Robotics and Autonomous Systems 153 (2022) 104077

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Improving obstacle boundary representations in predictive occupancy mapping

Erik Pearson [a], Kevin Doherty [b], Brendan Englot [a,*]

[a] Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA
[b] Massachusetts Institute of Technology, 77 Massachusetts Ave, Cambridge, MA 02139, USA

ARTICLE INFO

Article history:
Received 29 August 2021
Received in revised form 27 January 2022
Accepted 3 March 2022
Available online 15 March 2022

Keywords:
Mapping
Range sensing
Mobile robotics

ABSTRACT

Predictive, inference-based occupancy mapping has been used successfully in many instances to create accurate and descriptive maps from sparse data, defining occupied space in a manner suitable to support autonomous navigation. However, one key drawback of inferring occupancy based largely on the proximity of range sensor observations is inaccuracy at the boundary between occupied and free space, where sparse coverage by the sensor data can be misinterpreted. To obtain a more accurate representation of the boundary between free and occupied space, we propose several modifications to a recently published occupancy mapping algorithm that uses Bayesian generalized kernel inference. In particular, our proposed algorithm distinguishes between *unknown* map cells with insufficient observations, and those which are *uncertain* due to disagreement among numerous observations, in a predictive, inference-based occupancy map. This distinction is key to our improved ability to capture ambiguities arising at the boundary between free and occupied space. We validate our approach using synthetic range data from a simulated environment and demonstrate real-time mapping performance using range data acquired by a ground robot operating in an underground mine.

1. Introduction

Autonomous mobile robot exploration uses incomplete information about the surrounding environment to evaluate a performance metric that prioritizes the gathering of new data. However, the data gathered can be degraded by factors such as sensor noise, and gaps in coverage within the data itself. Recently, predictive occupancy grid mapping algorithms have been developed to address this issue. Such inference-based occupancy grid mapping algorithms can successfully filter noise and fill gaps, while also decreasing the memory required to maintain such data. Simultaneous localization and mapping (SLAM) can be used alongside such mapping tools to curb the localization error that may further exacerbate mapping errors.

While predictive mapping has addressed some of the central issues in occupancy mapping, it has yet to solve a few problematic consequences of applying predictive inference in this setting. One example is the over-estimation of free space caused by glancing rays, for which the gaps between sparse range observations may be filled with inaccurate predictions of free space. This can lead to incorrect assumptions in the planning and decision-making processes of robot exploration. The boundaries between free and occupied space within occupancy grid maps are also often characterized erroneously.

We present a new Bayesian generalized kernel (BGK) [1] inference mapping framework that addresses and attempts to rectify these issues. To avoid the erroneous prediction of free space due to glancing rays, we reduce the length of rays used in the training data. This produces fewer data to support free space estimation, but within the region between the sensor and ray endpoints, the rays are closer together, resulting in denser data that yields more accurate classification of free space (see Fig. 1).

On the boundary between free and occupied space, there exists a region where inference-based occupancy grid maps will produce erroneous predictions of occupancy. Even when there is plenty of data influencing the classification of this region, it is difficult to accurately define obstacle boundaries. Accordingly, the goal of our proposed algorithm is to accurately define free and occupied space near obstacle boundaries, while incorporating a new state, the *uncertain* state. The *uncertain* state occurs when there is conflicting data about a given map cell, not to be confused with *unknown*, which categorizes map regions where robots have yet to explore. The *variance* of each map cell plays a vital role in defining uncertainty, therefore this new successor algorithm to BKGOctoMap-L is named BGKOctoMap-LV.

2. Related work

Occupancy maps are typically produced from range data. Range data in the most basic form has a sensor origin linked to each

* Corresponding author.
E-mail addresses: epearson@stevens.edu (E. Pearson), kjd@csail.mit.edu (K. Doherty), benglot@stevens.edu (B. Englot).

https://doi.org/10.1016/j.robot.2022.104077
0921-8890/

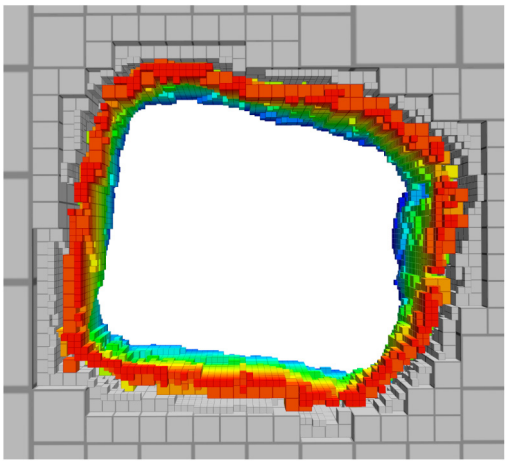


Fig. 1. Example of the proposed algorithm mapping a column in an underground mine, with free space outside the column shown in gray.

endpoint found during the sensing process. One common data structure used to define these endpoints is a point cloud [2]. As the name implies, a point cloud is made up of points within 3D Euclidean space that are located based on a predetermined frame of reference. Due to their simplicity, point clouds are the default data structure for a wide variety of sensors used for mapping. The Robot Operating System (ROS) [3] is frequently used to both capture and visualize point cloud data. Capturing point cloud data can be performed using real world sensors, or it can be simulated using environments such as Gazebo [4].

However, point cloud data presents a few limitations. Firstly, the data can be overwhelming in size. A 32-beam lidar can acquire over one million points per second, which rapidly accumulates when scanning a large environment. Secondly, there is no native free space representation within the point cloud data structure. Further computation is necessary to derive free space from each scan. Thirdly, there is no native representation of obstacle boundaries within a point cloud, and further computation is required to estimate them. Finally, point cloud data is susceptible to corruption by noise and other sources of error.

The errors induced by inaccurate robot localization corrupt the registration of the point cloud associated with a given range scan. This particular source of error has been studied extensively, with the most common remedy being SLAM [5]. In this paper, we will focus on errors that derive from additive noise in the sensing process itself, which is frequently assumed to be Gaussian [6], but can take on other characteristics. We will assume our robot's state is fully observable.

2.1. Occupancy grid mapping

Considering the limitations of point cloud data, researchers have pursued new data structures to define maps efficiently. A common method in use today is the occupancy grid map [7], which discretizes 3D space into voxels with a designated spatial resolution. By setting a low resolution we can drastically reduce the size of the data structure, as all points within a given voxel will be tallied together. However, low resolutions reduce our understanding of the environment. Therefore, a balance is necessary between memory efficiency and detail.

OctoMap [8], proposed more than a decade ago, continues to be widely used for memory-efficient 3D occupancy grid mapping. OctoMap is built to take point cloud data as an input, which is processed into a probabilistic map of free and occupied cells at a specified spatial resolution. Successors such as UFOMap [9] or the method described in [10] have been proposed in recent years to optimize both the process of introducing points into the map, as well as the structure of the map.

Occupancy mapping algorithms often permit cells to be represented at multiple spatial resolutions within a single map. One method used to achieve this is called *pruning*, where newly defined cells belonging to the same class are condensed into larger, lower-resolution cells. Pruning has been utilized in several mapping algorithms to reduce the map data while maintaining an accurate representation of the environment [1,11,12].

Regardless of the data structure used, sparse data from sensors will, across some resolutions, result in gaps within an occupancy grid map. These gaps are a common issue, as the distance between neighboring sensor rays increases the further an object is from the sensor origin. Therefore, it is often beneficial to implement predictive mapping techniques to infer the contents of an occupancy map between such gaps.

2.2. Learning-aided occupancy mapping

There are many methods for predicting the missing contents of 3D occupancy grid maps, however the majority of them frame occupancy mapping as a supervised learning problem. Gaussian processes have been widely used for this purpose, and they can accommodate a variety of kernel functions; of particular note is their compatibility with a sparse covariance function intended for exact inference over large datasets [13]. In most learning-aided mapping approaches, predicting the occupancy probability of cells not directly observed by sensor rays is influenced by the proximity of existing data to the query point. Each scan from a sensor can be interpreted as a set of "hit points" that define occupied regions and sensor rays that define free space.

Probabilistic inference can also address noise and other sources of uncertainty in occupancy mapping. Warped Gaussian processes [14] and Rao–Blackwellized particle filters [15] can account for the localization uncertainty captured by SLAM, while accurately representing occupancy probabilities across a grid map. Occupancy grid mapping has also been made more compatible with the map corrections required by SLAM in [16], where previous sensor observations are moved within an occupancy map as SLAM corrections are made. Due to the added expense of keeping track of all sensor observations so they can be moved if necessary, the majority of inference-based occupancy mapping algorithms, including ours, assume a robot's localization is accurate, but that its range sensor is noisy.

Many occupancy mapping algorithms use Gaussian Processes (GPs) to infer the missing contents of gaps caused by sensor inadequacies [11,17–20]. Due to the computational complexity of GPs, Bayesian generalized kernel (BGK) [21] inference has been proposed as an efficient alternative technique for applying Bayesian nonparametric inference to occupancy grid maps [1, 12]. Other inference methods that address the poor scalability of GPs include integral kernels [22], Markov random field ray propagation [23], Hilbert maps [24,25], Hilbert maps with deep learning [26], decomposable radial kernels [27], confidence rich grid mapping [28], and a method that employs spherical cells rather than voxels [29].

