checking.

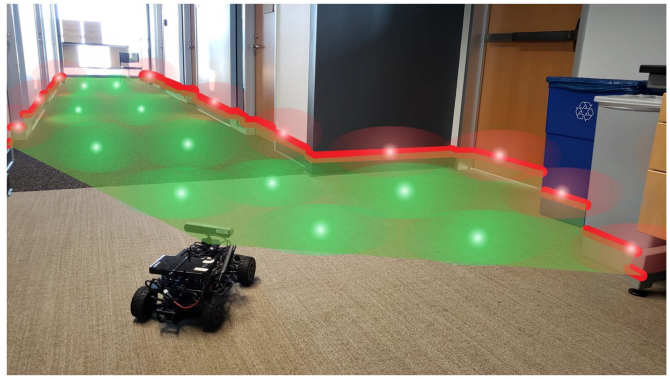


Fig. 1. Ground robot in an unknown environment relying on lidar scan data (red) for online occupancy mapping and collision-free trajectory planning. Our mapping algorithm learns a sparse set of occupied and free relevance vectors (light red and green dots, respectively) that represents the environment based on the lidar scans.

Our preliminary work [28] develops a kernel perceptron model for online occupancy mapping. The model uses support vectors and a kernel function to represent obstacle boundaries in configuration space. The number of support vectors scales with the complexity of the obstacle boundaries rather than the environment size. We develop an online training algorithm to update the support vectors incrementally as new range observations of the local surroundings are provided by the robot's sensors. To enable motion planning in the new occupancy representation, we develop efficient collision checking algorithms for piecewise-linear and piecewise-polynomial trajectories in configuration space. Our kernel perceptron model, however, provides occupancy labels without a probability distribution, making the classification accuracy susceptible to measurement noise. Since unknown regions are frequently assumed free for

Manuscript received 22 January 2022; accepted 26 April 2022. Date of publication 16 June 2022; date of current version 6 December 2022. This work was supported by ARL under Grant DCIST CRA W911NF-17-2-0181 and ONR under Grant SAI N00014-18-1-2828. This paper was recommended for publication by Associate Editor A. Kim and Editor F. Chaumette upon evaluation of the reviewers' comments. *(Corresponding author: Thai Duong.)*

The authors are with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: tduong@ucsd.edu; yip@ucsd.edu; natanasov@ucsd.edu).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TRO.2022.3177950.

Digital Object Identifier 10.1109/TRO.2022.3177950

motion planning purposes, the lack of probabilistic information also does not allow us to distinguish between well-observed and unseen regions. This is especially important in active exploration problems, where the robot autonomously chooses the unknown regions to explore.

This article develops a sparse Bayesian formulation of the occupancy mapping problem and introduces an incremental relevance vector machine (RVM) training algorithm to probabilistically model the environment. To make our sparse Bayesian kernel-based map (SBKM) compatible with motion planning algorithms, we derive collision checking algorithms for linear and general trajectories.

Contributions. This article introduces a sparse Bayesian kernel-based mapping method that does the following:

1) represents continuous-space probabilistic occupancy using a sparse set of relevance vectors stored in an R^*-tree data structure (Sections V-B and VIII-A);
2) allows online map updates from streaming partial observations using an incremental RVM training algorithm with the predictive distribution modeled by a probit function (Section V-B);
3) provides efficient and complete (without sampling) collision checking for robot trajectories (Sections VI and VIII-B).

II. Related Work

Occupancy grid mapping is a commonly used approach for modeling the free and occupied space of an environment. The space is discretized into a collection of cells, whose occupancy probabilities are estimated online using the robot's sensory data. While early work [29], [30] assumes that the cells are independent, Gaussian process (GP) occupancy mapping [31]–[33] uses a kernel function to capture the correlation among grid cells and predict the occupancy of unobserved cells. Online training of a GP model, however, does not scale well as its computational complexity grows cubically with the number of data points. Ramos *et al.* [34] improve on this by projecting the data points into Hilbert space and training a logistic regression model. Senanayake and Ramos [35] propose a Bayesian treatment of Hilbert maps, called sequential Bayesian Hilbert map (SBHM), that updates the map from sequential observations of the environment. They achieve sparseness by calculating feature vectors based on a sparse set of hinged points, e.g., on a coarse grid. Instead of a fixed set of hinged points, localized automatic relevance determination Hilbert maps (LARD-HM) [36] and efficient Hilbert maps [37] find the hinged points by clustering the training data points using *k*-means algorithms and calculate their kernel parameters using automatic relevance determination. Meanwhile, RVM [38]–[40] learns a sparse set of relevance vectors from the training dataset. The original RVM work [38] initially assumes that all data points are relevance vectors and prunes them down, incurring high computation cost. Tipping and Faul [40] derive a fast training algorithm that starts from an empty set of relevance vectors and adds points to the set gradually. Meanwhile, Lopez and How [41] propose an efficient deterministic alternative, which builds a *k-d* tree from point clouds and queries the nearest obstacles for collision checking. Using spatial partitioning similar to a *k-d* tree, octree-based maps [19], [42] offer efficient map storage by performing octree compression, while AtomMap [43] stores a collection of spheres in a *k-d* tree as a way to avoid grid cell discretization of the map. Instead of storing occupancy information, Voxblox [25] stores distance to obstacles in each cell and builds an Euclidean signed distance field, as a map representation, online from streaming sensor data.

Navigation, in an unknown environment, requires the safety of potential robot trajectories to be evaluated through a huge amount of collision checks with respect to the map representation [44]–[46]. Many works rely on sampling-based (SB) collision checking, simplifying the safety verification of continuous-time trajectories by evaluating only a finite set of samples along the trajectory [45], [47]. This may be undesirable in safety critical applications. Bialkowski *et al.* [44] propose an efficient collision checking method using safety certificates with respect to the nearest obstacles. Using a different perspective, learning-based collision checking methods [48]–[50] sample data from the environment and train machine learning models to approximate the obstacle boundaries. Pan *et al.* [49] propose an incremental support vector machine model for pairs of obstacles but train the models offline. Closely related to our work, Das *et al.* [48], [51] develop an online training algorithm, called Fastron, to train a kernel perceptron collision classifier. To handle dynamic environments, Fastron actively resamples the environment and updates the model globally. Geometry-based collision checking methods, such as the flexible collision library [52], are also related but rely on mesh representations of the environment which may be inefficient to generate from local observations.

Our preliminary work [28], summarized in Section IV, provides an approach to online occupancy mapping that supports efficient collision checking with guarantees. However, to handle noisy measurements and probabilistically model well-observed and unknown regions, we introduce a probabilistic formulation based on RVM inference that enables online sparse Bayesian kernel-based occupancy mapping. Inspired by GP mapping techniques, we utilize a kernel function to capture occupancy correlations but focus on a compact representation of obstacle boundaries by building an RVM model, i.e., a sparse set of relevance vectors, incrementally from streaming local sensor data. Specifically, only a local subset of the relevance vectors is updated each time using our incremental RVM training algorithm. Furthermore, motivated by the collision checking approach in [44], we derive our own efficient collision checking algorithms for our map representation. We develop an "inflated boundary" of the obstacle boundary that enables closed-form conditions for checking line segments and ellipsoids for collision. These key conditions allow us to check potential robot trajectories for motion planning purposes.

III. Problem Formulation

Consider a robot with state $\mathbf{s} \in \mathcal{S}$, consisting of the robot's position $\mathbf{p} \in [0, 1]^d$ and other variables such as orientation,

velocity, etc., navigating in an unknown environment (Fig. 1). Let $\mathcal{O} \subset [0,1]^d$ be a closed set representing occupied space and let $\mathcal{F}$ be its complement, representing free space. Assume that the robot can be enclosed by a sphere of radius $r \in \mathbb{R}_{>0}$ centered at $\mathbf{p}$. In configuration space (C-space), the robot body becomes a point $\mathbf{p}$, while the obstacle space and free space are transformed as $\bar{\mathcal{O}} = \cup_{\mathbf{x}\in\mathcal{O}} \mathcal{B}(\mathbf{x}, r)$, where $\mathcal{B}(\mathbf{x}, r) = \{\mathbf{x}' \in [0,1]^d : \|\mathbf{x} - \mathbf{x}'\|_2 \leq r\}$, and $\bar{\mathcal{F}} = [0,1]^d \backslash \bar{\mathcal{O}}$. Let $\bar{\mathcal{S}}$ be the subset of the robot state space that corresponds to the collision-free robot positions $\bar{\mathcal{F}}$.

Let $\dot{\mathbf{s}}(t) = f(\mathbf{s}(t), \mathbf{a}(t))$ characterize the continuous-time robot dynamics with control input trajectory $\mathbf{a}(t) \in \mathcal{A}$. We consider constant control inputs (zero-order hold) applied at discrete time steps t_k for $k = 0, 1, \ldots, N$ so that $\mathbf{a}(t) \equiv \mathbf{a}_k$ for $[t_k, t_{k+1})$. We assume that the state $\mathbf{s}(t)$ is known or estimated by a localization algorithm and let $\mathbf{s}_k := \mathbf{s}(t_k)$.

The robot is equipped with a sensor, such as lidar or depth camera, that provides distance measurements $\mathbf{z}_k$ at time t_k to the obstacle space $\mathcal{O}$ within its field of view. Our objective is to construct an occupancy map $\hat{m}_k : [0,1]^d \to \{-1, 1\}$ of the C-space based on accumulated observations $\mathbf{z}_{0:k}$, where "-1" and "1" mean "free" and "occupied," respectively. As the robot is navigating, new sensor data are used to update the map as a function, $\hat{m}_{k+1} = g(\hat{m}_k, \mathbf{z}_k)$, of the previous estimate $\hat{m}_k$ and a newly received range observation $\mathbf{z}_k$. Assuming unobserved regions are free, we rely on $\hat{m}_k$ to plan a robot trajectory to a goal region $\mathcal{G} \subseteq \bar{\mathcal{S}}$. Applying control action $\mathbf{a}$ at $\mathbf{s}$ incurs a motion cost $c(\mathbf{s}, \mathbf{a})$, e.g., based on traveled distance or energy expenditure, and we aim to minimize the cumulative cost of navigating safely to the goal $\mathcal{G}$.

Problem 1: Given a start state $\mathbf{s}_0 \in \bar{\mathcal{S}}$ and a goal region $\mathcal{G} \subseteq \bar{\mathcal{S}}$, find a sequence of control actions which leads the robot to $\mathcal{G}$ safely, while minimizing the motion cost

$$\begin{aligned} \min_{N, \mathbf{a}_0, \ldots, \mathbf{a}_N} \quad & \sum_{k=0}^{N-1} c(\mathbf{s}_k, \mathbf{a}_k) \\ \text{s.t.} \quad & \dot{\mathbf{s}} = f(\mathbf{s}, \mathbf{a}), \mathbf{a}(t) = \mathbf{a}_k \text{ for } t \in [t_k, t_{k+1}), \\ & \mathbf{s}(t_0) = \mathbf{s}_0,\ \mathbf{s}_N \in \mathcal{G},\ \hat{m}_{k+1} = g(\hat{m}_k, \mathbf{z}_k), \\ & \hat{m}_k(\mathbf{s}(t)) = -1 \text{ for } t \in [t_k, t_{k+1}), k = 0, \ldots, N. \end{aligned} \tag{1}$$

We next develop a SBKM representation, offering efficient collision checking for robot trajectories, and propose a complete solution to Problem 1.

IV. Sparse Kernel-Based Classifier for Occupancy Mapping

Our preliminary work [28] on sparse Kernel-based map (SKM) develops a sparse kernel perceptron model for online classification of occupied and free space in the environment. The model uses a set of support vectors and a kernel function to represent the obstacle boundaries in configuration space. The number of support vectors necessary for accurate classification scales with the complexity of the obstacle boundaries rather than the environment size. Our approach extends the Fastron algorithm [48], [49], which efficiently trains a kernel perceptron model using a training dataset collected globally from the environment. We develop an online training procedure (Algorithm 1) that updates the support vectors incrementally as new range observations $\mathbf{z}_k$ of the local surroundings arrive. Given a training dataset $\mathcal{D} = \{(\mathbf{x}_l, y_l)\}$ generated from $\mathbf{z}_k$ (e.g., see Section VIII-A for details), Algorithm 1 prioritizes updating misclassified points' weight based on their margins (lines 6 and 7) and remove the redundant support vectors (line 8) without affecting the model. When the next local dataset arrives, it looks for new misclassified points and incrementally adds them to the set of support vectors. Algorithm 1 returns a set of M^+ positive support vectors and their weight $\Lambda^+ = \{(\mathbf{x}_i^+, \alpha_i^+)\}_i$ and a set of M^- negative support vectors and their weight $\Lambda^- = \{(\mathbf{x}_j^-, \alpha_j^-)\}_j$. The classifier decision boundary is characterized by a score function

$$F(\mathbf{x}) = \sum_{i=1}^{M^+} \alpha_i^+ k\left(\mathbf{x}_i^+, \mathbf{x}\right) - \sum_{j=1}^{M^-} \alpha_j^- k(\mathbf{x}_j^-, \mathbf{x}) \tag{2}$$

where $k(\cdot, \cdot)$ is a kernel function and $\alpha_j^-, \alpha_i^+ > 0$. The occupancy of a query point $\mathbf{x}$ can be checked by evaluating the score function $F(\mathbf{x})$ in (2). Specifically, $\hat{m}_t(\mathbf{x}) = -1$ if $F(\mathbf{x}) < 0$ and $\hat{m}_t(\mathbf{x}) = 1$ if $F(\mathbf{x}) \geq 0$. The score calculation becomes slower when the number of support vectors increases. We improve on this by storing the support vectors in an R^*-tree data structure and efficiently query K^+ and K^- nearest positive and negative support vectors (line 1 in Algorithm 1) from the R^*-tree to approximate $F(\mathbf{x})$.

Motivated by the use of piecewise-linear and piecewise-polynomial trajectories in many robot motion planning and control algorithms [53]–[55], we derive conditions to classify lines and curves, i.e., to check if every point on the curve is free using the trained model. Checking that a curve $\mathbf{p}(t)$ is classified as free is equivalent to verifying that $F(\mathbf{p}(t)) < 0$, $\forall t \geq 0$. It is not possible to express this condition for t explicitly due to the nonlinearity of F. In Proposition 1, we show that an accurate upper bound $\bar{F}(\mathbf{p}(t))$ on the score $F(\mathbf{p}(t))$ exists and can be used to evaluate the condition $\bar{F}(\mathbf{p}(t)) < 0$ explicitly in t. The upper bound provides a conservative but fairly accurate "inflated boundary" and allows efficient classifications of curves $\mathbf{p}(t)$, assuming a radial basis function (RBF) kernel $k(\mathbf{x}, \mathbf{x}') = \eta \exp\left(-\gamma \|\mathbf{x} - \mathbf{x}'\|^2\right)$ is used.

Proposition 1 ([28]): For any $(\mathbf{x}_j^-, \alpha_j^-) \in \Lambda^-$, the score $F(\mathbf{x})$ is bounded above by $\bar{F}(\mathbf{x}) = k(\mathbf{x}, \mathbf{x}_*^+) \sum_{i=1}^{M^+} \alpha_i^+ - k(\mathbf{x}, \mathbf{x}_j^-)\alpha_j^-$ where $\mathbf{x}_*^+$ is the closest positive support vector to $\mathbf{x}$.

To check if a line $\mathbf{p}(t)$ collides with the inflated boundary, we find the first time t_u such that $\bar{F}(\mathbf{p}(t_u)) \geq 0$. This means that $\mathbf{p}(t)$ is classified as free for $t \in [0, t_u)$.

Proposition 2 ([28]): Consider a ray $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}$, $t \geq 0$ such that $\mathbf{p}_0$ is classified as free and $\mathbf{v}$ is constant. Let $\mathbf{x}_i^+$ and $\mathbf{x}_j^-$ be arbitrary positive and negative support vectors. Any point $\mathbf{p}(t)$ with $t \in [0, t_u) \subseteq [0, t_u^*)$ is free for

Algorithm 1: Incremental Kernel Perceptron Training [28].

