Bayesian Generalized Kernel Inference for Terrain Traversability Mapping

Tixiao Shan, Jinkun Wang and Brendan Englot
Department of Mechanical Engineering
Stevens Institute of Technology
{tshan3,jwang92,benglot}@stevens.edu

Kevin Doherty
MIT Computer Science &
Artificial Intelligence Lab
kdoherty@mit.edu

Abstract: We propose a new approach for traversability mapping with sparse lidar scans collected by ground vehicles, which leverages probabilistic inference to build descriptive terrain maps. Enabled by recent developments in sparse kernels, Bayesian generalized kernel inference is applied sequentially to the related problems of terrain elevation and traversability inference. The first inference step allows sparse data to support descriptive terrain modeling, and the second inference step relieves the burden typically associated with traversability computation. We explore the capabilities of the approach over a variety of data and terrain, demonstrating its suitability for online use in real-world applications.

Keywords: Traversability mapping, Range sensing, Autonomous navigation

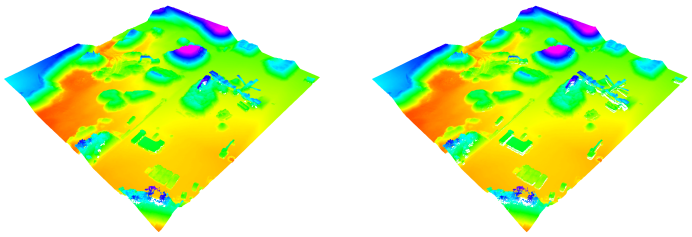
1 Introduction

Great efforts have been devoted to achieving autonomy for unmanned ground vehicles (UGVs), and an accurate representation of the environment is an essential prerequisite for such autonomy. Mapping methodologies for 2D planar UGV navigation [1], such as occupancy grid mapping, have achieved great success. However, assuming the surrounding world is planar limits the capabilities of UGVs in many circumstances. Traversability mapping [2], which classifies regions of variable-height terrain to be traversable or non-traversable, becomes a useful tool for enabling autonomous navigation in 3D environments, as it can address rough terrain and the presence of complex structures. Cameras [3, 4], lidar [5, 6], or a combination of the two [7, 8, 9] are typically utilized for mapping 3D terrain. Although vision-based methods have advantages in offering dense coverage, their sensitivity to illumination change may make captured data unreliable. On the other hand, lidar will function even at night, offering long-range visibility and a wide aperture. Therefore, this paper focuses on using 3D lidar to support real-time traversability mapping.

A typical traversability analysis derives a set of features over the available model of the terrain. Such features may include the slope and roughness of the terrain in different directions, and the capabilities of the robot also play an important role in the analysis, as they determine appropriate thresholds of such features with respect to traversability. Using a prior terrain model that is represented by a point cloud map, a tensor voting process was implemented for extracting a geometric representation of a local patch, whose structure was used to judge traversability in [5]. Similarly, the traversability assessment in [6] is performed by computing the roughness for a point with respect to its neighboring points. Learning-aided traversability estimation methods are attracting more attention due to their success in environments with highly variable appearance or complex geometries. [10] introduced a reinforcement learning based traversability analysis method, which is trained on obstacles composed of pallets, and tested in natural forest environments. A semi-supervised learning method was used in [11] for traversability analysis, in which the underlying model is inferred from positive labeled training data that is acquired from traversable terrain. A normal distributions transform traversability mapping (NDT-TM) approach was introduced in [12]. A support vector machine (SVM) was used to distinguish traversable and non-traversable areas in densely forested terrain. A convolutional neural network was built in [13] to solve traversability classification problems. A local terrain patch was converted to an image and then used as input to the neural network.

Despite the success of these methods, applying traversability mapping to real-world online navigation is still nontrivial, as a prior map is not always available and a UGV may lack the required

2nd Conference on Robot Learning (CoRL 2018), Zürich, Switzerland.



(a) Ground truth (b) BGK elevation map

Figure 1: **Applying BGK elevation inference to a terrain dataset:** The ground truth (a) of this dataset [15] consists of 10593395 points. When 50% of the points are used as training data for BGK elevation inference, the resulting root-mean-square error (RMSE) is 0.0659m. The total computation time is 2.9s. This inferred elevation map is shown in (b). The RMSE of VHR [15] in this instance is 0.065m, and the overall computation time of VHR on equivalent hardware is 10.5s.

perceptual and/or computational capability to navigate robustly. Lidar-based traversability mapping methods often suffer from sparse data, which limits their ability to provide sufficient coverage to support autonomous navigation. To solve this problem, a terrain modeling process can be introduced to predict terrain height in unknown locations before traversability analysis. Gaussian process (GP) regression has been applied to estimate terrain height in locations not directly observed by a robot's range sensor [14]. However, the complexity of GPs has limited their use in real-time computation requiring incremental updates. A variational Hilbert regression (VHR) framework is proposed in [15] to generate accurate terrain models. VHR outperforms GPs in terms of both accuracy and efficiency. However, the parameter tuning required, along with a potentially costly iterative optimization, may limit its application to real-world online mapping tasks. Tested on the same dataset (Fig. 1), the proposed method in this paper is able to achieve equivalent mapping accuracy in 28% of the computation time, and without the use of multi-threading, when compared with VHR.