Another recently proposed approach uses the agreement/disagreement of nearby sensor observations to respectively stretch or compress the kernel [30]. The adaptive kernel inference mapping algorithm (AKIMap) assumes the gaps between sensor observations to be a greater source of inaccuracy than sensor noise. Accordingly, the AKIMap algorithm achieves gap-filling by expanding the kernel's influence toward similar occupancy states and compressing the kernel's influence away from dissimilar occupancy states.

2.3. Free space representation

One aspect of occupancy grid mapping that distinguishes it from other classification problems is the typical use of three classes to categorize the contents of map cells: free, occupied, and unknown. To frame occupancy mapping as a binary classification problem amenable to solution by supervised learning algorithms, several learning-aided occupancy mapping algorithms focus specifically on classifying *occupied* cells, with the opposite class simply being "not occupied". Such algorithms typically do not quantify the accuracy with which free space can be predicted, focusing instead on the accuracy with which obstacles are mapped [17–19,23].

Predictive occupancy mapping algorithms that classify free space adopt a variety of approaches to interpreting a robot's range sensor observations. The most common method is to define an evenly spaced distribution of points between the sensor origin of a range scan and the "hit point" at the end of each sensor ray, which are interpreted to be sensor observations of free space [11,12,20,22,28]. AtomMap [29] also adopts this approach, but along the portion of a ray close to the hit point, it computes the normal to the nearby obstacle surface and produces two more free cells normal to the surface. Another method to define observations of free space is to randomly sample points between the sensor origin and the hit point of a ray. This approach requires certain attributes within the algorithm, such as kernel stretching to account for sparsity among the data, but was utilized by [24,25,30]. Guo et al. [27] used a Poisson distribution along each ray.

A final method for defining observations of free space is to utilize the sensor rays themselves, rather than representing free space observations using points. Both GP Occupancy Mapping (GPOM) [17] and BGKOctoMap-L [1] do this successfully by using the point-to-line distance, i.e., the shortest distance between a point of interest and a given sensor ray, to define a single free space observation contributed by that ray in the evaluation of a given query point. By adopting this approach, BGKOctoMap-L is able to mitigate the over-sampling of free space that would otherwise result from high-resolution interpolation along each sensor ray.

2.4. Unknown & uncertain representation

The third classification commonly used in occupancy grid mapping, *unknown*, typically applies to cells that have not yet been observed by a robot's sensor. In this paper, we will argue that predictive occupancy mapping benefits from two separate types of unknown classification: the standard definition of unknown in which insufficient data has been procured about a given region, and *uncertain*, where conflicting data makes accurate classification of map cells challenging. Although it has not been labeled as the uncertain class previously in predictive inference-based occupancy mapping, several previous algorithms have captured what they define as the unknown class, applying it when there is conflicting data. However, a few algorithms created methods to define unknown as missing data.

By explicitly defining free space, [10] was able to infer unknown cells as those remaining in the map apart from the cells classified as occupied and free. This "leftover" method is widely used in occupancy grid mapping, by which all map cells are defined to be unknown at the start of a mapping exercise, until sensor rays pass through them or near them, and they are gradually eliminated and replaced by free and occupied cells.

Another method to define unknown in occupancy mapping was proposed by Jadidi et al. [31], in which two separate maps are generated and then fused together. The first map estimates occupied and not occupied classes, while the second map estimates free and not free classes. When the two maps are merged together, the resulting map contains free and occupied cells contributed by each component map, and the remaining cells are categorized as unknown. This merging method originally encountered problems with overestimation or underestimation of a given class due to the selected kernel functions being applied to separate inference problems in isolation. A revised method was later proposed to solve this issue in [32], by merging the two component maps into a unique continuous occupancy map. Although most mapping algorithms ignore under- and over-estimation of the free and occupied classes, we aim to make an improvement specifically to address that issue. Additionally, in the sections that follow, we will describe a methodology to suitably define both unknown and uncertain classes separately within a predictive occupancy map, to achieve descriptive and accurate inference.

3. Background - Bayesian generalized kernel inference

3.1. Counting sensor model

Our proposed algorithm is structured similarly to the BGKOctoMap-L framework proposed by a subset of the authors, in [1]. Therefore, we make similar assumptions about our workspace, where measurements are defined as $\mathcal{Y} = \{y_1, \ldots, y_N \mid y_i \in \{0, 1\}\}$ and the values are assigned based on whether the measurements are a ray or hit point, which represent free space and occupied space respectively. Each $y_i \in \mathcal{Y}$ has a corresponding location denoted as $\mathcal{X} = \{x_1, \ldots, x_N \mid x_i \in \mathbb{R}^3\}$, such that all N range measurements can be described fully in the form of the tuple (x_i, y_i). Map cells are also assumed to be indexed by $j \in \mathbb{Z}^+$.

However, our proposed algorithm, BGKOctoMap-LV, differs from the prior work in the way we define the occupancy of a map cell. In [1], each cell was given a probability of occupancy defined by θ_j with Bernoulli likelihood

$$p(y_i \mid \theta_j) = \theta_j^{y_i}(1-\theta_j)^{1-y_i}, \tag{1}$$

where the posterior function $p(\theta_j \mid \mathcal{X}, \mathcal{Y})$ was used to estimate occupancy. This function was obtained using a conjugate prior, where θ_j was assumed to be a Beta distribution, resulting in the following occupancy estimator and variance:

$$\mathbb{E}[\theta_j] = \frac{\alpha_j}{\alpha_j + \beta_j} \tag{2}$$

$$\mathbb{V}[\theta_j] = \frac{\alpha_j\,\beta_j}{(\alpha_j + \beta_j)^2(\alpha_j + \beta_j + 1)} \tag{3}$$

The hyperparameters α_j and β_j were defined to simplify the joint measurement likelihood of each cell. Observations are only added to the "counts" of these parameters if they fall within the region R_j that influences cell j.

$$\alpha_j := \alpha_0 + \sum_{i,\, x_i \in R_j} y_i \tag{4}$$

$$\beta_j := \beta_0 + \sum_{i,\, x_i \in R_j} (1 - y_i). \tag{5}$$

By this definition, α_j represents the total count of hit points within region R_j, while β_j is the total number of occurrences of rays passing through the same region R_j. Each term also has a prior defined by α_0 and β_0.

In that previously used estimation procedure, θ_j is assumed to have a Beta distribution. However, for simplification, we instead assume a Bernoulli distribution. We also define a new parameter for each cell j to be the occupancy state $m_j \in \{0, 1\}$, which follows from the definition of measurements y_i, where 0 represents the free or unoccupied state while 1 is defined as occupied. This

distribution's probability mass function can leverage the same α_j and β_j parameters described previously.

$$P(m_j) = \begin{cases} \dfrac{\alpha_j}{\alpha_j + \beta_j}, & \text{if } m_j = 1 \\ \dfrac{\beta_j}{\alpha_j + \beta_j}, & \text{if } m_j = 0 \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

With this new distribution we can estimate occupancy and the variance of occupancy. We will use the term μ_j to describe the occupancy estimate in the equations below.

$$\mathbb{E}[m_j] = \sum_{m_j=\{0,1\}} m_j \, p(m_j) = \frac{\alpha_j}{\alpha_j + \beta_j} = \mu_j \tag{7}$$

$$\mathbb{V}[m_j] = \sum_{m_j=\{0,1\}} (m_j - \mu_j)^2 p(m_j) = \frac{\alpha_j \, \beta_j}{(\alpha_j + \beta_j)^2} \tag{8}$$

The expected value of m_j is equal to that of the Beta distribution θ_j, which suggests a reasonable assumption when using the new Bernoulli distribution. However, while the variance estimates are similar, they vary enough to create unique perspectives. The variance of a Beta distribution will tend to zero as more data is captured, even when α_j and β_j are equal. The variance of a Bernoulli distribution, however, will lie at a maximum when $\alpha_j = \beta_j$, regardless of how much data has been gathered, serving our goal of using map cell variance to characterize uncertain cells with conflicting observations.

3.2. Free space using rays

Using continuous rays permits an accurate representation of free space, and avoids the potential pitfalls of interpolating along a ray to generate evenly spaced free points from the sensor origin to a hit point. If the free points are discretized too finely, multiple free points will land within individual map cells, outweighing the influence of hit points. If they are discretized too coarsely, cells along a ray may have no points introduced into them, even though a sensor ray has passed through. These problems can be avoided by using the ray itself to represent free space, rather than interpolated points.

Our prior work on BGKOctoMap-L [1] used a continuous ray from the sensor origin to the hit point to define free space, where each map cell only sees the influence of the nearest point on each range beam. The nearest point was found by searching along the length of the ray using the following equations:

$$x_{free} = \begin{cases} P, & \text{if } \delta < 0 \\ P + \delta \dfrac{\overrightarrow{PQ}}{|PQ|}, & \text{if } 0 < \delta < 1 \\ Q, & \text{if } \delta > 1 \end{cases} \tag{9}$$

$$\delta = \frac{\overrightarrow{PQ} \cdot \overrightarrow{Px_*}}{|PQ|}, \tag{10}$$

where P is the sensor origin, Q is the hit point, and x_* is the map cell center. When a map cell lies behind the origin of a ray, the nearest point will be P, whereas if a map cell lies beyond the end of the ray, the nearest point will be Q. However, if the map cell occurs somewhere between P and Q within 3D space, then the nearest point will be determined using δ. These assumptions are also adopted in BGKOctoMap-LV.