Input: Support vectors $\Lambda^+ = \{(\mathbf{x}_i^+, \alpha_i^+)\}_i$ and $\Lambda^- = \{(\mathbf{x}_j^-, \alpha_j^-)\}_j$ stored in an R^*-tree; local dataset $\mathcal{D} = \{(\mathbf{x}_l, y_l)\}$; $\xi^+, \xi^- > 0$; $N_{\max}$.
Output: Updated Λ^+, Λ^-.
Query K^+, K^- nearest negative and positive support vectors from an R^*-tree data structure.
for $(\mathbf{x}_l, y_l)$ in $\mathcal{D}$ **do**
  Calculate $F_l = \sum_{i=1}^{K^+} \alpha_i^+ k(\mathbf{x}_i^+, \mathbf{x}_l) - \sum_{j=1}^{K^-} \alpha_j^- k(\mathbf{x}_j^-, \mathbf{x}_l)$
for $t = 1$ to $N_{\max}$ **do**
  if $y_l F_l > 0 \quad \forall l$ **then return** Λ^+, Λ^-
  $m = \text{argmin}_l y_l F_l$
  WEIGHT CORRECTION $F_m, y_m, \Lambda^+, \Lambda^-, \xi^+, \xi^-$
  REDUNDANCY REMOVAL $\Lambda^+, \Lambda^-, \mathcal{D}$
return Λ^+, Λ^-
function WEIGHT CORRECTION$F_m, y_m, \Lambda^+, \Lambda^-, \xi^+, \xi^-$
  $\xi = \xi^+$ if $y_m > 0$; and $\xi = \xi^-$, otherwise.
  Calculate $\Delta_\alpha = \xi y_m - F_m$.
  if $\exists(\mathbf{x}_m, \alpha_m) \in \Lambda^+ \cup \Lambda^-$ **then**
    Update weights: α_m+=$y_m \Delta\alpha$, F_l+=$k(\mathbf{x}_l, \mathbf{x}_m) y_m \Delta\alpha, \forall l$
  else
    Calculate $\alpha_m = y_m \Delta\alpha$
    Add $(\mathbf{x}_m, \alpha_m)$ to Λ^+ if $y_m > 0$ and Λ^-, otherwise.
function REDUNDANCY REMOVAL$\Lambda^+, \Lambda^-, \mathcal{D}$
  for $(\mathbf{x}_l, y_l) \in \mathcal{D}$ **do**
    if $\exists(\mathbf{x}_l, \alpha_l) \in \Lambda^+ \cup \Lambda^-$ and $y_l(F_l - \alpha_l^+) > 0$ **then**
      Remove $(\mathbf{x}_l, \alpha_l)$ from Λ^+ or Λ^-
      Update F_n-= $k(\mathbf{x}_l, \mathbf{x}_n)\alpha_l^+, \forall(\mathbf{x}_n, \cdot) \in \mathcal{D}$

$$t_u := \min_{i \in \{1,\dots,M^+\}} \rho(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-), \tag{3}$$

$$t_u^* := \min_{i \in \{1,\dots,M^+\}} \max_{j \in \{1,\dots,M^-\}} \rho(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) \tag{4}$$

where $\beta = \frac{1}{\gamma}\left(\log\left(\alpha_j^-\right) - \log\left(\sum_{i=1}^{M^+} \alpha_i^+\right)\right)$ and

$$\rho(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) = \begin{cases} +\infty, & \text{if } \mathbf{v}^T(\mathbf{x}_i^+ - \mathbf{x}_j^-) \le 0 \\ \frac{\beta - \|\mathbf{p}_0 - \mathbf{x}_j^-\|^2 - \|\mathbf{p}_0 + \mathbf{x}_i^+\|^2}{2\mathbf{v}^T(\mathbf{x}_j^- - \mathbf{x}_i^+)}, & \text{if } \mathbf{v}^T(\mathbf{x}_i^+ - \mathbf{x}_j^-) > 0 \end{cases}$$

.

For a line segment $(\mathbf{p}_A, \mathbf{p}_B)$, all points on the segment can be expressed as $\mathbf{p}(t_A) = \mathbf{p}_A + t_A \mathbf{v}_A$, $\mathbf{v}_A = \mathbf{p}_B - \mathbf{p}_A$, $0 \le t_A \le 1$ or $\mathbf{p}(t_B) = \mathbf{p}_B + t_B \mathbf{v}_B$, $\mathbf{v}_B = \mathbf{p}_A - \mathbf{p}_B$, $0 \le t_B \le 1$. Using the upper bound provided by (3) or (4), we find the free regions $[0, t_{uA})$ and $[0, t_{uB})$ starting from $\mathbf{p}_A$ and $\mathbf{p}_B$, respectively. If the free regions overlap, the segment is classified as free and vice versa.

We extend line segment classification to general curves by finding a Euclidean ball $\mathcal{B}(\mathbf{p}_0, r)$ whose interior is free.

Corollary 1 ([28]): Let $\mathbf{p}_0 \in \mathcal{C}$ be such that $\bar{F}(\mathbf{p}_0) < 0$ and let $\mathbf{x}_i^+$ and $\mathbf{x}_j^-$ be arbitrary positive and negative support vectors. Then, every point inside the Euclidean balls $\mathcal{B}(\mathbf{p}_0, r_u) \subseteq \mathcal{B}(\mathbf{p}_0, r_u^*)$ is free for

$$r_u := \min_{i \in \{1,\dots,M^+\}} \bar{\rho}(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-), \tag{5}$$

$$r_u^* := \min_{i \in \{1,\dots,M^+\}} \max_{j \in \{1,\dots,M^-\}} \bar{\rho}(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) \tag{6}$$

where $\bar{\rho}(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) = \frac{\beta - \|\mathbf{p}_0 - \mathbf{x}_j^-\|^2 + \|\mathbf{p}_0 - \mathbf{x}_i^+\|^2}{2\|\mathbf{x}_j^- - \mathbf{x}_i^+\|}$ and $\beta = \frac{1}{\gamma}(\log(\alpha_j^-) - \log(\sum_{i=1}^{M^+} \alpha_i^+))$.

Consider a polynomial $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{a}_1 t + \mathbf{a}_2 t^2 + \cdots + \mathbf{a}_d t^d, t \in [0, t_f]$ from $\mathbf{p}_0$ to $\mathbf{p}_f := \mathbf{p}(t_f)$. Corollary 1 shows that all points inside $\mathcal{B}(\mathbf{p}_0, r)$ are free for $r = r_u$ or r_u^*. If we can find the smallest positive t_1 such that $\|\mathbf{p}(t_1) - \mathbf{p}_0\| = r$, then all points on the curve $\mathbf{p}(t)$ for $t \in [0, t_1)$ are free. We classify the polynomial curve by iteratively covering it by Euclidean balls. If any ball's radius is smaller than a threshold ε, the curve is considered colliding. Otherwise, it is considered free.

The sparse kernel-based model [28] is accurate, updates efficiently from streaming range data, and evaluates curves $\mathbf{p}(t)$ for collisions without sampling. However, the model does not provide occupancy probability, which is desirable in autonomous navigation applications for distinguishing between unknown and well-observed free regions and for identifying map areas with large uncertainty. This observation motivates us to develop a sparse probabilistic model for online occupancy classification and efficient collision checking.

V. ONLINE PROBIT RVM TRAINING

In this section, we develop an online probit RVM training algorithm that builds a sparse probabilistic model for online occupancy mapping from streaming range observations.

A. Relevance Vector Machine Preliminaries

A RVM [40] is a sparse Bayesian approach for classification. Given a training dataset of N binary-labeled samples $\mathcal{D} = (\mathbf{X}, \mathbf{y}) = \{(\mathbf{x}_l, y_l)\}_l$, where $y_l \in \{-1, 1\}$, an RVM model maintains a sparse set of relevance vectors $\mathbf{x}_m$ for $m = 1, \dots, M$. The relevance vectors map a point $\mathbf{x}$ to a feature vector $\mathbf{\Phi}_\mathbf{x} = [k_1(\mathbf{x}), k_2(\mathbf{x}), \dots, k_M(\mathbf{x})]^\top \in \mathbb{R}^M$ via a kernel function $k_m(\mathbf{x}) := k(\mathbf{x}, \mathbf{x}_m)$. The likelihood of label y at point $\mathbf{x}$ is modeled by squashing a linear feature function

$$F(\mathbf{x}) := \mathbf{\Phi}_\mathbf{x}^\top \boldsymbol{w} + b \tag{7}$$

with weights $\boldsymbol{w} \in \mathbb{R}^M$ and bias $b \in \mathbb{R}$ through a function $\sigma : \mathbb{R} \mapsto [0, 1]$

$$\mathbb{P}(y = 1 | \mathbf{x}, \boldsymbol{w}) = \sigma(F(\mathbf{x})), \mathbb{P}(y = -1 | \mathbf{x}, \boldsymbol{w}) = 1 - \sigma(F(\mathbf{x})).$$

Note that (2) is a special case of (7) with $b = 0$. Examples of σ are the logistic function $\sigma(f) := \frac{1}{1 + \exp(-f)}$ and the probit function $\sigma(f) := \int_{-\infty}^{f} \varphi(z) dz$, where $\varphi(z) := \frac{1}{\sqrt{2\pi}} \exp(-z^2/2)$ is the standard normal probability density. The data likelihood of the

whole training set is

$$p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w}) = \prod_{l=1}^{N} \sigma(F(\mathbf{x}_l))^{\frac{1+y_l}{2}} \left(1-\sigma(F(\mathbf{x}_l))\right)^{\frac{1-y_l}{2}}. \quad (8)$$

An RVM model imposes a Gaussian prior on each weight w_m with zero mean and precision ξ_m (i.e., variance $1/\xi_m$)

$$p(\boldsymbol{w}|\boldsymbol{\xi}) = (2\pi)^{M/2} \prod_{m=1}^{M} \xi_m^{1/2} \exp\left(-\frac{\xi_m w_m^2}{2}\right). \quad (9)$$

The weight posterior is obtained via Bayes' rule

$$p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X},\boldsymbol{\xi}) = \frac{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\xi})}{p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\xi})}. \quad (10)$$

The precision $\boldsymbol{\xi}$ is determined via type-II maximum likelihood estimation, i.e., by maximizing the marginal likelihood

$$\mathcal{L}(\boldsymbol{\xi}) = \log p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{\xi}) = \log \int p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\xi})d\boldsymbol{w}. \quad (11)$$

Given a maximizer $\boldsymbol{\xi}$, the posterior $p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X},\boldsymbol{\xi})$ is generally intractable and approximated by a Gaussian distribution $p(\boldsymbol{w}|\mathbf{y},\mathbf{X},\boldsymbol{\mu},\boldsymbol{\Sigma})$ with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ using Laplace approximation [56]. Training consists in determining $\boldsymbol{\xi}$, $\boldsymbol{\mu}$, and $\boldsymbol{\Sigma}$.

At test time, due to the Laplace approximation, the predictive distribution of a query point $\mathbf{x}$ becomes

$$p(y|\mathbf{x},\boldsymbol{\xi}) \approx \int p(y|\mathbf{x},\boldsymbol{w})p(\boldsymbol{w}|\mathbf{y},\mathbf{X},\boldsymbol{\mu},\boldsymbol{\Sigma})d\boldsymbol{w}. \quad (12)$$

The usual formulation of RVM [40] uses a logistic function for σ, requiring additional approximations to the integral in (12). We emphasize that using a probit function, instead, enables a closed-form for the predictive distribution

$$\begin{aligned} p(y|\mathbf{x},\boldsymbol{\xi}) &\approx \int \sigma(y(\boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{w}+b))p(\boldsymbol{w}|\mathbf{y},\mathbf{X},\boldsymbol{\mu},\boldsymbol{\Sigma})d\boldsymbol{w} \\ &= \sigma\left(\frac{y(\Phi_{\mathbf{x}}^{\top}\boldsymbol{\mu}+b)}{\sqrt{1+\Phi_{\mathbf{x}}^{\top}\boldsymbol{\Sigma}\Phi_{\mathbf{x}}}}\right). \end{aligned} \quad (13)$$

This expression enables our results on closed-form classification of curves in Section VI.

We review the details of RVM training and then propose an online training algorithm that handles streaming training data.

1) Laplace Approximation: Approximation of the weight posterior $p(\boldsymbol{w}|\boldsymbol{y},\boldsymbol{X},\boldsymbol{\xi})$ is performed by fitting a Gaussian density function around its mode $\boldsymbol{\mu}$, the maximizer of

$$L(\boldsymbol{w}) := \log(p(\boldsymbol{y}|\boldsymbol{X},\boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{\xi})). \quad (14)$$

Substituting (8) and (9) in (14), we can obtain the gradient and Hessian of $L(\boldsymbol{w})$ for the probit function σ

$$\nabla L(\boldsymbol{w}) = \boldsymbol{\Phi}^{\top}\boldsymbol{\delta} - \mathbf{A}\boldsymbol{w}, \quad \nabla^2 L(\boldsymbol{w}) = -\boldsymbol{\Phi}^{\top}\mathbf{B}\boldsymbol{\Phi} - \mathbf{A} \quad (15)$$

where $\boldsymbol{\Phi} \in \mathbb{R}^{N\times M}$ is the feature matrix with entries $\boldsymbol{\Phi}_{i,j} := k_j(\mathbf{x}_i)$, $\boldsymbol{\delta} \in \mathbb{R}^N$ is a vector with entries $\delta_l := \frac{\varphi(y_l F(\mathbf{x}_l))}{\sigma(y_l F(\mathbf{x}_l))} y_l$, $\mathbf{A} := \operatorname{diag}(\boldsymbol{\xi}) \in \mathbb{R}^{M\times M}$, $\mathbf{B} := \operatorname{diag}(\mathbf{D}\boldsymbol{\Phi}^{\top}\boldsymbol{w} + b\boldsymbol{\delta} + \mathbf{D}\boldsymbol{\delta}) \in \mathbb{R}^{N\times N}$, and $\mathbf{D} := \operatorname{diag}(\boldsymbol{\delta}) \in \mathbb{R}^{N\times N}$. The Hessian is negative semidefinite, and, hence, $L(\boldsymbol{w})$ is concave. Setting $L(\boldsymbol{w}) = 0$, we obtain a Gaussian approximation $p(\boldsymbol{w}|\mathbf{y},\mathbf{X},\boldsymbol{\mu},\boldsymbol{\Sigma})$ with

$$\boldsymbol{\Sigma} = \left(\boldsymbol{\Phi}^{\top}\mathbf{B}\boldsymbol{\Phi} + \mathbf{A}\right)^{-1}, \quad (16)$$

$$\boldsymbol{\mu} = \boldsymbol{\Sigma}\boldsymbol{\Phi}^{\top}\mathbf{B}\left(\boldsymbol{\Phi}\boldsymbol{\mu} + \mathbf{B}^{-1}\boldsymbol{\delta}\right) \quad (17)$$

where $\boldsymbol{\mu}$ is defined implicitly and is obtained via first- or second-order ascent in practice [57]. Laplace approximation provides a closed-form approximated posterior, which enables efficient classifications of points, line segments, and curves, as shown in Section VI. When the true posterior is multimodal, Laplace approximation might not provide sufficient accuracy because it captures only one of the modes.

2) Sequential RVM Training: To determine the precision $\boldsymbol{\xi}$ of the weight prior in (9), Tipping and Faul [40] proposed a sequential training algorithm that starts from an empty set of relevance vectors, i.e., $\xi_l = \infty$, and incrementally introduces new vectors to maximize the marginal likelihood in (11)

$$\mathcal{L}(\boldsymbol{\xi}) \approx -\frac{1}{2}\left(N\log 2\pi + \log\det\mathbf{C} + \hat{\boldsymbol{t}}^{\top}\boldsymbol{C}^{-1}\hat{\boldsymbol{t}}\right) \quad (18)$$

where $\hat{\mathbf{t}} := \boldsymbol{\Phi}\boldsymbol{\mu} + \mathbf{B}^{-1}\boldsymbol{\delta}$ and $\mathbf{C} := \mathbf{B} + \boldsymbol{\Phi}\boldsymbol{A}^{-1}\boldsymbol{\Phi}^{\top}$. For each $(\mathbf{x}_l, y_l)$ in the training set $\mathcal{D}$, define $\theta_l = q_l^2 - s_l$ as follows:

$$s_l := \begin{cases} \frac{\xi_l S_l}{\xi_l - S_l}, & \text{if } \xi_l < \infty \\ S_l, & \text{else} \end{cases} \quad q_l := \begin{cases} \frac{\xi_l Q_l}{\xi_l - S_l}, & \text{if } \xi_l < \infty \\ Q_l, & \text{else} \end{cases} \quad (19)$$

where $S_l = \boldsymbol{\Phi}_l^{\top}\boldsymbol{C}^{-1}\boldsymbol{\Phi}_l$, $Q_l = \boldsymbol{\Phi}_l^{\top}\boldsymbol{C}^{-1}\hat{\boldsymbol{t}}$, and $\boldsymbol{\Phi}_l$ is the lth row of $\boldsymbol{\Phi}$. If $\theta_l > 0$, the point $\mathbf{x}_l$ is updated (if $\xi_l < \infty$) or added (if $\xi_l = \infty$) as a relevance vector with $\xi_l = \frac{s_l^2}{q_l^2 - s_l}$. If $\theta_l \le 0$ and $\xi_l < \infty$, the point $\boldsymbol{x_l}$ is removed from the RVM model. These steps are shown in lines 8–12 of Algorithm 2.

B. Online RVM Training Using Streaming Data

Existing techniques for RVM training assume that all data are available *a priori*. In this section, we develop an online RVM training algorithm (Algorithm 2) that updates the set of relevance vectors $\Lambda_k = \left\{\mathbf{x}_i^{(k)}, y_i^{(k)}, \xi_i^{(k)}\right)\right\}_i$ incrementally using streaming data. Suppose that Λ_k has been obtained based on prior data $\mathcal{D}_0, \ldots, \mathcal{D}_k$. At time $k+1$, a new training set $\mathcal{D}_{k+1}$ is received. The training set generation depends on the application. We construct $\mathcal{D}_{k+1}$ using a lidar scan $\mathbf{z}_{k+1}$ of an unknown environment as detailed in Section VIII-A. Relevance vectors are added or removed from Λ_k to correctly classify the latest changes, e.g., new or disappearing obstacles, in training set $\mathcal{D}_{k+1}$ without affecting the accuracy of the classification on the prior data and maintaining the sparsity of the model.

