Large-Scale 3D Scene Reconstruction with Hilbert Maps

Vitor Guizilini[1] and Fabio Ramos[2]

Abstract— **3D scene reconstruction involves the volumetric modeling of space, and it is a fundamental step in a wide variety of robotic applications, including grasping, obstacle avoidance, path planning, mapping and many others. Nowadays, sensors are able to quickly collect vasts amounts of data, and the challenge has become one of storing and processing all this information in a timely manner, especially if real-time performance is required. Recently, a novel technique for the stochastic learning of discriminative models through continuous occupancy maps was proposed: Hilbert Maps [1], that is able to represent the input space at an arbitrary resolution while capturing statistical relationships between measurements. The original framework was proposed for 2D environments, and here we extend it to higher-dimensional spaces, addressing some of the challenges brought by the curse of dimensionality. Namely, we propose a method for the automatic selection of feature coordinate locations, and introduce the concept of localized automatic relevance determination (LARD) to the Hilbert Maps framework, in which different dimensions in the projected Hilbert space operate within independent length-scale values. The proposed technique was tested against other state-of-the-art 3D scene reconstruction tools in three different datasets: a simulated indoors environment, RIEGL laser scans and dense LSD-SLAM pointclouds. The results testify to the proposed framework's ability to model complex structures and correctly interpolate over unobserved areas of the input space while achieving real-time training and querying performances.**

I. INTRODUCTION

Having a reliable model of surrounding space is of vital importance in robotics, with applications ranging from grasping and object manipulation to obstacle avoidance and autonomous navigation. At its core, such model should be able to distinguish between occupied and unoccupied areas, i.e. which ones are safe for traversing and which ones would produce a collision. Additional desired properties include:

- **Probabilistic reasoning,** so the model can take into account inherent imprecisions in sensor measurements.
- **Spatial relationships,** so the model can interpolate and extrapolate available information in order to improve its estimates in unknown areas.
- **Incremental learning,** so the model can adapt to new information as it is collected during navigation.
- **Update and query efficiency,** so the model can be accessed in real-time, both to incorporate new information and to obtain estimates as they become necessary.

Initial models [2], [3] would discretize the space, maintaining a grid of equally-sized cubic volumes (*voxels*) that store information about that area. However, this approach is very memory-intensive, especially at finer resolutions and in higher dimensions, and as a simplification each cell is treated independently, thus ignoring spatial relationships. Over the years substantial work has been done to improve this initial framework, trying to achieve the properties above mentioned under different sets of circumstances [3], [4], [5], [6].

Nowadays, a popular state-of-the-art grid-based approach for 3D scene reconstruction is OctoMap [7], which works by maintaining a tree-like structure (an *octree*) that recursively divides the space into smaller areas. Areas with similar classification are merged and/or pruned, to control access time and memory usage, and new information is incorporated by adding new internal nodes. The Gaussian Process Occupancy Map (GPOM) framework, introduced in [8], addresses spatial dependency by placing a GP prior over the space of functions mapping locations to the occupancy class. It does not require a prior discretization of space, since it produces a continuous function that can be sampled at arbitrary resolutions, but it scales cubically with the number of training points, which limits its applicability to large-scale, high-dimensional datasets. A similar drawback can be found in the use of Gaussian Process Implicit Surfaces (GPIS) [9], that switches from classification to regression and introduces a new covariance function, the thin-plate, that is particularly suitable for this sort of application.

Recently, a novel continuous occupancy mapping technique was introduced: Hilbert Maps [1], that is able to represent real world complexity in a linear fashion by operating on a high-dimensional feature vector, that projects observations into a reproducing kernel Hilbert space (RKHS) [10]. The result is an elegant probabilistic framework that allows for very efficient stochastic gradient descent optimization over its parameters, and querying can be performed at arbitrary resolutions. The original paper focuses on 2D datasets, and while the extension to 3D environments is straightforward, it comes with some challenges brought by the curse of dimensionality, namely the number of feature coordinates necessary to properly represent the environment.

