Random Mapping Method for Large-Scale Terrain Modeling

Xu Liu[1,2,3], **Decai Li**[1,2], **Yuqing He**[1,2]
[1]State Key Laboratory of Robotics, Shenyang Institute of Automation, Chinese Academy of Sciences
[2]Institutes for Robotics and Intelligent Manufacturing, Chinese Academy of Sciences
[3]University of Chinese Academy of Sciences
liuxu1@sia.cn, lidecai@sia.cn, heyuqing@sia.cn

Abstract

The vast amount of data captured by robots in large-scale environments brings the computing and storage bottlenecks to the typical methods of modeling the spaces the robots travel in. In order to efficiently construct a compact terrain model from uncertain, incomplete point cloud data of large-scale environments, in this paper, we first propose a novel feature mapping method, named random mapping, based on the fast random construction of base functions, which can efficiently project the messy points in the low-dimensional space into the high-dimensional space where the points are approximately linearly distributed. Then, in this mapped space, we propose to learn a continuous linear regression model to represent the terrain. We show that this method can model the environments in much less computation time, memory consumption, and access time, with high accuracy. Furthermore, the models possess the generalization capabilities comparable to the performances on the training set, and its inference accuracy gradually increases as the random mapping dimension increases. To better solve the large-scale environmental modeling problem, we adopt the idea of parallel computing to train the models. This strategy greatly reduces the wall-clock time of calculation without losing much accuracy. Experiments show the effectiveness of the random mapping method and the effects of some important parameters on its performance. Moreover, we evaluate the proposed terrain modeling method based on the random mapping method and compare its performances with popular typical methods and state-of-art methods.

Introduction

Mobile robot environment modeling is aimed at providing a suitable representation of the space of interest, and it plays a vital role in robot autonomous navigation and environment exploration. However, the construction of a lightweight environmental model for a large-scale, unstructured, and open environment is still a challenge. Specifically, the challenge results from several factors, such as: **data uncertainty** brought by the inevitable sensor noise and compounded localization uncertainty, which will make the data ambiguous; **data incompleteness** caused by the shielding or absorption of the sensor rays, and the gaps in the environments, which will also make the generated maps incomplete; **large amount of data** that results in high computation complexity and storage consumption, in which the former one prohibit the real-time mapping and the latter has become the bottleneck of large-scale environmental modeling.

Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

To obtain a clean, complete, and accurate environmental model efficiently and compactly, the traditional representation methods, such as the point cloud maps (Cole and Newman 2006), elevations maps (EM) (Krotkov and Hoffman 1994), multi-level surface maps (Triebel, Pfaff, and Burgard 2006), 3D voxel maps such as OctoMap (Hornung et al. 2013), and triangulated irregular networks (TIN) (Rekleitis et al. 2009), are incompetent. Point cloud maps directly store the measurements points, thus its memory consumption depends on how large the mapping area is and how sparse the point cloud is. Similarly, the grid-based method including 3D voxel maps, elevations maps, and multi-level surface maps, are not memory-efficient when there is a need for fine resolutions. TIN is a sample-based approach, which also suffers from a huge memory footprint when encountering highly textured surfaces. Furthermore, all of the above representations are obtained by pure geometric processing of the original measurements, which typically cannot deal with the data uncertainty in a statistically sound way and handle the data incompleteness through terrain inference from the information implied in the data.

The fact that the traditional methods can hardly deal with the challenges brought by the large-scale terrain modeling makes the state-of-art modeling techniques resort to probabilistic reasoning and machine learning. There are two main ideas for using machine learning to model terrain. One is to train a classifier to indicate if a three-dimensional (3D) point is occupied, and the other is to establish a regression model to fit the relationship between the two-dimensional (2D) positions and their elevations. Current methods are typically Bayesian-based, such as the Gaussian processes (GPs) (Williams and Rasmussen 2006; Lee et al. 2019; Jadidi, Miro, and Dissanayake 2018; Stork and Stoyanov 2020), variational inference (VI) (Bishop and Nasrabadi 2006; Guizilini and Ramos 2019), Bayesian kernel inference (Doherty et al. 2019), Gaussian mixture model (GMM) (Srivastava and Michael 2018; O'Meadhra, Tabib, and Michael 2018), and some others involve the deep learning (Bauer, Kuhnert, and Eckstein 2019). These techniques are sophisticated and possess good performances that they may incorporate uncertainty in a statistic way and replenish the incompleteness through generalized inference. However, those Bayesian-based non-parametric models need to keep the original training data for future predictions, which will also lead to the problem of excessive storage consumption. In addition, the standard GP suffers from a computation complexity of $\mathcal{O}(n^3)$ with the number of training points, which is prohibitive even using simple approximation methods when there are large amounts of data. Deep learning, regardless of its end-to-end manner, requires a lot of time and resources to iteratively train a complex model. Thus, the current methods can not deal with all of the problems of large-scale terrain modeling.

In fact, from the perspectives of environmental modeling and machine learning, a terrain modeling method is required to possess the following properties: **data fusion** that can consider the measurement uncertainty. The measurement noise is commonly at the centimeter level, and a robust and unbiased estimate can be statistically obtained; **terrain reasoning** that can infer the states of unobserved areas such as the occluded using spatial association of the available data; **efficient construction** that possess acceptable computation speed, memory consumption, and access time.

Attempting to solve all of the problems in large-scale terrain modeling, in this paper, we propose to build a concise linear regression model to represent the terrain. The essence of this method is to generate an approximately linearly distributed space, using random mapping, to enable the linear model. This method is efficient in computation and storage since it does not depend on iterative calculation and only needs to save limited model parameters. Furthermore, the models are obtained by parallel training using the extracted terrain surface data, which greatly reduces the training time and handles the data uncertainty well. Overall, the main contributions of this paper are: 1) We propose a novel method to rapidly generate linearly distributed high-dimensional feature space; 2) We propose a novel compact and efficient linear parameter representation for the large-scale terrain.

Related Work

Learning-based Robotics Mapping

Ramos and Ott (2016) regard the occupancy mapping problem as a classification task and train a discrimination model to present the occupancy maps using the kernel approximations and logistic regression classifier. This method have extended to 3D environments (Doherty, Wang, and Englot 2016; Guizilini and Ramos 2018a,b) and dynamic scenarios (Guizilini, Senanayake, and Ramos 2019; Senanayake et al. 2016). These methods can avoid assuming that the grids are independent of each other, thereby introducing spatial association among them. Another method often used in terrain modeling is to establish a regression model to fit the relationship between the 2D location and their elevations, and the most popular is the Gaussian process (Popović et al. 2017; Le Gentil et al. 2020; Li et al. 2020). This method can predict the terrain by information implied in the covariance functions. However, it suffers from a $\mathcal{O}(n^3)$ computation complexity with the sample number and needs to keep the original points. In addition to the terrain maps based on the geometric features, some others take advantage of the visual features to build the semantic terrain maps (Zhou et al. 2019; Schilling et al. 2017), and some others use the sound-based method (Valada, Spinello, and Burgard 2018). These feature-based methods may fail in the environments that lake of features and these methods are commonly based on the deep convolutional networks (Long, Shelhamer, and Darrell 2015), which always requires more consumption time and computing resources. Taking these into account, we use compact and efficient linear parametric models to model terrain. So far, to our best knowledge, no one has done so.

