Fast 3D Modeling with Approximated Convolutional Kernels

Vitor Guizilini and Fabio Ramos
School of Information Technologies
University of Sydney, Australia
{vitor.guizilini;fabio.ramos}@sydney.edu.au

Abstract: This paper introduces a novel regression methodology for 3D reconstruction, with applications in robotics tasks such as terrain modeling and implicit surface calculation. The proposed methodology is based on projections into a high-dimensional space, that is able to fit arbitrarily complex data as a continuous function using a series of kernel evaluations within a linear regression model. We avoid direct kernel calculation by employing a novel sparse random Fourier feature vector, that approximates any shift-invariant kernel as a series of dot products relative to a set of inducing points placed throughout the input space. The varying properties of these inducing points produce non-stationarity in the resulting model, and can be jointly learned alongside linear regression weights. Furthermore, we show how convolution with arbitrary kernels can be performed directly in this high-dimensional continuous space, by training a neural network to learn the Fourier transform of the convolutional output based on information from the input kernels. Experimental results in terrain modeling and implicit surface calculation show that the proposed framework is able to outperform similar techniques in terms of computational speed without sacrificing accuracy, while enabling efficient convolution with arbitrary kernels for tasks such as global localization and template matching within these applications.

Keywords: Kernel Methods, Random Fourier Features, High Dimensional Projections, Terrain Modeling, Implicit Surfaces

1 Introduction

Any autonomous task is predicated on a system's ability to sense its surroundings, and more specifically on its ability to process collected data in order to generate an accurate internal representation of the external world. For example, in autonomous navigation an accurate terrain model is crucial [1, 2], since it determines which areas are safe for the vehicle to traverse and which ones would result in an unfavorable outcome. Similarly, tasks such as grasping [3] or shape estimation [4] require proper surface modeling, so the system is aware of how it should interpret and interact with the observed object. Amongst the main challenges in creating such models are the incorporation of spatial dependencies, data incompleteness and handling details in unstructured areas, which becomes even more challenging when dealing with large-scale scenarios, which is the case for most current real-world applications. The evolution of sensor technology has also recently led to a jump from two to three-dimensional datasets, that are better in representing real world structures than a single slice. Unfortunately, this creates an increase in computational complexity that many of the currently available techniques are still unable to properly manage, especially under real-time constraints, which is often the case in robotics applications.

The Gaussian Process (GPs) framework [5] has become increasingly popular for such modeling tasks in recent years [2, 3, 6], since it incorporates spatial dependencies using covariance functions, naturally encodes uncertainty and is able to produce a continuous function that can be queried at arbitrary resolutions. The standard GP framework, however, scales cubically with the number of training points, thus rendering it computationally too expensive for large-scale datasets, and most kernel covariance functions are too smooth to correctly details sudden discontinuities. Non-stationary [7, 8] and energy-based [9] covariance functions can be used to produce variable smoothness throughout the input space, and sparse approximations [10, 11, 1] reduce computational complexity by projecting data into a subset of inducing points, or using model ensembles. The Hilbert Maps framework, initially proposed in [12], addresses these limitations by introducing high-dimensional projections and using simple classifiers, combined with stochastic gradient descent [13], to achieve efficient continuous non-stationary occupancy mapping in large-scale 3D environments [14]. This methodology has recently been extended to address terrain modeling tasks in [15] using variational Bayesian inference.

This paper introduces a novel regression methodology for 3D reconstruction, based on high-dimensional projections, that further improves on the computational efficiency and modeling capabilities of currently available techniques. We propose a novel feature vector that approximates kernel evaluations using random Fourier features relative to a set of inducing points, while enforcing sparsity by considering only a subset of nearest neighbors. Different kernels are approximated using samples drawn from specific probability distributions [16], and by selecting the number of random samples we can create a trade-off between accuracy and performance for different tasks. Furthermore, we show how convolution between any two kernels can be achieved within the proposed framework by training a neural network to learn the transformation between input functions and the resulting activation values directly in the random Fourier space. In contrast to standard discrete convolution, which scales exponentially relative to the number of input dimensions [17], the proposed continuous convolution methodology only depends on feature vector dimensionality and is orders of magnitude faster, while allowing the convolution of any two shift-invariant kernels. We apply these convolutional results to the tasks of global localization within terrain modeling and template matching within implicit surfaces, using raw sensor data as input, which would be infeasible using discrete convolution due to the large mask sizes and need for large-scale space discretization.

2 Preliminaries

This section provides a brief overview of the theoretical background that constitutes the basis for the proposed regression and convolution methodology. We start by describing the technique for high-dimensional data projection, including the choice of feature vector and the training and inference equations. We then move on to an overview of random Fourier features and how they can be applied to the approximation of kernels via the dot product of randomized vectors.

2.1 High-Dimensional Projections

Regression within the proposed framework is based on projections into a high-dimensional Reproducing Kernel Hilbert Space (RKHS) [18], using a feature vector whose dot product approximates popular kernels found in the literature [5]. By operating only in terms of kernel evaluations we never have to explicitly perform calculations in this high-dimensional (and potentially infinite) feature space. Furthermore, it can be shown [19] that in sufficiently high-dimensional spaces any function can be expressed using a a simple linear regression model:

$$f(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \Phi(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is the input point, $\Phi(\mathbf{x})$ is its corresponding feature vector and $\mathbf{w}$ contains weight parameters. Given a training dataset $\mathcal{D} = \{X, \mathbf{y}\} = \{\mathbf{x}_n, y_n\}_{n=1}^N$, where $\mathbf{x}_n \in \mathcal{R}^D$ are input points (i.e. spatial coordinates of observed data) and $y_n \in \mathcal{R}^1$ are their corresponding outputs (i.e. elevation values for terrain modeling or surface levels for implicit surface calculation), these weight parameters are optimized by minimizing the least-squares error function, using stochastic gradient descent [13] on individual training points or mini-batches:

$$\mathcal{L} = \sum_{n=1}^{N} \left(f(\mathbf{x}, \mathbf{w}) - y_n \right)^2 + R(\mathbf{w}). \tag{2}$$