3.3. Kernel estimation

While the discretization provided by occupancy grid maps is useful in many respects, it can also reduce the accuracy and precision captured by the underlying sensor data. Therefore, we assume there is a smooth distribution of sensor data over any given map cell which can be considered a region of influence around each cell. BGKOctoMap-L uses the following sparse kernel to assume a distribution from a cell center [13]: $k(x, x') =$

$$\begin{cases} \sigma \left[\frac{2+\cos(2\pi \frac{d}{l})}{3}(1 - \frac{d}{l}) + \frac{1}{2\pi} \sin(2\pi \frac{d}{l}) \right] & \text{if } d < l \\ 0 & \text{if } d > l, \end{cases} \tag{11}$$

where $d = \|x - x'\|$ is the distance between a map cell's center x and a nearby sensor observation x'. A constant hyperparameter σ is used to define the strength of influence while l declares the size of the region of influence. As seen in Fig. 2, when d is small the influence on the predicted occupancy of a map cell will be largest. This procedure for prediction dictated by spatial proximity can improve the accuracy of occupancy map estimation, using relevant data from outside a cell's boundaries to influence its classification.

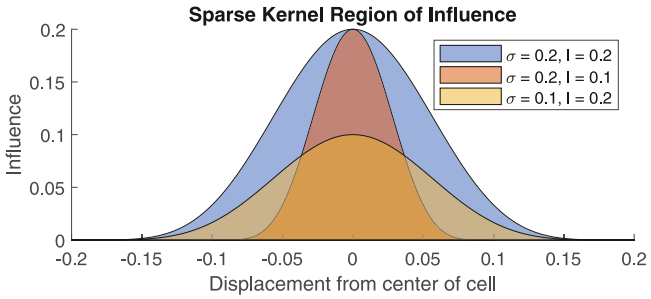


Fig. 2. An illustrative example of the sparse kernel function (Eq. (11)) with a variety of values for σ and l.

Eqs. (4) and (5) describing the counting sensor model must be updated to incorporate the above kernel function. In those equations, y_i represents the observations of occupancy from the incoming data, such that a hit point will have $y_i = 1$ while a ray passing only through free space will have $y_i = 0$.

$$\alpha_j := \alpha_0 + \sum_{i=1}^{N} k(x_j, x_i) y_i \tag{12}$$

$$\beta_j := \beta_0 + \sum_{i=1}^{N} k(x_j, x_i)(1 - y_i) \tag{13}$$

Eqs. (12) and (13) are used by BGKOctoMap-L and define the influence of range sensor observations on the classification of occupancy map cells. However, there are two other states in our newly proposed variant of this algorithm, unknown and uncertain. Unknown has typically been assumed to apply when $m_j = 0.5 \pm \epsilon$ for some small ϵ, however, the current Eq. (7) for expected value will return $m_j \approx 0.5$ for all instances where $\alpha_j \approx \beta_j$, even when α_j and β_j are very large and the state of cell j should be uncertain. Therefore, new classifications will be introduced in BGKOctoMap-LV, both to account for unknown cells, and to separately define cells containing conflicting information as uncertain.

4. Introducing new classes into BGKOctoMap-L

4.1. Defining and weighting the unknown classification

The unknown classification represents regions of a robot's workspace that are yet to be explored. Even when some initial exploration has occurred, a map cell may still be considered

partly unknown. Therefore, we propose to incorporate pseudo-evidence into our estimation procedure via a threshold for the unknown classification. Above that threshold, a cell's contents are assumed to be observed, and it must be designated occupied, free or uncertain. To ensure that unknown is only counted when α_j and β_j are small, we define a threshold w_{MIN} such that:

$$w_j = \begin{cases} w_{MIN} & \text{if } \alpha_j + \beta_j < w_{MIN} \\ \alpha_j + \beta_j & \text{otherwise} \end{cases} \tag{14}$$

The unknown classification is assumed to apply when $m_j = 0.5$, therefore, we will assume that there is pseudo-evidence γ_j corresponding to this class, in the same way that α_j accounts for occupied space and β_j accounts for free space. Based on our threshold w_{MIN}, we can make the assumption that $w_j = \alpha_j + \beta_j + \gamma_j$, which means we can solve for γ_j given that we know the other three terms. However, now we need to modify the probability mass function and update the occupancy and variance equations used in our proposed estimation process.

$$P(m_j) = \begin{cases} \dfrac{\alpha_j}{\alpha_j + \beta_j + \gamma_j}, & \text{if } m_j = 1 \\ \dfrac{\gamma_j}{\alpha_j + \beta_j + \gamma_j}, & \text{if } m_j = 0.5 \\ \dfrac{\beta_j}{\alpha_j + \beta_j + \gamma_j}, & \text{if } m_j = 0 \\ 0, & \text{otherwise} \end{cases} \tag{15}$$

With this new distribution we can update our estimates for occupancy and the variance of occupancy accordingly.

$$\mathbb{E}[m_j] = \sum_{m_j \in \{0, 0.5, 1\}} m_j \, p(m_j) = \frac{\alpha_j + 0.5\gamma_j}{\alpha_j + \beta_j + \gamma_j} = \mu_j \tag{16}$$

$$\mathbb{V}[m_j] = \sum_{m_j \in \{0, 0.5, 1\}} (m_j - \mu_j)^2 p(m_j) = \frac{\alpha_j \, \beta_j + 0.25(\alpha_j + \beta_j)\gamma_j}{(\alpha_j + \beta_j + \gamma_j)^2} \tag{17}$$

When $\gamma_j = 0$, we can see that Eq. (16) reverts back to Eq. (7), which considered only free/occupied classes. Including γ_j via the w_{MIN} term will not change the final state of a cell's occupancy, but will produce a result that includes unknown when $\alpha_j + \beta_j < w_{MIN}$. However, this particular equation still allows for the probability of occupancy to be in the unknown range when $\alpha_j \approx \beta_j$, even for large values. To separate unknown from uncertain, we define a new estimator to push the probability of occupancy to 1 or 0 as γ_j shrinks:

$$\hat{\mathbb{E}}[m_j] = \begin{cases} \dfrac{\alpha_j + 0.5\gamma_j}{\alpha_j + \gamma_j} & \text{if } \alpha_j \geq \beta_j \\ \dfrac{0.5\gamma_j}{\beta_j + \gamma_j} & \text{if } \alpha_j < \beta_j \end{cases} \tag{18}$$

In typical applications of occupancy grid mapping, the majority consensus of the sensor observations that land within a map cell will dictate whether that cell is labeled free or occupied. Following the same philosophy, Eq. (18) checks the consensus of past sensor observations, and then pushes the occupancy probability toward the corresponding class. As γ_j eventually approaches 0 with the exploration of a robot's workspace, a cell's occupancy probability will approach 1 or 0, indicating that its contents are no longer unknown.

4.2. Defining the uncertain classification

Unfortunately the threshold w_{MIN} does not address the issue that $\alpha_j \approx \beta_j$ should represent uncertainty when they are large, indicating disagreement among sensor observations. There are a few methods capable of dealing with this issue, such as assigning more weight to newer data than to old data [33]. However, our proposed solution allows us to push the probability of occupancy to its upper and lower limits while also allowing the occupancy probability to flip from occupied to free and vice versa without nearing unknown.

We update the variance Eq. (17) based on Eq. (18), as we need to use μ_j, which yields:

$$\hat{\mathbb{V}}[m_j] = \begin{cases} \dfrac{(\alpha_j^2 + \alpha_j\gamma_j)(4\beta_j + \gamma_j) + \beta_j\gamma_j^2}{4(\alpha_j + \gamma_j)^2(\alpha_j + \beta_j + \gamma_j)} & \text{if } \alpha_j \geq \beta_j \\ \dfrac{\alpha_j(4\beta_j^2 + 4\beta_j\gamma_j + \gamma_j^2) + \beta_j\gamma_j(\beta_j + \gamma_j)}{4(\beta_j + \gamma_j)^2(\alpha_j + \beta_j + \gamma_j)} & \text{if } \alpha_j < \beta_j \end{cases} \tag{19}$$

When $\gamma_j \gg \alpha_j, \beta_j$, we can see both forms of variance converge asymptotically to $\frac{1}{\gamma_j}$, which indicates our starting variance when $w_{MIN} \gg \alpha_0, \beta_0$. Therefore, if w_{MIN} is sufficiently large, resulting in a large quantity of pseudo-evidence indicating unknown, the starting variance will be small.

However, when $\gamma_j = 0$, the new variance equation simplifies:

$$\hat{\mathbb{V}}'[m_j] = \begin{cases} \dfrac{\beta_j}{\alpha_j + \beta_j} & \text{if } \alpha_j \geq \beta_j \\ \dfrac{\alpha_j}{\alpha_j + \beta_j} & \text{if } \alpha_j < \beta_j \end{cases} \tag{20}$$

Known cells will be updated according to this equation. While Eq. (20) is different from the traditional variance shown in Eq. (8), we can still use this as an uncertainty metric, as it usefully captures the proportionality of the lesser observation count to the total observation count.

We do not apply the "pushing to extremity" premise of Eq. (18) to a map cell's variance, as that would result in the variance dropping to zero once $\alpha_j + \beta_j > w_{MIN}$, reducing the information we have to describe the occupancy of that cell and our confidence in its prediction.

4.3. Free space representation from sensor data

Rays beginning at the sensor and ending at a hit point are used to define a robot's observations of free space. Due to the nature of range observations radiating at fixed angles from a sensor origin, the free space around the sensor will be more densely represented with data than the surrounding obstacles observed at various hit points. Additionally, with their continuous representation, the free observations along a ray are likely to be closer to a given query point than the single hit point at the end of the ray. Therefore, overestimation of free space occurs if there are no checks in place to avoid it.