Algorithm 2 presents our online probit RVM training approach. The algorithm starts with the existing set of relevance vectors Λ_k and adds new relevance vectors based on the samples in $\mathcal{D}_{k+1}$ using the sequential training approach in Section V-A. Instead of using the feature matrix $\boldsymbol{\Phi}$ (line 4) associated with all prior relevance vectors, we use a feature matrix approximation based on a local set Λ_{local} of K nearest relevance vectors (line 2). Section VII provides a discussion on the computational improvements and assumptions of the score function approximation resulting from using Λ_{local} instead of Λ_k. For test time

Algorithm 2: Online Probit RVM Training.

Input: Relevance vectors $\Lambda_k = \left\{\left(\boldsymbol{x}_i^{(k)}, y_i^{(k)}, \xi_i^{(k)}\right)\right\}$; training set $\mathcal{D}_{k+1} = \{(\mathbf{x}_l, y_l)\}_l$; number of nearest relevance vectors to use K (optional)

Output: Relevance vectors $\Lambda_{k+1} = \left\{\left(\boldsymbol{x}_i^{(k+1)}, y_i^{(k+1)}, \xi_i^{(k+1)}\right)\right\}$; weight posterior mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$

1: Initialize $\Lambda_{k+1} = \Lambda_k$.
2: **if** K is defined **then** $\Lambda_{\text{local}} = K$ nearest relevance vectors from Λ_k
3: **else** $\Lambda_{\text{local}} = \Lambda_k$.
4: $\boldsymbol{\Phi} = \text{FEATUREMATRIX}(\Lambda_{\text{local}}, \mathcal{D}_{k+1})$
5: $\xi_l = \infty$ for each $(\mathbf{x}_l, y_l)$ in $\mathcal{D}_{k+1}$
6: $\boldsymbol{\Sigma}, \boldsymbol{\mu} = \text{LAPLACEAPPROXIMATION}(\Lambda_{\text{local}}, \mathcal{D}_{k+1})$.
7: **while** not converged and max number iterations not reached **do**
8: Pick a candidate $(\mathbf{x}_m, y_m)$ from $\mathcal{D}_{k+1}$.
9: Calculate $S_m, Q_m, s_m, q_m, \theta_m$.
10: If $\theta_m > 0$ and $\xi_m = \infty$, add $(\mathbf{x}_m, y_m, \xi_m)$ to Λ_{local}.
11: If $\theta_m \leq 0$ and $\xi_m < \infty$, remove $(\mathbf{x}_m, y_m, \xi_m)$ from Λ_{local}.
12: If $\theta_m > 0$ and $\xi_m < \infty$, re-estimate $\xi_m = \frac{s_m^2}{q_m^2 - s_m}$ in Λ_{local}.
13: $\boldsymbol{\Sigma}, \boldsymbol{\mu} = \text{LAPLACEAPPROXIMATION}(\Lambda_{\text{local}}, \mathcal{D}_{k+1})$.
14: $\Lambda_{k+1} = \Lambda_{k+1} \cup \Lambda_{\text{local}}$.
15: $\boldsymbol{\Sigma}, \boldsymbol{\mu} = \text{GLOBALPOSTERIORAPPROXIMATION}(\Lambda_{k+1})$
16: **return** Λ_{k+1}, $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$
17:
18: **function** FEATUREMATRIX($\Lambda, \mathcal{D}$)
19: Calculate $\boldsymbol{\Phi}_{i,j} = k(\mathbf{x}_i, \mathbf{x}_j)$ for all $\mathbf{x}_j \in \Lambda$ and all $\mathbf{x}_i \in \mathcal{D}$
20: return $\boldsymbol{\Phi}$
21: **function** LAPLACEAPPROXIMATION($\Lambda, \mathcal{D}$)
22: Calculate $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$ for relevance vectors Λ using $\mathcal{D}$ ((16) and (17)).
23: **return** $\boldsymbol{\Sigma}$, $\boldsymbol{\mu}$.
24: **function** GLOBALPOSTERIORAPPROXIMATION(Λ)
25: **return** LAPLACEAPPROXIMATION(Λ, Λ).

classification, we compute the mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the Laplace approximation to the weight posterior according to (16) and (17). Laplace approximation requires all data $\mathcal{D} = \cup_{i=1}^{k+1} \mathcal{D}_i$, used for training up to time $k+1$, but only the local dataset $\mathcal{D}_{k+1}$ is available. Interestingly, the set Λ_{k+1} of relevance vectors itself globally and sparsely represents all the data used for training and, therefore, can be used for Laplace approximation (line 15). If additional computation for Laplace approximation is not feasible, one might directly store the weight mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ (line 15) over time. The memory requirements for either case are discussed in Section VII.

Fig. 2(a) depicts a ground robot equipped with a lidar scanner whose goal is to build an occupancy map of the environment. Fig. 2(c) plots the training set $\mathcal{D}_1$ generated from the first lidar scan $\mathbf{z}_1$, assuming the current set of relevance vectors Λ_k is empty. Fig. 2(d) shows the trained RVM model as a sparse set of relevance vectors, serving as a sparse probabilistic occupancy map of the environment, incrementally updated via the streaming lidar scans. To illustrate map updates based on the latest depth measurements, we consider a second scan where part of the obstacles in the first scan disappears and a new obstacle appears in the second scan [Fig. 2(g)]. The RVM model, trained with the first laser scan, is updated with the second scan using our online RVM training algorithm (Algorithm 2), reflecting the obstacle changes in the environment as plotted in Fig. 2(h). A map representation is useful for autonomous navigation (Problem 1) only if it allows checking potential robot trajectories $\mathbf{s}(t)$ for collisions. We propose classification methods for points, line segments, and general curves next.

VI. RVM Classification of Points, Lines, and Curves

This section discusses classification using the predictive distribution in (13) and makes a connection with our preliminary results in [28]. Commonly, machine learning models are only able to classify point queries but applications, such as robot trajectory planning, may require classification of general curves. This can be done by successively checking a dense set of points, sampled along the curve. However, we show that under certain assumptions on the kernel function and the decision threshold, line and general curve classification based on the RVM decision boundary can be performed directly and efficiently, based on the closed-form of the predictive distribution in (13), without the need to sample.

A. RVM Classification of Points

Consider a set Λ of M relevance vectors with prior weight precision $\boldsymbol{\xi}$ and mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$ of the approximate weight posterior $p(\boldsymbol{w}|\mathbf{y}, \mathbf{X}, \boldsymbol{\mu}, \boldsymbol{\Sigma})$. To classify a query point $\mathbf{x}$ using the RVM model, we place a threshold $\bar{e} = \sigma(e)$ on the probability $\mathbb{P}(y = 1|\mathbf{x}, \boldsymbol{\xi})$ (Definition 1). Fig. 2(d) illustrates the decision boundary defined by Definition 1 with $\bar{e} = \sigma(e) = 0.494$, i.e., $e = -0.01$.

Definition 1: Let $\bar{e} \in [0, 1]$ and $e := \sigma^{-1}(\bar{e})$. A point $\mathbf{x}$ is classified as "–1" (free) if

$$\mathbb{P}(y = 1|\mathbf{x}, \boldsymbol{\xi}) = \sigma\left(\frac{\Phi_{\mathbf{x}}^\top \boldsymbol{\mu} + b}{\sqrt{1 + \Phi_{\mathbf{x}}^\top \boldsymbol{\Sigma} \Phi_{\mathbf{x}}}}\right) \leq \bar{e} \tag{20}$$

or, equivalently, if

$$G_1(\mathbf{x}) := \boldsymbol{\Phi}_{\mathbf{x}}^\top \boldsymbol{\mu} + b - e\sqrt{1 + \boldsymbol{\Phi}_{\mathbf{x}}^\top \boldsymbol{\Sigma} \boldsymbol{\Phi}_{\mathbf{x}}} \leq 0. \tag{21}$$

The condition in (21) can be verified for a given point, but it is challenging to obtain an explicit expression in terms of $\mathbf{x}$. If, instead of a point $\mathbf{x}$, we consider a time-parameterized curve $\mathbf{p}(t)$, then (21) becomes a nonlinear programming feasibility problem in t. To avoid nonlinear programming, we develop a series of upper bounds for $G_1(\mathbf{x})$ that make the condition for classifying a point as free (i.e., $y = -1$) more conservative but with a simpler dependence on $\mathbf{x}$.

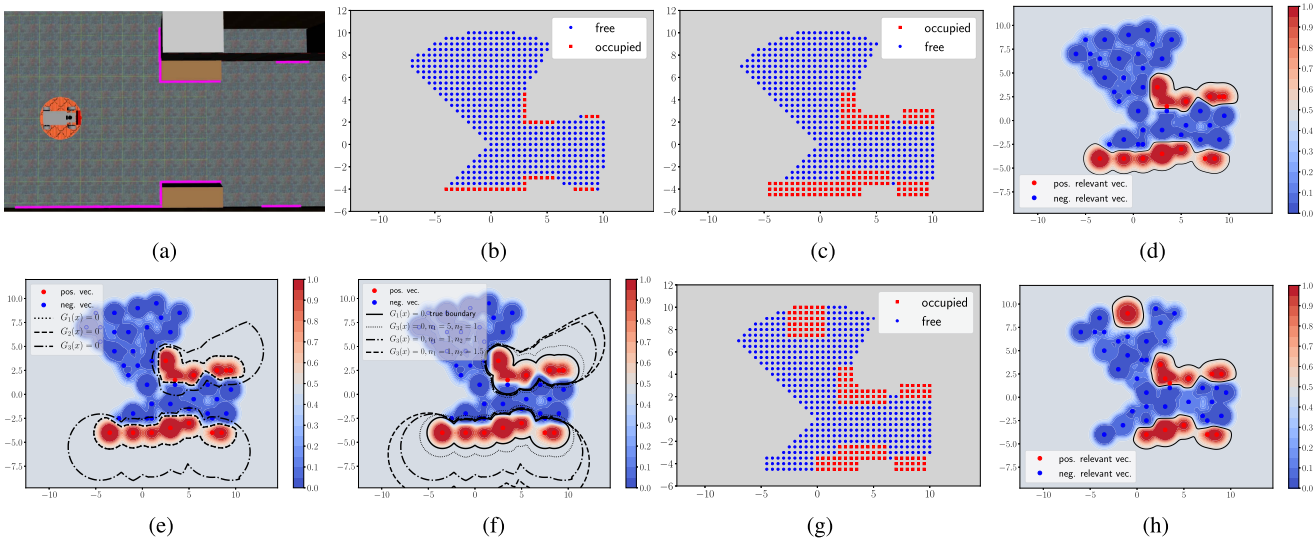


Fig. 2. Example of our mapping method. (a) Robot observing the environment via a laser scan (magenta). (b) Work-space samples generated from the laser scan $\mathbf{z}_k$. (c) Configuration-space samples used as a training set at time k. (d) Exact decision boundary with bias $b = -0.05$ and classification threshold $e = -0.01$, $\bar{e} = 0.494$. (e) Inflated boundaries (c.f., Section VI) generated by $G_1(\mathbf{x}), G_2(\mathbf{x}), G_3(\mathbf{x})$ with $n_1 = n_2 = 1$. (f) Inflated boundary $G_3(\mathbf{x}) = 0$ with various n_1, n_2. Example of map updates with changes in the environment. (g) Configuration-space samples from a second laser scan where part of the obstacles in the first scan disappears and a new obstacle appears. (h) Our updated RVM model reflecting the latest changes.

Proposition 3: For a nonnegative kernel function $k_m(\mathbf{x}) := k(\mathbf{x}, \mathbf{x}_m)$, a point $\mathbf{x}$ is classified as "–1" if

$$G_2(\mathbf{x}) := \sum_{m=1}^{M} \left(\mu_m - e\mathbb{1}_{\{e\leq 0\}}\sqrt{\lambda_{\max}}\right) k_m(\mathbf{x}) + b - e \leq 0 \tag{22}$$

where $\lambda_{\max} \geq 0$ is the largest eigenvalue of the covariance $\boldsymbol{\Sigma}$, μ_m is the mth element of the mean $\boldsymbol{\mu}$, and $\mathbb{1}_{\{e\leq 0\}}$ is an indicator function which equals 1 if $e \leq 0$ and 0, otherwise.

Proof: Please refer to Appendix A. □

The relaxed condition in (22) adjusts the weights of the relevance vectors by an amount of $\delta\mu = -e\mathbb{1}_{\{e\leq 0\}}\lambda_{\max} \geq 0$. Intuitively, this increases the effect of the positive relevance vectors, leading to a more conservative condition than Definition 1. Proposition 3 also allows us to use only the largest eigenvalue $\lambda_{\max}$ of $\boldsymbol{\Sigma}$ for point classification, which is easier to obtain and store than the whole covariance matrix $\boldsymbol{\Sigma}$. Methods for computing $\lambda_{\max}$ are discussed in Section VII-A.

To simplify the notation, let $\nu_m := \mu_m - e\mathbb{1}_{\{e\leq 0\}}\lambda_{\max}$ be the corrected relevance vector weights and split Λ into M^+ positive relevance vectors $\Lambda^+ = \{(\mathbf{x}_m^+, \nu_m^+)\}$ and M^- negative relevance vectors $\Lambda^- = \{(\mathbf{x}_m^-, \nu_m^-)\}$, where $\nu_m^+ = \nu_m$ if $\nu_m > 0$ and $\nu_m^- = -\nu_m$ if $\nu_m < 0$. Now, (22) can be rewritten as

$$G_2(\mathbf{x}) = \sum_{i=1}^{M^+} \nu_i^+ k\left(\mathbf{x}, \mathbf{x}_i^+\right) - \sum_{j=1}^{M^-} \nu_j^- k(\mathbf{x}, \mathbf{x}_j^-) + b - e \leq 0. \tag{23}$$

Hence, Proposition 3 allows us to make an important connection between sparse kernel classification with a "hard" decision threshold (Section IV) and its Bayesian counterpart (Section V-B). Specifically, after the relevance vector weight correction, (22) is equivalent to the kernel perceptron score in (2) except for the bias term $b - e$.

1) Role of the Bias Term: One of the motivations for developing a Bayesian map representation is to distinguish between observed and unobserved regions in the environment. Intuitively, as a query point $\mathbf{x}$ is chosen further away from "observed" regions, where training data has been obtained, its correlation with existing relevance vectors, measured by $k(\mathbf{x}, \mathbf{x}_m)$, decreases. To capture and exploit this property, we assume that the kernel has a common RBF structure that depends only on a quadratic norm $\|\boldsymbol{\Gamma}(\mathbf{x} - \mathbf{x}_m)\|$.

Assumption 1: Let $k(\mathbf{x}, \mathbf{x}_m) := \eta \exp(-\|\boldsymbol{\Gamma}(\mathbf{x} - \mathbf{x}_m)\|^2)$ with parameters $\eta > 0$ and $\boldsymbol{\Gamma} \in \mathbb{R}^{d\times d}$.

In our application, the kernel parameters η and $\boldsymbol{\Gamma}$ may be optimized offline via automatic relevance determination [58] using training data from known occupancy maps. Under this assumption, the feature vector $\boldsymbol{\Phi}_{\mathbf{x}}$ tends to 0 as $\mathbf{x}$ goes toward unobserved regions and the occupancy probability $\mathbb{P}(y = 1|\mathbf{x}, \boldsymbol{\xi})$ tends to $\sigma(b)$ in (20). Therefore, the value of $\sigma(b)$ represents the occupancy probability of points in the unknown regions. In other words, $\sigma(b)$ specifies how much we trust that unknown regions are occupied and should be a constant. For this reason, the bias b is fixed in our online RVM training algorithm. If we are optimistic about the unknown regions, the parameter b can be set to a large negative number, i.e., $\sigma(b) \approx 0$, and the decision boundary shrinks toward the occupied regions. If we want the robot to be cautious about the unknown regions, the parameter b can be set to a large positive number, i.e., $\sigma(b) \approx 1$, and the decision boundary expands toward the unknown regions. A common assumption in motion planning [59] is to treat unknown regions as free in order to allow trajectory planning to goals in the unknown space. In the context of this article, this means that the occupancy probability of points in unknown regions $\sigma(b)$ should be lower than or equal to the decision threshold $\bar{e} = \sigma(e)$ in Definition 1.

Assumption 2: Assume that $e \geq b$, and, hence, $\bar{e} \geq \sigma(b)$.

2) RVM Classification With $e = b$: A natural choice for the occupancy probability of unknown regions, $\sigma(b)$, is to set it exactly equal to the decision threshold between free and occupied space, i.e., $e = b$. While the SKM in our preliminary work [28] does not have either the bias parameter b and the threshold e in its model, (23) has the same form as (2) when $b = e$ and is exactly equivalent when $b = e = 0$. Therefore, all results in Section IV for classification of points, lines, and curves can be reused.