In the above equation, $R(\mathbf{w})$ is a regularizer, such as the elastic net [20], used to prevent overfitting and promote sparseness in $\mathbf{w}$. The selection of which feature vector to use is critical, since it determines how input data will be projected into this high-dimensional space. Here we employ a non-stationary sparse feature vector, as described in [21], which is defined as:

$$\Phi(\mathbf{x}, \mathcal{M}) = [k(\mathbf{x}, \mathcal{M}_1), k(\mathbf{x}, \mathcal{M}_2), \ldots, k(\mathbf{x}, \mathcal{M}_M)]^T, \tag{3}$$

where $\mathcal{M} = \{\boldsymbol{\mu}_i, \boldsymbol{\theta}_i\}_{i=1}^M$ is a set of inducing points used to correlate different portions of the input space, each composed of a location $\boldsymbol{\mu} \in \mathcal{R}^D$ and a vector $\boldsymbol{\theta}$ containing parameters for kernel k, both of which can also be optimized during the training process alongside $\mathbf{w}$. The choice of kernel is task-dependent, with different kernels having their own parameter sets and properties that are better suitable for certain applications [5]. Inducing points can be placed throughout the input space on a grid or using clustering techniques [21], and to improve computational speed only the k-nearest neighbors can be considered for feature vector calculation (all other coefficients are set to zero).

2.2 Random Fourier Features

Formally, a kernel $k(\mathbf{x}_i, \mathbf{x}_j)$ with parameters $\boldsymbol{\theta}$ defines a Hilbert space with inner product $\langle ., . \rangle$ based on a feature vector $\zeta_{\mathbf{s}}(\mathbf{x})$, such that $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \zeta_{\mathbf{s}}(\mathbf{x}_i), \zeta_{\mathbf{s}}(\mathbf{x}_j) \rangle$. If said kernel is shift-invariant (i.e. stationary) it can be rewritten as $k(\boldsymbol{\tau})$, where $\boldsymbol{\tau} = \mathbf{x}_i - \mathbf{x}_j$. The Bochner's Theorem [22] can then be applied to create a representation in terms of the probability distribution $p_k(\mathbf{s}|\boldsymbol{\theta})$ defined by its Fourier transform. Defining $\zeta_{\mathbf{s}}(\mathbf{x}) = e^{i\mathbf{s}^T\mathbf{x}}$, we have:

$$k(\boldsymbol{\tau}) = \int_{\mathcal{R}^D} p_k(\mathbf{s}|\boldsymbol{\theta}) e^{i\mathbf{s}^T\boldsymbol{\tau}} d\mathbf{s} = E_{\mathbf{s}}[\zeta_{\mathbf{s}}(\mathbf{x}_i)^T \zeta_{\mathbf{s}}(\mathbf{x}_j)], \tag{4}$$

so that $\zeta_{\mathbf{s}}(\mathbf{x}_i)^T \zeta_{\mathbf{s}}(\mathbf{x}_j)$ is an unbiased estimate of $k(\mathbf{x}_i, \mathbf{x}_j)$ when $\mathbf{s} \in \mathcal{R}^D$ is drawn from $p_k(\boldsymbol{\theta})$. In order to lower the variance of this estimate, a set $\mathbf{S} = \{\mathbf{s}_1, \dots, \mathbf{s}_R\}$ of R randomly chosen features $\zeta_{\mathbf{s}}(\mathbf{x})$ can be concatenated and normalized by $\sqrt{R}$. The inner product of $\boldsymbol{\zeta}_{\mathbf{s}}(\mathbf{x}_i)^T \boldsymbol{\zeta}_{\mathbf{s}}(\mathbf{x}_j) = \frac{1}{R}\sum_{u=1}^{R} \zeta_{\mathbf{s}_u}(\mathbf{x}_i)^T \zeta_{\mathbf{s}_u}(\mathbf{x}_j)$ is a sampled average of $\zeta_{\mathbf{s}}(\mathbf{x}_i)^T \zeta_{\mathbf{s}}(\mathbf{x}_j)$ and will therefore be a more robust approximation to Equation 4. One possible mapping ϕ_k that produces real values and satisfies the condition $E_{\mathbf{s}}\left[\phi_k(\mathbf{x}_i, \boldsymbol{\theta}), \phi_k(\mathbf{x}_j, \boldsymbol{\theta})\right] = k(\mathbf{x}_i, \mathbf{x}_j, \boldsymbol{\theta})$ is the following:

$$\phi_k(\mathbf{x}, \boldsymbol{\theta}) = \phi_k(\mathbf{x}, \mathbf{S} \sim p_k(\boldsymbol{\theta})) = \frac{1}{\sqrt{R}} \left[\cos(\mathbf{s}_1^T\mathbf{x}), \dots, \cos(\mathbf{s}_R^T\mathbf{x}), \sin(\mathbf{s}_1^T\mathbf{x}), \dots, \sin(\mathbf{s}_R^T\mathbf{x})\right], \tag{5}$$

Different probabilistic distributions for $p_k(\boldsymbol{\theta})$ approximate different kernels, and a comprehensive list can be found in [23]. This is similar to the work of Raimi and Recht [24], with approximation bounds for general learning problems presented in [16].