In particular, glancing rays, occurring nearly tangential to an obstacle surface, can be a problematic cause of overestimation in predictive occupancy mapping. When using such rays to estimate free space, it is possible to erroneously define free space inside obstacles, as seen at the top of Fig. 3. On the left of the Figure we can see the true environment, and at right some of the resulting errors introduced into an occupancy map. When rays are nearly tangential to an obstacle wall, part of the wall is assumed to be free space in the resulting map, even with a small standoff from the ray. This is caused by unbiased inference over the contents of the sensor ray, which weighs free space more heavily than occupied space.

To reduce overestimation of free space, the proposed algorithm reduces the length of a ray based on nearby hits. To reduce the ray length, we search within the ray's region of influence for

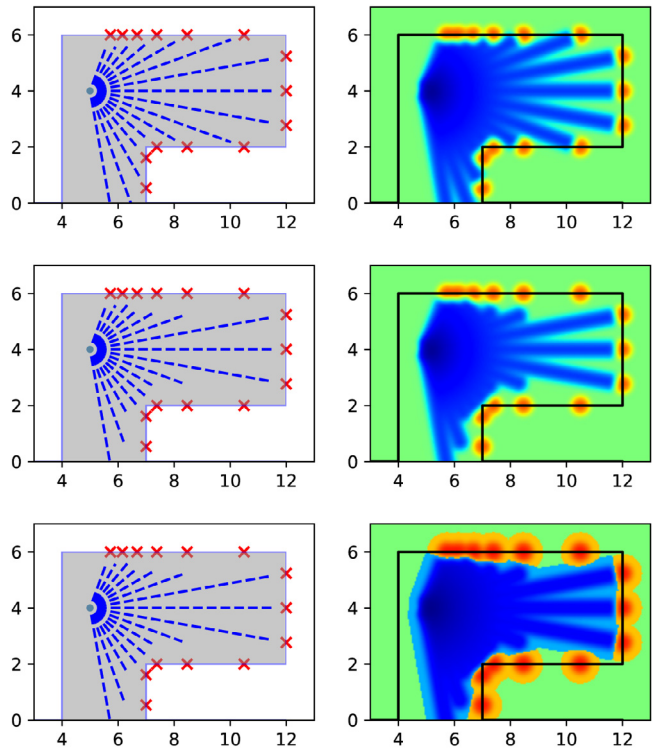


Fig. 3. An illustrative comparison of occupancy representations. The left shows training data while the right shows the inference result. From top to bottom: (1) point-to-line distance kernel evaluation used in BGKOctoMap-L, (2) reducing the length of glancing rays to minimize overestimation of free space, (3) combining reduced glancing rays and the proposed occupancy predictions from Eq. (18).

any hits that occur closer to the sensor origin than the hit at the end the ray. If a hit is found, the ray length is shortened to that distance. While this is an exhaustive search, there are a few ways to minimize computational complexity.

By reducing the length of glancing rays passing nearly tangent to nearby obstacles, we are able to minimize the overestimation of free space when the available data is sparse. This procedure complements our proposed Eq. (18) for updating occupancy probabilities, which forces occupancy probability toward the free and occupied classes. Fig. 3 shows the resulting estimation of occupancy probability from a single range scan. At the bottom of Fig. 3, the border between free and occupied is sharp, well-defined, and can quickly adapt to the arrival of new observations, unlike previous mapping methods. Map variance is not included in this figure, as the map only visualizes the occupancy probabilities.

5. Algorithm summary

Two key updates to BGKOctoMap-L have been proposed, and their implementation is discussed in this section. The first reduces the length of glancing rays, augmenting the training data that will be subsequently used to infer occupancy. This method requires the initial training data $\mathcal{X}, \mathcal{Y}$, which contain the positions and occupancy of the raw sensor data. When $y_i \in \mathcal{Y}$ is equal to 1, the corresponding state $x_i \in \mathcal{X}$ is considered occupied, and when $y_i = 0$, the corresponding state is free.

The ray-shortening procedure is summarized in Algorithm 1. To shorten free space rays $\overrightarrow{ox_i}$, which are defined between the sensor origin and a hit, we must compare all rays against all hit points in the current point cloud on which the map is based. After computing the nearest point on the ray to a given occupied point

Algorithm 1 Reducing Free-Space Training Data

```
1:  Input: Training data: 𝒳, 𝒴;
2:  for each (x_i, y_i) ∈ (𝒳, 𝒴) do
3:      if y_i = 0 then
4:          Use ox⃗_i ray
5:          for each (x_i', y_i') ∈ (𝒳, 𝒴) do
6:              if y_i' = 1 then
7:                  Compute x_free from Eq. (9) using x_i' as x_*
8:                  d = ‖x_free − x_i'‖
9:                  if d < l from Eq. (11) then
10:                     x_i ← reduce_ray(ox⃗_i, x_i')
```

Algorithm 2 Bayesian Generalized Kernel (BGK) Inference

```
1:  Input: Training data: 𝒳, 𝒴; Query point: x_*
2:  Initialize: α_*, β_* ←≈ 0
3:  for each (x_i, y_i) ∈ (𝒳, 𝒴) do
4:      if y_i = 0 then
5:          Compute x̂_i ← x_free from Eq. (9)
6:      else
7:          x̂_i ← x_i
8:      k_i ← k(x_*, x̂_i)  Sparse kernel, Eq. (11)
9:      α_* ← α_* + k_i y_i
10:     β_* ← β_* + k_i(1 − y_i)
11: return α_*, β_*
```

x_i' on line 7, the minimum distance between the ray and point is calculated. If the distance calculated is less than the influence distance l, then there is an overlap between the ray and occupied point, which would ordinarily result in overestimation of free space. To mitigate overestimation of free space, the ray is reduced each time a new occupied point is found within its region of influence. This process occurs repeatedly, where the ray and thus its region of influence is updated each time $d < l$, ensuring that the ray is reduced as far as necessary and the search order does not matter.

By only changing the input data and the way we process cell data, we are able to continue to use the same inference procedure as BGKOctoMap-L, summarized in Algorithm 2. The objective to compute the posterior parameters given observations $\mathcal{X}$ and $\mathcal{Y}$ remains the same. However, the input data has been altered per Alg. 1, reducing the length of the rays where needed to avoid free-space overestimation. Also unlike BGKOctoMap-L, the posterior data α and β are updated per Eqs. (14)–(19) to predict both occupancy and the variance of occupancy.

As described earlier, the proposed BGKOctoMap-LV algorithm will use Eq. (18) to solve for expected occupancy and (19) to solve for variance. While the equations themselves differ, the structure of the algorithm remains the same as the BGKOctoMap-L algorithm. However, these changes alter the way cell states are defined. Cells are defined as unknown if the expected occupancy lies within the range between the occupied and free thresholds, typically a range of $[0.5 - \epsilon, 0.5 + \epsilon]$. By pushing the occupancy values to their extremes, we can predict the contents of newly discovered cells early and never revert back to unknown. Using Eq. (19) for variance, the uncertain state was created by defining a variance threshold. When the variance is above that threshold, regardless of expected occupancy, the state is considered uncertain.

Table 1
List of parameters which are the same in all tests performed.

Parameter	Description	Value	Algorithms
occ_thresh	Occupancy Threshold	0.7	All
free_thresh	Free Threshold	0.3	All
var_thresh	Variance Threshold	0.02 0.15 0.2	GP BGK-L BGK-LV
α_0, β_0	Beta Prior Parameters	0.001	BGK-L/BGK-LV
w_{MIN}	Unknown Prior Parameter	0.001	BGK-LV

Table 2
List of parameters used in the simulated mapping tests.

Parameter	Description	Value	Algorithms
resolution	Resolution	0.1 m	All
ds_resolution	Down-sample Resolution	0.1 m	All
block_depth	Block Depth	1	All
free_resolution	Free Sample Resolution	0.1 m 0.85 m	GP/BGK-LV BGK-L
l	Kernel Length	1.0 m 0.2 m	GP BGK-L/BGK-LV
σ_0	Kernel Scale	1.0 0.1	GP BGK-L/BGK-LV

6. Experimental results

Using OctoMap [8] as a baseline, we compare our proposed algorithm, BGKOctoMap-LV,[1] against other inference-based maps including Gaussian Process OctoMaps (GPOctoMap) [11], the original learning-aided Bayesian generalized kernel map (BGKOctoMap-L) [1], and a newer framework for adaptive kernel inference (AKIMap) [30]. While these algorithms differ in many ways, a few key components were kept constant throughout our tests. The range of *known* occupancy states was separated into two regions, known free and known occupied. Known free was declared to be the region of occupancy where $p = \hat{\mathbb{E}}[m_j] \leq 0.3$ while the known occupied region was defined by $0.7 \leq p = \hat{\mathbb{E}}[m_j]$ for all algorithms.

GPOctoMap, BGKOctoMap-L and our proposed variant BGKOctoMap-LV all use variance thresholds to define unknown and uncertain occupancy states. Due to the uniqueness in how variance is treated among each separate algorithm, they all required different values for their thresholds. The values used for our tests can be found in Table 1. A few other parameters were kept constant during all our testing for BGKOctoMap-L and the proposed BGKOctoMap-LV. Namely, the beta prior parameters α_0 and β_0 were assigned very small values to assume virtually no prior information about the environment. To ensure fairness in the results and follow the same process, w_{MIN} was assigned a very small value as well, resulting in minimal usage of the unknown class when evaluating the accuracy of map predictions.