Corollary 2: If the bias b in (7) is used as the decision threshold for classification in Definition 1, i.e., $e = b$, then, by Proposition 3 and (23), a point $\mathbf{x}$ is classified as "–1" if

$$\sum_{i=1}^{M^+} \nu_i^+ k(\mathbf{x}, \mathbf{x}_i^+) - \sum_{j=1}^{M^-} \nu_j^- k(\mathbf{x}, \mathbf{x}_j^-) \leq 0. \tag{24}$$

Hence, Proposition 2 and Corollary 1 hold for line and curve classification using an RVM model.

3) RVM Classification With $e \geq b$: For a general decision threshold, $e \geq b$, and a kernel function $k_m(\mathbf{x})$ satisfying Assumption 1, we develop an explicit condition for classifying a point $\mathbf{x}$ as free.

Proposition 4: For integers $n_1, n_2 \geq 1$, define $\rho(a, b) := (n_1 + n_2)(\frac{a}{n_1})^{\frac{n_1}{n_1+n_2}}(\frac{b}{n_2})^{\frac{n_2}{n_1+n_2}}$. A point $\mathbf{x}$ is classified as "–1" if

$$G_3(\mathbf{x}) := \left(\sum_{i=1}^{M^+} \nu_i^+\right) k(\mathbf{x}, \mathbf{x}_*^+) - \rho(e - b, \nu_j^- k(\mathbf{x}, \mathbf{x}_j^-)) \leq 0 \tag{25}$$

where $\mathbf{x}_*^+$ is the closest positive relevance vector to $\mathbf{x}$ and $\mathbf{x}_j^-$ is any negative relevance vector.

Proof: Please refer to Appendix B. □

Fig. 2(e) illustrates the exact RVM decision boundary from (21), $G_1(\mathbf{x}) = 0$, and the boundaries $G_2(\mathbf{x}) = 0$ and $G_3(\mathbf{x}) = 0$ resulting from the upper bounds in Propositions 3 and 4. Note that the boundary generated by $G_2(\mathbf{x})$ is very close to the true boundary from $G_1(\mathbf{x})$, empirically showing that the bound $G_2(\mathbf{x})$ is tight. The upper bound $G_3(\mathbf{x})$ provides a conservative "inflated boundary," whose accuracy can be controlled via the integers n_1, n_2 in Proposition 4. Note that $G_3(\mathbf{x})$ is inaccurate mainly in the unknown regions because the arithmetic mean-geometric mean inequality used in Proposition 4's proof (Appendix B) effectively replaces the kernel function $k(\mathbf{x}, \mathbf{x}_j^-)$ by a slower decaying one $k(\mathbf{x}, \mathbf{x}_j^-)^{\frac{n_2}{n_1+n_2}}$. This suits the intuition that unknown regions should be categorized as free more cautiously. Fig. 2(f) shows that increasing the ratio n_2/n_1 makes the "inflated boundary" closer to the true decision boundary in the unknown regions but slightly looser in the well-observed regions and vice versa. Next, based on Proposition 4, we develop conditions for classification of lines and curves when $e \geq b$ without the need for sampling.

B. RVM Classification of Lines

Consider a linear trajectory described by a ray $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}, t \geq 0$ such that $\mathbf{p}_0$ is obstacle-free according to Proposition 4, i.e., $G_3(\mathbf{p}_0) \leq 0$, and $\mathbf{v}$ is a constant. To check if $\mathbf{p}(t)$ collides with the inflated boundary $G_3(\mathbf{x}) = 0$, we find a time t_u such that any point $\mathbf{p}(t)$ is classified free for $t \in [0, t_u)$.

Algorithm 3: RVM Line Classification.

Input: Line segment $(\mathbf{p}_A, \mathbf{p}_B)$; relevance vectors $\Lambda = \{(\mathbf{x}_i, y_i, \boldsymbol{\xi}_i)\}$; weight posterior mean $\boldsymbol{\mu}$ and max covariance eigenvalue $\lambda_{\max}$

1: $\mathbf{v}_A = \mathbf{p}_B - \mathbf{p}_A$, $\mathbf{v}_B = \mathbf{p}_A - \mathbf{p}_B$
2: Calculate t_{uA} and t_{uB} using (26) or (27).
3: **if** $t_{uA} + t_{uB} > 1$ **then return** True (Free)
4: **else return** False (Colliding)

Proposition 5: Consider a ray $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}$, $t \geq 0$. Let $\mathbf{x}_i^+$ and $\mathbf{x}_j^-$ be arbitrary positive and negative relevance vectors. Then, any point $\mathbf{p}(t)$ with $t \in [0, t_u) \subseteq [0, t_u^*)$ is free for

$$t_u := \min_{i=1,\ldots,M^+} \tau(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) \tag{26}$$

$$t_u^* := \min_{i=1,\ldots,M^+} \max_{j=1,\ldots,M^-} \tau(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) \tag{27}$$

where $\tau(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-)$

$$= \begin{cases} +\infty, & \text{if } V(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has less than two roots} \\ +\infty, & \text{if } V(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has two roots} t_1 < t_2 \leq 0 \\ t_1 & \text{if } V(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has two roots} 0 \leq t_1 < t_2 \\ 0 & \text{if } V(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has two roots} t_1 \leq 0 \leq t_2 \end{cases}$$

and $V(t, \mathbf{x}_i^+, \mathbf{x}_j^-) = at^2 + b(\mathbf{x}_i^+, \mathbf{x}_j^-)t + c(\mathbf{x}_i^+, \mathbf{x}_j^-)$ with

$$a := -n_1 \|\boldsymbol{\Gamma}\mathbf{v}\|^2,$$

$$b(\mathbf{x}_i^+, \mathbf{x}_j^-) := -2\mathbf{v}^\top \boldsymbol{\Gamma}^\top \boldsymbol{\Gamma}(n_1\mathbf{p}_0 - (n_1 + n_2)\mathbf{x}_i^+ + n_2\mathbf{x}_j^-),$$

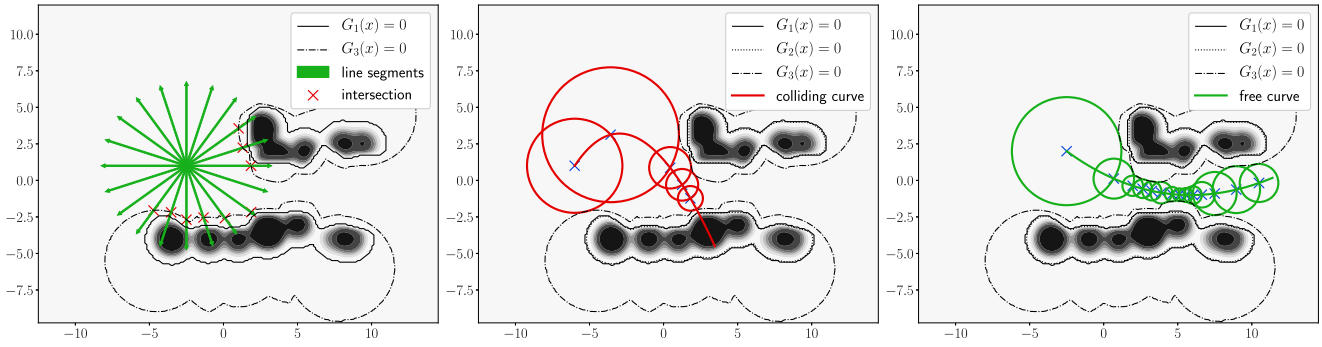
$$c(\mathbf{x}_i^+, \mathbf{x}_j^-) := -(n_1+n_2)\|\boldsymbol{\Gamma}(\mathbf{p}_0 - \mathbf{x}^+{}_i)\|^2 + n_2\|\boldsymbol{\Gamma}(\mathbf{p}_0 - \mathbf{x}_j^-)\|^2 - (n_1 + n_2)\log \frac{\rho(e - b, \nu_j^-)}{\eta^{\frac{n_1}{n_1+n_2}} \sum_{i=1}^{M^+} \nu_i^+}.$$

Proof: Please refer to Appendix C. □

For a line segment $(\mathbf{p}_A, \mathbf{p}_B)$, all points on the segment can be expressed as $\mathbf{p}(t_A) = \mathbf{p}_A + t_A\mathbf{v}_A$, $\mathbf{v}_A = \mathbf{p}_B - \mathbf{p}_A$, $0 \leq t_A \leq 1$, or $\mathbf{p}(t_B) = \mathbf{p}_B + t_B\mathbf{v}_B, \mathbf{v}_B = \mathbf{p}_A - \mathbf{p}_B, 0 \leq t_B \leq 1$. Using the upper bound t_{uA} on t_A provided by (26) or (27), we find the free region on $(\mathbf{p}_A, \mathbf{p}_B)$ starting from $\mathbf{p}_A$. Likewise, we calculate t_{uB} which specifies the free region from $\mathbf{p}_B$. If $t_{uA} + t_{uB} > 1$, the entire line segment is free; otherwise, the segment is considered colliding. The proposed approach is summarized in Algorithm 3 and illustrated in Fig. 3(a) for the trained RVM model in Fig. 2.

C. RVM Classification of Curves

Instead of a constant velocity $\mathbf{v}$ representing the direction of motion, we can define a general curve $\mathbf{p}(t)$ by considering a time-varying term $\mathbf{v}(t)$. We extend the collision checking conditions in Proposition 5 by finding an ellipsoid $\mathcal{E}(\mathbf{p}_0, r) := \{\mathbf{x} : \|\boldsymbol{\Gamma}(\mathbf{x} - \mathbf{p}_0)\| \leq \mathbf{r}\}$ around $\mathbf{p}_0$ whose interior is free of obstacles, where Γ is the kernel parameter defined in Assumption



(a) (b) (c)

Fig. 3. Illustration of our classification algorithms for the trained RVM model in Fig. 2 with $b=-0.05, e=-0.01$, and $n_1=n_2=1$ (a) Checking line segments. (b) Checking a colliding curve. (c) Checking a free curve.

1. This specific form of the ellipsoid leads a closed-conditions as shown in Proposition 6.

Proposition 6: Let $\mathbf{p}_0$ be such that $G_3(\mathbf{p}_0)<0$ and let $\mathbf{x}_i^+$ and $\mathbf{x}_j^-$ be arbitrary positive and negative support vectors. Then, every point inside the ellipsoids $\mathcal{E}(\mathbf{p}_0, r_u) \subseteq \mathcal{E}(\mathbf{p}_0, r_u^*)$ is free for

$$r_u = \min_{i=1,\dots,M^+} r(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) \tag{28}$$

$$r_u^* = \min_{i=1,\dots,M^+} \max_{j=1,\dots,M^-} r(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-) \tag{29}$$

where $r(\mathbf{p}_0, \mathbf{x}_i^+, \mathbf{x}_j^-)$

$$= \begin{cases} +\infty, & \text{if } \bar{V}(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has less than two roots} \\ +\infty, & \text{if } \bar{V}(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has two roots } t_1 < t_2 \le 0 \\ t_1 & \text{if } \bar{V}(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has two roots } 0 \le t_1 < t_2 \\ 0 & \text{if } \bar{V}(t, \mathbf{x}_i^+, \mathbf{x}_j^-) \text{ has two roots } t_1 \le 0 \le t_2 \end{cases}$$

and $\bar{V}(t, \mathbf{x}_i^+, \mathbf{x}_j^-) = \bar{a}t^2 + \bar{b}(\mathbf{x}_i^+, \mathbf{x}_j^-)t + \bar{c}(\mathbf{x}_i^+, \mathbf{x}_j^-)$ with

$$\bar{a} := -n_1,$$

$$\bar{b}(\mathbf{x}_i^+, \mathbf{x}_j^-) := 2\|\mathbf{\Gamma}(n_1\mathbf{p}_0 - (n_1+n_2)\mathbf{x}_i^+ + n_2\mathbf{x}_j^-)\|,$$

$$\bar{c}(\mathbf{x}_i^+, \mathbf{x}_j^-) := c(\mathbf{x}_i^+, \mathbf{x}_j^-).$$

Proof: Please refer to Appendix D. □

Consider a general time-parameterized curve $\mathbf{p}(t), t \in [0, t_f]$ from $\mathbf{p}_0 := \mathbf{p}(0)$ to $\mathbf{p}_f := \mathbf{p}(t_f)$. Proposition 6 shows that all points inside the ellipsoid $\mathcal{E}(\mathbf{p}_0, r)$ are free for $r = r_u \le r_u^*$. If we can find the smallest positive t_1 such that

$$\|\mathbf{\Gamma}(\mathbf{p}(t_1) - \mathbf{p}_0)\| = r \tag{30}$$

then all points on the curve $\mathbf{p}(t)$ for $t \in [0, t_1)$ are free. This is equivalent to finding the smallest positive solution of (30). We perform curve classification by iteratively covering the curve by free ellipsoids. If the value of r is smaller than a threshold ε, the curve is considered colliding. Otherwise, it is considered free. The classification process for curves is shown in Algorithm 4 and illustrated in Fig. 3(b) and (c) for the trained RVM model in Fig. 2 for a colliding curve and a free curve, respectively.

In Propositions 5 and 6, calculating t_u and r_u takes $O(M)$ time, while the computational complexity of calculating t_u^* and r_u^* are $O(M^2)$, where $M = M^+ + M^-$. If the line segments or curves are limited to the neighborhood of the starting point $\mathbf{p}_0$, the bound t_u and r_u can reasonably approximate t_u^* and r_u^*, respectively, if $\mathbf{x}_j^-$ is chosen as the negative support vector, closest to $\mathbf{p}_0$. Calculation of t_u and r_u in Propositions 5 and 6 is efficient in the sense that it has the same complexity as classifying a point; yet, it can classify an entire line segment for $t \in [0, t_u)$ and an entire ellipsoid $\mathcal{E}(\mathbf{p}_0, r_u)$, respectively.

Algorithm 4: RVM Curve Classification.

Input: Curve $\mathbf{p}(t), t \in [0, t_f]$; threshold ε; relevance vectors $\Lambda = \{(\mathbf{x}_i, y_i, \boldsymbol{\xi}_i)\}$; weight posterior mean $\boldsymbol{\mu}$ and max covariance eigenvalue $\lambda_{\max}$

while True **do**

Calculate r_k using (28) or (29).

if $r_k < \varepsilon$ **then return** False (Colliding)

Solve $\|\mathbf{\Gamma}(\mathbf{p}(\mathbf{t}) - \mathbf{p}(\mathbf{t_k}))\| = \mathbf{r_k}$ for $t_{k+1} \ge t_k$

if $t_{k+1} \ge t_f$ **then return** True(Free)

VII. Computational and Storage Improvements

A. Computational Improvements

In the context of autonomous navigation, as a robot explores new regions of its environment, the number of relevance vectors required to represent the obstacle boundaries increases. Since the score function in (7) depends on all relevance vectors, the training time (Algorithm 2) and the classification time (Definition 1 for points, Algorithm 3 for lines, and Algorithm 4 for curves) increase as well. We propose an approximation to the score function $F(\mathbf{x})$ for the radial basis kernel in Assumption 1. Since $k(\mathbf{x}, \mathbf{x}_m)$ approaches zero rapidly as the distance between $\mathbf{x}$ and $\mathbf{x}_m$ increases, the value of $F(\mathbf{x})$ is not affected significantly by relevance vectors far from $\mathbf{x}$. We use R^*-tree data structures constructed from the relevance vectors Λ^+, Λ^- to allow efficient lookup of the nearest K^+ and K^- positive and negative relevance vectors. Approximating the score function $F(\mathbf{x})$ using the nearest K^+ and K^- relevance vectors improves its computational complexity from $O(M)$ to $O(\log M)$. To classify a point $\mathbf{x}$, the M-dimensional feature vector $\mathbf{\Phi}_x$ is

approximated by a K-dimensional one using the K relevance vectors closest to $\mathbf{x}$. Classification of a line segment or a curve in Propositions 5 and 6 can be approximated by using the K^+ and K^- nearest positive and negative relevance vectors. The computational complexities of (26)–(29) improve from $O(M)$ and $O(M^2)$ to $O(\log M)$.

The line and curve classification algorithms depend on Proposition 3 which requires the largest eigenvalue $\lambda_{\max}$ of the weight posterior covariance matrix $\mathbf{\Sigma}$. Obtaining $\lambda_{\max}$ from $\mathbf{\Sigma}$ can be expensive as the number of relevance vectors grows. Under Assumption 1, the entries in the feature matrix $\mathbf{\Phi}$ for relevance vectors that are far from each other go to 0 quickly and can be set to zero, e.g., using a cutoff threshold for the kernel values or only keeping the kernel values for the K nearest relevance vectors. This leads to sparse matrices $\mathbf{\Phi}$ and $\mathbf{\Sigma}^{-1}$ in (16) whose smallest eigenvalue $1/\lambda_{\max}$ can be approximated efficiently (e.g., [60]).