3 Methodology

This section describes the proposed continuous modeling and convolution technique, for arbitrary kernels. Initially a novel feature vector is introduced, that combines the properties of the sparse feature vector in Equation 3 and the random Fourier feature vector in Equation 5. Afterwards, we show how convolution can be performed using this novel feature vector, by learning a function that maps two sets of random Fourier features into a third set that approximates convolutional results, within the proposed regression framework.

3.1 Sparse Random Fourier Feature Vector

The main insight of the proposed feature vector is that, for a query point $\mathbf{x}_*$, rather than directly performing its kernel evaluations in relation to $\mathcal{M}$, as shown in Equation 3, we maintain two separate components, one pertaining to the inducing set and another to the query point. The feature vector for the inducing set is defined as:

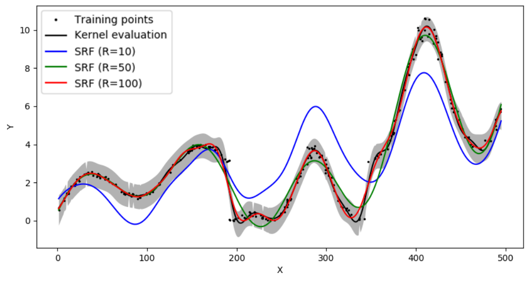
$$\Phi_k(\Lambda, \Theta) = \Phi_k(\mathcal{M}) = [\phi_k(\boldsymbol{\mu}_1, \boldsymbol{\theta}_1), \phi_k(\boldsymbol{\mu}_2, \boldsymbol{\theta}_2), \dots, \phi_k(\boldsymbol{\mu}_M, \boldsymbol{\theta}_M)]^T, \tag{6}$$

where $\Lambda = \{\boldsymbol{\mu}_1, \dots, \boldsymbol{\mu}_M\}$ and $\Theta = \{\boldsymbol{\theta}_1, \dots, \boldsymbol{\theta}_M\}$ are respectively the sets containing all kernel locations and parameters for $\mathcal{M}$. This feature vector is of size $M \times R$ and contains the random Fourier features for each inducing point location relative to all the kernels we are approximating, based on the information contained in Θ. Similarly, the feature vector for the query point is defined as:

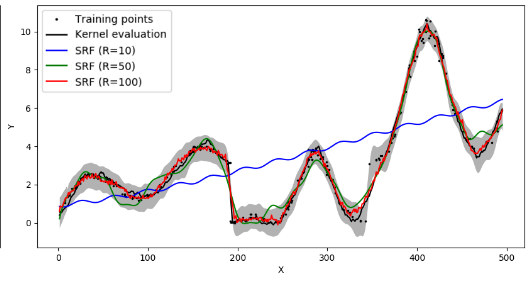
$$\Phi_k(\mathbf{x}_*, \Theta) = [\phi_k(\mathbf{x}_*, \boldsymbol{\theta}_1), \phi_k(\mathbf{x}_*, \boldsymbol{\theta}_2), \dots, \phi_k(\mathbf{x}_*, \boldsymbol{\theta}_M)]^T, \tag{7}$$

which contains the random Fourier features for $\mathbf{x}_*$ relative to all the kernels we are approximating, based on the information contained in Θ. The output value for a query point is now simply a matter of calculating the dot product between these two feature vectors multiplied by the weight parameters that define the function within this high-dimensional projection (see Equation 1):

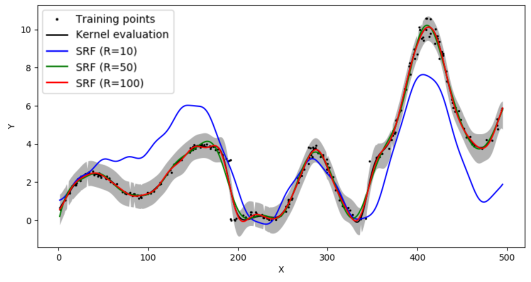
$$f(\mathbf{x}, \mathbf{w}, \mathcal{M}) = \sum_{m=1}^{M} w_m \phi_k(\boldsymbol{\mu}_m, \theta_m)^T \phi_k(\mathbf{x}_*, \theta_m) = ||\mathbf{w} \cdot \Phi(\mathcal{M}) \cdot \Phi(\mathbf{x}_*, \Theta)||_1. \tag{8}$$



(a) Gaussian kernel.



(b) Matérn 1/2 kernel.



(c) Cauchy kernel.

# Random	Gauss	Mat.1/2	Cauchy
$R = 5$	1.553	2.123	1.568
$R = 10$	0.423	1.782	1.545
$R = 25$	0.377	0.544	0.383
$R = 50$	0.341	0.494	0.328
$R = 100$	0.311	0.334	0.319
$R = 250$	0.294	0.279	0.295
Kernel	0.279	0.253	0.287

(d) Final RMSE values (average of 20 runs).

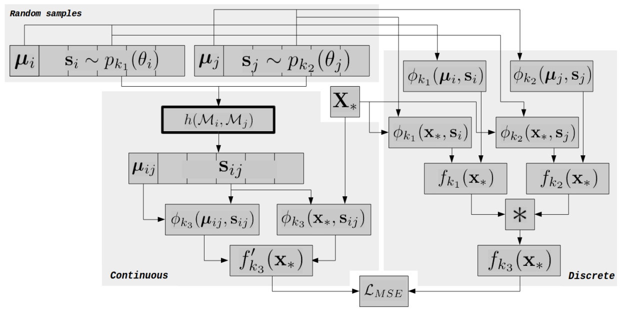
Figure 1: Regression results using the proposed methodology, using different kernels and number of random samples (gray areas are 95% confidence boundaries from a Sparse Gaussian Process [11], with the same inducing set and choice of kernel).