Random Mapping

In the past many years, there are similar concepts to random mapping. In the artificial neural network, Pao and Takefuji (1992) and Pao, Park, and Sobajic (1994) proposed Random Vector Functional Link Networks, in which the input layer is directly connected to both the hidden layer and the output layer, and the weights between the input layer and hidden layer are randomly selected. Schmidt et al. (1992) proposed another standard feed-forward neural network with random weights between the input layer and hidden layer. These methods accelerate the training of the network and reduce the hardware requirements. Kaski (1998) applied the random mapping to dimensionality reduction for fast similarity computation in document classification and Bingham and Mannila (2001) applied it to the processing of both noisy and noiseless images, and information retrieval in text documents. To accelerate the training of kernel machines, Rahimi, Recht et al.(2007) proposed to map the input high-dimensional feature space to a randomized low-dimensional one and then apply existing fast linear methods for classification and regression tasks. Our initial inspiration for random mapping originates from this. In this paper, we consider random mapping from a new perspective, which is used to generate a linearly distributed high-dimensional feature space, which enables a linear regression model in it. Particularly, we prove this property mathematically rigorously.

Random Mapping Method

Random Mapping Concept

We begin the representation of random mapping method by first introducing the notions and concepts. Assuming that we have an arbitrary set of samples $\{\boldsymbol{x}_i, t_i\}_{i=1}^{L}$, where $\boldsymbol{x}_i \in \boldsymbol{R}^{N\times 1}$ is the feature vector and $t_i \in \boldsymbol{R}^m$ is the target value. In the case when the data in low-dimensional feature space is not linearly separable, a popular and effective machine learning technique we often use is kernel trick $k(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})\phi(\boldsymbol{x}')$ (Boser, Guyon, and Vapnik 1992; Hofmann, Schölkopf, and Smola 2008) or kernel approximation $k(\boldsymbol{x}, \boldsymbol{x}') \approx \phi(\boldsymbol{x})\phi(\boldsymbol{x}')$ (Williams and Seeger 2001), which allows us to implicitly use the features in a high, even infinite dimensional reproducing kernel Hilbert space. However, we may struggle to find a suitable kernel or mapping function, which is time-consuming or difficult.

Considering that we may be trapped in the dilemmas when constructing a valid kernel or mapping function, we propose the random method to efficiently generate a feature mapping function $\phi(\boldsymbol{x}_i)$ for a vector $\boldsymbol{x}_i$ as follows:

$$\phi(\boldsymbol{x}_i) = \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x_i}) = \boldsymbol{g}(\boldsymbol{v}_i) = \boldsymbol{s}_i \tag{1}$$

where $\boldsymbol{W}$ is a $M \times N$ matrix denoting a linear transformation. Particularly, as the essential property of our proposed method, all elements of $\boldsymbol{W}$ are generated at random from a probability distribution, such as a uniform distribution. Here, M is called random mapping dimension which is always determined by trials and far larger than N but much less than L. As a result, $\boldsymbol{v}_i \triangleq \boldsymbol{W}\boldsymbol{x_i}$ denotes a $M \times 1$ random vector. The vector-valued function $\boldsymbol{g}(\boldsymbol{v}) : \boldsymbol{R}^M \rightarrow \boldsymbol{R}^M$ is called activation function which is defined based on a scalar-valued function $g(v) : \boldsymbol{R} \rightarrow \boldsymbol{R}$ as follows:

$$\boldsymbol{g}(\boldsymbol{v}) = [g(v_1), g(v_2), \cdots, g(v_M)]^T \tag{2}$$

Mathematically, $g(v)$ is required to be a nonlinear function possessing derivatives of at least $L-1$ order, such as the sine function $g(v) = sin(v)$ and one kind of the sigmoid functions $g(v) = (1 + exp(-v))^{-1}$. Thus, $\boldsymbol{s}_i$ is a $M \times 1$ random feature vector that has a nonlinear relationship with the input vector $\boldsymbol{x}_i$. Here, we introduce a matrix $\boldsymbol{V} = [\boldsymbol{v}_1, \boldsymbol{v}_2, \cdots, \boldsymbol{v}_L]$. Further, we define a matrix-valued function $\boldsymbol{G}(\boldsymbol{V}) : \boldsymbol{R}^{M\times L} \rightarrow \boldsymbol{R}^{M\times L}$ as follows:

$$\boldsymbol{G}(\boldsymbol{V}) = \boldsymbol{G}([\boldsymbol{v}_1, \cdots, \boldsymbol{v}_L]) = [\boldsymbol{g}(\boldsymbol{v}_1), \cdots, \boldsymbol{g}(\boldsymbol{v}_L)] \tag{3}$$

As a result, for an input set $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^L \triangleq [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L]$, which can be written as an $N \times L$ matrix, we can obtain the random mapping matrix $\boldsymbol{S}$ simply according to the following formula:

$$\boldsymbol{S} = \boldsymbol{G}(\boldsymbol{W}\boldsymbol{X}) = \boldsymbol{G}(\boldsymbol{V}) \tag{4}$$

Here, we will call the matrix $\boldsymbol{S}$ random mapping set (RMS) because it is generated randomly using random mapping (RM) and each column of it can be treated as a feature vector. Its dimension $M \times L$ is determined by the mapping dimension M and the number of samples L. Particularly, for a sample set $\boldsymbol{X}$, the nonlinear transformation $\boldsymbol{G}(\boldsymbol{W}\boldsymbol{X})$ can generate a specified and commonly higher dimensional RMS in which the data is approximately linearly distributed, providing M is large enough. Intuitively, it is uncommon that a randomly obtained set possesses the favorable statistical property. We will explain this property rigorously soon.

Feasibility and Existence Analysis

As aforementioned, the derived RMSs possess the tractable linearly distributed property. We now study the micro details to understand why it has such macro performance. Here, we only provide qualitative analysis at some minor points by omitting obscure derivations but meanwhile try to keep the processes serious. Assuming that a linear (or a generalized linear (Bishop and Nasrabadi 2006)) classifier $f(\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x})$ is learnt in a RMS, and our goal is to find the optimal weight vector $\boldsymbol{\beta} \in \boldsymbol{R}^{M\times 1}$ that can minimize the sum-of-squares error $\sum_{i=1}^L (f(\boldsymbol{x}_i, \boldsymbol{\beta}) - t_i)^2$, i.e., $\min ||\boldsymbol{\beta}^T\boldsymbol{S} - \boldsymbol{T}||$ where $\boldsymbol{T} = [t_i]_{i=1}^L$ is a $1 \times L$ target vector. Then we have the following claim that we are going to prove.