Specifically, we propose Bayesian generalized kernel (BGK) inference [16] for solving the traversability mapping problem, with the aid of sparse kernels [17]. We first apply BGK elevation inference to solve the sparse data problem encountered during terrain mapping. Then we relieve the typical computational burden by only performing traversability computations over the elevation data at selected locations. The traversability of locations elsewhere is estimated by BGK traversability inference. This framework enables us to perform online traversability mapping with sparse lidar data and hardware that is compatible with a small UGV. To the best of our knowledge, this is the first application of Bayesian generalized kernel inference to the problem of terrain mapping. The rest of the paper is organized as follows. Section II introduces Bayesian generalized kernel inference and the proposed framework in detail. Section III presents a set of experiments over a variety of real and simulated outdoor environments in which UGVs map their surroundings using lidar.

2 Bayesian Generalized Kernel Inference for Traversability Mapping

Here we define the traversability mapping problem and give the details of our solution. Given a sampled 3D point cloud, we first represent the environment as an elevation map. Finely and uniformly discretized planar "grid cells" on the ground are each assigned a height value. We then classify each cell as traversable or non-traversable. To solve this problem efficiently and precisely, we employ the Bayesian kernel inference method of Vega-Brown et al. [16] in two forms: regression, to obtain a dense elevation map m_e; and classification, to determine traversability map m_v.

2.1 Bayesian Generalized Kernel Inference

Given observations $\mathcal{D} = \{(\mathbf{x}_i, y_i)_{i=1:N}\}$, we seek to infer a probability distribution of a target value parameterized on the latent space Θ: $p(y^*|\mathbf{x}^*, \mathcal{D}) \propto \int p(y^*|\boldsymbol{\theta}^*)p(\boldsymbol{\theta}^*|\mathbf{x}^*, \mathcal{D})d\boldsymbol{\theta}^*$, where $p(\boldsymbol{\theta}^*|\mathbf{x}^*, \mathcal{D}) \propto \int_{\boldsymbol{\theta}_{1:N}} \prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}_i)p(\boldsymbol{\theta}_{1:N}, \boldsymbol{\theta}^*|\mathbf{x}_{1:N}, \mathbf{x}^*)d\boldsymbol{\theta}_{1:N}$ is the posterior distribution of the latent parameters associated with the target input. Unlike Gaussian Processes, which assume all parameters Θ are correlated, in Bayesian generalized kernel inference, parameters over the observation input are conditionally independent given target input $p(\boldsymbol{\theta}_{1:N}, \boldsymbol{\theta}^*|\mathbf{x}_{1:N}, \mathbf{x}^*) =$

$\prod_{i=1}^{N} p(\boldsymbol{\theta}_i|\mathbf{x}_i, \boldsymbol{\theta}^*, \mathbf{x}^*)p(\boldsymbol{\theta}^*|\mathbf{x}^*)$, which enables us to marginalize latent parameters $p(\boldsymbol{\theta}^*|\mathbf{x}^*, \mathcal{D}) \propto \prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}^*, \mathbf{x}^*, \mathbf{x}_i)p(\boldsymbol{\theta}^*|\mathbf{x}^*)$. If we further construct a smooth extended likelihood model, we are able to represent the posterior parameters as follows [16],

$$p(\boldsymbol{\theta}^*|\mathbf{x}^*, \mathcal{D}) \propto \prod_{i=1}^{N} p(y_i|\boldsymbol{\theta}^*)^{k(\mathbf{x}_i, \mathbf{x}^*)} p(\boldsymbol{\theta}^*|\mathbf{x}^*), \tag{1}$$

where $k(\cdot, \cdot)$ is a kernel function. The posterior can be exactly determined if the likelihood model is from the exponential family and the corresponding conjugate prior is assumed. Two applications of this inference model are employed in the paper.