Note that $\mathbf{w} \cdot \Phi(\mathcal{M}) = \Phi(\mathbf{w}, \mathcal{M})$ is independent of the selection of query point, and thus can be stored for multiple calculations. Examples of regression results obtained using the proposed sparse random Fourier (SRF) feature vector, for different kernels and number of random samples, can be found in Figure 1, alongside results using direct kernel calculation. As expected, the approximations quickly converge to the true kernel values, with $R = 100$ already achieving root mean squared errors (RMSE) below 10%. The number of random samples can be used as a trade-off between accuracy (better approximated values) and performance (faster computation), depending on the application.

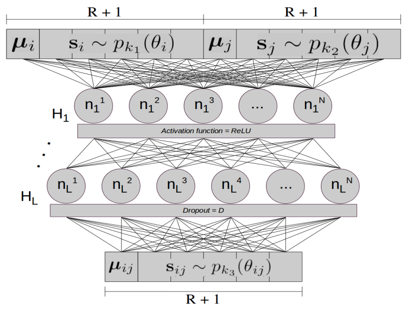
3.2 Convolution with Sparse Random Fourier Features

The work of [25] introduced a methodology for continuous convolution in high-dimensional projections that exploits the well-known mathematical fact [26] that the convolution of two Gaussian distributions $\mathcal{N}_i$ and $\mathcal{N}_j$ is also a Gaussian distribution $\mathcal{N}_{ij}$, composed by the sum of input mean and variance values. Although this results in a very efficient closed-form solution to the convolution problem, it is limited to that particular type of kernel, since others do not have similar properties. In this paper we circumvent this limitation by introducing a function $h(.,.)$ that is trained using random samples $\{\boldsymbol{\mu}_i, \mathbf{S}_i \sim p_{k_1}(\theta_i)\}$ and $\{\boldsymbol{\mu}_j, \mathbf{S}_j \sim p_{k_2}(\theta_j)\}$, generated from the possible parameters in both input kernels k_1 and k_2, to approximate the random samples $\{\boldsymbol{\mu}_{ij}, \mathbf{S}_{ij} \sim p_{k_3}(\theta_{ij})\}$ corresponding to a hypothetical kernel k_3 with parameters θ_{ij} that model the convolutional output. As reference for training we use discrete convolution over the two input kernels in relation to a vector $\mathbf{X}_*$, composed of query points equally spaced throughout the input space, as shown in Figure 2a. The mean squared error loss function $\mathcal{L}_{MSE}$ between discrete $f_{k_3}(\mathbf{X}_*)$ and continuous $f'_{k_3}(\mathbf{X}_*)$ approximations is minimized to produce an optimized function $h(.,.)$ capable of performing convolution directly in the random feature space, without discretization or any assumptions regarding the input kernels.

In fact, note that we do not have direct access to the nature of k_3 or its parameters θ_{ij}, they are encoded into the random samples $\mathbf{S}_{ij}$ that directly approximate convolution between the two selected input kernels. An extension of this concept would allow the convolution of two unknown input kernels, given only their locations $\boldsymbol{\mu}_i$ and $\boldsymbol{\mu}_j$ and random samples $\mathbf{S}_i$ and $\mathbf{S}_j$. The convolution between two SRF feature vectors $\Phi_{k_1}(\mathbf{w}_i, \mathcal{M}_i)$ and $\Phi_{k_2}(\mathbf{w}_j, \mathcal{M}_j)$, as depicted in Algorithm 1, produces a third feature vector $\Phi_{k_3}(\mathbf{w}_{ij}, \mathcal{M}_{ij}) = \Phi_{k_3}(\mathbf{w}_{ij}, \Lambda_{ij}, \Sigma_{ij})$, where $\Sigma_{ij} = \{\mathbf{S}_1, \cdots, \mathbf{S}_M\}$ is the collection of random samples that approximate the unknown output kernel k_3 for each inducing point in $\mathcal{M}_{ij}$.


Random samples
$\boldsymbol{\mu}_i$
$\mathbf{s}_i \sim p_{k_1}(\theta_i)$
$\boldsymbol{\mu}_j$
$\mathbf{s}_j \sim p_{k_2}(\theta_j)$
$\phi_{k_1}(\boldsymbol{\mu}_i, \mathbf{s}_i)$
$\phi_{k_2}(\boldsymbol{\mu}_j, \mathbf{s}_j)$
$\mathbf{X}_*$
$h(\mathcal{M}_i, \mathcal{M}_j)$
$\phi_{k_1}(\mathbf{x}_*, \mathbf{s}_i)$
$\phi_{k_2}(\mathbf{x}_*, \mathbf{s}_j)$
$\boldsymbol{\mu}_{ij}$
$\mathbf{s}_{ij}$
$f_{k_1}(\mathbf{x}_*)$
$f_{k_2}(\mathbf{x}_*)$
$\phi_{k_3}(\boldsymbol{\mu}_{ij}, \mathbf{s}_{ij})$
$\phi_{k_3}(\mathbf{x}_*, \mathbf{s}_{ij})$
$*$
$f'_{k_3}(\mathbf{x}_*)$
$f_{k_3}(\mathbf{x}_*)$
Continuous
$\mathcal{L}_{MSE}$
Discrete

(a) Diagram of the proposed training convolutional methodology.