Claim: For every real number $\epsilon > 0$, there exists a natural number P such that for all M meeting $M > P$, we will have at least one weight vector $\boldsymbol{\beta}$ that enables $||\boldsymbol{\beta}^T\boldsymbol{S} - \boldsymbol{T}|| < \epsilon$. Here, every symbol is defined as above, i.e., M is the random mapping dimension and $\boldsymbol{S}$ is an $M \times L$ RMS.

To prove the above claim, we first define two lemmas:

Lemma 1: When the random mapping dimension M is sufficiently large, there exists a weight vector $\boldsymbol{\beta}$ that makes $||\boldsymbol{\beta}^T\boldsymbol{S} - \boldsymbol{T}|| = 0$ with probability 1.

Proof of Lemma 1: Here, without loss of generality, we prove that the RMS $\boldsymbol{S}$ is column full rank when the mapping dimension M is large enough. It is worth noting that this is a stronger condition since even if $\boldsymbol{S}$ is not full rank, we can get the same conclusion as long as $r(\boldsymbol{S}) = r(\boldsymbol{S}^T, \boldsymbol{T}^T)$ (Greub 2012), where $r(\boldsymbol{S})$ is the rank of matrix $\boldsymbol{S}$.

First of all, we define a sequence of $1 \times N$ random weight vectors as $\{\boldsymbol{w}_i\}_{i=1}^{+\infty}$. Using these vectors, we further define a sequence of $i \times N$ weight matrix as $\{\boldsymbol{W}_i\}_{i=1}^{+\infty}$ by stacking the first i element in $\{\boldsymbol{w}_i\}_{i=1}^{+\infty}$. According to (4), given a dataset $\boldsymbol{X}$, we can then acquire a sequence of RMS as $\{\boldsymbol{S}_i\}_{i=1}^{+\infty}$. Represent $\boldsymbol{S}_i$ by its row vectors as $\boldsymbol{S}_i = [\boldsymbol{r}_1, \cdots, \boldsymbol{r}_i]^T$, where $\boldsymbol{r}_i = [g(\boldsymbol{w}_i \cdot \boldsymbol{x}_1), \cdots, g(\boldsymbol{w}_i \cdot \boldsymbol{x}_L)]$, comprising the inner product between $\boldsymbol{w}_i$ and each sample. Obviously, we have the following result

$$r(\boldsymbol{S}_{i+1}) \geq r(\boldsymbol{S}_i) \tag{5}$$

which means that the sequence $\{r(\boldsymbol{S}_i)\}_{i=1}^{\infty}$ is monotonically increasing. Then ,we will prove that L is its reachable supremum. To this end, we equivalently prove that when the mapping dimension is sufficiently large, i.e., when i is large enough, there must be L linearly independent vectors $\{\boldsymbol{r}_k\}_{k=1}^L$ in $\boldsymbol{S}_i$. Here, we prove this by contradiction. Supposing that no matter how big i is, there are at most $L-1$ linearly independent vectors in $\boldsymbol{S}_i$. Under this condition, we can acquire that all of those vectors in $\boldsymbol{S}_i$ are from a subspace whose dimension is $L-1$, spanned by $L-1$ linearly independent vectors. Then, there will exists at least one nonzero $L \times 1$ vector $\boldsymbol{n}$ orthogonal to this subspace, yielding

$$\boldsymbol{r}_i \cdot \boldsymbol{n} = g(\boldsymbol{w}_i \cdot \boldsymbol{x}_1) \cdot n_1 + \cdots + g(\boldsymbol{w}_i \cdot \boldsymbol{x}_L) \cdot n_L = 0 \tag{6}$$

Since $g(v)$ is at least (L-1)-order derivable, we have

$$\sum_{j=1}^{L} n_j \cdot g^{(p)}(\boldsymbol{w}_i \cdot \boldsymbol{x}_j) = 0 \tag{7}$$

for $p = 1, 2, \cdots, L-1$, where $g^{(p)}$ is the p-order derivative of g with respect to $\boldsymbol{w}_i$. This is a system of linear equations comprising L equations and L variables n_i for $i = 1, \cdots, L$, whose coefficient matrix is

$$\boldsymbol{C} = \begin{pmatrix} g(\boldsymbol{w}_i \cdot \boldsymbol{x}_1) & \cdots & g(\boldsymbol{w}_i \cdot \boldsymbol{x}_L) \\ \vdots & \ddots & \vdots \\ g^{(L-1)}(\boldsymbol{w}_i \cdot \boldsymbol{x}_1) & \cdots & g^{L-1}(\boldsymbol{w}_i \cdot \boldsymbol{x}_L) \end{pmatrix} \tag{8}$$

Assuming a week condition that $\boldsymbol{x}_l \neq \boldsymbol{x}_{l'}$ for $l \neq l'$, we can acquire that $\boldsymbol{w}_i \cdot \boldsymbol{x}_l \neq \boldsymbol{w}_i \cdot \boldsymbol{x}_{l'}$ for $l \neq l'$ almost everywhere (Dudley 2018). Then, given the functions $g^{(p)}(v)$ which are unequal almost everywhere, we can obtain that

$g^{(p)}(\boldsymbol{w}_i \cdot \boldsymbol{x}_l) \neq g^{(p)}(\boldsymbol{w}_i \cdot \boldsymbol{x}_{l'})$ for $l \neq l'$ almost everywhere, and their function value can be arbitrary one belonging to the value domain of $g^{(p)}(v)$. Considering that the determinant of the square matrix $\boldsymbol{C}$ is a multivariable polynomial function $f : \boldsymbol{R}^{L^2} \rightarrow \boldsymbol{R}$ of its elements (Horn and Johnson 2012) and the zero set of a polynomial has Lebesgue measure zero (Fleming 2012; Caron and Traynor 2005), we have that the set of matrices with zero determinant has Lebesgue measure zero. From the perspective of probability theory, we construct a probability space $(\boldsymbol{\Omega}, \boldsymbol{F}, P)$, where $\boldsymbol{\Omega}$ is the set of all matrices generated by (8). Divide the elements in $\boldsymbol{\Omega}$ into two subsets $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$, where $\boldsymbol{A}_1$ comprising the matrices with determinant zero and $\boldsymbol{A}_2$ with determinant non-zero. $\boldsymbol{F}$ is the σ-algebra, defined as $\boldsymbol{F} = \{\boldsymbol{\Omega}, \boldsymbol{A}_1, \boldsymbol{A}_2, \varnothing\}$. Then, define $P(\boldsymbol{A}) = m(\boldsymbol{A})/m(\boldsymbol{\Omega})$, where m is the Lebesgue measure and $\boldsymbol{A} \in \boldsymbol{F}$. Then we have $P(\boldsymbol{A}_1) = 0$, so $\boldsymbol{C}$ is singular with probability zero. Noticeably, it is possible for the sets being non-empty even infinite to still have Lebesgue measure zero. Therefore, the system of linear equations only has zero solution, which is contradictory to the fact that $\boldsymbol{n}$ is non-zero. Thus, there must exist L linearly independent vectors in $\boldsymbol{S}_k$ when k is large enough.