2.2 Bayesian Kernel Elevation Regression

For elevation regression, we assume a Gaussian model $y \sim \mathcal{N}(\mu, \sigma^2)$ with fixed and known variances σ^2. The conjugate prior is also Gaussian $\mu \sim \mathcal{N}(\mu_0, \sigma^2/\lambda)$, where we define λ as a hyperparameter reflecting our confidence in the prior, with $\lambda = 0$ indicating no confidence, and $\lambda \to \infty$ indicating a state of perfect knowledge. Applying Eq. (1) with the above assumptions,

$$p(\mu^*|\mathbf{x}^*, \mathcal{D}) \propto \prod_{i=1}^{N} \exp\left\{ -\frac{1}{2}\frac{(y_i - \mu)^2}{\sigma^2} k(\mathbf{x}_i, \mathbf{x}^*) \right\} \exp\left\{ -\frac{1}{2}\frac{(\mu - \mu_0)^2}{\sigma^2}\lambda \right\}.$$

The mean and variance of the posterior parameters can be shown as

$$\mathbb{E}[\mu^*|\lambda, \mathcal{D}, \mathbf{x}^*] = \frac{\lambda\mu_0 + \sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}^*)y_i}{\lambda + \sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}^*)}, \quad \text{Var}[\mu^*|\lambda, \mathcal{D}, \mathbf{x}^*] = \frac{\sigma^2}{\lambda^*}, \tag{2}$$

where $\lambda^* = \lambda + \sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}^*)$. Therefore, we can derive the mean of the posterior predictive distribution, which is given by $\mathbb{E}[y^*|\lambda, \mathcal{D}, \mathbf{x}^*] = \mathbb{E}[\mu^*|\lambda, \mathcal{D}, \mathbf{x}^*]$. Consequently, applying this method to incremental elevation inference is straightforward. At each time instance, the elevation y^* at new location $\mathbf{x}^*$ can be estimated using Eq. 2 with the new training data $\mathcal{D}$, where $\mathbf{x}$ indicates the discrete locations in m_e that are currently observed, y is the observed elevation, and $\mathbf{x}^*$ represents the map locations that are within distance l of $\mathbf{x}$ in m_e.

2.3 Bayesian Kernel Traversability Classification

To perform classification, we similarly treat traversability as a Bernoulli distributed binary random variable, i.e. $y \sim \text{Ber}(\theta)$, and we seek to estimate the value of the parameter θ^*. We again adopt a conjugate prior formulation where $\theta \sim \text{Beta}(\alpha_0, \beta_0)$, in which α_0 and β_0 are hyperparameters. The posterior is also a Beta distribution,

$$p(\theta^*|\mathbf{x}^*, \mathcal{D}) \propto \theta^{\alpha^*-1}(1-\theta)^{\beta^*-1},$$

and we have

$$\alpha^* = \alpha_0 + \sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}^*)y_i, \quad \beta^* = \beta_0 + \sum_{i=1}^{N} k(\mathbf{x}_i, \mathbf{x}^*)(1 - y_i). \tag{3}$$

The mean and variance of the posterior predictive distribution are

$$\mathbb{E}[y^*|\alpha_0, \beta_0, \mathcal{D}, \mathbf{x}^*] = \frac{\alpha^*}{\alpha^* + \beta^*}, \quad \text{Var}[y^*|\alpha_0, \beta_0, \mathcal{D}, \mathbf{x}^*] = \frac{\alpha^*\beta^*}{(\alpha^* + \beta^*)^2}. \tag{4}$$

2.3.1 Traversability Training Data

Compared with the process of elevation regression, the training dataset $\mathcal{D}$ for traversability classification is not directly observed - instead our traversability training data are derived from the results of elevation inference. The $\mathbf{x}$ values in $\mathcal{D}$ are the same as those used in Section 2.2 for elevation inference. The y values in $\mathcal{D}$, which represent the traversability of these cells, are computed by

adapting the traversability estimation framework of [3]. The traversability of a cell is determined by three criteria: the step height h, the slope s and the roughness r:

$$v = \alpha_1 \frac{h}{h_{\text{crit}}} + \alpha_2 \frac{s}{s_{\text{crit}}} + \alpha_3 \frac{r}{r_{\text{crit}}}, \tag{5}$$

where α_1, α_2 and α_3 are weights which sum to 1. h_{crit}, s_{crit} and r_{crit}, which represent the maximum allowable step height, slope and roughness respectively, are critical values that may cause the robot to tip over or become stuck. The traversability v has a range of $[0, 1]$. A small value means the local terrain is flat and smooth, while a large value indicates rough terrain. When the traversability of a cell is estimated, all three criteria must be computed from m_e. If one of the criteria exceeds its critical value, the corresponding cell is labeled non-traversable. For the sake of brevity, the detailed procedures of obtaining step height h, slope s and roughness r can be found in [3].

We note that when the elevation of a grid cell is changed due to the arrival of new measurements, the traversability of all neighboring cells, within at least the radius of the robot, needs to be re-computed. However, the direct computation of traversability using Eq. 5, which involves plane fitting and eigendecomposition, over all affected cells, is intractable for use in real-time. Thus we only perform this computation for the cells intersected directly by lidar points. We can also incorporate the estimated *elevation variance* into Eq. 5 for a conservative traversability estimate in the regions where measurements are sparse. Although the elevation variance is not considered in Eq. 5, we incorporate it into traversability inference in Section 2.3.2 below.