R + 1
R + 1
$\boldsymbol{\mu}_i$
$\mathbf{s}_i \sim p_{k_1}(\theta_i)$
$\boldsymbol{\mu}_j$
$\mathbf{s}_j \sim p_{k_2}(\theta_j)$
H_1
n_1^1 n_1^2 n_1^3 ... n_1^N
Activation function = ReLU
H_L
n_L^1 n_L^2 n_L^3 n_L^4 ... n_L^N
Dropout = D
$\boldsymbol{\mu}_{ij}$
$\mathbf{s}_{ij} \sim p_{k_3}(\theta_{ij})$
R + 1

(b) Neural network topology for $h(.,.)$.

Figure 2: Training methodology for the proposed continuous convolution technique for arbitrary kernels using SRF feature vectors.

L	N	Gauss			Matérn 1/2			Cauchy		
		1D	2D	3D	1D	2D	3D	1D	2D	3D
1	100	3.18	4.11	4.91	3.84	4.75	5.45	3.67	4.58	4.75
	75	6.55	8.09	17.85	4.18	8.12	21.22	3.45	6.07	18.60
	50	15.34	26.60	44.13	6.79	45.23	67.08	5.18	40.38	48.72
	25	21.06	33.93	59.32	24.12	60.51	77.36	21.65	56.25	65.93
2	100	0.81	1.14	1.41	0.97	1.22	1.54	0.86	1.22	1.57
	75	0.86	2.03	7.78	0.99	3.29	9.79	0.91	2.24	8.19
	50	1.48	17.34	21.14	2.31	21.57	32.39	1.54	19.44	23.36
	25	9.35	25.88	44.89	11.06	34.07	72.67	9.74	27.15	46.54

Table 1: Approximated $\mathcal{L}_{MSE}$ ($\times 10^{-3}$) convolutional test results for different network topologies, using $R = 100$ as the number of random samples and $D = 80\%$ as dropout value.

For each inducing point in $\mathcal{M}_i$, the inducing set $\mathcal{M}_j$ is translated to its location and the approximated convolution is calculated between each of its points and their corresponding nearest neighbors in $\mathcal{M}_i$, which is used to update output weights. The resulting feature vector can be used to query the approximated convolutional result in any location $\mathbf{x}_*$, according to Equation 7.

In this work we propose a neural network as the mapping function $h(.,.)$, that takes as input the concatenation of both input vectors $\{\boldsymbol{\mu}_i, \mathbf{s}_i\}$ and $\{\boldsymbol{\mu}_j, \mathbf{s}_j\}$, which goes through a series of fully-connected layers with ReLU [27] activation functions and dropout [28], and returns as output another vector containing location and random samples $\{\boldsymbol{\mu}_{ij}, \mathbf{s}_{ij}\}$ approximating convolutional results. In the interest of computational efficiency, we focused on simpler architectures, with a template shown in Figure 2b and results using different network parameters available in Table 1. From these results we can see that even such simple architectures are already able to produce accurate convolution approximations, especially with the introduction of a second hidden layer (adding further layers did not produce significant improvements). Higher dimensions also require a larger number of hidden

Algorithm 1 Pseudo-code for the continuous convolution between two SRF feature vectors

```
 1: Input: Two feature vectors $\Phi_{k_1}(\mathbf{w}_i, \mathcal{M}_i)$ and $\Phi_{k_2}(\mathbf{w}_j, \mathcal{M}_j)$ representing input functions
 2: Output: Feature vector $\Phi_{k_3}(\mathbf{w}_{ij}, \mathcal{M}_{ij})$ representing the convolutional output
 3: $\Lambda_{ij}$ , $\Sigma_{ij}$ , $\mathbf{w}_{ij} = \Lambda_i$ , $\Sigma_i$ , $\mathbf{0}$   % Initialize output values
 4: for $u = 0$ to $M_i$ do
 5:     for $v = 0$ to $M_j$ do
 6:         $\boldsymbol{\mu}^{uv} \leftarrow \boldsymbol{\mu}_i^u + \boldsymbol{\mu}_j^v$   % Calculate translated location values
 7:         $\mathbf{n} \leftarrow k$-nearest neighbors of $\mathcal{M}_i$ in $\boldsymbol{\mu}^{uv}$   % Index vector of $k$ nearest neighbors
 8:         for $n$ in $\mathbf{n}$ do
 9:             $\boldsymbol{\mu}_{ij}$ , $\mathbf{s}_{ij}$ $\leftarrow$ $h(\boldsymbol{\mu}_i^n, \mathbf{s}_i^n, \boldsymbol{\mu}^{uv}, \mathbf{s}_j^v)$   % Calculate convolution approximation
10:             $\mathbf{w}_{ij}^u$ += $\mathbf{w}_i^n \cdot \mathbf{w}_j^v \cdot \phi_{k_3}(\boldsymbol{\mu}_{ij}, \mathbf{s}_{ij})$   % Update output weights
11:         end for
12:     end for
13: end for
```