B. Storage Improvements

Algorithm 2 returns a set of M relevance vectors $\boldsymbol{x_m}$ with labels y_m and weight prior precision ξ_m. This set represents the RVM model parameters and its memory requirements are linear in M. However, the predictive distribution in (13) needs to be obtained via Laplace approximation [see (16) and (17)] when the RVM model is used for classification. If additional computation for Laplace approximation is not feasible during test time, the weight posterior mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$ may be stored also, but $\mathbf{\Sigma}$ requires $O(M^2)$ storage. Fortunately, the approximate decision boundary $G_2(\mathbf{x}) = 0$ in Proposition 3 used for point, line, and curve classification only requires the largest eigenvalue $\lambda_{\max}$ of $\mathbf{\Sigma}$. Hence, only the value of $\lambda_{\max}$ needs to be stored in addition to the relevance vectors $\boldsymbol{x}_m$, labels y_m, and weight mean μ_m. In this case, line 15 in Algorithm 2 returns the weight posterior mean $\boldsymbol{\mu}$ and $\lambda_{\max}$ instead of $\boldsymbol{\mu}$ and $\mathbf{\Sigma}$.

VIII. Application to Occupancy Mapping and Autonomous Navigation

A. Online Mapping

We consider a robot placed in an unknown environment at time t_k as illustrated in Fig. 1. It is equipped with a lidar scanner measuring distances to nearby obstacles. Samples generated from the lidar range scan $\mathbf{z}_k$ are shown in Fig. 2(b). Since the robot body is bounded by a sphere of radius r, each laser ray end point in configuration space becomes a ball-shaped obstacle, while the robot body becomes a point. To generate local training data, the occupied and free C-space areas observed by the lidar are sampled (e.g., on a regular grid). As shown in Fig. 2(c), this generates a set $\bar{\mathcal{D}}_k$ of points with label "1" (occupied) in the ball-shaped occupied areas and with label "-1" (free) between the robot position and each laser end point. To accelerate training, only the difference between two consecutive local datasets $\mathcal{D}_k = \bar{\mathcal{D}}_k \backslash \bar{\mathcal{D}}_{k-1}$ is used in our online RVM training algorithm (Algorithm 2). Storing the sets of relevance vectors Λ_k over time requires significantly less memory than storing the training data $\cup_k \mathcal{D}_k$. The occupancy of a query point $\mathbf{x}$ can be estimated from the relevance vectors by evaluating the function $G_1(\mathbf{x})$ in (21). Specifically, $\hat{m}_k(\mathbf{x}) = -1$ if $G_1(\mathbf{x}) \leq 0$ and $\hat{m}_k(\mathbf{x}) = 1$ if $G_1(\mathbf{x}) > 0$. Fig. 2(d) illustrates the boundaries generated by Algorithm 2.

Algorithm 5: GetSuccessors and ObstacleFree Subroutines in A^*[61] and RRT^* [62], Respectively.

Input: Current position $\mathbf{p}_k$; set of relevance vectors $\Lambda = \{(\mathbf{x}_i, y_i, \boldsymbol{\xi}_i)\}$ with posterior weight mean $\boldsymbol{\mu}$ and covariance $\mathbf{\Sigma}$; set $\mathcal{N}(\mathbf{p}_k)$ of potential reference trajectories $\mathbf{p}(t - t_k)$ with $\mathbf{p}(t_k) = \mathbf{p}_k$.
Output: Set of collision-free trajectories S.
$S \leftarrow \emptyset$;.
for $\mathbf{p}'$ in $\mathcal{N}(\mathbf{p}_k)$ **do**
  if $\mathbf{p}'$ is a line **and** CheckLine($\mathbf{p}_k, \mathbf{p}', \Lambda$) **then** ▷Algorithm 3
    $S \leftarrow S \cup \{\mathbf{p}'\}$
  if $\mathbf{p}'$ is a curve **and** CheckCurve($\mathbf{p}_k, \mathbf{p}', \Lambda$) **then** ▷Algorithm 4
    $S \leftarrow S \cup \{\mathbf{p}'\}$
return S

B. Autonomous Navigation

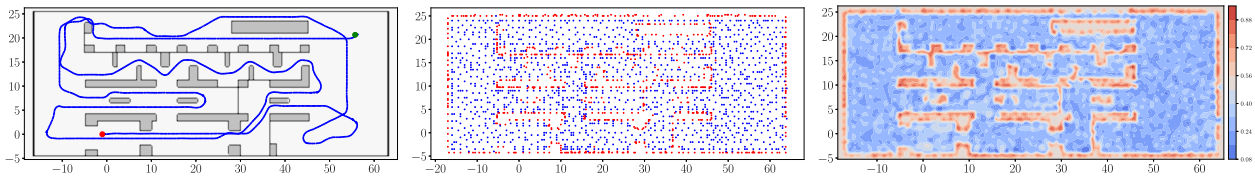
Finally, we present a complete online mapping and navigation approach that solves Problem 1. Given the SBKM $\hat{m}_k$ proposed in Section VIII-A, a motion planning algorithm such as A^* [61] or RRT^* [62] may be used with our collision-checking algorithms to generate a path that solves the autonomous navigation problem (Algorithm 5). The robot follows the path for some time and updates the map estimate $\hat{m}_{k+1}$ with new observations. Using the updated map, the robot replans the path and follows the new path instead. This process is repeated until the goal is reached or a time limit is exceeded (Algorithm 6).

The use of the "inflated boundary" $G_3(\mathbf{x}) = 0$ from Proposition 4 for collision checking might block the motion planning task if it is not tight enough in certain regions of the environment (e.g., unobserved regions as discussed in Section VI-A). For such regions, a different ratio of n_2/n_1 can be used in Proposition 4 to achieve a tighter bound $G_3(\mathbf{x})$. Increasing the decision threshold $\bar{e}$ (Definition 1) can also improve the accuracy of $G_3(\mathbf{x})$ if a tradeoff with robot safety is allowed. Another resort is to use SB collision checking, selecting points along the curve $\mathbf{p}(t)$ and using Definition 1.

We consider robots with two different motion models. In simulation, we use a first-order fully actuated robot, $\dot{\mathbf{p}} = \mathbf{v}$, where the state $\mathbf{s}$ is the robot position $\mathbf{p} \in [0, 1]^3$, with piecewise-constant velocity $\mathbf{v}(t) \equiv \mathbf{v}_k \in \mathcal{V}$ for $t \in [t_k,\ t_{k+1})$, leading to piecewise-linear trajectories

$$\mathbf{p}(t) = \mathbf{p}_k + (t - t_k)\mathbf{v}_k, \qquad t \in [t_k,\ t_{k+1}) \tag{31}$$

where $\mathbf{p}_k := \mathbf{p}(t_k)$. In this case, the classification algorithm for line segments (Algorithm 3) is used during motion planning.



(a) (b) (c)

Fig. 4. Sparse map representation (with $\eta = 1, \Gamma = \sqrt{\gamma} I,$ and $\gamma = 3.0$) built from local streaming laser scans along the robot trajectory. (a) Ground truth map and robot trajectory. (b) Set of 2141 relevance vectors. (c) The final SBKM map.

Algorithm 6: Autonomous Mapping and Navigation With a Sparse Bayesian Kernel-Based Map.

Input: Initial state $\mathbf{s}_0 \in \bar{\mathcal{S}}$; goal region $\mathcal{G}$; prior relevance vectors Λ_0.

for $k = 0, 1, \ldots$ **do**
  if $\mathbf{s}_k \in \mathcal{G}$ **then break**
  $\mathbf{z}_k \leftarrow$ new range sensor observation
  $\mathcal{D}_\mathbf{k} \leftarrow$ Training data generation $(\mathbf{z}_k, \mathbf{s}_k)$ ▷ Section VIII-A
  $\Lambda_{k+1} \leftarrow$ Online RVM training $(\Lambda_k, \mathcal{D}_k)$ ▷ Algorithm 2
  Path planning $(\Lambda_{k+1}, \mathbf{s}_k, \mathcal{G})$ ▷ Algorithm 5
  Move to the first state $\mathbf{s}_{k+1}$ along the path

In the real experiments, we consider a ground wheeled Ackermann-drive robot with dynamics model

$$\dot{\mathbf{p}} = v \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \qquad \dot{\theta} = \frac{v}{\ell} \tan \phi \tag{32}$$

where the state $\mathbf{s}$ consists of the position $\mathbf{p} \in \mathbb{R}^2$ and orientation $\theta \in \mathbb{R}$, the control input $\mathbf{u}$ consists of the linear velocity $v \in \mathbb{R}$ and the steering angle $\phi \in \mathbb{R}$, and ℓ is the distance between the front and back wheels. The nonlinear car dynamics can be transformed into a second-order fully actuated system $\ddot{\mathbf{p}} = \mathbf{a}$ via feedback linearization [54], [55]. Using piecewise-constant acceleration $\mathbf{a}(t) \equiv \mathbf{a}_k \in \mathcal{A}$ for $t \in [t_k,\ t_{k+1})$ leads to piecewise-polynomial trajectories

$$\mathbf{p}(t) = \mathbf{p}_k + (t - t_k) v_k \begin{bmatrix} \cos(\theta_k) \\ \sin(\theta_k) \end{bmatrix} + \frac{(t - t_k)^2}{2} \mathbf{a}_k \tag{33}$$

where $\mathbf{p}_k := \mathbf{p}(t_k)$, $\theta_k := \theta(t_k)$, and $v_k := v(t_k)$. In our experiments, the input set $\mathcal{A}$ is finite and the classification algorithm for curves (Algorithm 4) is used to get successor nodes in an A^* motion planning algorithm.

IX. Experimental Results

This section presents an evaluation of our autonomous mapping and navigation method using a fully actuated robot (31) in a simulated environment (Section IX-A), the Intel Research Lab dataset [63] (Section IX-B), and a car-like robot (Fig. 1) with Ackermann-drive dynamics (32) in real experiments (Section IX-D). We examined the obstacle boundary with respect to the bias parameter b and the threshold e in Section IX-C and carried out an active mapping experiment using our map uncertainty in Section IX-E. We used a RBF kernel with parameters $\eta = 1$ and $\Gamma = \sqrt{\gamma}\mathbf{I}$. The bias parameter b is set to -0.05 in Sections IX-A, IX-D, and IX-E, and 0.0 in Section IX-B. Timing results are reported from an Intel i9 3.1-GHz CPU with 32-GB RAM.

A. Comparison With Binary Map Representation

In this section, we compared the accuracy, the recall, and storage requirements of our SBKM with those of the non-Bayesian SKM from our preliminary work [28], the popular occupancy mapping algorithm OctoMap [19], and the SBHM [35]. Since SKM only provides binary maps, binary maps are used to calculate accuracy and recall. As the ground-truth map [Fig. 4(a)] represents the work space instead of C-space, a point robot ($r = 0$) was used for an accurate comparison. Lidar scans were simulated along the robot trajectory shown in Fig. 4(a) and used to build our SBKM, the non-Bayesian SKM [28], OctoMap, and SBHM. An R^*-tree approximation of the score $F(\mathbf{x})$ was used with $K^+ + K^- = 200$ nearest support vectors around the robot location $\mathbf{p}_k$ for map updating and with $K^+ + K^- = 10$ nearest support vectors for collision checking. OctoMap's resolution was set to 0.25 m to match that of the grid used to sample our training data from. As SBHM depends on a grid of hinge points to generate feature vectors, we chose the grid's resolution so that the number of hinge points is similar to our SBKM's number of relevance vectors for a fair comparison.

Table I compares the accuracy, the recall, and the storage requirements of our SBKM maps, our SKM maps, SBHM maps [35], and OctoMap's binary and probabilistic maps. The SBKM map and its sparse set of relevance vectors are shown in Fig. 4. To calculate map accuracy, we used different thresholds $\bar{e}$ to generate binary versions of our map and compare with the ground truth. The ground truth map was sampled on a grid with the same resolution of 0.25 m and the accuracy was calculated as the number of correct predictions divided by the total number of samples. The interior (gray regions in the ground-truth map) of the obstacles was considered occupied for our map—since it was surrounded by positive relevance vectors—but were considered free in SKM and OctoMap maps.

Table I shows that SBKM (with threshold $\bar{e} = 0.5$), SKM, OctoMap's binary map, and SBHM (with threshold $\bar{e} = 0.5$) led to a similar accuracy of $\sim 99\%$ ($\gamma = 2.0$ and 3.0) and $\sim 97\%$ ($\gamma = 1.0$) and a similar recall of $\sim 99\%$ ($\gamma = 2.0$and3.0) and $\sim 98\%$ ($\gamma = 1.0$). SKM has $\sim 0.5\%$ higher accuracy than SBKM since the support vectors lie around the obstacle boundary,

TABLE I
COMPARISON AMONG OUR SPARSE BAYESIAN KERNEL-BASED MAP (SBKM), OUR SPARSE KERNEL-BASED MAP (SKM) [28], OCTOMAP (OM) [19], AND SEQUENTIAL BAYESIAN HILBERT MAP (SBHM) [35]

Methods	Kernel param. γ	Threshold $\bar{e}$	Accuracy	Recall	Vectors/nodes	Storage
SBKM	1.0	0.5	97.8%	97.9%	1115	9 kB†
SBKM	2.0	0.5	99.0%	99.3%	1642	13 kB†
SBKM	3.0	0.45	99.2%	99.7%	2141	17 kB†
SBKM	3.0	0.5	99.3%	99.4%	2141	17 kB†
SBKM	3.0	0.55	99.5%	98.7%	2141	17 kB†
SKM	3.0	–	99.9%	99.0%	2463	20 kB
SKM	2.0	–	99.8%	98.3%	2613	21 kB
SKM	1.0	–	99.8%	98.5%	3064	25 kB
OM	–	0.5	99.9%	99.7%	12 432 nonleafs and 34 756 leafs	25 kB (binary)/236 kB (full)
SBHM	1.0	0.5	97.0%	98.0%	1156	9 kB‡
SBHM	2.0	0.5	99.0%	99.4%	1676	13 kB‡
SBHM	3.0	0.45	98.6%	99.0%	2205	17 kB‡
SBHM	3.0	0.5	99.0%	98.6%	2205	17 kB‡
SBHM	3.0	0.55	99.5%	98.2%	2205	17 kB‡

†The storage requirements for SBKM are calculated for two storing approaches mentioned in Section VII: 1) with Laplace approximation at test time, i.e., storing the relevance vectors' location with their label and precision $\boldsymbol{\xi}$; 2) without Laplace approximation at test time, i.e., storing the relevance vectors' location with their label and weight's mean μ, and the largest eigenvalue of the covariance matrix $\lambda_{\max}$. As we only need an extra float to store $\lambda_{\max}$, both approaches offer similar storage requirements. ‡ The storage requirements for SBHM are calculated as if the hinge points are stored using our storing approach. An RBF kernel with $\eta = 1, \Gamma = \sqrt{\gamma} I$ was used for SBHM map and our SBKM and SKM maps.

leading to sharper decision boundaries. As the decision threshold $\bar{e}$ decreases, the accuracy decreases because more free cells are classified as "occupied," and the recall increases because more occupied cells are classified as "occupied." When the parameter γ decreases, the support of the kernel expanded, leading to fewer relevance vectors, i.e., less storage but lower accuracy and recalls. This illustrates the tradeoff between storage gains and accuracy when the details of the obstacles' boundaries can be reduced via a lower value of γ to achieve higher compression rate.

We compared the storage requirements for our SBKM and SKM representations and OctoMap. OctoMap's binary map required a compressed octree with 12 432 nonleaf nodes with 2 b per node, leading to a storage requirement of ~ 25 kB. Its fully probabilistic map required to store 47 188 leaf and nonleaf nodes with 5 b per node, leading to a storage requirement of ~ 236 kB. As the space consumption depends on the computer architecture and how the relevance vector information is compressed, we provide only a rough estimate of storage requirements for our maps. For the SKM map, each support vector required 8 b, including an integer for the support vector's location on the underlying grid and a float for its weight. As a result, ~ 20 kB were needed to store the 2463 resulting support vectors for $\gamma = 3.0$. As discussed in Section VII-B, the SBKM map could be stored in two ways: 1) the relevance vectors' location, their label, and their weight prior precision if Laplace approximation was allowed at test time; 2) the relevance vectors' location, their label, their weight mean, and the largest eigenvalue $\lambda_{\max}$ of the covariance matrix Σ if Laplace approximation was not allowed at test time and our collision checking methods were used. The former stored an integer representing a relevance vector's location on the underlying grid and a float representing its weight prior's precision and its label (using the float sign). This required 8 b on a 32-b architecture per relevance vector. Our SBKM map with $\Gamma = \sqrt{3.0} I$ contained 2141 relevance vectors, leading to storage requirements of ~ 17 kB. The latter also needed 17 kB to store the relevance vectors' location, their weight's mean and label. Besides, an extra float (4 b) is needed to store $\lambda_{\max}$ leading to a similar total storage requirement of 17 kB. These requirements were 32% and 15% better than those of OctoMap and our (non-Bayesian) SKM, respectively. To achieve a sparse Bayesian map representation, more computation is needed, leading to slower map update for SBKM compared to SKM and OctoMap. Since SBHM and SBKM are both Bayesian online mapping methods and share similar settings, we compared their map update time in Section IX-B. As γ decreases, the number of relevance vectors of SBKM decreases, while the number of support vectors of SKM increases. This is because the relevance vectors spread out in the environment while the support vectors are placed on both sides of the obstacle boundaries. Therefore, as γ decreases, a relevance vector represents more space leading to fewer relevance vectors in the SBKM models and more support vectors, maintaining a sharp decision boundary, for the SKM models.