2.3.2 Traversability Inference

The estimated traversability of cells at new locations $\mathbf{x}^*$, which is the same set used in Section 2.2, can be obtained from Eq. 4. The hyperparameters α^* and β^* at each new location are updated using Eq. 3. Note that since $\mathbf{x}$ and $\mathbf{x}^*$ remain the same as in Section 2.2, some results of Eq. 2 can be reused here to save computational resources.

We can also take advantage of variance predictions in a similar fashion to the occupancy mapping problem in [18]. The state of a grid cell in m_v is modeled as follows:

$$\text{state} = \begin{cases} \text{traversable}, & \text{if } v < v_{th},\ \sigma^2 < \sigma_{th}^2 \\ \text{non-traversable}, & \text{otherwise} \end{cases} \tag{6}$$

in which v is the mean of the predicted traversability at this cell, and v_{th} is the traversability threshold. Additionally thresholded by σ_{th}^2, the cells with variance σ^2 larger than σ_{th}^2 will also be labeled as non-traversable. Incorporating the variance into Eq. 6 naturally gives us conservative traversability estimation in regions where observations are sparse.

2.4 Sparse Kernel

Exact inference is permitted by Equations 2 and 4 provided that the requisite kernel computation can be performed exactly. Data structures like k-d trees offer logarithmic time radius queries, which lend themselves to efficient inference if we can limit the search neighborhood. Kernels like the radial basis function kernel have infinite support, leading to approximation error in truncation. Instead, we opt for the sparse kernel [17]:

$$k(\mathbf{x}, \mathbf{x}^*) = \begin{cases} \frac{2+\cos(2\pi\frac{d}{l})}{3}(1-\frac{d}{l}) + \frac{1}{2\pi}\sin(2\pi\frac{d}{l}), & \text{if } d \leq l \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

where d is the L^2 norm $\|\mathbf{x} - \mathbf{x}^*\|_2$. The kernel has support on the interval $[0, l]$, which allows exact inference to be performed in log-linear time.

2.5 BGK Traversability Mapping Process

The proposed BGK traversability mapping framework is shown in Fig. 2. Upon receiving lidar data in the form of a point cloud, we perform BGK elevation inference to obtain a dense height map m_e. Then we directly compute traversability for the cells that were explicitly intersected by the point cloud. BGK traversability inference then estimates the traversability of all cells whose elevation was inferred in the previous step, and the traversability map m_v is produced as output.

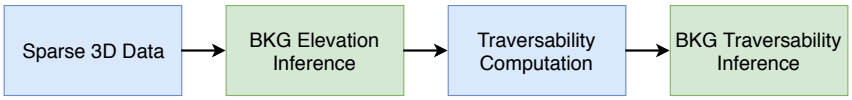


Figure 2: **Traversability mapping process:** Incoming lidar data, in the form of a point cloud, is incorporated as training data, and terrain elevation is estimated for all cells that lie within a designated distance threshold of the points. Traversability is then directly computed for the cells intersected by lidar points. This is used as the training data for traversability inference, applied to all grid cells that are within the same distance threshold used in the previous inference step.

We note that an alternative approach for traversability mapping is to perform BGK traversability inference directly, without an elevation inference step. Compared with the proposed framework, however, we have encountered inferior results, due in part to the fact that the resulting traversability estimates are less accurate when their training data is supported by limited, sparse elevation data.

3 Experimental Results

We evaluate the proposed terrain traversability mapping framework quantitatively and qualitatively in simulated and real-world environments. The method is implemented in C++ and executed using the robot operating system (ROS) [19] in Ubuntu Linux. The computational hardware is a laptop with an i7 2.5GHz CPU and 16GB memory. Throughout all the experiments, no multi-threading or GPU parallel computation is used for speed improvements.

3.1 Simulated Data

Gazebo [20] is utilized for two simulated experiments, which feature structured and unstructured environments, since the ground truth of the environment can be known precisely. These two environments are referred to as $City$, an urban environment that features buildings, trees, roads and sidewalks, and $Aerial$, a mountainous environment that features rough terrain and hills. Two volumetric scanning lidar sensors, the Velodyne VLP-16 and HDL-32E, are simulated in Gazebo for data gathering (applied to the *City* and *Aerial* maps respectively). Both sensors operate at 10Hz in all experiments; the real-time experiments are shown in full in the video attachment[1].