Similar to past comparisons [1,11,12], two simulated datasets were utilized with known ground truth to check each algorithm for accuracy. The first environment was "structured" with predominantly rectangular features, while the second environment was "unstructured" and contained cylindrical objects surrounded by a rectangular perimeter. Both of these datasets have virtually no overlapping point cloud data, which challenges inference-based algorithms using sparse data.

However, accuracy is not the only way to compare inference-based mapping algorithms. Rather, how effective the inference-based algorithms are at parsing sparse data into a reasonable representation of the environment should also be checked. To do so, we used data captured from our own Jackal ground robot of the Strataspace mine in Louisville, KY using a VLP-16 Lidar. Lidar sensors result in very dense data, therefore to rigorously compare inference capabilities, we artificially sparsified the data by downsampling the incoming point clouds.

Each test was performed on a desktop with an 8-core 3.60 GHz Intel i9 CPU with 16 threads running Ubuntu Linux. Every algorithm was run via the Robot Operating System (ROS) [3] and utilized the Point Cloud Library (PCL) [2].

[1] The code for this paper is implemented in a branch of the Learning-aided 3D Mapping Library, available at https://github.com/RobustFieldAutonomyLab/la3dm/tree/feature/bgklv.

6.1. Simulated data

Both the structured and unstructured environments are $10.0 \times 7.0 \times 2.0$ m in size, with twelve non-overlapping range scans provided as input. Only the cells that are seen in Figs. 4 and 5 were used to evaluate the accuracy of each algorithm. The colored cells indicate the known occupied state while the gray cells indicate the known free state. Both unknown and uncertain cells were not considered for the purpose of plotting the receiver operating characteristic (ROC) curves, to accurately capture this mapping scenario as a binary classification problem.

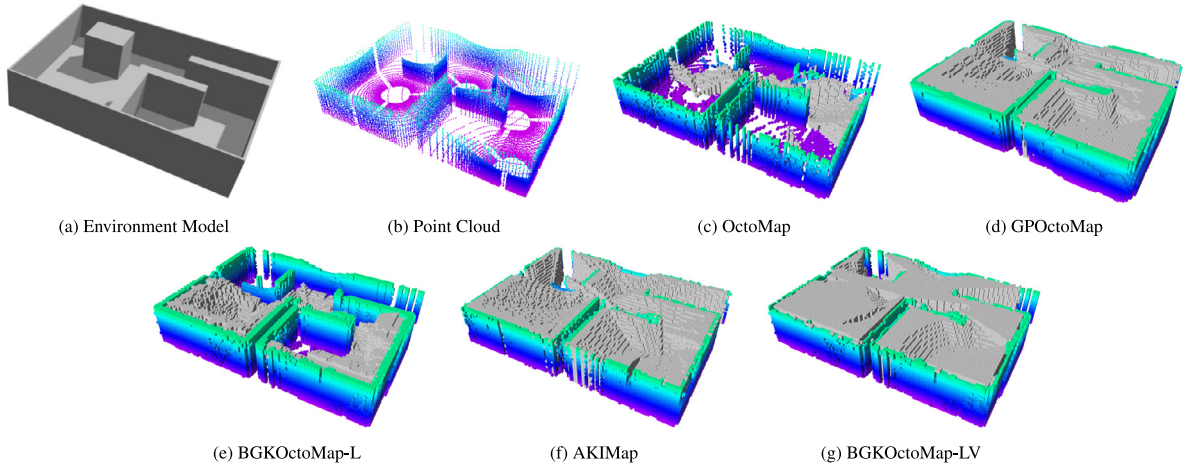
To ensure consistency in the data, each algorithm used 0.1 m resolution without any pruning. Various other parameters used specifically for this simulated mapping test can be found in Table 2. Each of the parameters used was either tuned to approximately optimize the performance of each algorithm, or were defined by their original authors as the default values. In particular, the free sample resolution used by BGKOctoMap-L was set to maximize the known free cells, while minimizing the losses incurred by the known occupied cells when the data conflicts. Both the GPOctoMap and BGKOctoMap-LV algorithms were able to use any value less than or equal to the kernel length values for their free sample resolutions. The kernel length and scale values used were the default values for that respective resolution. As the BGKOctomap-LV algorithm was based on BGKOctomap-L [1], we used the same default values for our kernel length and scale. These were chosen based on previous experiments, where larger kernel lengths led to excessive homogeneity within the maps, while the scale changed how quickly the unknown cells disappear.

Receiver operating characteristic (ROC) curves were used to check each algorithm for its accuracy with respect to each environment. Only the known free and known occupied cells shown in Figs. 4 & 5 were used in the ROC curves, resulting in Figs. 6 & 7 respectively. The ROC curve of the structured environment shows how the proposed algorithm BGKOctoMap-LV has comparable accuracy to new techniques such as AKIMap and makes improvements on the prior BGKOctoMap-L. The same conclusion can be drawn from the ROC curve describing the unstructured environment. Thus we believe that the accuracy of the proposed algorithm is sufficient for use in the field.

6.2. Lidar mapping experiments

We collected VLP-16 lidar data from the Strataspace mine in Louisville, KY to compare predictive mapping algorithms.[2] The subset of the data we selected for these experiments resulted in a $130 \times 130 \times 7$ m environment. To necessitate the use of predictive mapping, the point cloud data was sparsified, downsampling

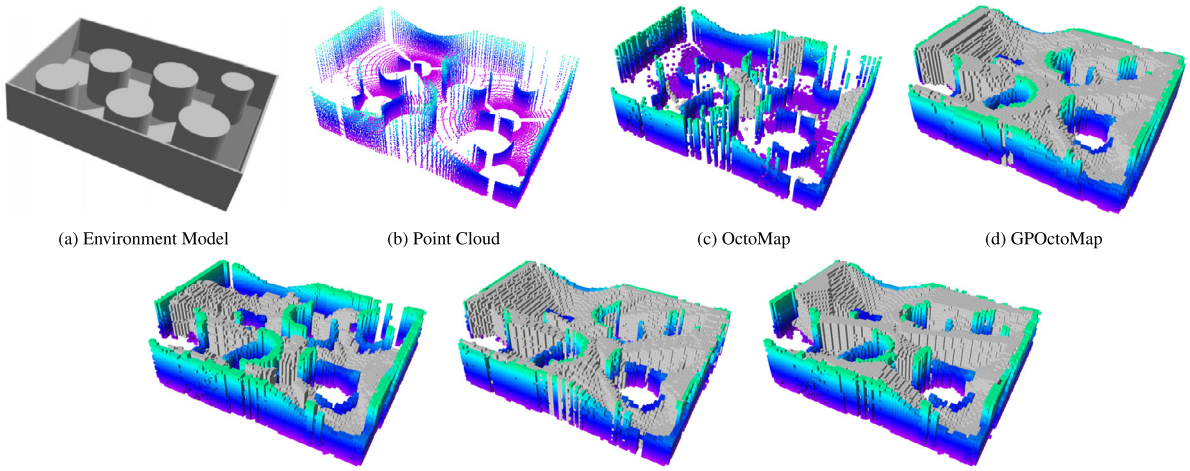
[2] A video showing the process of each mapping algorithm performing on this data is available at https://youtu.be/VTUc4lQ2en4.



(a) Environment Model (b) Point Cloud (c) OctoMap (d) GPOctoMap

(e) BGKOctoMap-L (f) AKIMap (g) BGKOctoMap-LV

Fig. 4. Results from using the same Structured map used in [1,30] to test mapping inference on rectangular obstacles.



(a) Environment Model (b) Point Cloud (c) OctoMap (d) GPOctoMap

(e) BGKOctoMap-L (f) AKIMap (g) BGKOctoMap-LV

Fig. 5. Results from using the same Unstructured map used in [1,30] to test mapping inference on curved obstacles.

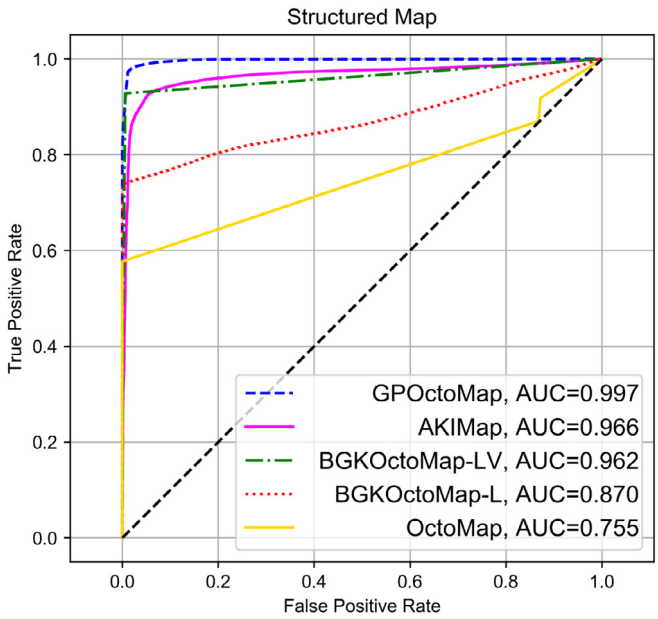


Fig. 6. ROC curves for the Structured Map.