Thus, we can naturally acquire that $\{r(\boldsymbol{S}_i)\}_{i=1}^{+\infty}$ have a reachable supremum L. Recall that $\{r(\boldsymbol{S}_i)\}_{i=1}^{+\infty}$ is monotonically increasing, so we have the following result

$$\lim_{i \to \infty} r(\boldsymbol{S}_i) = \sup(r(\boldsymbol{S}_i) | i \in \boldsymbol{N}) = L \tag{9}$$

according to the monotone bounded convergence theorem (Fitzpatrick 2009). Thus, we have that matrix $\boldsymbol{S}$ is column full rank almost everywhere in the case when the mapping dimension M is large enough. Q.E.D.

Lemma 2: Given two arbitrary RMSs $\boldsymbol{S}_1$ and $\boldsymbol{S}_\triangle$ that are $M_1 \times L$ and $M_\triangle \times L$, where $M_1 \ll L$ and $M_\triangle \ll L$, defining

$$\boldsymbol{S}_2 = \begin{pmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_\triangle \end{pmatrix} \tag{10}$$

that is $(M_1 + M_\triangle) \times L$ by stacking $\boldsymbol{S}_1$ and $\boldsymbol{S}_\triangle$, and

$$E(\boldsymbol{S}_1) = \min ||\boldsymbol{\beta}_1^T \boldsymbol{S}_1 - \boldsymbol{T}|| = ||\boldsymbol{\beta}_1^{*T} \boldsymbol{S}_1 - \boldsymbol{T}|| \tag{11}$$

denoting the training error of the linear model trained by $\boldsymbol{S}_1$, in which $\boldsymbol{\beta}_1^* = \arg\min ||\boldsymbol{\beta}_1^T \boldsymbol{S}_1 - \boldsymbol{T}||$. Define $E(\boldsymbol{S}_2)$ and $\boldsymbol{\beta}_2^*$ in the similar way. Then we will have $E(\boldsymbol{S}_2) \leq E(\boldsymbol{S}_1)$.

Proof of Lemma 2: In term of the above definitions, we can represent $E(\boldsymbol{S}_2)$ based on $\boldsymbol{S}_1$ and $\boldsymbol{S}_\triangle$ as follows:

$$E(\boldsymbol{S}_2) = ||\boldsymbol{\beta}_2^{*T} \boldsymbol{S}_2 - T|| = ||\boldsymbol{\beta}_2^{*T} \begin{pmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_\triangle \end{pmatrix} - \boldsymbol{T}|| \tag{12}$$

Since $\boldsymbol{\beta}_2^*$ has the smallest error among all $\boldsymbol{\beta}_2$, yielding

$$E(\boldsymbol{S}_2) \leq || \begin{pmatrix} \boldsymbol{\beta}_1^* \\ \boldsymbol{0} \end{pmatrix}^T \begin{pmatrix} \boldsymbol{S}_1 \\ \boldsymbol{S}_\triangle \end{pmatrix} - \boldsymbol{T}|| \tag{13}$$

where $\boldsymbol{0}$ is the zero vector of $M_\triangle \times 1$. Using the block matrix multiplication, (13) can further yield

$$E(\boldsymbol{S}_2) \leq ||\boldsymbol{\beta}_1^{*T} \boldsymbol{S}_1 - \boldsymbol{T}|| = E(\boldsymbol{S}_1) \tag{14}$$

Q.E.D.

Based on Lemma 1 and Lemma 2, our initial claim can be proved as follows:

Proof of Claim: Given a sequence $\{E(\boldsymbol{S}_i)\}_{i=1}^{\infty}$, where $\boldsymbol{S}_i$ is an $i \times L$ RMS and $E(\boldsymbol{S}_i) = ||\boldsymbol{\beta}_i^{*T} \boldsymbol{S}_i - \boldsymbol{T}||$. According to Lemma 1 and Lemma 2, $\{E(\boldsymbol{S}_i)\}_{i=1}^{\infty}$ is monotonically decreasing as i increases and has a lower bound zero when i is large enough. According to the monotone convergence theorem again, we have the following result

$$\lim_{i \to \infty} E(\boldsymbol{S}_i | i \in \boldsymbol{N}) = 0 \tag{15}$$

Thus, for $\forall \epsilon > 0$, $\exists P \in \boldsymbol{N}$, when $M > P$, there is $\boldsymbol{\beta} \in \boldsymbol{R}^{M \times 1}$ that makes $||\boldsymbol{\beta}^T \boldsymbol{S} - \boldsymbol{T}|| < \epsilon$. Q.E.D.

This result enables us to choose a natural number $M \ll L$ to reduce the computational complexity in practical applications. Moreover, since the weight matrix $\boldsymbol{W}$ is obtained at random, we avoid the training and construction of kernels. Thus, this method is very time-saving. In this paper, we first applied the RM method to the terrain modeling that can be regarded as a regression problem. Even so, RM can simply extend to many other classification and regression tasks with arbitrary dimensionalities and sizes.

Terrain Models Learning

Assume that a robot has captured a data set $\boldsymbol{D} = \{\boldsymbol{x}_i, t_i\}_{i=1}^{L}$ by the end points of a laser ranger finder or depth cameras when moving in the environments, where $\boldsymbol{x}_i$ is a 2D location and t_i is its elevation. Then, treating the terrain modeling for the unstructured environments as a regression problem, we attempt to build a linear parametric model between $\{\boldsymbol{x}_i\}_{i=1}^{L}$ and $\{t_i\}_{i=1}^{L}$, based on the RM method, which is

$$y = f(\boldsymbol{x}, \boldsymbol{\beta}) = \boldsymbol{\beta}^T \boldsymbol{g}(\boldsymbol{W}\boldsymbol{x}) + b = \boldsymbol{\beta}^T \boldsymbol{s} + b = \boldsymbol{\beta}^T \boldsymbol{s} \tag{16}$$

Here, the symbols are defined as above, and to make the notations simple, we expand $\boldsymbol{\beta}$ with b and $\boldsymbol{s}$ with 1. In this problem, our goal is to find the optimal weight $\boldsymbol{\beta}$ for terrain inference. Here, depending on if online learning is required, we provide two solutions.