We compared the average collision checking time over one million random line segments $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t$ and one million random second-order polynomial curves $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t + \mathbf{a}t^2$ for $t \in [0, t_f]$ using our complete method [Algorithm 3 with (27) for line segments, Algorithm 4 with (29) for curves, $K^+ + K^- = 10$ for score approximation, and $e = -0.01$ for occupancy threshold] and SB methods with different sampling resolutions using the ground truth map. Fig. 5(a) and (b) shows that the SB collision checking time increased as the time length t_f increased or the sampling resolution decreased. Meanwhile, our method's time was stable at ~ 3 μs for checking line segments and at ~ 11 μs for checking polynomial curves suggesting our collision checking algorithms' suitability for real-time applications.

B. Comparison With Probabilistic Map Representations

In this section, we compared our SBKM approach with other probabilistic occupancy mapping techniques: OctoMap [19],

TABLE II
COMPARISON AMONG OUR SPARSE BAYESIAN KERNEL-BASED MAP (SBKM), SEQUENTIAL BAYESIAN HILBERT MAP (SBHM) [35] (WITH FULL AND DIAGONAL COVARIANCE MATRICES), LOCALIZED AUTOMATIC RELEVANCE DETERMINATION HILBERT MAP (LARD-HM) [36], AND OCTOMAP [19] ON THE INTEL RESEARCH LAB DATASET [63]

Test data	Methods	SBHM	SBHM	LARD-HM	LARD-HM	LARD-HM	SBKM	SBKM	OM
–	γ	6.71	6.71	–	–	–	6.71	6.71	–
–	Σ	full	diag.	–	–	–	Full	$\lambda_{\max}$ only	–
–	Feature dim.	5600	5600	3640	5460	7280	3492	3492	–
Uniformly sampled	AUC	0.98	0.98	0.90	0.96	0.97	0.96	0.95	0.95
Uniformly sampled	NLL	0.24	0.24	0.41	0.30	0.24	0.36	0.52	0.27
Near boundary + rooms	AUC	0.82	0.77	0.57	0.55	0.56	0.62	0.61	0.75
Near boundary + rooms	NLL	0.54	0.60	0.83	0.83	0.78	0.73	0.84	0.67
–	Map update time/scan	11.8 s	0.03 s	0.03 s	0.03 s	0.03 s	0.76 s	0.43 s	0.01 s
Line	Collision checking time	380 ms	38 ms	5.6 ms	6.2 ms	6.2 ms	7 μs	7 μs	21 μs
Curve	Collision checking time	384 ms	37 ms	6.3 ms	6.7 ms	6.5 ms	18 μs	18 μs	23 μs

An RBF kernel with $\eta = 1, \Gamma = \sqrt{\gamma} I$ was used for SBHM and our SBKM maps. The metrics are the area under the receiver operating characteristic curve (AUC) and the negative log-likelihood loss (NLL). Collision checking time was measured for line segments $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t$ and second-order polynomial curves $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}T + \mathbf{a}t^2$ for $t \in [0, t_f]$ with $t_f = 2$s. Our proposed collision checking described in Section VI was used with our SBKM map while sampling-based collision checking with sampling resolution $\Delta = 0.005$ was used with the other maps.

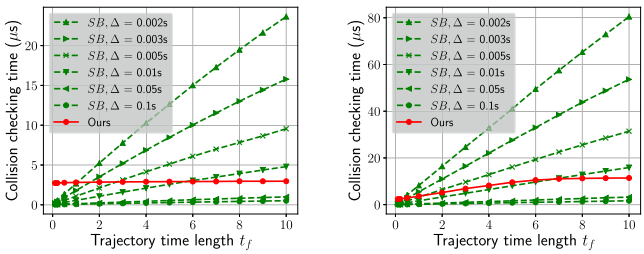
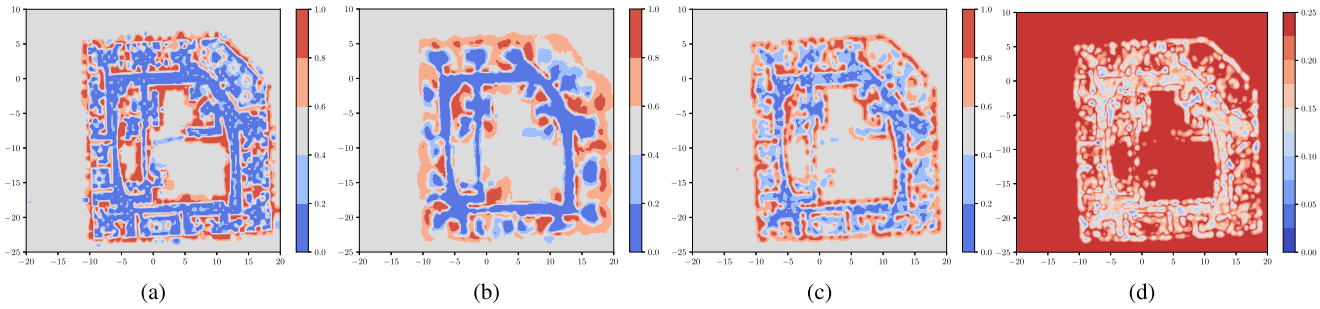


Fig. 5. Collision checking time comparison between our methods and sampling-based ones with different sampling interval Δ for (a) line segments $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t$ and (b) second-order polynomial curves $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t + \mathbf{a}t^2$ for $t \in [0, t_f]$ with various values of t_f.

LARD-HM [36], and SBHM [35]. The Intel Research Lab dataset [63] was used with the four methods to build the map of the environment in an online manner. Both SBKM and SBHM are kernel-based Bayesian probabilistic mapping methods from streaming local observations. While SBHM achieves sparseness by calculating feature vectors based on a sparse set of hinged points, e.g., on a coarse grid, our SBKM method learns the hinged points, i.e., the relevance vectors, from data. We tried our best to match the parameters for a fair comparison, e.g., using the same kernel parameter $\Gamma = \sqrt{\gamma}\mathbf{I}$ with $\gamma = 6.71$ as provided by SBHM code [35]. Our online training data (Section V-B) were generated from a grid with resolution 0.2 m. Meanwhile, the LARD-HM method determines the hinged points by clustering the training data points using *k*-means algorithms and calculates their kernel parameters by automatic relevance determination. To build a LARD-HM map from streaming depth measurements, we used the publicly available LARD-HM code in [36] to incrementally add the centroids of the clusters and their kernel parameters to the maps. OctoMap code [19] was used with its default parameters. Since we only considered probabilistic occupancy maps in this section, the metrics for comparison were the area under the receiver operating characteristic curve (AUC) and the negative log-likelihood loss (NLL) of a point $\mathbf{x}$, defined as $\text{NLL}(y|\mathbf{x}, \boldsymbol{\xi}) = -\log p(y|\mathbf{x}, \boldsymbol{\xi})$, where $y \in \{-1, 1\}$ is the true label and $p(y|\mathbf{x}, \boldsymbol{\xi})$ is the predictive distribution in (13). The AUC score and NLL loss were calculated over two test sets: one sampled uniformly from the whole dataset to capture overall reconstruction accuracy and one sampled near the room area boundaries to capture detail reconstruction accuracy.

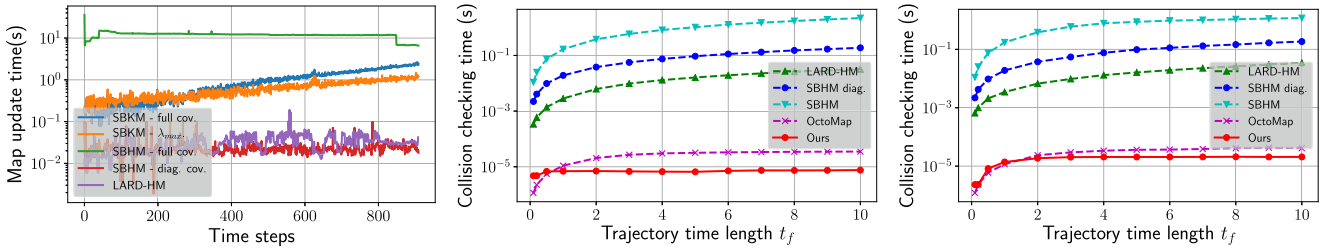
Table II presents the metrics for the four mapping methods. SBHM used a fixed grid of 5600 hinged points with resolution 0.5 m for feature vector calculation. Meanwhile, our approach incrementally learned a sparse set of 3492 relevance vectors from the training dataset, not requiring a set of fixed hinged points. Similarly, LARD-HM determined the hinged points from the data by incrementally adding the centroids of k_f free and k_o occupied clusters and their kernel parameters from each laser scan, where $k_f = k_o = 2, 3, 4$ leading to 3640 (similar to the SBKM's feature dimension), 5460 (similar to the SBHM's feature dimension), and 7280 hinged points in Table II, respectively. Fig. 6(a)–(c) shows the final maps from the SBHM, LARD-HM approaches, and our SBKM method, respectively. Fig. 6(d) plots our SBKM map's variance, distinguishing between known (low variance) and unknown (high variance) regions.

Our final map's AUC score and NLL loss were slightly worse than those of SBHM with full covariance matrix while maintaining $\sim 35\%$ fewer points to represent the environment and having faster map updates with less than 1 s per scan, on average, as shown in Table II and Fig. 7(a). Our training algorithm incrementally built the set of relevance vectors and only updated the weights of the local vectors due to the use of K nearest relevance vectors in Algorithm 2. Consequently, it did not have a fixed global set of points to optimize over as done by SBHM, leading to suboptimality in tradeoff for sparseness. Note that both our map and the SBHM map estimated the mean $\boldsymbol{\mu}$ and the full covariance matrix $\boldsymbol{\Sigma}$ of the weights' posterior for test time. If our collision checking algorithms are used for planning, only the largest eigenvalue $\lambda_{\max}$ of $\boldsymbol{\Sigma}$ is needed and can be calculated efficiently using the sparsified inverse covariance matrix as shown in Section VII-B. In this case, Table II shows that our map update time was reduced by half to about ~ 0.43 s per scan [Table II and Fig. 7(a)] while offering similar AUC score to that of our SBKM map with full covariance matrix. The higher NLL loss was due to the upper bound used in Proposition 3 for point classification instead of the true occupancy probability.



(a) (b) (c) (d)

Fig. 6. Comparison among our sparse Bayesian kernel-based map (SBKM), localized automatic relevance determination Hilbert map (LARD-HM) [36] with 7280 clusters, and sequential Bayesian Hilbert map (SBHM) [35] with $\eta = 1, \Gamma = \sqrt{6.71}I$ built online using lidar scans from the Intel Research Lab dataset [63] (a) SBHM map. (b) LARD-HM map. (c) Our SBKM map. (d) Our SBKM maps variance.



(a) (b) (c)

Fig. 7. (a) Map update times and collision checking time comparison for (b) line segments $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t$ and (c) second-order polynomial curves $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t + \mathbf{a}t^2$ for $t \in [0, t_f]$ with various values of t_f. Our proposed collision checking described in Section VI is used with our SBKM map while sampling-based collision checking with sampling resolution $\Delta = 0.005$ is used with the other maps.

A variant of SBHM that only uses diagonal covariance matrix updated the map 25 times faster than our SBKM method with full covariance matrix. While SBKM time can be improved further using a diagonal covariance matrix, we leave this investigation for future work.

Our map's AUC score and NLL loss, calculated using test data sampled uniformly from the dataset, were better than those of LARD-HM with 3640 features (similar to our map's feature size 3492), comparable to those of LARD-HM with 5460 features, and worse than those of LARD-HM with 7280 features, shown in Fig. 6(b). Our map update time was 14 times (with $\lambda_{\max}$ only) and 25 times (with full covariance matrix) slower. The main reason for the slower speed is that our method uses Bayesian updates with a full posterior covariance matrix while LARD-HM is not Bayesian. The speed of our method can be accelerated by using a diagonal-only covariance matrix formulation or learning different kernel parameters from data using the key ideas in LARD-HM. With similar number of features, LARD-HM tended to preserve less details of the obstacle boundary compared to our method in the room areas of the Intel Lab dataset, as shown in Fig. 6(b) and in Table II by the better AUC score and NLL loss of our map when the test set was sampled in the rooms near the boundary. This reflects the difference that our SBKM approach adds relevance vectors, which can be very close to the boundary, while LARD-HM maps choose cluster centroids, which intuitively are farther from the boundary.

OctoMap's AUC score, calculated from test points sampled uniformly from the complete dataset, was lower than ours as the default maximum (0.97) and minimum (0.12) values of the occupancy probability were used. Meanwhile, OctoMap's performance was better than ours in preserving boundary details in the room areas. A feature dimension is not reported for OctoMap since it is not a kernel-based method.

An advantage of our SBKM map representation is that it can utilize the collision-checking techniques for lines and curves developed in Section VI. We checked 1000 random line segments $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t$ and 1000 s-order polynomial curves $\mathbf{p}(t) = \mathbf{p}_0 + \mathbf{v}t + \mathbf{a}t^2$ for $t \in [0, t_f]$ for collisions using our method [Algorithm 3 with (27) for line segments, Algorithm 4 with (29) for curves, $K^+ + K^- = 20$ for score approximation, and $e = -0.01$ for occupancy threshold] and SB methods with sampling resolution $\Delta = 0.005$ s using the OctoMap, SBHM, and LARD-HM maps. Fig. 7(b) and (c) shows that the collision checking time for SB methods increased as t_f increased. Meanwhile, our method's time was stable at ~ 7 μs for checking line segments and at ~ 20 μs for checking polynomial curves, as shown in Table II.

C. Decision Boundary's Conservativeness

As the decision boundary between free and occupied spaces affects the area for robot navigation, we examined its conservativeness with respect to the bias b, threshold e, and measurement noise variance l^2 parameters. Fig. 8(a) plots the occupied area in our map, i.e., the area with occupancy probability greater than the threshold $e = 0$, built from the Intel Lab dataset [63] for

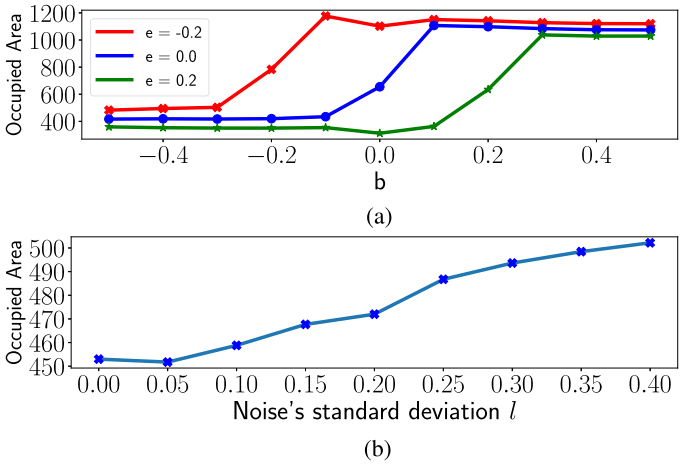


Fig. 8. Occupied area versus the bias b and the threshold e (a) and versus noise level (b). (a) The occupied area (with different values of e) versus the bias b. (b) The occupied area (with $b = -0.01$, $e = 0.0$) versus the noise's standard deviation.

different values of the bias b and threshold e. As expected, the occupied area increases, i.e., smaller navigable area, if e decreases and/or b increases. Note that our previous SKM model [28] does not offer similar tuning for the decision boundary because it does not provide parameters such as b and e.

The decision boundary's conservativeness should also be affected by the measurement noise since a robot should proceed carefully around the obstacle boundary if the depth measurements are noisy. To illustrate the conservativeness of the boundary generated by our approach against measurement noise, we added Gaussian noise with zero mean and variance l^2 to the laser endpoints in the dataset and trained our model. The occupied area versus the noise's standard deviation l is plotted in Fig. 8(b). As the noise level l increases, the occupied area increases, i.e., the navigable area decreases, making the robot more cautious around obstacles in the environment.

D. Real Experiments

Real experiments were carried out on a $1/10$th scale Racecar robot equipped with a Hokuyo UST-10LX Lidar and Nvidia TX2 computer. The robot body was modeled by a ball of radius $r = 0.25$ m. The online training data (Section V-B) were generated from a grid with resolution 0.25 m. We used an RBF kernel parameter $\Gamma = \sqrt{\gamma}\mathbf{I}$ with $\gamma = 3.0$ and an R^*-tree approximation of the score $F(\mathbf{x})$ with $K^+ + K^- = 20$ nearest support vectors around the robot location $\mathbf{p}_k$ for map updating. For motion planning, second-order polynomial motion primitives were generated with time discretization of $\tau = 1$ s as described in Section VIII-B. The motion cost was defined as $c(\mathbf{s}, \mathbf{a}) := (\|\mathbf{a}\|^2 + 2)\tau$ to encourage both smooth and fast motion [53]. Algorithm 4 with (29), $\varepsilon = 0.1$, score approximation with $K^+ = K^- = 2$, and threshold $e = -0.01$ were used for collision checking in Algorithm 5. The trajectory generated by an A^* motion planner was tracked using a closed-loop controller [64]. The robot navigated in an unknown hallway to two destinations consequently chosen by a human operator. Fig. 9(a) shows the learned relevance vectors representing the environment. Fig. 9(b) shows the probabilistic map recovered from the relevance vectors together with the robot trajectory and the two chosen destinations.