This paper addresses these challenges, and proposes a set of modifications that allows the use of the Hilbert Maps framework in higher-dimensional spaces with real-time update and query performances, while achieving results comparable to other state-of-the-art 3D scene reconstruction tools. The technical contributions of this paper are:

1) An automatic coordinate selection technique for Hilbert Maps feature vectors, that is scalable to higher-dimensional datasets.
2) A sparsification technique for feature vector calculation, that brings computational complexity down to

[1]Vitor Guizilini is with the School of Information Technologies, University of Sydney, Australia. vitor.guizilini@gmail.com
[2]Fabio Ramos is with the School of Information Technologies, University of Sydney, Australia. fabio.ramos@sydney.edu.au

Preprint submitted to 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received February 27, 2016.

$\mathcal{O}(\log M)$, where M is the number of feature coordinates.

3) The introduction of localized automatic relevance determination (LARD) into the Hilbert Maps framework, in which each feature coordinate maintains its own length-scale values, thus naturally inducing non-stationarity in different areas of the input space.

The remainder of this paper is structured as follows: Section II starts by describing the standard Hilbert Maps framework, and then introduces the proposed modifications for higher-dimensional processing. Section III describes the new training and updating methodology, necessary to incorporate to modifications here described. Section IV discusses experimental results using simulated and real large-scale datasets, obtained from laser scanners and a dense visual SLAM technique, and finally Section V concludes and presents future research directions.

II. HILBERT MAPS

A. Overview

For the task at hand, we assume a training dataset $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^N$, where $\mathbf{x}_i \in \Re^3$ is a point in the three-dimensional space[1] and $y_i = \{-1, +1\}$ is a classification variable that indicates the occupancy property of $\mathbf{x}_i$. This dataset is obtained incrementally, as the robot moves throughout the environment collecting distance measurements from nearby surfaces (i.e. with a laser scanner or camera triangulation). The point of contact of each distance beam is labeled as $+1$ (occupied), while the distance traversed by the beam is labeled as -1 (free). As in [1], a random sample from every one to two meters is enough to ensure a good coverage of empty space.

This dataset is used to incrementally learn a discriminative model $p(y|\mathbf{x}, \mathbf{w})$, parametrized by a vector $\mathbf{w}$, to predict the occupancy property of new query points $\mathbf{x}_*$. A simple *Logistic Regression classifier* (LR) is used [11], due to its training speed and direct extension to online learning. The probability of non-occupancy for a query point $\mathbf{x}_*$ is:

$$p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w}) = \frac{1}{1 + \exp\left(\mathbf{w}^T \Phi(\mathbf{x}_*)\right)}, \qquad (1)$$

and, conversely, $p(y_* = +1|\Phi(\mathbf{x}_*), \mathbf{w}) = 1 - p(y_* = -1|\Phi(\mathbf{x}_*), \mathbf{w})$ is the probability of occupancy. However, in contrast to conventional LR, this model has a nonlinear decision boundary, since it operates on feature vectors $\Phi(\mathbf{x})$ rather than directly on the value $\mathbf{x}$ of data points. In order to estimate the optimal parameters $\mathbf{w}$, we minimize the regularized negative log-likelihood (NLL) function, given by:

$$\begin{aligned} NLL(\mathbf{w}) &= \sum_{i=1}^{N} -\log p\left(y_i|\Phi(\mathbf{x}_i), \mathbf{w}\right) + R(\mathbf{w}) & (2) \\ &= \sum_{i=1}^{N} \Big(1 + \exp\Big(-y_i \mathbf{w}^T \dot{\Phi}(\mathbf{x}_i)\Big)\Big) + R(\mathbf{w}), & (3) \end{aligned}$$

[1]Even though not explored here, this framework can be trivially extended to higher-dimensional spaces, to address other sorts of classification problems.