Closed-form Solution of $\boldsymbol{\beta}$

Here, we use all of the data at once to train the model. To optimize $\boldsymbol{\beta}$, we minimize the objective function with L2-norm regularizer, given by:

$$J(\boldsymbol{\beta}) = \frac{1}{2} \sum_{i=1}^{L} (t_i - y_i)^2 + \frac{1}{2} \alpha ||\boldsymbol{\beta}||_2^2 \tag{17}$$

where α is the regularization parameter. The gradient of $J(\boldsymbol{\beta})$ with respect to $\boldsymbol{\beta}$ can be calculated as

$$\nabla J(\boldsymbol{\beta}) = \boldsymbol{S}\boldsymbol{S}^T \boldsymbol{\beta} - \boldsymbol{S}\boldsymbol{T}^T + \alpha \boldsymbol{\beta} \tag{18}$$

Let $\nabla J(\boldsymbol{\beta}) = 0$, which yields

$$(\boldsymbol{S}\boldsymbol{S}^T + \alpha \boldsymbol{I})\boldsymbol{\beta} = \boldsymbol{S}\boldsymbol{T}^T \tag{19}$$

where $\boldsymbol{I}$ denotes the identity matrix. The linear equation (19) has a symmetric and positive definite coefficient matrix, so it can be solved using Cholesky decomposition or singular value decomposition (SVD).

Otherwise, since $\boldsymbol{S}$ is a nonsquare matrix, a more direct solution to solve the linear equation $\boldsymbol{\beta}^T\boldsymbol{S} = \boldsymbol{T}$ is given by

$$\boldsymbol{\beta} = (\boldsymbol{T}\boldsymbol{S}^{\dagger}) \tag{20}$$

where $\boldsymbol{S}^{\dagger}$ is the Penrose-Moore generalized inverse of $\boldsymbol{S}$. This solution is a least squares solution with smallest norm, playing a similar role to the regularization items.

Iterative Solution of $\boldsymbol{\beta}$

To ensure the model can be learnt online, the fast stochastic gradient descent (SGD) method can be used. SGD optimizes $\boldsymbol{\beta}$ using one sample at a time by

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \frac{\eta_t}{2}\left(\frac{\partial((t_i - \boldsymbol{\beta}^T\boldsymbol{s}_i)^2 + \alpha||\boldsymbol{\beta}||_2^2)}{\partial\boldsymbol{\beta}}\right) \tag{21}$$

where η_t is the learning rate that can be constant or gradually decaying with the time step t. Convergence analysis usually requires it to satisfy the conditions $\sum_t \eta_t^2 < \infty$ and $\sum_t \eta_t = \infty$ (Zhang 2004; Bottou 2012). Commonly, it is given by $\eta_t = 1/(\alpha(t_0 + t))$, where t_0 is determined based on a heuristic (Bottou 2012). Thanks to the generalization performance analysis done by Buttoud Bottou(2010) and Bottou and Bousquet(2011), the unregularized version of (21) still possesses good generalization performance.

The computation complexity of SGD is $\mathcal{O}(kL\overline{p})$, where k is the number of iterations, L is the sample number, and $\overline{p}$ is the average number of nonzero attributes per sample. So, SGD can efficiently update the parameters in linear time.

Experiments Setup

Used Dataset

The used point cloud datasets are ample, shown in Table 1. *Planet* (Tong et al. 2013) and *quarry* are publicly available in http://asrl.utias.utoronto.ca/datasets/3dmap/index.html and http://www.pointcab-software.com/en/downloads/, respectively. The *mountain* is collected by ourselves. Datasets and code are open source in https://github.com/LiuXuSIA/rmm.

Experiments Settings

Some settings will be explained in the specific section of experiments result and *technical appendix*. Here are the important general settings.

- The results are averaged of 10-fold cross-validation.
- The model weight parameters $\boldsymbol{\beta}$ are solved by SVD.
- The models are trained by the terrain surfaces extracted from the noisy original data (Liu, Li, and He 2021).
- The experiments are conducted on a computer having an Intel i7 CPU with 16 cores of 3.8GHz, 64GB memory, and Windows 10 OS.

Evaluation Metrics

Mean of coefficient of determination (MR^2): fit goodness.
Mean of mean square error (MMSE): inference accuracy.
Mean of standard deviation (SMSE): inference stability.
Mapping time (MT): time spent in modeling the terrain.
Memory consumption (MC): occupied disk size.
Access time (AT): time spent in accessing the map.
unit: MMSE (m^2) MT&AT (s) MC (Mb)

name	data size	region size	elevation range
planet	731,937	24.9×28.5	-2.3-1.1
quarry	39,460,480	140×140	5.1-21.2
mountain	43,364,312	3172×3332	372.1-923.9

Table 1: The main properties of the used datasets.

Experiments Results

Validity of Random Mapping Method

We first conduct experiments to validate the effectiveness of the RM method, then evaluate the effects of activation function $g(v)$ and mapping dimension M on inference accuracy and time consumption. The following are the main results.

RM method is effective for regression tasks. Table 2 presents quantitative results of random mapping regression (RMR) and other regression techniques (see the *technical appendix*) when applied to the *planet* dataset, using two n-fold cross-validation experiments. In these experiments, the mapping dimension used by RMR is 850 and the random weights are sampled from the uniform distribution [-4,4]. Except for the training time, all data are acquired from test sets. MMSE and SMSE measure the accuracy and stability of these techniques' interpolation over the available data, and R^2 is a more intuitive representation of model fitting ability. As expected, the standard interpolation methods are unable to deal with this complex regression task for the bigger residual errors, regardless of the faster speed. The RMR and some kernel-based methods can achieve much better fitting results and much lower MMSEs, however, RMR consistently outperforms the kernel-based method in terms of MMSE while obtaining these results significantly faster in both training time and test time.

Noting that the kernel-based methods, especially the GP, suffer from a high computation complexity, even using an approximation method (Quinonero-Candela, Rasmussen, and Williams 2007) such as *kd*-trees (Vasudevan et al. 2009). In our implementation, we divide the whole areas into several sub-regions, using GMM, and train an individual GP model for each sub-region (see the *technical appendix*). This considerably decreases the training time and increases the accuracy, however, the GPR still needs much more time to generate the models. The time consumption presented in Table 2 validates RMR's scalability to the much larger datasets. We attribute the efficiency of RMR to its randomness and the closed-form solution, while the others depend on the sophisticated iterative solution.