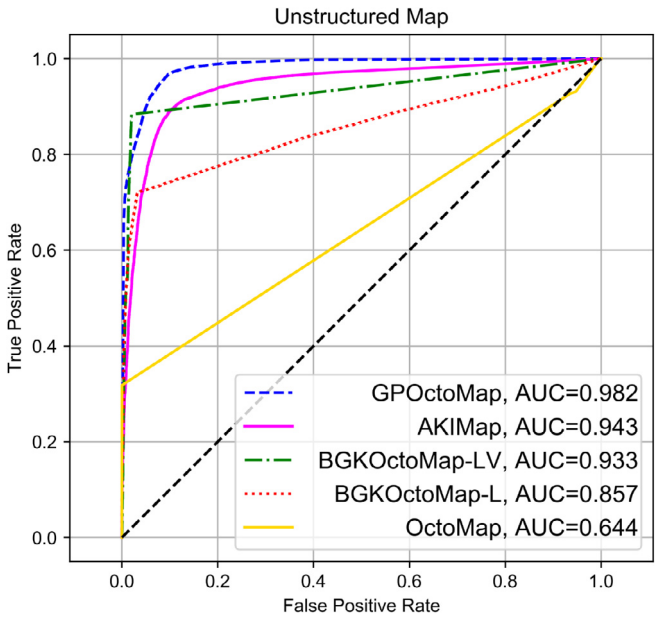


Fig. 7. ROC curves for the Unstructured Map.

Table 3
List of parameters used in the mine tests.

Parameter	Description	Value	Algorithms
resolution	Resolution	0.2 m	All
ds_resolution	Down-sample Resolution	0.5 m	All
max_range	Maximum Sensing Range	30 m	All
block_depth	Block Depth	4 5 6	GP BGK-L BGK-LV
free_resolution	Free Sample Resolution	0.1 m 6.5 m	GP/BGK-LV BGK-L
l	Kernel Length	1.0 m 0.6 m	GP BGK-L/BGK-LV
σ_0	Kernel Scale	1.0 0.1	GP BGK-L/BGK-LV

the original scans to 0.5 m leaves. Without downsampling, the data was so dense that all maps performed similarly.

While map accuracy is important, it often ignores some defining features of a "good" map. For instance, a map may have very high accuracy, but only define two or three cells as occupied or free. Other factors need to be taken into account when considering what defines a "good" map, such as coverage. In a live experiment, point cloud data is often gathered very quickly and not every scan can be utilized if a mapping algorithm wishes to perform in real-time. However, not all scans are necessary if there is sufficient overlap. Therefore the frequency at which a map updates is important to mapping effectively. Another useful metric is how much pruning has occurred. Funk et al. in [10] prove that successful pruning can be used to speed up motion planning. When more pruning occurs, fewer cells are used to cover the same volume.

The data structure used by GPOctoMap, BGKOctoMap-L and BGKOctoMap-LV has limitations on its pruning capabilities. "Blocks" are defined as a set of neighboring cells, where block depth is a parameter used to define how many cells exist within a given block. Each increase in block depth is equivalent to another level of possible pruning, however there is also increased computational complexity as block depth increases. Due to the direct link between block depth and how much pruning can occur, each algorithm was tested at differing block depths over the same subset of the mine data used to compare all the algorithms later. Comprehensive testing resulted in the data shown in Table 4, where the largest feasible block depth is emboldened and used in the final experimental results. The lower update frequencies at block depths one and two are a result of being unable to fully utilize the multi-threaded functionality of each algorithm at those depths.

Besides block depth, which was varied for these experiments, the remaining parameters were kept constant; all values are summarized in Table 3. The resolution was set to 0.2 m due to the size of the environment used. That resolution size resulted in our choice for down-sample resolution, which affected the kernel characteristic length used by BGKOctoMap-L and BGKOctoMap-LV. The kernel length dictates how far away from a cell we should consider data to be influential, thus the kernel length must be larger than the down-sample resolution. However, the value chosen for the two BGKOctoMap algorithms was still not as large as the one used by the GPOctoMap algorithm, which was left at the default value.

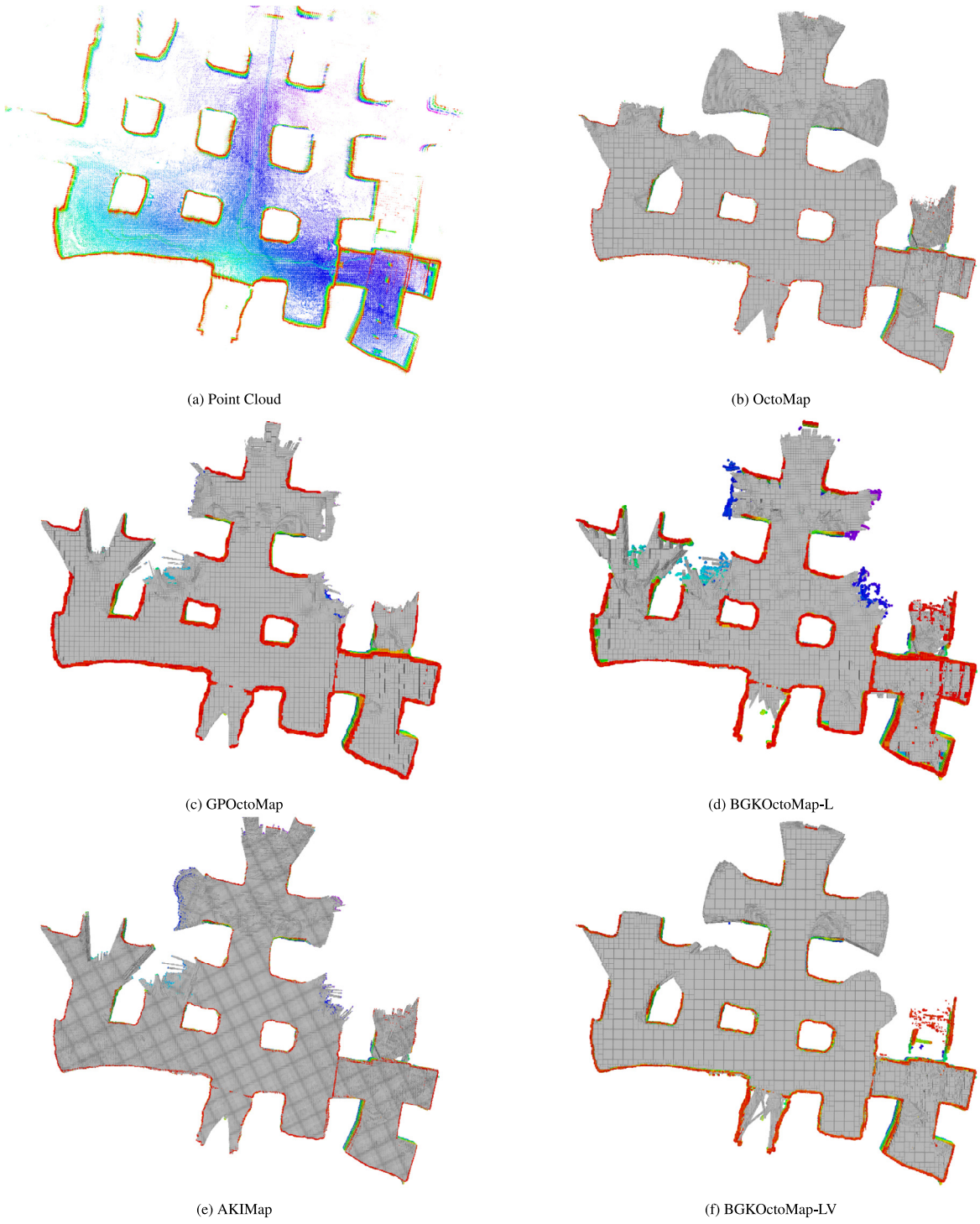
OctoMap and AKIMap do not explicitly use block depth, thus they were parameterized separately. AKIMap has no pruning system in place and was left that way for these tests. OctoMap does have a pruning function, but it can perform that function without the data structure built via block depth used by the GP and BGK algorithms, which allowed OctoMap to heavily prune free space in this experiment without depth parameterization.

Table 4
Varying block depth to decide which to use for lidar mapping experiments.

Algorithm	Block depth	Total coverage (m^3)	Average cell size	Average update frequency (Hz)
OctoMap	N/A	**38382**	**0.0296**	**3.27**
Akimap	N/A	**35364**	**0.0080**	**2.24**
GPOctoMap	1	28827	0.0080	0.91
	2	37083	0.0434	2.39
	3	42699	0.0872	3.73
	4	**43749**	**0.0806**	**2.63**
	5	18034	0.0259	0.07
	6	1031	0.0144	<0.01
BGKOctoMap-L	1	7862	0.0080	1.19
	2	18153	0.0253	3.32
	3	25229	0.0331	5.54
	4	31386	0.0407	5.05
	5	**35956**	**0.0488**	**2.73**
	6	32159	0.0444	0.59
BGKOctoMap-LV	1	33796	0.0080	0.13
	2	38451	0.0342	0.61
	3	38928	0.0508	0.86
	4	38932	0.0536	0.83
	5	38911	0.0543	0.72
	6	**38326**	**0.0544**	**0.51**

Coverage is the sum total of the volume defined by known free and known occupied cells. Average cell size is the coverage divided by the total number of cells defined by known free and occupied. The average update frequency was calculated by finding the total number of maps published by each algorithm over the course of the dataset, and dividing by the full duration of the dataset. While all algorithms were fastest at block depth three, more pruning occurred at larger block depths. Therefore, the metric for choosing which block depth to use for each algorithm was based primarily on total coverage. GPOctoMap had the most coverage at block depth four, while BGKOctoMap-L had the most coverage at block depth five. The coverage decreased when the algorithms were too slow to process enough lidar scans, resulting in missed data. BGKOctoMap-LV used block depth 6 due to the increase in average cell size with a minimal loss of coverage.