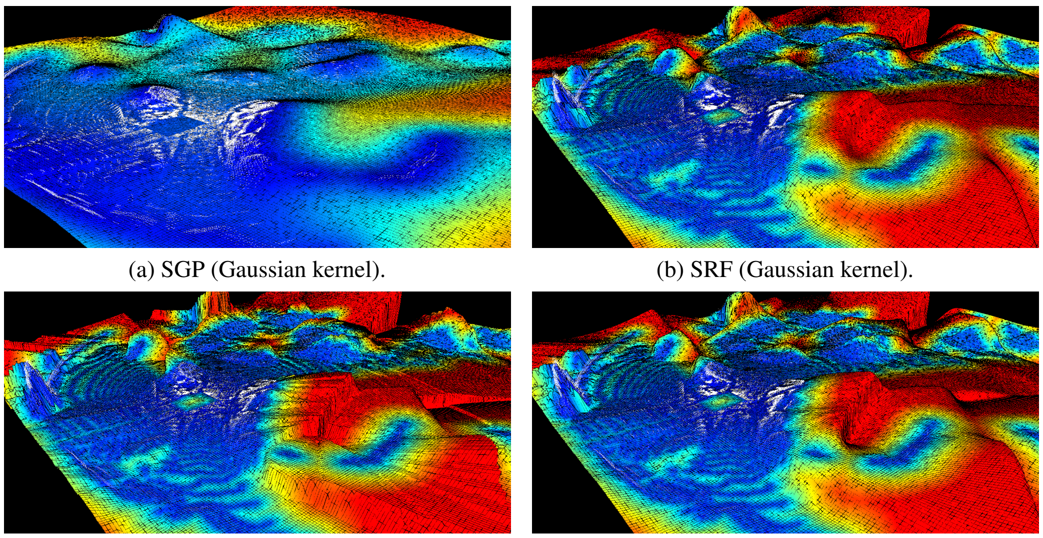
nodes for a proper convergence, regardless of kernel type. The introduction of different normalization, regularization and optimization techniques might lead to further improvements, however a deeper analysis is left for future work.

4 Experiments

Two key tasks in robotics are explored as experimental validation for the proposed SRF framework: terrain modeling and implicit surface calculation. These two applications are very similar, in the sense that they both map input spatial coordinates, respectively 2D and 3D points, into a corresponding value, respectively elevation values and surface levels. In both cases, we explore the benefits produced by the proposed SRF framework relative to other commonly used techniques, including a comparison with direct kernel calculation, the use of different kernels and convolution in random feature space, as a way to achieve tasks such as global localization and template matching.

4.1 Terrain Modeling

To test the terrain modeling capabilities of the proposed SRF regression methodology, we used a dataset obtained from the website `http://asrl.utias.utoronto.ca/datasets/3dmap/a100_dome.html`, entitled *Terrain*. It contains 252616 points obtained from a laser sensor with a range of roughly $30m$, however due to terrain obstructions a large portion remains unobserved (i.e. without data points). The final terrain model obtained using a Sparse Gaussian Process (SGP) framework [11] is depicted in Figure 3, alongside the corresponding terrain models obtained using the proposed SRF framework for different kernel types. As a measure of variance for the SRF framework we used the Euclidean distance to the nearest cluster, so areas further away from available data are considered to have higher uncertainty. From these results we can see that the SRF framework produces more visually detailed representations, due to its natural non-stationarity, and that different kernels produce sharper or smoother models, depending on their individual properties. Quantitative results can be found in Table 2 for the Gaussian kernel and varying levels of data sparsity (percentage of available data used for training, while the remaining is used for testing), including comparisons with the VHR framework [15] and the SRF framework with direct kernel calculation, rather than using random features for approximation ($R = 200$ was set for all experiments). These results show that the SRF framework is able to achieve accuracies comparable to other techniques, while doing so more efficiently, and the introduction of approximated kernel calculation further decreases computing times by a factor of roughly 15% relative to direct kernel calculation. Decreasing R would further improve computing times, at the expense of approximation accuracy (see Figure 1).



(a) SGP (Gaussian kernel).

(b) SRF (Gaussian kernel).

(c) SRF (Matérn 1/2).

(d) SRF (Cauchy kernel).

Figure 3: Terrain modeling results using different techniques (colored by variance). For kernel approximations, $R = 200$ random samples were used with an average cluster distance of $0.25m$ (the Quick-Means algorithm [25] was used to generate inducing points location).

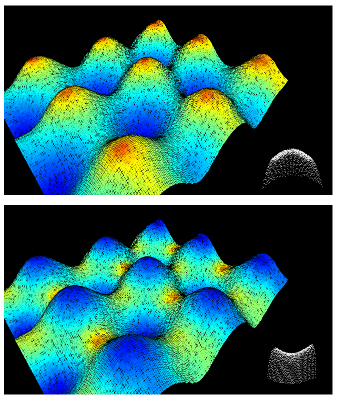
Table 2: Quantitative results for different regression techniques, with varying levels of data sparsity (RMSE values are in meters and time is measured in seconds, including training and querying).

	Method \ Sparsity	*95%*		*90%*		*70%*		*50%*	
		RMSE	*Time*	*RMSE*	*Time*	*RMSE*	*Time*	*RMSE*	*Time*
Terrain	**SGP**	0.186	470.22	0.194	357.36	0.217	198.59	0.239	100.19
	VHR	0.149	6.44	0.152	6.21	0.162	6.14	0.171	5.55
	SRF (kern.)	0.153	4.12	0.157	3.68	0.168	3.24	0.175	2.98
	SRF (appr.)	0.155	3.59	0.162	3.12	0.166	2.76	0.178	2.55
Objects	**SGP**	0.322	811.19	0.335	606.49	0.391	352.79	0.504	179.25
	SRF (kern.)	0.206	5.14	0.211	4.59	0.232	4.04	0.265	3.71
	SRF (appr.)	0.211	4.55	0.220	3.95	0.236	3.49	0.271	3.23

Similar regression results are depicted in Table 3, with a fixed 70% sparsity level and varying kernel types, where as expected we see that sharper representations, such as the Matérn 1/2 kernel, are better capable of capturing details in observed data. We also present convolution results, in which the observed environment is convolved with randomly selected $1.0m$-sided cubic subsets, simulating sensor data from a particular location, to produce a continuous function approximating activation values, based on different kernel types (see Figure 4). Pre-trained neural networks for convolution were used (see Section 3.2), and as expected introducing the wrong pre-trained model (i.e. based on different kernel types) significantly compromises RMSE values relative to a baseline discrete convolution with $0.1m$ space discretization. For comparison, computation times are also provided for discrete convolution, which as expected is considerably slower, due to the large mask sizes considered and need for space discretization, while the proposed SRF framework works directly on high-dimensional continuous representations. Results using direct kernel calculation, that use the closed-form solution for Gaussian convolution and thus eliminate the need for neural network approximations, are also presented, which even though relatively faster are restricted to that particular kernel type. These convolution results can be used as probability distributions for tasks such as global localization [25], in which the vehicle's current sensor data is convolved relative to the entire observed environment in search for areas with high similarity.