Increasing RM dimensions can reinforce the model. Figure 1 shows the effects of the mapping dimension and activation function on the inference accuracy of RMR, both in the training set and the test set. As proved, the MMSE of RMR continues to increase until it approaches one as the mapping dimension increases. Generally, increasing the feature dimension will alleviate the problem of the model under-fitting, which is consistent with the Cover's Theorem (Cover 1965). Note that there is no evident difference between the performance of MMSE on the training set and the test set, enabling RMR a good generalization ability that

	Ten-fold cross-validation					Ten two-fold cross-validation				
	MR^2	MMSE	SMSE	TraT	TeT	R^2	MMSE	SMSE	TraT	TeT
RMR(sin 850)	**0.94**	**0.009**	**0.0003**	**0.826**	**0.036**	**0.94**	**0.009**	**0.0002**	**0.456**	**0.020**
RMR(sigmod 850)	**0.92**	**0.013**	**0.0010**	**0.804**	**0.035**	**0.91**	**0.014**	**0.0028**	**0.443**	**0.020**
GPR(rbf)	-2.39	0.564	0.0057	716.48	37.31	-2.42	0.565	0.0021	147.33	11.58
GPR(rq)	0.94	0.009	0.0002	15456.5	33.95	0.94	0.010	0.0002	3243.60	10.34
KRR(rbf)	0.94	0.010	0.0002	76.76	15.46	0.94	0.010	0.0001	18.99	4.81
NNR(relu)	0.79	0.035	0.0041	306.41	0.957	0.76	0.039	0.0063	118.87	0.541
SVR(rbf)	0.61	0.065	0.0012	9.43	3.90	0.60	0.066	0.0007	3.016	1.20
LR	0.44	0.091	0.0021	0.005	0.0001	0.45	0.090	0.0007	0.0015	0.0001
QL	0.49	0.085	0.0019	0.003	0.0003	0.49	0.085	0.0006	0.0014	0.0001
CL	0.52	0.079	0.0017	0.005	0.0004	0.52	0.079	0.0007	0.0026	0.0003

Table 2: The performances of different regression techniques in *planet*, including the Gaussian process regression (GPR) (Williams and Rasmussen 2006), kernel ridge regression (KRR) (Vovk 2013), neural network regression (NNR), support vector regression (KRR) (Smola and Schölkopf 2004), linear regression (LR), quadratic regression (QR), and cubic regression (CR) (Draper and Smith 1998). The brackets after the method are the *kernel function* or *activation function&mapping dimension* used by the method. (TraT: training time, TeT: test time, rq: *rational quadratic* kernel (refer to the *technical appendix*))

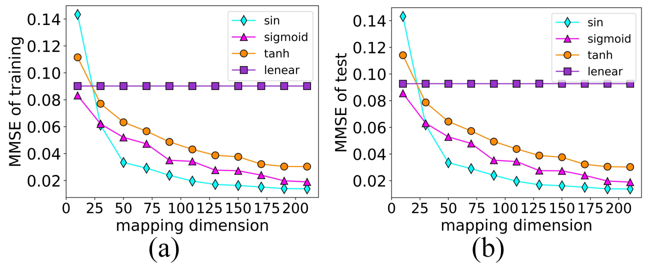

Figure 1: The effects of mapping dimension and activation function to the model inference accuracy. (a) MMSE of the training set. (b) MMSE of the test set.

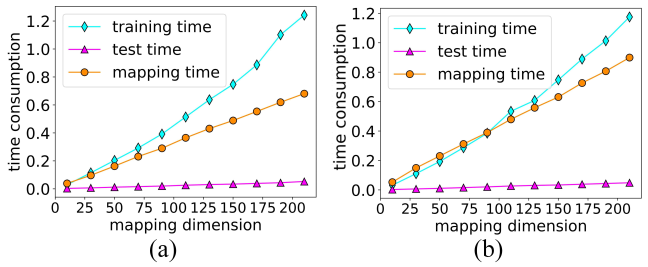

Figure 2: The effects of mapping dimension and activation function to time consumption. (a) Time consumption using *sine* function. (b) Time consumption using *tanh* function.

can infer the unobserved areas. But meanwhile, the time consumption, including the random mapping time, training time, and test time, also increases, as shown in Figure 2. This is reasonable since the larger mapping dimension will account for more complex matrix computing, particularly the matrix inversion. This requires us to make a trade-off between time and accuracy. In our experiments, once the activation function is selected, we treat the mapping dimension M as a hyper-parameter determined by trials. Note that a too large mapping dimension may account for over-fitting.

		MMSE	MC	MT	AT
P	**RMR(sin 600)**	**0.009**	**0.016**	**0.556**	**0.022**
	GPR(NN)	0.036	38.29	142.5	4.634
	VI	0.022	38.29	0.975	1.587
Q	**RMR(sin 800)**	**0.030**	**0.017**	**5.739**	**0.723**
	GPR(NN)	0.008	1576	967.7	60.74
	VI	0.003	1576	9.266	8.694

Table 3: Comparisons between RMR and other terrain modeling methods in *planet* (P) and *quarry* (Q). (NN: *neural network* kernel, Chapter 4 in (Williams and Rasmussen 2006))

Functions having at least (L-1)-order derivative can be the activation function. As shown in Figure 1, we use several different functions as the activation function, including the infinitely derivable and the one-order derivable. As expected, the linear function is incompetent to be an activation function for its terrible performance. The reason can be found in the proof section, that is, the linear function cannot guarantee the nonsingularity of the coefficient matrix $\boldsymbol{C}$. For the models with infinitely derivable activation functions, though the resulting performance is a little different, they are all generally competent, especially the *sine* function.

Evaluation of Terrain Modeling

The second set of experiments is to evaluate the capacity of RMR for terrain modeling, using *planet* and *quarry* datasets. Except for basic fidelity, we are concerned with the modeling efficiency and the processing of unobserved areas and uncertainties. We compare the results with the typical GP (Vasudevan et al. 2009) and state-of-the-art VI (Guizilini and Ramos 2019). Since the source code of VI is not available, we quote the results from the original paper (Table 1). Although RMR can model terrain online using SGD, we perform it offline to compare it with the two offline methods.

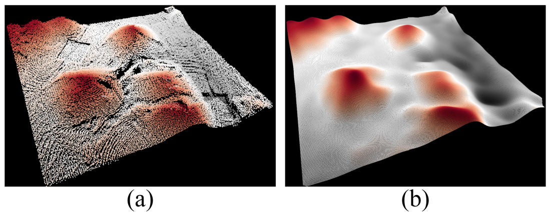
Figure 3: The *planet* terrain. (a) Terrain surface extrated from the original data. (b) Terrain restored by RMR.

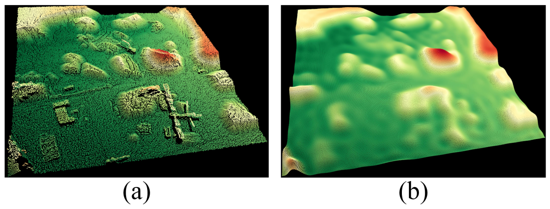
Figure 4: The *quarry* terrain. (a) Terrain surface extracted from the original data. (b) Terrain restored by RMR.