Most of these algorithms operated successfully around the 2–3 Hz range while the proposed algorithm managed to succeed at $>$ 0.5 Hz. While more processing is required, BGKOctoMap-LV achieved accurate inference over the incoming data. Our proposed algorithm did not see much of a decrease in processing time as block depth increased, because incoming data was processed using a different method which is no longer an exponential function of block depth, which differs from previous algorithms. GPOctoMap and BGKOctoMap-L used the size of a block to partition incoming data into sets, which became superfluous at larger block depths. For smaller block depths, that system improved processing speeds, which was necessary for GPOctoMap to run in real-time by utilizing multi-threading capabilities. By comparison, the proposed BGKOctoMap-LV algorithm defines sets of incoming data for each map cell, regardless of block size, which allows us to successfully utilize larger block depths. The difference can be seen by the drop from 2.63 Hz to 0.07 Hz as GPOctoMap goes from block depth four to five. Similar behavior occurs as BGKOctoMap-L goes from block depth five to six, where the average update frequency drops from 2.73 Hz to 0.59 Hz. Our proposed algorithm, BGKOctoMap-LV, does not change as drastically as block depth increases, with the worst case from block depth five to six resulting in a 0.21 Hz decrease in processing speed.



(a) Point Cloud (b) OctoMap

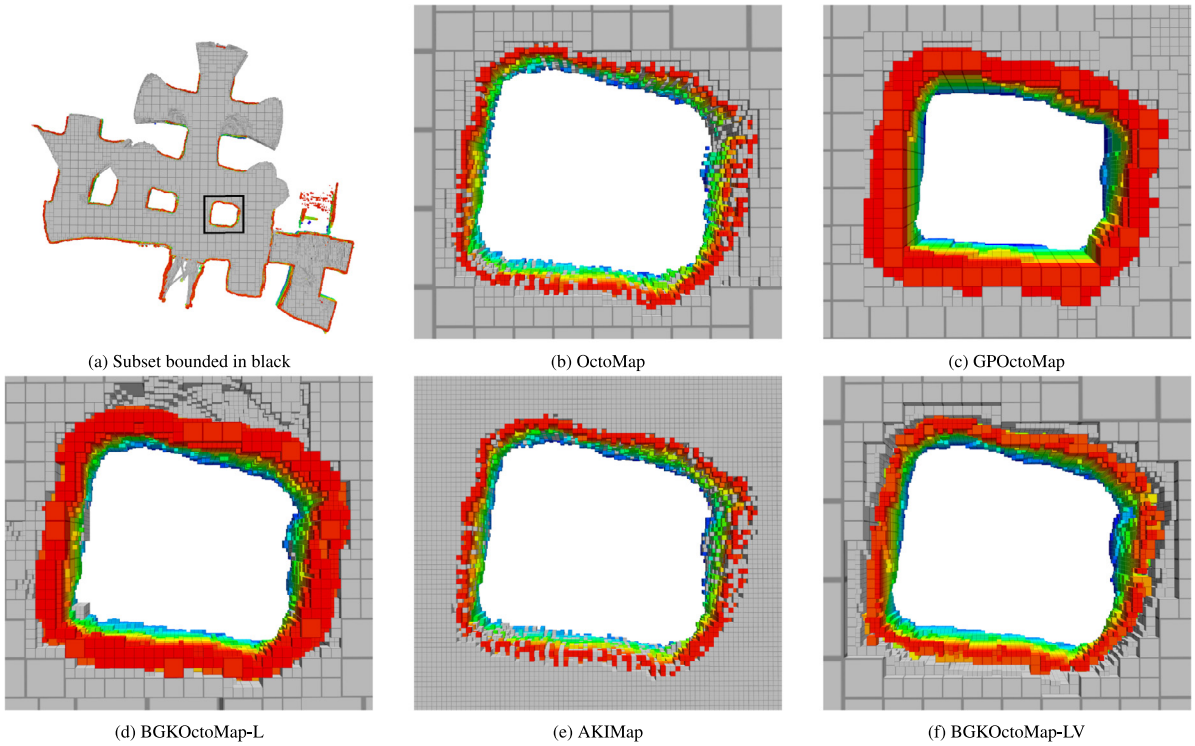
(c) GPOctoMap (d) BGKOctoMap-L

(e) AKIMap (f) BGKOctoMap-LV

Fig. 8. Each mapping algorithm was tested using the same lidar dataset collected from an underground limestone mine. Lidar odometry was employed while mapping to minimize drift. To help visualize the data, we only show data below z = 3 m. Gray cells indicate free space, while colored cells indicate occupied space. The color change of occupied cells indicates their height.

While GPOctoMap produced a very large average cell size at both block depth three and four, the data may be misleading. GPOctoMap performed with strong inference capabilities, producing a much smoother representation of the environment. When such smoothing occurs, very few smaller cells remain as most are pruned, being merged together with their similar neighbors. This behavior actually reduces the accuracy of certain real-world features, such as bumpy surfaces or protrusions. However, those same features would decrease the average cell size of a completely accurate occupancy grid map. Therefore, even though it seems GPOctoMap is vastly outperforming at lower block depths, the map produced is likely to under-perform in path planning exercises compared to the larger block depths used by BGKOctoMap-L and BGKOctoMap-LV.

Using the parameters shown in Table 3, each algorithm was tested over the data visualized in Fig. 8a. The point cloud data was not range-limited, showing more than what was utilized by each mapping algorithm. Color is used to indicate height, with red



(a) Subset bounded in black (b) OctoMap (c) GPOctoMap

(d) BGKOctoMap-L (e) AKIMap (f) BGKOctoMap-LV

Fig. 9. For closer inspection, we chose a subset of the mine data that represents the qualitative performance of each mapping algorithm with our predetermined parameters, showing how each map defines occupied space, free space, and the boundary between them. Too much inference results in overestimation, while too little results in an incomplete representation of the wall.

indicating the top and blue the bottom. For ease of visualization, each point was given a non-zero size. All of the images in Fig. 8 were truncated at a height of 3m to better illustrate how each algorithm inferred free space. However, when testing each block depth, we did not impose a cutoff height.

Each algorithm accurately captured the global characteristics of the mine, with some aliasing occurring in the AKIMap image due to its lack of pruning. By mapping everything at the finest available resolution, AKIMap produced thin obstacle boundaries and far too many map cells to permit rapid path planning. OctoMap performed one task that was implemented into the proposed BGKOctoMap-LV algorithm but does not exist in the inference-based competitors, which is utilizing rays up to max range from obstacles observed beyond the max range. By utilizing every ray, OctoMap was able to successfully fill in free space quite effectively even with sparse data. GPOctoMap was able to capture the majority of free space with the largest cell size available at the respective block depth used. This is due to its aggressive inference capabilities, which also resulted in an inflated representation of obstacle boundaries. BGKOctoMap-L also resulted in inflated obstacle boundaries, however we believe the cause to be different. The free resolution of BGKOctoMap-L used for this experiment was quite large, at 6.5 m. Free space would be significantly over-represented if it was not directly curbed. By using such a large value, we were able to reduce the overestimation of free space while still acquiring an accurate map, which can be seen in Fig. 8d. However, a consequence of curbing free space was the overestimation of occupied space.

The proposed BGKOctoMap-LV produced a very rich and well-pruned free space representation of the mine data. However, this result is not unique to BGKOctoMap-LV. Therefore, we will zoom in on a subset of the map to better understand the boundary between occupied and free cells. In Fig. 9, we inspect a single pillar within the mine that enables us to more clearly visualize the differences between all algorithms. OctoMap has gaps in the occupied data due to the lack of inference applied to the range data. AKIMap performed similarly, likely due to limits on the range of influence of sensor data, which was based on its fine resolution. Both GPOctoMap and BGKOctoMap-L had similarly inflated representations of occupied space, however, GPOctoMap added thickness atop free space observations, while BGKOctoMap-L expanded into the free space due to pulling the free space back from the walls. The proposed algorithm BGKOctoMap-LV resulted in well-defined free space and accurate estimation of occupied space. There are some small gaps in the data between known free and known occupied cells due to the uncertainty caused by high variance, however those gaps would be at most a min-resolution cell thick. The proposed algorithm is able to accurately represent the unambiguously free and occupied regions of the workspace, while capturing uncertainty on the boundary between them.

Reducing the inflated representation of occupied space in BGKOctoMap-LV may have unexplored benefits. Uneven surfaces, such as those caused by furniture or certain wall features, are expected to be captured accurately with less generalization. One example to consider is a thin wall that is seen from both sides. In the BGKOctomap-L case, we would expect the wall to be estimated significantly thicker, as overestimation of occupancy occurs from the wall surface toward the sensor on each side of the wall. GPOctomap pushes the overestimation behind the surface, resulting in a more accurate yet still excessively thick wall, where newer data may fix the currently viewed side of the wall at the cost of expanding the reverse side. BGKOctomap-LV minimizes this overestimation, resulting in the most accurate representation of a thin wall seen from both sides. Thin walls are very common within indoor environments, which indicates the frequency and potential relevance of this scenario.