Three approaches are compared here to evaluate the proposed method. The first approach implemented is the $baseline$ approach. It only processes the raw point cloud data; no inference is used. The second approach, which performs BGK elevation inference and directly computes traversability for all cells where elevation is inferred, is referred to as $BGK + Trav$ for convenience. The proposed framework, which utilizes both BGK elevation and traversability inference in sequence, is referred to as BGK^+. For BGK elevation inference, we set $\lambda = 0$, which means that we have no prior knowledge of elevation at any position. For traversability inference, we apply the parameters $\alpha_0 = \beta_0 = 0.001$ to enforce a weak uninformative prior on all cells. The distance threshold l is selected to be 0.3m and 1.0m for our two simulated tests, respectively. The UGV is assumed to have a radius of 0.3m. Note that when we estimate the state of a cell using BGK$^+$, the variance in Eq. 6 is not used, for the sake of fair comparison with the ground truth.

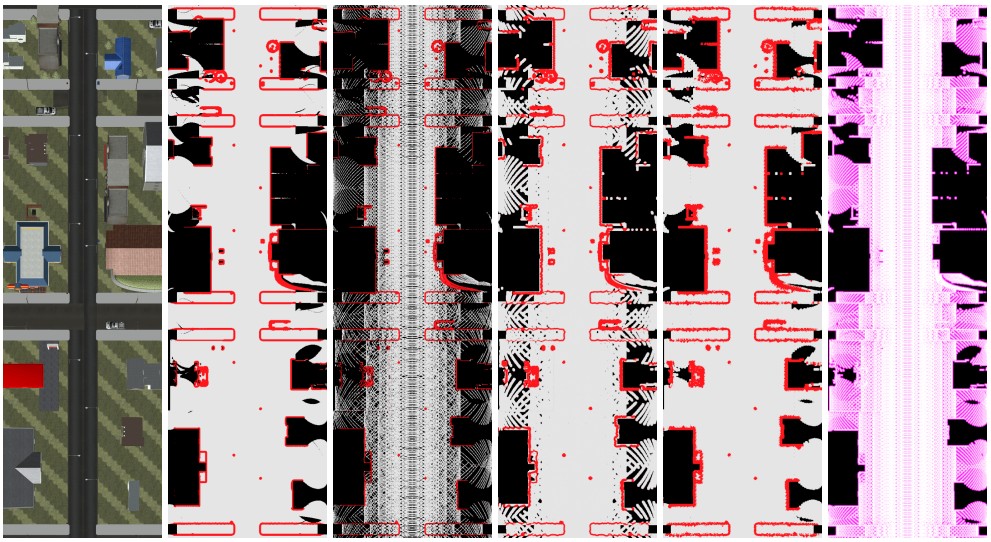
We implement the traversability mapping task as two independent processes. One process performs raw point cloud registration and BGK elevation inference, and the other performs the traversability computation of Eq. 5 and traversability inference. Since we acquire lidar scans at a rate of 10Hz, scans may be dropped if they take any one of the two processes longer than 0.1 seconds to complete.

3.1.1 Structured Environment

The simulated structured environment[2], which is shown in Fig. 3(a), spans 50 x 210 meters. The ground truth representation of the environment is obtained by taking lidar scans along the center of the road, from top to bottom, at a constant velocity of 1 m/s. The trajectory of the lidar is a straight line with a length of 210 meters, and there are 0.1 meters between scans. As a result, a total number of 2101 scans comprise our ground truth data. However, it is often undesirable to drive this slowly

[1] https://youtu.be/ewrCyDiWi-8

[2] Simulated structured environment: https://bitbucket.org/osrf/citysim



(a) Gazebo (b) GT (c) Baseline (d) BGK+Trav (e) BGK$^+$ (f) Variance

Figure 3: **Structured environment simulation:** The above plots illustrate (a) top view of the simulated urban environment in Gazebo, (b) ground truth of the traversability map, (c) traversability map produced by the baseline approach, (d) map produced by the BGK+Trav approach, (e) the results of the proposed method, BGK$^+$, and (f) the variance map of BGK$^+$ calculated by Eq. 4, where white color indicates low variance, and magenta indicates high variance.

in real-world mapping scenarios. Thus, we will only use scans at 1m intervals for traversability mapping, equivalent to a vehicle that moves at a speed of 36 km/h with a scan rate of 10Hz. As a result, we obtain 211 scans for the traversability mapping comparison.

The ground truth for the traversability map is shown in Fig. 3(b). Traversable, non-traversable and unknown regions are colored gray, red and black respectively. Fig. 3(c) shows the traversability mapping result by applying the baseline approach. As no inference is performed, it only covers 57% of the area covered by the ground truth. When the BGK+Trav approach is applied, it can cover 93% of the area with the aid of BGK elevation inference. However, due to the intensive traversability computation, real-time performance is not achieved, and 114 out of 211 scans are skipped. At last, we test BGK$^+$, which performs both elevation and traversability inference, on the same data. It closes many of the gaps in Figs. 3(c) and 3(d), achieving 100% map coverage. As expected, the variance map of BGK$^+$, which is obtained from Eq. 2, shows that the regions covered by fewer observations have higher variance values (colored in magenta).