Modeling efficiency. Specifically, this contains three meanings, namely creation speed, access time, and memory consumption. Table 3 presents the modeling efficiency of different terrain modeling methods. As above, RMR consistently outperforms the other two methods in terms of mapping time, memory consumption, and access time. This attributes to the function approximation ability and fast solving process of RMR. More than this, as Bayesian methods, the GP and VI need to save all the original data for later inference, which causes bulky storage. From a practical point of view, this ought to be prohibited. Indeed, it is important to build a compact model in memory so that the large-scale environmental model can be kept in the robots itself and transmitted efficiently among multiple robots. As a result, RMR is more attractive for its parametric representation with limited model parameters to storage. These parametric models can be regarded as implicit representations of the terrains, from which the terrain maps can be efficiently accessed by substituting the querying point $\boldsymbol{g}(\boldsymbol{W}\boldsymbol{x} + b)$ into the parametric model. By contrast, even in the inference stage, the computation complexity of GP is $\mathcal{O}(n^2m)$, in which n and m are the numbers of training and test data, respectively.

Processing of unobserved areas. Occlusion and gaps, etc. within the environments to be modeled will account for the unobserved areas that may lead to potential hazards in the robot trajectory optimization. The powerful inference ability of RMR in the test datasets inspires us to complete the unobserved areas, using accurate interpolations. Figure 3 and Figure 4 shows the terrain maps of *planet* and *quarry* visualized by the terrain parameter models, using the points uniformly generated in the modeling areas. As expected, the unobserved areas can be completed. In the context of robotics path planning, this is helpful for the robots to plan a collision-free path. Otherwise, in exploration, it can guild the robot to the previously unobserved regions.

Particularly, the data uncertainty has been handled in the surface extraction process (Liu, Li, and He 2021). We won't describe it in detail here but in the *technical appendix*.

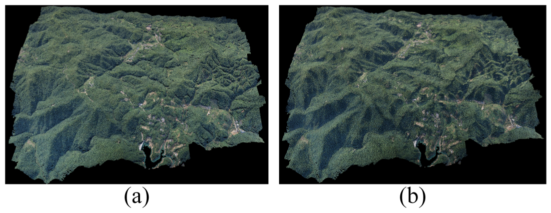
Figure 5: The *mountain* terrain. (a) Terrain surface extracted from the original data. (b) Terrain restored by RMR.

	1	4	8	12	16
R^2	0.994	0.993	0.994	0.994	0.994
MSE	3.220	3.631	3.219	3.244	3.270
Clock time	12.051	3.109	1.185	0.786	0.604
CPU time	12.051	11.345	9.012	8.823	8.957

Table 4: Model results with different amounts of parallelism.

Practical Application

We show the practicability of RMR for real large-scale terrain modeling with parallel computing. Here, the used *mountain* dataset is generated by applying structure from motion and stitching techniques (Hartley and Zisserman 2003) to multi-view images captured by a monocular camera attached to a UAV. Here, we randomly divide the training set into several equal subsets (Meng et al. 2019), and then simultaneously train an RMR model for each subset, using the multiple cores of CPU. The final model is obtained by averaging the parameters of all models. Table 4 presents the results with different numbers of division. For 1, all points are used for training one model, while for 16, the points are very sparse. These results validate the time-saving feature of parallel training, without losing much accuracy. Figure 5 shows the generated terrain maps, using the trained model with 16 divisions. High-fidelity terrain maps show the practicality of RMR toward modeling the large-scale environments. Particularly, the attached color information was obtained by the k-nearest neighbor in the original colored point cloud, which can be used to terrain classification (Silver et al. 2006).

Conclusion

In this paper, we first propose RMR and prove its linear distribution property mathematically strictly. The randomness and closed-form solutions make RMR very efficient. We validate the effectiveness of RMR and present its superiorities in terms of inference accuracy and computation time. Meanwhile, we illustrate the effects of the mapping dimension and activation function. Further, based on RMR, we propose to represent terrain in the form of a linear parameter model. This can outperform other methods in terms of mapping time, memory consumption, and access time. In practical applications, we show that RMR can resort to parallel training to greatly reduce the mapping time without losing much accuracy. In the future, we will first explore the relationship between RMR and data scale. Next, we will conduct path planning using this particular terrain representation.

Acknowledgments

This study was supported by the National Key R&D Program of China under Grant 2019YFB1310604, and in part by the National Natural Science Foundation of China under Grants 91948303, 91848203, and 61821005.

References

Bauer, D.; Kuhnert, L.; and Eckstein, L. 2019. Deep, spatially coherent Occupancy Maps based on Radar Measurements. In *AmE 2019-Automotive meets Electronics; 10th GMM-Symposium*, 1–6. VDE.

Bingham, E.; and Mannila, H. 2001. Random projection in dimensionality reduction: applications to image and text data. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 245–250.

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Boser, B. E.; Guyon, I. M.; and Vapnik, V. N. 1992. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, 144–152.

Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, 177–186. Springer.

Bottou, L. 2012. Stochastic gradient descent tricks. In *Neural networks: Tricks of the trade*, 421–436. Springer.

Bottou, L.; and Bousquet, O. 2011. 13 the tradeoffs of large-scale learning. *Optimization for machine learning*, 351.

Caron, R.; and Traynor, T. 2005. The zero set of a polynomial. *WSMR Report*, 05–02.

Cole, D. M.; and Newman, P. M. 2006. Using laser range data for 3D SLAM in outdoor environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 1556–1563. IEEE.

Cover, T. M. 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3): 326–334.

Doherty, K.; Shan, T.; Wang, J.; and Englot, B. 2019. Learning-aided 3-D occupancy mapping with Bayesian generalized kernel inference. *IEEE Transactions on Robotics*, 35(4): 953–966.

Doherty, K.; Wang, J.; and Englot, B. 2016. Probabilistic map fusion for fast, incremental occupancy mapping with 3d hilbert maps. In *2016 IEEE international conference on robotics and automation (ICRA)*, 1011–1018. IEEE.

Draper, N. R.; and Smith, H. 1998. *Applied regression analysis*, volume 326. John Wiley & Sons.

Dudley, R. M. 2018. *Real analysis and probability*. CRC Press.

Fitzpatrick, P. 2009. *Advanced calculus*, volume 5. American Mathematical Soc.

Fleming, W. 2012. *Functions of several variables*. Springer Science & Business Media.

Greub, W. H. 2012. *Linear algebra*, volume 23. Springer Science & Business Media.

Guizilini, V.; and Ramos, F. 2018a. Learning to reconstruct 3D structures for occupancy mapping from depth and color information. *The International Journal of Robotics Research*, 37(13-14): 1595–1609.

Guizilini, V.; and Ramos, F. 2018b. Towards real-time 3D continuous occupancy mapping using Hilbert maps. *The International Journal of Robotics Research*, 37(6): 566–584.

Guizilini, V.; and Ramos, F. 2019. Variational Hilbert regression for terrain modeling and trajectory optimization. *The International Journal of Robotics Research*, 38(12-13): 1375–1387.