4.2 Implicit Surfaces

Similar tests were also conducted for implicit surface calculation, using object pointclouds available in `http://graphics.stanford.edu/data/3Dscanrep/`, namely the *Bunny*, *Dragon*, *Armadillo* and *Lucy* datasets. For each one, normal orientations were calculated and used to generate points inside or outside the object (random samples were collected, with a downsample of 100 relative to the original number of points). These received values of respectively $+1$ and -1 and were used to train regression models, where values of 0 represent the surface levels of the observed object. Some reconstruction examples using the proposed SRF framework can be found in Figure 5, colored by activation values relative to the convolution with randomly selected $0.1m$-sided cubic subsets


(a) Synthetic data (sine waves).

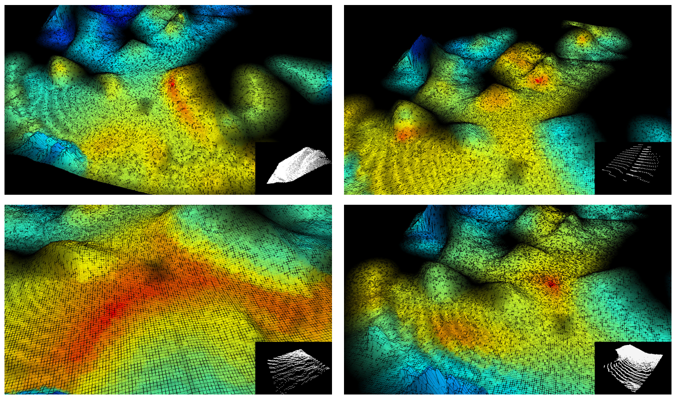

(b) *Terrain* dataset.

Figure 4: Convolution results in terrain modeling using the proposed SRF framework (original terrain model colored by activation value, relative to the pointcloud on the bottom right, with darker areas representing higher variance values).

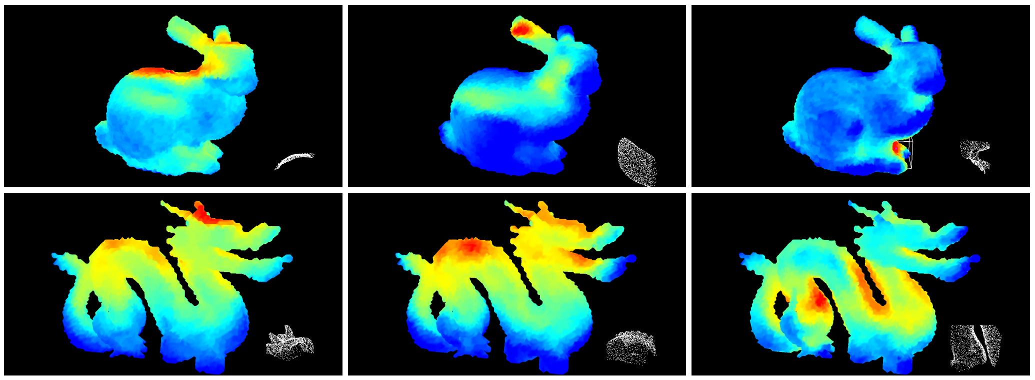


Figure 5: Convolution results in implicit surface calculation using the proposed SRF framework (original implicit surface models generated using the Marching Cubes algorithm [29] at 0 levels colored by activation value, relative to the pointcloud on the bottom right).

(all datasets were scaled to a larger dimension of $1m$, with an average cluster distance of $0.05m$ and space discretization of $0.02m$ for the purposes of discrete convolution). Quantitative results for regression and convolution tasks within these implicit surface models are provided in Tables 2 and 3, where all four datasets have been combined into a single *Objects* dataset and average values are provided. From these results we can see that the trends established in the terrain modeling experiments, such as faster computational times using the proposed SRF framework and better representations using sharper kernels, are maintained and become even more pronounced. This is due to a higher input dimensionality, that introduces more complexity in observed structures and makes discrete convolution in the 3D space more computationally demanding. The proposed SRF framework, on the other hand, experienced a much smaller increase in computational complexity from 2D to 3D data, which is to be expected since it only depends on the number of inducing points used for feature vector generation. These convolutional results can be used in tasks such as template matching [30], where discrete convolution becomes prohibitively expensive.

Table 3: Regression and convolution results for different techniques using various kernel types (RMSE values are in meters and time is measured in seconds, including training and querying).