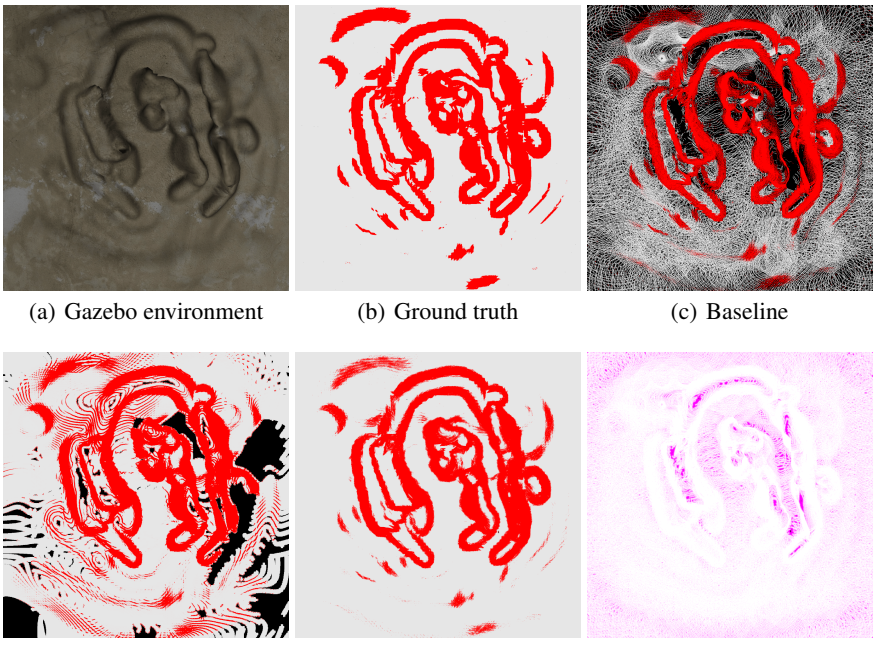
3.1.2 Unstructured Environment

The simulated unstructured environment[3], which is shown in Fig. 4(a), spans 120 x 120 meters. In this test, we simulate an unmanned aerial vehicle's fixed-altitude flyover of the environment to produce a map for a UGV, in which it captures a scan every 10 meters along the latitude and longitude directions. Thus we obtain a total of 169 (13 x 13) scans for this mapping comparison. The resulting traversability maps of each method are shown in Fig. 4 (d), (e) and (f) respectively. The same color scheme of Fig. 3 is also applied here. The baseline approach can only cover 50% of the area with the available scans. Because the BGK+Trav method suffers from prohibitive computational cost, it skips 84% of the scans. However, 89% of the map is covered due to elevation inference. BGK$^+$ is able to cover 100% of the map while maintaining real-time performance.

3.2 Benchmarking Results

The receiver operating characteristic (ROC) curves in Fig. 5 show a predictive performance comparison between BGK+Trav and BGK$^+$. The ROC curves plot the true positive rate against the false positive rate. We compare the predicted traversability to a ground-truth traversability of 1 for

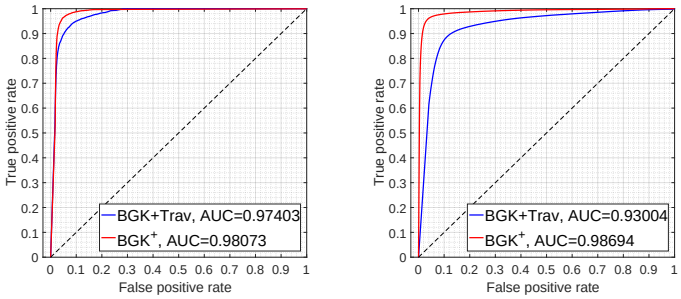
[3]Simulated unstructured environment: `http://wiki.ros.org/hector_quadrotor`



(a) Gazebo environment (b) Ground truth (c) Baseline

(d) BGK+Trav (e) BGK$^+$ (f) Variance Map

Figure 4: **Unstructured environment simulation:** The *Aerial* simulated terrain model is shown in (a). Traversability maps of the ground truth, baseline, BGK+Trav and BGK$^+$ methods are shown in (b), (c), (d) and (e) respectively. The variance map of BGK$^+$ is shown in (f).