where $R(\mathbf{w}) = \lambda_1 \|\mathbf{w}\|_2^2 + \lambda_2 \|\mathbf{w}\|_1$ is a regulariser term, used to prevent overfitting and to enforce sparseness in $\mathbf{w}$. Respectively, $\|.\|_1$ and $\|.\|_2$ are the L1 and L2 norms and λ_1 and λ_2 are parameters balancing the quadratic term and the degree of sparseness. One particularly interesting property of Eq. 3 is that it is suitable for stochastic gradient descent optimization (SGD) [12], since its negative log-likelihood value is the sum of the negative log-likelihoods of individual points. Under the SGD framework, the information contained in each training point provides one small step towards a local minimum [13], given by:

$$\mathbf{w}_t = \mathbf{w}_{t-1} - \eta_t A_t^{-1} \frac{\delta}{\delta \mathbf{w}} NNL(\mathbf{w}), \qquad (4)$$

where $\eta > 0$ is the learning rate and the matrix A is a preconditioner to accelerate convergence rate. In most cases, A can be set to the identity matrix, while η is either constant or asymptotically decay with the number of iterations. Note that this technique naturally lends itself to online learning, since new information can be added to the current model by incrementally performing the stochastic update step given by Eq. 4. This also ensures that computational costs remain constant regardless of dataset size, since each point is processed only once and can then be discarded.

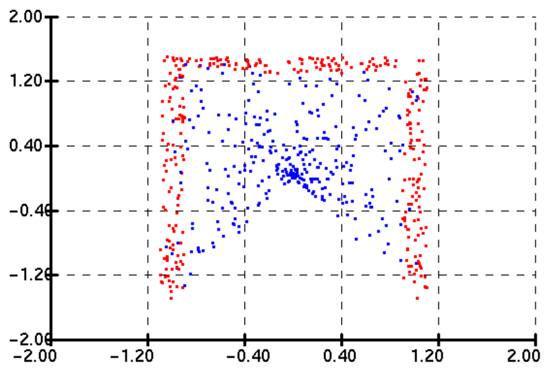
B. Feature Selection

A crucial insight in the Hilbert Maps (HM) framework is the application of its discriminative model not directly to the inputs $\mathbf{x}_i$, but rather on a high-dimensional feature vector $\Phi(\mathbf{x}_i)$ computed directly from $\mathbf{x}_i$. It is shown that the dot product of these feature vectors can approximate popular kernels used in the literature [14], e.g. $\Phi(\mathbf{x}_i)\Phi(\mathbf{x}_j) \approx k(\mathbf{x}_i, \mathbf{x}_j)$. In other words, they define a Hilbert space, and thus are able to represent a nonlinear mapping of the inputs to a space of potentially infinite dimension, while maintaining computational efficiency. In [1] several different features with the above mentioned property are presented and discussed, alongside their relationship to other well-known classification models.

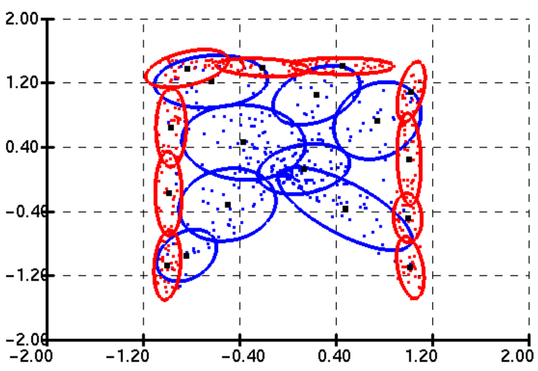
However, one important question remains: Where are these features located? Or, more specifically, what are the coordinates of its components in the input space? These are necessary in order to produce the feature vector $\Phi(\mathbf{x})$ that describes each point $\mathbf{x}$ in the Hilbert space. In essence, they act as anchors[2], correlating different input points based on a common distance metric: the kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$.

We start by augmenting the parameter set $\mathbf{w}$ to also include each respective coordinate $\mathbf{z}$, thus creating the parameter set $\mathcal{P} = \{\mathbf{z}_i, w_i\}_{i=1}^M$, where M is the number of feature coordinates and $\mathbf{z}_i \in \Re^3$ (note the similarity with the training dataset $\mathcal{D}$). For any point $\mathbf{x}$, its respective feature vector is

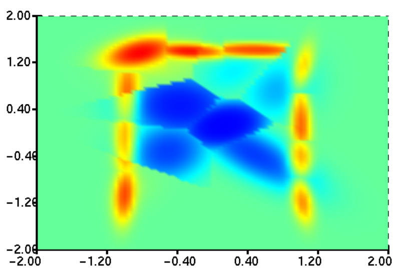
[2]Or inducing points, a well-known concept for sparse approximation in other learning frameworks [15].