Guizilini, V.; Senanayake, R.; and Ramos, F. 2019. Dynamic hilbert maps: Real-time occupancy predictions in changing environments. In *2019 International Conference on Robotics and Automation (ICRA)*, 4091–4097. IEEE.

Hartley, R.; and Zisserman, A. 2003. *Multiple view geometry in computer vision*. Cambridge university press.

Hofmann, T.; Schölkopf, B.; and Smola, A. J. 2008. Kernel methods in machine learning. *The annals of statistics*, 36(3): 1171–1220.

Horn, R. A.; and Johnson, C. R. 2012. *Matrix analysis*. Cambridge university press.

Hornung, A.; Wurm, K. M.; Bennewitz, M.; Stachniss, C.; and Burgard, W. 2013. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous robots*, 34(3): 189–206.

Jadidi, M. G.; Miro, J. V.; and Dissanayake, G. 2018. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, 42(2): 273–290.

Kaski, S. 1998. Dimensionality reduction by random mapping: Fast similarity computation for clustering. In *1998 IEEE International Joint Conference on Neural Networks Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98CH36227)*, volume 1, 413–418. IEEE.

Krotkov, E.; and Hoffman, R. 1994. Terrain mapping for a walking planetary rover. *IEEE Transactions on Robotics and Automation*, 10(6): 728–739.

Le Gentil, C.; Vayugundla, M.; Giubilato, R.; Stürzl, W.; Vidal-Calleja, T.; and Triebel, R. 2020. Gaussian process gradient maps for loop-closure detection in unstructured planetary environments. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1895–1902. IEEE.

Lee, B.; Zhang, C.; Huang, Z.; and Lee, D. D. 2019. Online continuous mapping using gaussian process implicit surfaces. In *2019 International Conference on Robotics and Automation (ICRA)*, 6884–6890. IEEE.

Li, B.; Wang, Y.; Zhang, Y.; Zhao, W.; Ruan, J.; and Li, P. 2020. GP-SLAM: laser-based SLAM approach based on regionalized Gaussian process map reconstruction. *Autonomous Robots*, 1–21.

Liu, X.; Li, D.; and He, Y. 2021. Multiresolution Representations for Large-Scale Terrain with Local Gaussian Process Regression. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 5497–5503. IEEE.

Long, J.; Shelhamer, E.; and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3431–3440.

Meng, Q.; Chen, W.; Wang, Y.; Ma, Z. M.; and Liu, T. Y. 2019. Convergence analysis of distributed stochastic gradient descent with shuffling. *Neurocomputing*, 337(APR.14): 46–57.

O'Meadhra, C.; Tabib, W.; and Michael, N. 2018. Variable resolution occupancy mapping using gaussian mixture models. *IEEE Robotics and Automation Letters*, 4(2): 2015–2022.

Pao, Y.-H.; Park, G.-H.; and Sobajic, D. J. 1994. Learning and generalization characteristics of the random vector functional-link net. *Neurocomputing*, 6(2): 163–180.

Pao, Y.-H.; and Takefuji, Y. 1992. Functional-link net computing: theory, system architecture, and functionalities. *Computer*, 25(5): 76–79.

Popović, M.; Vidal-Calleja, T.; Hitz, G.; Sa, I.; Siegwart, R.; and Nieto, J. 2017. Multiresolution mapping and informative path planning for UAV-based terrain monitoring. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1382–1388. IEEE.

Quinonero-Candela, J.; Rasmussen, C. E.; and Williams, C. K. 2007. Approximation methods for Gaussian process regression. In *Large-scale kernel machines*, 203–223. MIT Press.

Rahimi, A.; Recht, B.; et al. 2007. Random Features for Large-Scale Kernel Machines. In *NIPS*, volume 3, 5. Citeseer.

Ramos, F.; and Ott, L. 2016. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. *The International Journal of Robotics Research*, 35(14): 1717–1730.

Rekleitis, I.; Bedwani, J.-L.; Gingras, D.; and Dupuis, E. 2009. Experimental Results for Over-the-Horizon Planetary exploration using a LIDAR sensor. In *Experimental Robotics*, 65–77. Springer.

Schilling, F.; Chen, X.; Folkesson, J.; and Jensfelt, P. 2017. Geometric and visual terrain classification for autonomous mobile navigation. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2678–2684. IEEE.

Schmidt, W. F.; Kraaijveld, M. A.; Duin, R. P.; et al. 1992. Feed forward neural networks with random weights. In *International Conference on Pattern Recognition*, 1–1. IEEE Computer Society Press.

Senanayake, R.; Ott, L.; O'Callaghan, S.; and Ramos, F. 2016. Spatio-temporal hilbert maps for continuous occupancy representation in dynamic environments. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, 3925–3933.

Silver, D.; Sofman, B.; Vandapel, N.; Bagnell, J. A.; and Stentz, A. 2006. Experimental analysis of overhead data processing to support long range navigation. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2443–2450. IEEE.

Smola, A. J.; and Schölkopf, B. 2004. A tutorial on support vector regression. *Statistics and computing*, 14(3): 199–222.

Srivastava, S.; and Michael, N. 2018. Efficient, multifidelity perceptual representations via hierarchical gaussian mixture models. *IEEE Transactions on Robotics*, 35(1): 248–260.

Stork, J. A.; and Stoyanov, T. 2020. Ensemble of sparse gaussian process experts for implicit surface mapping with streaming data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 10758–10764. IEEE.

Tong, C. H.; Gingras, D.; Larose, K.; Barfoot, T. D.; and Dupuis, É. 2013. The Canadian planetary emulation terrain 3D mapping dataset. *The International Journal of Robotics Research*, 32(4): 389–395.

Triebel, R.; Pfaff, P.; and Burgard, W. 2006. Multi-level surface maps for outdoor terrain mapping and loop closing. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, 2276–2282. IEEE.

Valada, A.; Spinello, L.; and Burgard, W. 2018. Deep feature learning for acoustics-based terrain classification. In *Robotics research*, 21–37. Springer.

Vasudevan, S.; Ramos, F.; Nettleton, E.; and Durrant-Whyte, H. 2009. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics*, 26(10): 812–840.

Vovk, V. 2013. Kernel ridge regression. In *Empirical inference*, 105–116. Springer.

Williams, C.; and Seeger, M. 2001. Using the Nyström method to speed up kernel machines. In *Proceedings of the 14th annual conference on neural information processing systems*, CONF, 682–688.

Williams, C. K.; and Rasmussen, C. E. 2006. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA.

Zhang, T. 2004. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In *Proceedings of the twenty-first international conference on Machine learning*, 116.

Zhou, R.; Ding, L.; Gao, H.; Feng, W.; Deng, Z.; and Li, N. 2019. Mapping for Planetary Rovers from Terramechanics Perspective. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1869–1874. IEEE.