(a) $City$ dataset (b) $Aerial$ dataset

Figure 5: **Receiver operating characteristic curves:** The ROC curves of BGK+Trav and BGK$^+$ methods for the $City$ and $Aerial$ datasets are shown in (a) and (b) respectively.

a non-traversable cell and 0 for a traversable cell, so that the comparison of inference accuracy is independent of the choice of threshold in Eq. 6. The plot can be viewed as a plot of predictive performance as a function of the threshold. The area under the curve (AUC) is also provided in each case for comparison of prediction accuracy. BGK$^+$ outperforms BGK+Trav in both simulated environments with a higher AUC. We also perform tests for the one-step alternative approach that only performs BGK traversability inference without performing BGK elevation inference. The AUCs of this approach are 0.9617 and 0.9581 for the two tests respectively. Thus BGK$^+$, which performs two-step inference, achieves a higher AUC than either one-step inference approach.

Quantitative results for the different methods compared in simulation are summarized in Table 1. BGK$^+$ infers the contents of 100% of the visible terrain area despite sparse lidar coverage in both experiments. Since BGK$^+$ only explicitly computes the traversability of cells where new points arrive, it requires less computation time. Mean squared error (MSE) is provided for the elevation inference step of all methods. Since BGK$^+$ skips fewer scans, the MSE of BGK$^+$ is lower than the MSE of BGK+Trav. BGK$^+$ shows advantages in terms of both efficiency and accuracy. We also note that BGK+Trav yields the best results when unlimited computation time is available.

Table 1: Quantitative results for different mapping approaches

Dataset	Method	Map Coverage (%)	Skipped Scans	BGK Elevation Inference Time (s)	Traversability Calculation Time (s)	BGK Traversability Inference Time (s)	Mean Squared Error (sq m)
City	Baseline	57	32/211	N/A	0.093	N/A	N/A
	BGK+Trav	93	114/211	0.043	0.174	N/A	0.0089
	BGK+	100	1/211	0.042	0.029	0.017	0.0052
Aerial	Baseline	50	82/169	N/A	0.149	N/A	N/A
	BGK+Trav	89	142/169	0.069	0.583	N/A	0.039
	BGK+	100	3/169	0.066	0.045	0.020	0.011

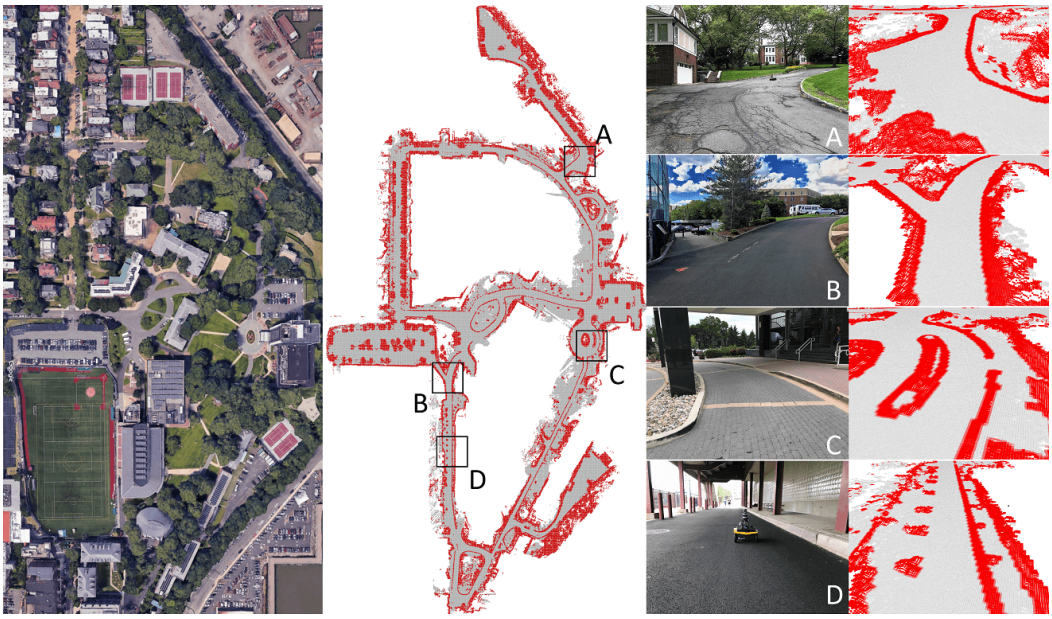


Figure 6: **Traversability map of a large-scale urban area using BGK$^+$:** A satellite image of the mapped area is shown at left. The traversability map from applying BGK$^+$ is shown at center. Representative scenes of the mapped environment are shown at right.

3.3 Large-scale Urban Environment

We also evaluate our framework in a large-scale urban area. We manually drove a Jackal UGV with a VLP-16 lidar across the area for about 39 min. at a speed of about 1.5 m/s. The mapped area has a maximum elevation change of 15 meters, and spans 285 x 550 meters. A satellite image of the mapped area is shown at the left of Fig. 6. A total number of 22,636 scans, which are captured at a rate of 10Hz, are used for traversability mapping. The final traversability map from BGK$^+$ is shown in the center of the figure. Four representative images of the environment, along with the traversability map, are shown at right. The proposed system successfully distinguishes non-traversable areas and traversable areas - we note in particular that curbs, which are challenging for autonomous urban navigation, are precisely marked as non-traversable throughout the map.