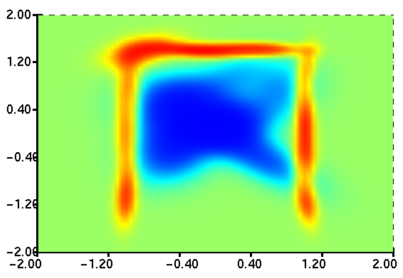
(a) Training points (labelled as free and occupied)



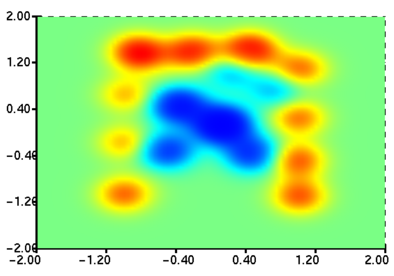
(b) Clustering results (10 occupied and 8 free clusters)



(c) LARD-HM results with $k = 1$



(d) LARD-HM results with $k = 5$



(e) HM results with $k = 5$

Fig. 1: 2D example of the proposed clustering and feature generation method. The training points in (a) are clustered using the *mini-batch k-means* algorithm, and mean and covariance values are extracted from each cluster, as shown in (b). A value of k nearest neighbors is then selected and used to produce the sparse feature vectors for the Hilbert Maps training and inference steps. In (c) only one nearest neighbor is used, which explains the jagged lines as sudden changes between one cluster and another. In (d) five nearest neighbors are used, thus producing a more smooth transition between different regions of the input space. Finally, (e) shows the effects of removing Localized Automatic Relevance Determination (LARD) from the Hilbert Maps framework.

defined as:

$$\Phi(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{z}_1) \\ k(\mathbf{x}, \mathbf{z}_2) \\ \vdots \\ k(\mathbf{x}, \mathbf{z}_M) \end{bmatrix}. \qquad (5)$$

In [1] these coordinates are sampled in a grid-like manner from a predetermined initial number, equally covering the entire input space. However, the curse of dimensionality quickly renders this approach infeasible for higher-dimensional problems, requiring an exceedingly large number of feature components in order to properly model the environment. Most of these components, however, will be irrelevant, since they correspond to empty areas of the input space, and therefore could be removed without impacting results.

Here we propose using available data to estimate feature coordinate location, to avoid generating irrelevant coordinates for any specific dataset. Clustering is a natural way to achieve that, and in particular the *k-means* algorithm [16] provides a quick and reliable technique to generate points that can be used as feature coordinates. Furthermore, to ensure scalability to large-scale datasets, the *mini-batch k-means* algorithm [17] has been shown to achieve a good compromise between speed and accuracy, being orders of magnitude faster than the standard implementation while consistently producing better results than its stochastic gradient descent counterpart [13]. *Mini-batch k-means* is also based on the principles of stochastic gradient descent such that our method can be seen as a version of a doubly stochastic gradient algorithm [18]. Pseudo-code for the *mini-batch k-means* algorithm can be found in Alg. 1, and Fig. 1b shows an example of clustering in a simple 2D dataset. Furthermore, empirical tests have shown that clustering each class (occupied and free) independently tends to yield better results.

Additionally, we can enforce sparsity by limiting the number of feature coordinates that are relevant for each data point. This approach is intuitive, since data points that are far away are expected to have less impact on each other's estimate. We achieve this by maintaining a *KD-tree* [19] with the position of all feature coordinates. For each data point $\mathbf{x}$, the k nearest neighbors are searched, and these are the coordinates that will receive their corresponding $k(\mathbf{x}, \mathbf{z}_m)$ kernel values (see Eq. 5), while the all the others are set to