7. Conclusion and future work

In this paper, we have proposed BGKOctoMap-LV, enhancing the BGKOctoMap-L inference-based occupancy grid mapping algorithm to accurately capture known free, known occupied, unknown, and uncertain states. The framework's accuracy was confirmed using simulated mapping scenarios, comparing against recently proposed algorithms in inference-based occupancy grid mapping, while real lidar data was also used to qualitatively illustrate their different outcomes intuitively.

We expect a future version of this algorithm to prove useful for exploration and inspection in 3D environments. Separating *uncertain* from *unknown* will allow us to develop new methods of exploration using inference based mapping that focus exclusively on the *unknown* classification and disregard those map cells considered *uncertain*, for the purpose of efficiently completing the exploration mission. A different approach could utilize both states for a two step exploration and interrogation method. With the ability to drive down uncertainty without depending on the unknown classification, mobile robot exploration algorithms could interrogate structures until they have been completely observed, before continuing to explore.

We believe there are a few more avenues for improving this algorithm's performance, and there needs to be further examination of the unknown classification. One such method would be a varying kernel length based on how far data is from the sensor. Due to the radial nature of sensor data, it will be corrupted by larger errors the farther the data is from the sensor. Therefore, we believe we can create more accurate maps if we adjust the kernel characteristic length for data that lies farther from the sensor. The unknown classification is expected to play a larger role in noisy mapping scenarios, where the boundary between occupied and free is unclear from a single sensor observation. One other update would be to favor new incoming data over older data. This has been utilized previously [33,34] to remove dynamic obstacles from static maps, and would potentially improve the accuracy and adaptability of this framework.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported in part by the National Science Foundation, USA, grant number IIS-1723996. We also thank Yewei Huang and Paul Szenher for collecting the Strataspace mine dataset and obtaining its underlying localization solution.

Appendix A. Supplementary data

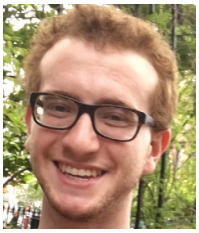
Supplementary material related to this article can be found online at https://doi.org/10.1016/j.robot.2022.104077.

References

[1] K. Doherty, T. Shan, J. Wang, B. Englot, Learning-aided 3-D occupancy mapping with Bayesian generalized kernel inference, IEEE Trans. Robot. 35 (4) (2019) 953–966.
[2] R. Rusu, S. Cousins, 3D is here: Point cloud library (PCL), in: Proceedings of the IEEE International Conference on Robotics and Automation, 2011.
[3] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: An open-source robot operating system, in: IEEE ICRA Workshop on Open Source Software, 2009.
[4] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. 3 (2004) 2149–2154.
[5] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, J.J. Leonard, Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age, IEEE Trans. Robot. 32 (6) (2016) 1309–1332.
[6] S. Izadi C.V. Nguyen, D. Lovell, Modeling kinect sensor noise for improved 3D reconstruction and tracking, in: 2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, 2012, pp. 524–530.
[7] A. Elfes, Using occupancy grids for mobile robot perception and navigation, Computer 22 (6) (1989) 46–57.
[8] A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, W. Burgard, Octomap: An efficient probabilistic 3D mapping framework based on octrees, Auton. Robots 34 (3) (2013) 189–206.
[9] D. Duberg, P. Jensfelt, UFOMap: An efficient probabilistic 3D mapping framework that embraces the unknown, IEEE Robot. Autom. Lett. (2020) 6411–6418.
[10] N. Funk, J. Tarrio, S. Papatheodorou, M. Popović, P.F. Alcantarilla, S. Leutenegger, Multi-resolution 3D mapping with explicit free space representation for fast and accurate mobile robot motion planning, IEEE Robot. Autom. Lett. 6 (2) (2021) 3553–3560.
[11] J. Wang, B. Englot, Fast, accurate Gaussian process occupancy maps via test-data octrees and nested Bayesian fusion, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2016, pp. 1003–1010.
[12] K. Doherty, J. Wang, B. Englot, BayesIan generalized kernel inference for occupancy map prediction, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2017, pp. 3118–3124.
[13] A. Melkumyan, F. Ramos, A sparse covariance function for exact Gaussian process inference in large datasets, Proc. Int. Joint Conf. Artif. Intell. Organ. 9 (2009) 1936–1942.
[14] M.G. Jadidi, J.V. Miro, G. Dissanayake, Warped Gaussian processes occupancy mapping with uncertain inputs, IEEE Robot. Autom. Lett. 2 (2) (2017) 680–687.
[15] G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters, IEEE Trans. Robot. 23 (1) (2007) 34–46.
[16] P. Sodhi, B.J. Ho, M. Kaess, Online and consistent occupancy grid mapping for planning in unknown environments, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 7879–7886.
[17] S.T. O'Callaghan, F.T. Ramos, Gaussian process occupancy maps, Int. J. Robot. Res. 31 (1) (2012) 42–62.
[18] S. Kim, J. Kim, Gpmap: A unified framework for robotic mapping based on sparse Gaussian processes, in: Proceedings of the 9th International Conference on Field and Service Robotics, 2013.
[19] S. Kim, J. Kim, Recursive Bayesian updates for occupancy mapping and surface reconstruction, in: Proceedings of the Australasian Conference on Robotics and Automation, 2014.
[20] L. Gan, R. Zhang, J.W. Grizzle, R.M. Eustice, M. Ghaffari, BayesIan spatial kernel smoothing for scalable dense semantic mapping, IEEE Robot. Autom. Lett. 5 (2) (2020) 790–797.
[21] W.R. Vega-Brown, M. Doniec, N.G. Roy, Nonparametric Bayesian inference on multivariate exponential families, Adv. Neural Inf. Process. Syst. 27 (2014) 2546–2554.
[22] S.T. O'Callaghan, F.T. Ramos, Continuous occupancy mapping with integral kernels, in: Proceedings of the AAAI Conference on Artificial Intelligence, 2011, pp. 1494–1500.
[23] K.S. Shankar, N. Michael, MRFMap: Online probabilistic 3D mapping using forward ray sensor models, Robotics: Sci. Syst. (2020).
[24] F. Ramos, L. Ott, Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent, in: Proceedings of Robotics: Science and Systems, 2015.
[25] V. Guizilini, F. Ramos, Large-scale 3D scene reconstruction with Hilbert maps, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2016, pp. 3247–3254.
[26] V. Guizilini, F. Ramos, Learning to reconstruct 3D structures for occupancy mapping, in: Proceedings of Robotics: Science and Systems, 2017.
[27] S. Guo, N.A. Atanasov, Information filter occupancy mapping using decomposable radial kernels, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2019, pp. 7887–7894.
[28] A.-A. Agha-Mohammadi, E. Heiden, K. Hausman, G. Sukhatme, Confidence-rich grid mapping, in: Proceedings of the 18th International Symposium on Robotics Research, 2017.
[29] D. Fridovich-Keil, E. Nelson, A. Zakhor, Atommap: A probabilistic amorphous 3D map representation for robotics and surface reconstruction, in: IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 3110–3117.
[30] Y. Kwon, B. Moon, S.E. Yoon, Adaptive kernel inference for dense and sharp occupancy grids, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

[31] M.G. Jadidi, J.V. Miro, R. Valencia, J. Andrade-Cetto, Exploration on continuous Gaussian process frontier maps, in: Proceedings of the IEEE International Conference on Robotics and Automation, 2014, pp. 6077–6082.
[32] M.G. Jadidi, J.V. Miro, G. Dissanayake, Gaussian processes autonomous mapping and exploration for range sensing mobile robots, Auton. Robots 42 (2) (2018) 273–290.
[33] T. Teixeira, F. Mutz, K.S. Komati, L. Veronese, V.B. Cardoso, C. Badue, T. Oliveira-Santos, A.F. De Souza, Memory-like map decay for autonomous vehicles based on grid maps, 2018, arXiv preprint, arXiv:1810.02355.
[34] S. Macenski, D. Tsai, M. Feinberg, Spatio-temporal voxel layer: A view on robot perception for the dynamic world, Int. J. Adv. Robot. Syst. 17 (2) 2020.

Erik Pearson received a Bachelor of Engineering in Mechanical Engineering and a Bachelor of Science in Mathematics from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2015. He is currently a Ph.D. student in the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA.



Kevin Doherty received a Bachelor of Engineering in Electrical Engineering from Stevens Institute of Technology, Hoboken, NJ, USA, in 2017. He is currently a Ph.D. student in the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA, USA, and a member of the MIT-Woods Hole Oceanographic Institute Joint Program in Applied Ocean Science and Engineering.

Brendan Englot received S.B., S.M., and Ph.D. degrees in Mechanical Engineering from Massachusetts Institute of Technology, Cambridge, MA, USA, in 2007, 2009, and 2012, respectively.

He was a research scientist with United Technologies Research Center, East Hartford, CT, USA, from 2012 to 2014. He is currently an Associate Professor with the Department of Mechanical Engineering, Stevens Institute of Technology, Hoboken, NJ, USA. His research interests include motion planning, localization, and mapping for mobile robots, learning-aided autonomous navigation, and marine robotics. He is the recipient of a 2017 National Science Foundation CAREER award and a 2020 ONR Young Investigator Award.