4 Conclusion

We have proposed applying Bayesian generalized kernel inference to terrain traversability mapping. Our framework is unique in its composition, with two sequential inference steps. The first step performs elevation inference to address the sparsity of the available point clouds, and the second step performs traversability inference to relieve the burden of exhaustive traversability computation. The proposed framework is validated using both simulated and real-world data and provides efficiency and accuracy for real-time terrain mapping with lidar. Future work will consider the practical use of variance information to better-support safe navigation in complex environments.

Acknowledgments

This research has been supported in part by the National Science Foundation, grant number IIS-1723996.

References

[1] J.M. Santos, D. Portugal, and R. P. Rocha. An Evaluation of 2D SLAM Techniques Available in Robot Operating System. *Proceedings of the IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 1-6, 2013.

[2] P. Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 4:1373-1385, 2013

[3] A. Chilian and H. Hirschmuller. Stereo Camera Based Navigation of Mobile Robots on Rough Terrain. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4571-4576, 2009.

[4] I. Bogoslavskyi, O. Vysotska, O. Serafin, J. Grisetti, and C. Stachniss. Efficient Traversability Analysis for Mobile Robots Using the Kinect Sensor. *Proceedings of the IEEE European Conference on Mobile Robots*, pp. 158-163, 2013.

[5] F. Colas, S. Mahesh, F. Pomerleau, M. Liu, and R. Siegwart. 3D Path Planning and Execution for Search and Rescue Ground Robots. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 722-727, 2013.

[6] P. Krsi, P. Furgale, M. Bosse, and R. Siegwart. Driving on Point Clouds: Motion Planning, Trajectory Optimization, and Terrain Assessment in Generic Nonplanar Environments. *Journal of Field Robotics*, 34(5):940-984, 2017.

[7] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert. Autonomous Navigation for BigDog. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 4736-4741, 2010.

[8] J. Sock, J. Kim, J. Min, and K. Kwak. Probabilistic Traversability Map Generation Using 3D-LIDAR and Camera. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 5631-5637, 2016.

[9] F. Schilling, X. Chen, J. Folkesson, and P. Jensfelt. Geometric and visual terrain classification for autonomous mobile navigation. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2678-2684, 2017.

[10] K. Zimmermann,, P. Zuzanek, M. Reinstein, and V. Hlavac,. Adaptive Traversability of Unknown Complex Terrain with Obstacles for Mobile Robots. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 5177-5182, 2014.

[11] B. Suger, B. Steder, and W. Burgard. Traversability Analysis for Mobile Robots in Outdoor Environments: A Semi-supervised Learning Approach Based on 3D-Lidar Data. *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3941-3946, 2015.

[12] J. Ahtiainen, T. Stoyanov, and J. Saarinen. Normal Distributions Transform Traversability Maps: LIDAROnly Approach for Traversability Mapping in Outdoor Environments. *Journal of Field Robotics*, 34(3):600-621, 2017.

[13] R. O. Chavez-Garcia, J. Guzzi, L. M. Gambardella, and A. Giusti. Learning Ground Traversability From Simulations. *IEEE Robotics and Automation Letters*, 3:1695-1702, 2018.

[14] S. Vasudevan, F. Ramos, E. Nettleton, and H. DurrantWhyte. Gaussian process modeling of largescale terrain. *Journal of Field Robotics*, 26(10):812-840, 2009.

[15] V. Guizilini and F. Ramos. Variational Hilbert Regression with Applications to Terrain Modeling. *Proceedings of the International Symposium on Robotics Research*, 2017.

[16] W. R. Vega-Brown, M. Doniec, and N. G. Roy. Nonparametric Bayesian inference on multivariate exponential families. *Advances in Neural Information Processing Systems*, pp. 2546-2554, 2014.

[17] A. Melkumyan and F. Ramos. A sparse covariance function for exact Gaussian process inference in large datasets. *International Joint Conference on Artificial Intelligence*, 9:1936-1942. 2009.

[18] K. Doherty, J. Wang, and B. Englot. Bayesian generalized kernel inference for occupancy map prediction. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3118-3124, 2017.

[19] N. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source Robot Operating System. *IEEE International Conference on Robotics and Automation, Workshop on Open Source Software*, 2009.

[20] N. Koenig and A. Howard. Design and Use Paradigms for Gazebo, an Open-Source Multi-Robot Simulator. *Proceedings of the IEEE International Conference on Intelligent Robots and Systems*, pp. 2149-2154, 2004.