The time taken by Algorithm 2 to update the relevance vectors from one lidar scan is shown in Fig. 9(c). Map updates implemented in Python took 0.4 s on average. It took a longer time ($\sim$1 s) to update the map when the robot observed new large parts of the environment, e.g., at the beginning and toward the end of our experiment. To evaluate collision checking time, the A^* planning time was normalized by the number of motion primitives being checked to account for differences in planning to nearby and far goals. The planning time per motion primitive [Fig. 9(d)] was $\sim$15 μs on average and $\sim$30 μs at most, suggesting our method's suitability for real-time applications.

In both the simulations and the real experiment, the free area contains multiple blobs of points with low occupancy probability, caused by the sparse map representation and the fixed kernel parameters of SBKM. To improve this, kernel parameter learning, e.g., using variational inference [65], clustering, and automatic relevance determination [36], [37], or kernel parameter dictionaries, e.g., pretrained in small and simple environments [66], can be used to to adaptively update the kernel parameters at different locations based on the depth measurements. This is a promising avenue for future research.

E. Active Mapping

Our SBKM representation enables uncertainty quantification which, besides collision checking, can be used for active mapping. This ability is not offered by non-Bayesian mapping methods, such as SKM. In this section, we demonstrate active mapping of an unknown simulated environment using SBKM. Our approach estimates the map uncertainty in different regions and chooses the region with the highest uncertainty as the goal region. Specifically, we maintain a frontier, defined as a list of L candidate poses $\mathcal{P}_l$, $l = 1, 2, \ldots, L$. For each pose $\mathcal{P}_l$, we calculate the map uncertainty $H(\mathcal{S}_l)$ of the field of view $\mathcal{S}_l$ of the depth sensor. The map uncertainty of a region $\mathcal{S}$ is measured as the average marginal entropy over the region

$$H(\mathcal{S}) = \frac{1}{|\mathcal{S}|}\int_{\mathcal{S}} h(\mathbf{x})d\mathbf{x} \tag{34}$$

where $h(\mathbf{x})$ is the marginal entropy of a point $\mathbf{x}$ in the region, calculated using the predictive distribution in Definition 1 as

$$\begin{aligned} h(\mathbf{x}) = & -P(y=1|\mathbf{x},\boldsymbol{\xi})\log_2 P(y=1|\mathbf{x},\boldsymbol{\xi}) \\ & -P(y=0|\mathbf{x},\boldsymbol{\xi})\log_2 P(y=0|\mathbf{x}_i,\boldsymbol{\xi}) \end{aligned} \tag{35}$$

$y \in \{-1, 1\}$ is the predictive label of the point $\mathbf{x}$, and $|\mathcal{S}|$ denotes the area of the region $\mathcal{S}$. We choose the region S_{l^*} with the largest average marginal entropy to explore

$$l^* = \underset{l=1,2,\ldots,L}{\operatorname{argmax}}\ H(\mathcal{S}_l). \tag{36}$$

In our active mapping experiment, the candidate poses $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_L$ in the frontier were sampled from the laser endpoints up to the current time t with four different yaw

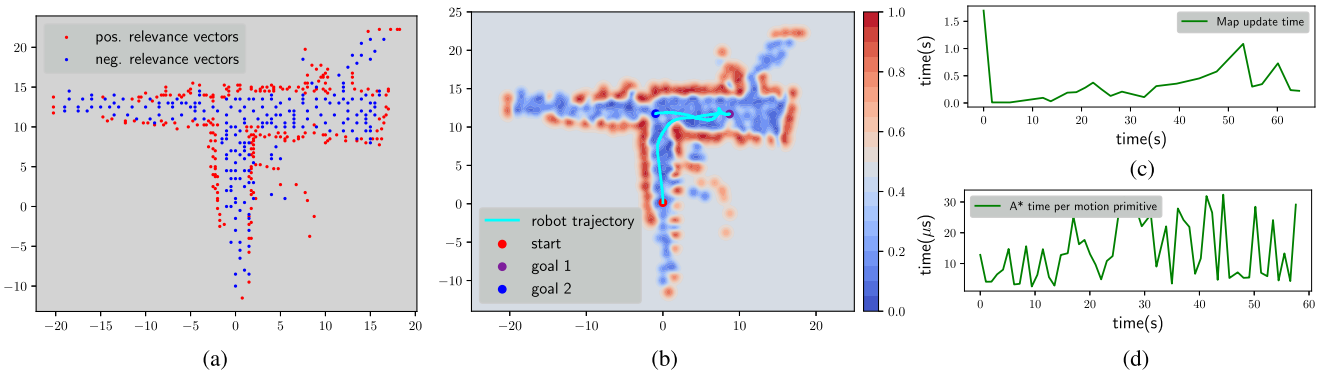


Fig. 9. Real experiment with an autonomous Racecar robot navigating in an unknown hallway environment. (a) Final 453 relevance vectors. (b) Final probabilistic map. (c) Map update time. (d) Planning time per motion primitive. Please refer to our website https://thaipduong.github.io/sbkm for the experiment video.

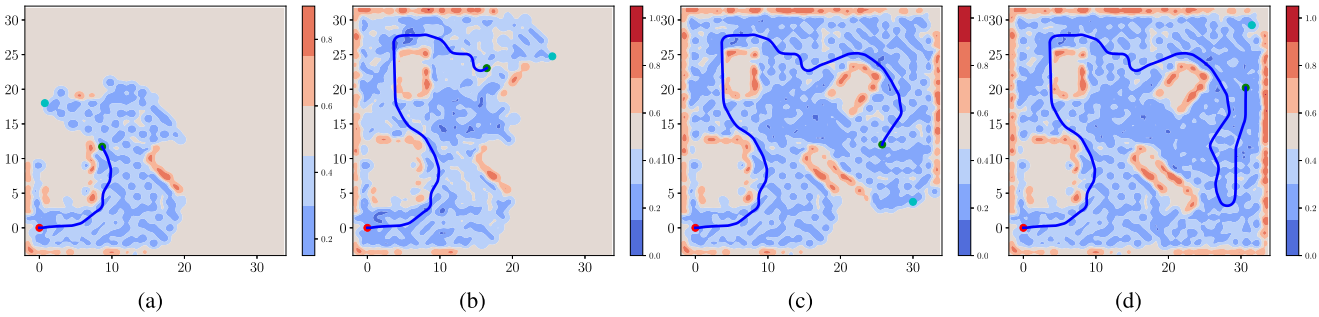


Fig. 10. Active mapping at times $t = 76, 209, 301,$ and 414 s. The red, green, and cyan dots are the initial and current robot positions, and the goal actively picked to reduce the map entropy, respectively. The robot trajectory is in blue. (a) $t = 76$ s. (b) $t = 209$ s. (c) $t = 301$ s. (d) $t = 414$ s.

angles: $0, \frac{\pi}{4}, \frac{\pi}{2}, \frac{3\pi}{4}$. Since the laser scans could not see through obstacles, we gained little information of the environment by placing the robot near the occupied regions. Therefore, only the endpoints with maximum lidar range, i.e., the laser ray did not hit an obstacle, and at least 2 m away from the positive relevance vectors were considered. A hypothetical lidar field of view $\mathcal{S}_l$ [similar to Fig. 2(b) without the obstacles], simulating a Hokuyo UST-10LX lidar, was placed at each candidate pose $\mathcal{P}_l$. We sampled $N = 100$ points $\mathbf{x}_i, i = 1, \ldots, N$, from $\mathcal{S}_l$ and computed the marginal entropy $H(\mathbf{x}_i)$. The average marginal entropy (34) of the region $\mathcal{S}_l$ was approximated as

$$H(\mathcal{S}_l) \approx \frac{1}{N} \sum_{i=1}^{N} h(\mathbf{x}_i). \tag{37}$$

The robot picked the goal region $\mathcal{S}_{l^*}$ with the highest map uncertainty from the set $\{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_L\}$ every 0.5 s and planned a trajectory to reach the goal using our collision checking methods and the same A^* planner used in Section IX-D. Fig. 10 shows the SBKM map, the robot trajectory, and the candidate pose associated with the goal region at different times as the robot successfully explored and actively built the map. The average marginal entropy of the map (Fig. 11), estimated using (37) with $N = 20\,736$ points sampled on a regular grid of resolution 0.25 m, shows that our active mapping approach reduced the map uncertainty over time.

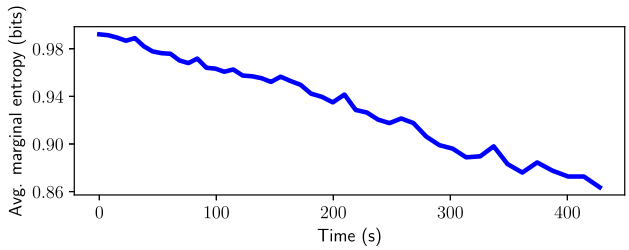


Fig. 11. Average marginal entropy of a point in the map over time.

X. Conclusion

This article proposed a sparse Bayesian kernel-based mapping method for efficient online generation of large occupancy maps, supporting autonomous robot navigation in unknown environments. Our map representation, as a sparse set of relevance vectors learned from streaming range observations of the environment, is efficient to store. It supports efficient and complete collision checking for general curves modeling potential robot trajectories. Our experiments demonstrated the potential of this model at generating compressed, yet accurate, probabilistic environment models. Our results offered a promising venue for quantifying safety and uncertainty and enabling real-time long-term autonomous navigation in unpredictable environments. Future work will further explore active exploration and map uncertainty reduction, kernel parameter learning, as well as

simultaneous localization and mapping using the proposed map representations.

APPENDIX A
PROOF OF PROPOSITION 3

Proof: A point $\mathbf{x}$ is considered free if

$$\boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{\mu} + b < e\sqrt{1+\boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{\Sigma}\boldsymbol{\Phi}_{\mathbf{x}}} \tag{38}$$

where $\boldsymbol{\Phi}_{\mathbf{x}}$ is the feature vector $\boldsymbol{\Phi}_{\mathbf{x}} = [k_1(\mathbf{x}), k_2(\mathbf{x}), \ldots, k_M(\mathbf{x})]^{\top}$. We use the following lower bound and upper bound on $\boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{\Sigma}\boldsymbol{\Phi}_{\mathbf{x}}$: $0 \le \boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{\Sigma}\boldsymbol{\Phi}_{\mathbf{x}} \le \lambda_{\max}\sum_{m=1}^{M}(k_m(\mathbf{x}))^2$ where $\lambda_{\max} \ge 0$ is the largest eigenvalue of the covariance matrix $\boldsymbol{\Sigma}$. Since $k_m(\mathbf{x}) > 0$ for all m, we have

$$1 \le \sqrt{1+\boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{\Sigma}\boldsymbol{\Phi}_{\mathbf{x}}} \le 1 + \sqrt{\lambda_{\max}}\sum_{m=1}^{M}(k_m(\mathbf{x})). \tag{39}$$

Therefore, the point $\mathbf{x}$ is still free if

$$\boldsymbol{\Phi}_{\mathbf{x}}^{\top}\boldsymbol{\mu} + b \le e(1 + \mathbb{1}_{\{e\le 0\}}\sqrt{\lambda_{\max}}\sum_{m=1}^{M}(k_m(\mathbf{x}))) \tag{40}$$

or $\sum_{m=1}^{M}(\boldsymbol{\mu}_m - e\mathbb{1}_{\{e\le 0\}}\sqrt{\lambda_{\max}})k_m(\mathbf{x}) + b - e \le 0$. □

APPNDIX B
PROOF OF PROPOSITION 4

Proof: A point $\mathbf{x}$ is free if (23) holds. Let $\mathbf{x}_*^+$ be the closest positive relevance vector to $\mathbf{x}$ and $\mathbf{x}_j^-$ be any negative relevance vector. We have

$$\sum_{i=1}^{M^+}\nu_i^+ k(\mathbf{x},\mathbf{x}_i^+) - \sum_{j=1}^{M^-}\nu_j^- k(\mathbf{x},\mathbf{x}_j^-) + b - e \le$$

$$\le \left(\sum_{i=1}^{M^+}\nu_i^+\right) k(\mathbf{x},\mathbf{x}_*^+) - \nu_j^- k(\mathbf{x},\mathbf{x}_j^-) + b - e.$$

Under Assumptions 1 and 2, both terms $\nu_j^- k_j(\mathbf{x})$ and $e-b$ are nonnegative. By the arithmetic mean- geometric mean inequality, we have

$$\begin{aligned}\nu_j^- k(\mathbf{x},\mathbf{x}_j^-) + e - b &= n_2\frac{\nu_j^- k(\mathbf{x},\mathbf{x}_j^-)}{n_2} + n_1\frac{e-b}{n_1}\\ &\ge (n_1+n_2)\left(\frac{\nu_j^- k(\mathbf{x},\mathbf{x}_j^-)}{n_2}\right)^{\frac{n_2}{n_1+n_2}}\left(\frac{e-b}{n_1}\right)^{\frac{n_1}{n_1+n_2}}\\ &= \rho(e-b,\nu_j^- k(\mathbf{x},\mathbf{x}_j^-)).\end{aligned}$$

Therefore, a point $\mathbf{x}$ is free if

$$\left(\sum_{i=1}^{M^+}\nu_i^+\right) k(\mathbf{x},\mathbf{x}_*^+) - \rho(e-b,\nu_j^- k(\mathbf{x},\mathbf{x}_j^-)) \le 0. \tag{41}$$

□

APPENDIX C
PROOF OF PROPOSITION 5

Proof: By plugging $k(\mathbf{x},\mathbf{x}_*^+) = \eta e^{-\|\boldsymbol{\Gamma}(\mathbf{x}-\mathbf{x}_*^+)\|^2}$, and $k(\mathbf{x},\mathbf{x}_j^-) = \eta e^{-\gamma\|\boldsymbol{\Gamma}(\mathbf{x}-\mathbf{x}_j^-)\|^2}$ into (41), a point $\mathbf{x}$ is free if

$$e^{-\|\boldsymbol{\Gamma}(\mathbf{x}-\mathbf{x}_*^+)\|^2 + \frac{n_2}{n_1+n_2}\|\boldsymbol{\Gamma}(\mathbf{x}-\mathbf{x}_j^-)\|^2} \le \frac{\rho(e-b,\nu_j^-)}{\eta^{\frac{n_1}{n_1+n_2}}\sum_{i=1}^{M^+}\nu_i^+}. \tag{42}$$

Substituting the test point $\mathbf{x}$ by $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}$ in (42), the point $\mathbf{p}(t)$ is free if

$$\begin{aligned}V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) = &-(n_1+n_2)\|\boldsymbol{\Gamma}(\mathbf{p}_0+t\mathbf{v}-\mathbf{x}_*^+)\|^2\\ &+n_2\|\boldsymbol{\Gamma}(\mathbf{p}_0+t\mathbf{v}-\mathbf{x}_j^-)\|^2 - (n_1+n_2)\beta \le 0\end{aligned}$$

where $\beta = \log\frac{\rho(e-b,\nu_j^-)}{\eta^{\frac{n_1}{n_1+n_2}}\sum_{i=1}^{M^+}\nu_i^+}$. By expanding the quadratic norms in $V(t,\mathbf{x}_*^+,\mathbf{x}_j^-)$, the point $\mathbf{p}(t)$ is free if

$$V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) = at^2 + b(\mathbf{x}_*^+,\mathbf{x}_j^-)t + c(\mathbf{x}_*^+,\mathbf{x}_j^-) \le 0$$

$$\begin{aligned}\text{where}\qquad a = &-n_1\|\boldsymbol{\Gamma}\mathbf{v}\|^2,\\ b(\mathbf{x}_*^+,\mathbf{x}_j^-) = &-2\mathbf{v}^{\top}\boldsymbol{\Gamma}^{\top}\boldsymbol{\Gamma}(n_1\mathbf{p}_0 - (n_1+n_2)\mathbf{x}_*^+ + n_2\mathbf{x}_j^-),\\ c(\mathbf{x}_*^+,\mathbf{x}_j^-) = &-(n_1+n_2)\|\boldsymbol{\Gamma}(\mathbf{p}_0-\mathbf{x}_*^+)\|^2\\ &+n_2\|\boldsymbol{\Gamma}(\mathbf{p}_0-\mathbf{x}_j^-)\|^2 - (n_1+n_2)\beta.\end{aligned} \tag{43}$$

Note that $V(t,\mathbf{x}_*^+,\mathbf{x}_j^-)$ is a quadratic polynomial in t and the point $\mathbf{p}(t)$ is free if $V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \le 0$.

1) If it has less than two roots, (43) is satisfied for all t.
2) If it has two roots $t_1 < t_2$, then $V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \le 0$ for $t \ge t_2$ or $t \le t_1$. There are three cases:

a) $t_1 < t_2 \le 0$: $V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \le 0$ for all $t \ge 0$ or the entire ray $s(t)$ is free;
b) $0 \le t_1 < t_2$: $V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \le 0$ for $t \in [0,t_1]$ or the ray $s(t)$ is free for $t \in [0,t_1]$;
c) $t_1 \le 0 \le t_2$: $V(0,\mathbf{x}_*^+,\mathbf{x}_j^-) \ge 0$ or the ray $s(t)$ is colliding.