		Regression			**Convolution** (average of 500 random masks)						
	Kernel	*SGP*	*SRF (kern.)*	*SRF (appr.)*	*SRF (kern.)*		*SRF (appr.)*				*Discrete*
					Gauss	*Time*	*Gauss*	*Mat12*	*Cauchy*	*Time*	*Time*
Terrain	**Gauss.**	0.162	0.168	0.166	0.257	1.08	0.264	0.481	0.439	1.21	19.41
	Mat.1/2	0.148	0.153	0.155	$--$	$--$	0.420	0.275	0.478	1.18	22.34
	Cauchy	0.155	0.164	0.161	$--$	$--$	0.369	0.504	0.261	1.23	20.77
Objects	**Gauss.**	0.391	0.232	0.236	0.281	2.23	0.302	0.672	0.689	2.54	75.32
	Mat.1/2	0.322	0.201	0.198	$--$	$--$	0.789	0.248	0.777	2.57	83.11
	Cauchy	0.359	0.212	0.217	$--$	$--$	0.742	0.805	0.279	2.41	81.88

5 Conclusion

This paper introduces a novel regression methodology for terrain modeling and implicit surface calculation, based on high-dimensional data projection using a feature vector that relies on sparse random Fourier (SRF) approximations. We show that the proposed SRF framework outperforms similar techniques with direct kernel calculation, while being highly scalable to higher input dimensionalities, which is particularly useful in 3D applications. Additionally, a novel continuous convolution methodology within the SRF framework is proposed, that approximates convolutional results between any two shift-invariant kernels by learning a function that approximates the convolution operation directly in the random Fourier feature space. These results are used in tasks of global localization and template matching, achieving computational speeds orders of magnitude faster than standard discrete convolution while allowing the use of arbitrary kernels. Future work will focus on exploring different applications for the proposed regression framework, including simultaneous localization and mapping (SLAM) [31, 32] and automatic feature selection [33].

Acknowledgments

This research was supported by funding from the Faculty of Engineering & Information Technologies, The University of Sydney, under the Faculty Research Cluster Program.

References

[1] C. Plagemann, S. Mischke, S. Prentice, K. Kersting, N. Roy, and W. Burgard. Learning predictive terrain models for legged robot locomotion. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[2] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte. Non-stationary dependent gaussian processes for data fusion in large-scale terrain modeling. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[3] S. Dragiev, M. Toussaint, and M. Gienger. Gaussian process implicit surfaces for shape estimation and grasping. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2845–2850, 2011.

[4] J. Bloomenthal and B. Wyvill. *Introduction to Implicit Surfaces*. Morgan Kaufmann Publishers Inc., 1997.

[5] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[6] M. Jadidi, J. Miro, and G. Dissanayake. Gaussian processes autonomous mapping and exploration for range-sensing mobile robots. *Autonomous Robots*, 2017.

[7] S. Vasudevan, F. Ramos, E. Nettleton, and H. Durrant-Whyte. Gaussian process modeling of large-scale terrain. *Journal of Field Robotics (JFR)*, 26(10):812–840, 2010.

[8] C. Plagemann, K. Kersting, and W. Burgard. Nonstationary Gaussian process regression using point estimates of local smoothness. In *Proceedings of the European Conference on Machine Learning (ECML)*, 2008.

[9] O. Williams and A. Fitzgibbon. Gaussian process implicit surfaces. In *Gaussian Processes in Practice*, 2007.

[10] J. Quinonero-Candelas and C. Rasmussen. An unifying view of sparse approximate Gaussian process regression. *Journal of Machine Learning Research (JMLR)*, 6:1939–1959, 2005.

[11] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1257–1264, 2006.

[12] F. Ramos and L. Ott. Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent. In *Proceedings of Robotics: Science and Systems (RSS)*, 2015.

[13] L. Bottou. Stochastic gradient descent tricks. In *Neural Networks: Tricks of the Trade*, volume 7700, chapter 18, pages 421–436. Springer Berlin Heidelberg, 2nd edition, 2012.

[14] V. Guizilini and F. Ramos. Towards real-time 3d continuous occupancy mapping using hilbert maps. *International Journal of Robotics Research (IJRR)*, 37(6):566–584, 2018.

[15] V. Guizilini and F. Ramos. Variational hilbert regression with applications to terrain modeling. In *Proceedings of the International Symposium on Robotics Research (ISRR)*, 2017.

[16] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1313–1320, 2009.

[17] K. Pavel and S. David. Algorithms for efficient computation of convolution. In *esign and Architectures for Digital Signal Processing*, chapter 8, pages 179–209. InTech, 2013.

[18] B. Schölkopf, K. Muandet, K. Fukumizu, S. Harmeling, and J. Peters. Computing functions of random variables via reproducing kernel Hilbert space representations. *Statistics and Computing*, 25(4):755–766, 2015.

[19] P. Komarek. Logistic regression for data mining and high-dimensional classification. Technical report, Carnegie Mellon University, 2004.

[20] H. Zou and T. Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*, 2005.

[21] V. Guizilini and F. Ramos. Large-scale 3d scene reconstruction with Hilbert maps. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[22] I. Gihman and A. Skorohod. *The Theory of Stochastic Processes*. Springer Verlag, Berlin, Germany, 1974.

[23] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. Balcan, and L. Song. Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems (NIPS)*, volume 27, pages 3041–3049, 2014.

[24] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, 2008.

[25] V. Guizilini and F. Ramos. Iterative continuous convolution for 3d template matching and global localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[26] P. Bromiley. Products and convolutions of Gaussian probability density functions. Technical report, TINA Vision, 2003.

[27] B. Xu, N. Wang, T. Chen, and M. Li. Empirical evaluation of rectified activations in convolutional network. In *http://arxiv.org/abs/1505.00853*, 2015.

[28] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research (JMLR)*, 15:1929–1958, 2014.

[29] W. Lorensen and H. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *Proceedings of the Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, volume 14, pages 163–169, 1987.

[30] R. Brunelli. *Template Matching Techniques in Computer Vision: Theory and Practice*. John Wiley & Sons, 2009.

[31] M. Hasan and M. Abdellatif. Fast template matching of objects for visual slam. In *Proceedings of the International Conference on Intelligent Robotics and Applications*, 2012.

[32] K. Tateno, F. Tombari, I. Laina, and N. Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[33] V. Guizilini and F. Ramos. Unsupervised feature learning for 3d scene reconstruction with occupancy maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.