Let $\tau(\mathbf{p}_0,\mathbf{x}_*^+,\mathbf{x}_j^-)$

$$= \begin{cases}+\infty, & \text{if } V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \text{ has less than two roots}\\ +\infty, & \text{if } V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \text{ has two roots } t_1 < t_2 \le 0\\ t_1 & \text{if } V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \text{ has two roots } 0 \le t_1 < t_2\\ 0 & \text{if } V(t,\mathbf{x}_*^+,\mathbf{x}_j^-) \text{ has two roots } t_1 \le 0 \le t_2\end{cases}.$$

Note that $\mathbf{x}_*^+$ varies with t but belongs to a finite set; we calculate $\tau(\mathbf{p}_0,\mathbf{x}_i^+,\mathbf{x}_j^-)$ for all positive vectors $\mathbf{x}_i^+$ and take the minimum value. Therefore, $\mathbf{p}(t)$ is free as long as: $t \le t_u = \min_{i=1,\ldots,M^+}\tau(\mathbf{p}_0,\mathbf{x}^+,\mathbf{x}_j^-)$, which holds for any negative relevance vector $\mathbf{x}_j^-$. Therefore, the point $\mathbf{p}(t)$ is free as long as $t \le t_u^* = \max_{j=1,\ldots,M^-}\min_{i=1,\ldots,M^+}\tau(\mathbf{p}_0,\mathbf{x}^+,\mathbf{x}_j^-)$. □

APPENDIX D
PROOF OF PROPOSITION 6

Proof: Consider an arbitrary ray $\mathbf{p}'(t) = \mathbf{p}_0 + t\mathbf{v}', 0 \le t < \infty$. If we scale the velocity $\mathbf{v}'$ by a positive constant λ, i.e., $\mathbf{v} = \lambda\mathbf{v}'$, the ray $\mathbf{p}(t) = \mathbf{p}_0 + t\mathbf{v}, 0 \le t < \infty$ represents the

same ray as $\mathbf{p}'(t)$. If we scale the vector v' by $\lambda = \frac{1}{\|\mathbf{\Gamma v}'\|}$, the velocity vector $\mathbf{v}$ satisfies $\|\mathbf{\Gamma v}\| = \mathbf{1}$. Using the Cauchy–Schwarz inequality in (43) in Appendix C, we have

$$-2t\mathbf{v}^\top \mathbf{\Gamma}^\top \mathbf{\Gamma}(n_1\mathbf{p}_0 - (n_1+n_2)\mathbf{x}_*^+ + n_2\mathbf{x}_j^-)$$
$$\leq 2t\|\mathbf{\Gamma}(n_1\mathbf{p}_0 - (n_1+n_2)\mathbf{x}_*^+ + n_2\mathbf{x}_j^-)\|.$$

Therefore, the point $\mathbf{p}(t)$ is free if $\bar{V}(t, \mathbf{x}_*^+, \mathbf{x}_j^-) \leq 0$. Following the same reasoning as Proposition 5, the point $\mathbf{p}(t)$ is free for $0 < t < r_u$ or $0 < t < r_u^*$. In other words, the interior of the ellipsoids $\mathcal{E}(\mathbf{p}_0, r_u) \subseteq \mathcal{E}(\mathbf{p}_0, r_u^*)$ is free. □

REFERENCES

[1] J. Guzzi, A. Giusti, L. M. Gambardella, G. Theraulaz, and G. A. D. Caro, "Human-friendly robot navigation in dynamic environments," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2013, pp. 423–430.

[2] L. Janson, T. Hu, and M. Pavone, "Safe motion planning in unknown environments: Optimality benchmarks and tractable policies," in *Proc. Robot.: Sci. Syst.*, Pittsburgh, PA, USA, 2018.

[3] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox, "RGB-D mapping: Using kinect-style depth cameras for dense 3D modeling of indoor environments," *Int. J. Robot. Res.*, vol. 31, no. 5, pp. 647–663, 2012.

[4] M. Klingensmith, I. Dryanovski, S. Srinivasa, and J. Xiao, "Chisel: Real time large scale 3D reconstruction onboard a mobile device," in *Proc. Robot.: Sci. Syst.*, 2015.

[5] T. Whelan, R. Salas-Moreno, B. Glocker, A. Davison, and S. Leutenegger, "ElasticFusion: Real-time dense SLAM and light source estimation," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016.

[6] K. Wang, F. Gao, and S. Shen, "Real-time scalable dense Surfel mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 6919–6925.

[7] J. Behley and C. Stachniss, "Efficient Surfel-based SLAM using 3D laser range data in urban environments," in *Proc. Robot.: Sci. Syst.*, 2018.

[8] M. Kaess, "Simultaneous localization and mapping with infinite planes," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 4605–4611.

[9] S. Bowman, N. Atanasov, K. Daniilidis, and G. Pappas, "Probabilistic data association for semantic SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 1722–1729.

[10] L. Nicholson, M. Milford, and N. Sünderhauf, "QuadricSLAM: Dual quadrics from object detections as landmarks in object-oriented SLAM," *IEEE Robot. Automat. Lett.*, vol. 4, no. 1, pp. 1–8, Jan. 2019.

[11] S. Yang and S. Scherer, "CubeSLAM: Monocular 3-D object SLAM," *IEEE Trans. Robot.*, vol. 35, no. 4, pp. 925–938, Aug. 2019.

[12] M. Shan, Q. Feng, and N. Atanasov, "OrcVIO: Object residual constrained visual-inertial odometry," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5104–5111.

[13] L. Teixeira and M. Chli, "Real-time mesh-based scene estimation for aerial inspection," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 4863–4869.

[14] E. Piazza, A. Romanoni, and M. Matteucci, "Real-time CPU-based large-scale three-dimensional mesh reconstruction," *IEEE Robot. Automat. Lett.*, vol. 3, no. 3, pp. 1584–1591, Jul. 2018.

[15] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 1689–1696.

[16] A. Elfes, "Using occupancy grids for mobile robot perception and navigation," *Computer*, vol. 22, no. 6, pp. 46–57, 1989.

[17] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3D reconstruction with loop closure," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 500–516.

[18] M. Muglikar, Z. Zhang, and D. Scaramuzza, "Voxel map for visual SLAM," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 4181–4187.

[19] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Auton. Robots*, vol. 34, no. 3, pp. 189–206, 2013.

[20] M. Zeng, F. Zhao, J. Zheng, and X. Liu, "Octree-based fusion for realtime 3D reconstruction," *Graph. Models*, vol. 75, no. 3, pp. 126–136, 2013.

[21] E. Vespa, N. Nikolov, M. Grimm, L. Nardi, P. H. J. Kelly, and S. Leutenegger, "Efficient octree-based volumetric SLAM supporting signed-distance and occupancy mapping," *IEEE Robot. Automat. Lett.*, vol. 3, no. 2, pp. 1144–1151, Apr. 2018.

[22] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proc. Conf. Comput. Graph. Interactive Techn.*, 1996, pp. 303–312.

[23] M. Kazhdan, M. Bolitho, and H. Hoppe, "Poisson surface reconstruction," in *Proc. Eurographics Symp. Geometry Process.*, 2006, pp. 61–70.

[24] S. Izadi *et al.*, "KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. ACM Sym. User Interface Softw. Technol.*, 2011, pp. 559–568.

[25] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D euclidean signed distance fields for on-board MAV planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1366–1373.

[26] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: A fast incremental euclidean distance fields for online quadrotor motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 4423–4430.

[27] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3D reconstruction at scale using Voxel hashing," *ACM Trans. Graph.*, vol. 32, no. 6, pp. 1–11, 2013.

[28] T. Duong, N. Das, M. Yip, and N. Atanasov, "Autonomous navigation in unknown environments using sparse kernel-based occupancy mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9666–9672.

[29] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA, USA: MIT Press, 2005.

[30] G. Grisetti, C. Stachniss, and W. Burgard, "Improved techniques for grid mapping with Rao-Blackwellized particle filters," *IEEE Trans. Robot.*, vol. 23, no. 1, pp. 34–46, Feb. 2007.

[31] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *Int. J. Robot. Res.*, vol. 31, no. 1, pp. 42–62, 2012.

[32] J. Wang and B. Englot, "Fast, accurate gaussian process occupancy maps via test-data octrees and nested Bayesian fusion," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1003–1010.

[33] M. G. Jadidi, J. V. Miro, and G. Dissanayake, "Warped Gaussian processes occupancy mapping with uncertain inputs," *IEEE Robot. Automat. Lett.*, vol. 2, no. 2, pp. 680–687, Apr. 2017.

[34] F. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent," *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1717–1730, 2016.

[35] R. Senanayake and F. Ramos, "Bayesian Hilbert maps for dynamic continuous occupancy mapping," in *Proc. Conf. Robot Learn.*, 2017, pp. 458–471.

[36] V. Guizilini and F. Ramos, "Large-scale 3D scene reconstruction with hilbert maps," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2016, pp. 3247–3254.

[37] V. Guizilini and F. Ramos, "Towards real-time 3D continuous occupancy mapping using Hilbert maps," *Int. J. Robot. Res.*, vol. 37, no. 6, pp. 566–584, 2018.

[38] M. E. Tipping, "The relevance vector machine," in *Proc. Adv. Neural Inf. Process. Syst.*, 2000, pp. 652–658.

[39] M. E. Tipping, "Sparse Bayesian learning and the relevance vector machine," *J. Mach. Learn. Res.*, vol. 1, no. 6 pp. 211–244, 2001.

[40] M. E. Tipping *et al.*, "Fast marginal likelihood maximisation for sparse Bayesian models," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2003, pp. 276–283.

[41] B. T. Lopez and J. P. How, "Aggressive 3-D collision avoidance for high-speed navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 5759–5765.

[42] J. Chen and S. Shen, "Improving octree-based occupancy maps using environment sparsity with application to aerial robot navigation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3656–3663.

[43] D. Fridovich-Keil, E. Nelson, and A. Zakhor, "AtomMap: A probabilistic amorphous 3D map representation for robotics and surface reconstruction," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2017, pp. 3110–3117.

[44] J. Bialkowski, M. Otte, S. Karaman, and E. Frazzoli, "Efficient collision checking in sampling-based motion planning via safety certificates," *Int. J. Robot. Res.*, vol. 35, no. 7, pp. 767–796, 2016.

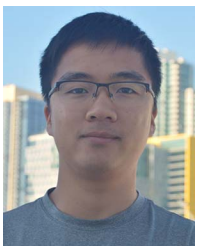
[45] J. Luo and K. Hauser, "An Empirical study of optimal motion planning," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2014, pp. 1761–1768.

[46] K. Hauser, "Lazy collision checking in asymptotically-optimal motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2015, pp. 2951–2957.

[47] E. G. Tsardoulias, A. Iliakopoulou, A. Kargakos, and L. Petrou, "A review of global path planning methods for occupancy grid maps regardless of obstacle density," *J. Intell. Robot. Syst.*, vol. 84, no. 1, pp. 829–858, 2016.

[48] N. Das, N. Gupta, and M. Yip, "Fastron: An online learning-based model and active learning strategy for proxy collision detection," in *Proc. Conf. Robot Learn.*, 2017, pp. 496–504.

[49] J. Pan and D. Manocha, "Efficient configuration space construction and optimization for motion planning," *Engineering*, vol. 1, no. 1, pp. 046–057, 2015.
[50] J. Huh and D. D. Lee, "Learning high-dimensional mixture models for fast collision detection in rapidly-exploring random trees," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 63–69.
[51] N. Das and M. Yip, "Learning-based proxy collision detection for robot motion planning applications," *IEEE Trans. Robot.*, vol. 36, no. 4, pp. 1096–1114, Aug. 2020.
[52] J. Pan, S. Chitta, and D. Manocha, "FCL: A general purpose library for collision and proximity queries," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2012, pp. 3859–3866.
[53] S. Liu, N. Atanasov, K. Mohta, and V. Kumar, "Search-based motion planning for quadrotors using linear quadratic minimum time control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 2872–2879.
[54] A. De Luca, G. Oriolo, and M. Vendittelli, "Stabilization of the unicycle via dynamic feedback linearization," *IFAC Proc. Volumes*, vol. 33, no. 27, pp. 687–692, 2000.
[55] J. Franch and J. Rodriguez-Fortun, "Control and trajectory generation of an Ackerman vehicle by dynamic linearization," in *Proc. Eur. Control Conf.*, 2009, pp. 4937–4942.
[56] D. J. MacKay, "The evidence framework applied to classification networks," *Neural Comput.*, vol. 4, no. 5, pp. 720–736, 1992.
[57] I. T. Nabney, "Efficient training of RBF networks for classification," *Int. J. Neural Syst.*, vol. 14, no. 3, pp. 1–8, 2004.
[58] R. M. Neal, *Bayesian Learning for Neural Networks*, vol. 118, Berlin, Germany: Springer, 2012.
[59] S. Koenig and Y. Smirnov, "Sensor-based planning with the free space assumption," in *Proc. IEEE Int. Conf. Robot. Automat.*, 1997, pp. 3540–3545.
[60] R. B. Lehoucq, D. C. Sorensen, and C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems With Implicitly Restarted Arnoldi Methods*. Philadelphia, PA, USA: SIAM, 1998.
[61] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2009.
[62] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proc. Robot.: Sci. Syst.*, 2010.
[63] A. Howard and N. Roy, "The robotics data set repository (Radish)," 2003. [Online]. Available: http://radish.sourceforge.net/
[64] O. Arslan and D. E. Koditschek, "Exact robot navigation using power diagrams," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2016, pp. 1–8.
[65] R. Senanayake, A. Tompkins, and F. Ramos, "Automorphing Kernels for nonstationarity in mapping unstructured environments," in *Proc. 2nd Conf. Robot Learn.*, 2018, pp. 443–455.
[66] A. Tompkins, R. Senanayake, and F. Ramos, "Online domain adaptation for occupancy mapping," in *Proc. Robot.: Sci. Syst.*, 2020.



Thai Duong (Student Member, IEEE) received the B.S. degree in electronics and telecommunications from the Hanoi University of Science and Technology, Hanoi, Vietnam, in 2011 and the M.S. degree in electrical and computer engineering from Oregon State University, Corvallis, OR, USA, in 2013. He is currently working toward the Ph.D. degree in electrical and computer engineering with the University of California, San Diego, La Jolla, CA, USA.

His research interests include machine learning in robotics, mapping and active exploration using mobile robots, robot dynamics learning, and decision making under uncertainty.

Michael Yip (Senior Member, IEEE) received the B.Sc. degree in mechatronics engineering from the University of Waterloo, Waterloo, ON, Canada, in 2009, the M.S. degree in electrical engineering from the University of British Columbia, Endowment Lands, BC, Canada, in 2011, and the Ph.D. degree in bioengineering from Stanford University, Stanford, CA, USA, in 2015.

He is currently an Assistant Professor of electrical and computer engineering at University of California, San Diego, La Jolla, CA, USA, and IEEE RAS Distinguished Lecturer, Hellman Fellow, and Director with the Advanced Robotics and Controls Laboratory (ARCLab). His group currently focuses on solving problems in data-efficient and computationally efficient robot control and motion planning through the use of various forms of learning representations, including deep learning and reinforcement learning strategies. His lab applies these ideas to surgical robotics and the automation of surgical procedures. Previously, his research has investigated different facets of model-free control, planning, haptics, soft robotics, and computer vision strategies, all toward achieving automated surgery.

Dr. Yip's work has been recognized through several best paper awards at ICRA, including the inaugural best paper award for *IEEE Robotics and Automation Letters*. He has previously been a Research Associate with Disney Research Los Angeles in 2014, a Visiting Professor with Stanford University in 2019, and a Visiting Professor with Amazon Robotics' Machine Learning and Computer Vision group in Seattle, WA, USA, in 2018.

Nikolay Atanasov (Member, IEEE) received the B.S. degree in electrical engineering from Trinity College, Hartford, CT, USA, in 2008 and the M.S. and Ph.D. degrees in electrical and systems engineering from the University of Pennsylvania, Philadelphia, PA, USA, in 2012 and 2015, respectively.

He is currently an Assistant Professor of electrical and computer engineering with the University of California, San Diego, La Jolla, CA, USA. His research focuses on robotics, control theory, and machine learning, applied to active sensing using ground and aerial robots. He works on probabilistic environment models that unify geometry and semantics and on optimal control and reinforcement learning approaches for minimizing uncertainty in these models.

Dr. Atanasov's work has been recognized by the Joseph and Rosaline Wolf award for the best Ph.D. dissertation in Electrical and Systems Engineering at the University of Pennsylvania in 2015 and the best conference paper award at the *International Conference on Robotics and Automation* in 2017.

Authorized licensed use limited to: Univ of Calif San Diego. Downloaded on February 24,2023 at 22:19:03 UTC from IEEE Xplore. Restrictions apply.