Algorithm 1 Mini-Batch K-Means

```
Input: data points $X$
       number of clusters $m$ and iterations $t$ , batch size $b$
       distance function $d(\mathbf{x}_i, \mathbf{x}_j)$
Output: cluster centers $C$
 1: $C \leftarrow m$ points picked randomly from $X$
 2: $\mathbf{v} \leftarrow \mathbf{0}$   // Initialise per-cluster counter vector
 3: for  i = 1 to $t$  do
 4:     $B \leftarrow b$ points picked randomly from X
 5:     for  $\mathbf{x} \in B$  do
 6:         $C[\mathbf{x}] \leftarrow \min d(\mathbf{x}, C)$   // Store center nearest to $\mathbf{x}$
 7:     end for
 8:     for  $\mathbf{x} \in B$  do
 9:         $\mathbf{c} \leftarrow C[\mathbf{x}]$   // Get stored center for $\mathbf{x}$
10:         $\mathbf{v}[\mathbf{x}] \leftarrow \mathbf{v}[\mathbf{x}] + 1$   // Update per-cluster counter
11:         $\eta \leftarrow \frac{1}{\mathbf{v}[\mathbf{x}]}$   // Calculate learning rate
12:         $\mathbf{c} \leftarrow (1 - \eta)\mathbf{c} + \eta\mathbf{x}$   // Take gradient step
13:     end for
14: end for
```

zero. Using this approach, the cost for calculating one feature is now $\mathcal{O}(k \log M)$ instead of $\mathcal{O}(M)$, due to the search cost for nearest neighbors.

C. Local Automatic Relevance Determination

Automatic Relevance Determination (ARD) is a common technique in Bayesian learning [20], used to scale the input space according to how relevant each dimension is for predictive purposes. The larger a length-scale l is, the less a function $f(\frac{x}{l^2})$ responds to changes in x, and in the limit $l \to \infty$ the function f does not change at all, essentially rendering any knowledge of x useless for the discriminative model.

In this section we address the effects of introducing independent length-scales for each feature coordinate, and how it produces non-stationarity in the input space. We start by augmenting the parameter dataset introduced in Section II-B to also include a 3×3 length-scale matrix Σ_i, such that $\mathcal{P} = \{\mathbf{z}_i, \Sigma_i, w_i\}_{i=1}^{M}$. Similarly to how the values of $\mathbf{z}$ were obtained from the centroids of each cluster, the length-scale matrices can be calculated as the covariance matrix of the points belonging to each cluster:

$$\mathbf{z}^m = \{\bar{x}^m, \bar{y}^m, \bar{z}^m\} = \frac{1}{N^m} \sum_{i=0}^{N^m} \mathbf{x}_i^m \tag{6}$$

$$\Sigma^m = \frac{\sum_{i=0}^{N_m}}{N^m - 1} \begin{bmatrix} (\Delta x_i^m)^2 & \Delta y_i^m \Delta x_i^m & \Delta z_i^m \Delta x_i^m \\ \Delta x_i^m \Delta y_i^m & (\Delta y_i^m)^2 & \Delta z_i^m \Delta y_i^m \\ \Delta x_i^m \Delta z_i^m & \Delta y_i^m \Delta z_i^m & (\Delta z_i^m)^2 \end{bmatrix} \tag{7}$$

where $\Delta \mathbf{x}_i^m = \mathbf{x}_i^m - \mathbf{z}^m$. Following this approach, the length-scale of each dimension is tied to the corresponding variance of cluster points. Larger variances will produce slower changes and vice-versa, which is to be expected, since a larger variance indicate less certainty, rendering the transition between classes blurrier. The standard RBF kernel, with the introduction of multi-dimensional length-scales, is expressed as:

$$k(\mathbf{x}_i, \mathbf{x}_j, \Sigma) = \exp\left(-\frac{1}{2}(\mathbf{x}_i - \mathbf{x}_j)^T \Sigma^{-1} (\mathbf{x}_i - \mathbf{x}_j)\right), \tag{8}$$

while the feature vector $\Phi(\mathbf{x}_i)$, introduced in Eq. 5, is augmented to become:

$$\Phi(\mathbf{x}) = \begin{bmatrix} k(\mathbf{x}, \mathbf{z}_1, \Sigma_1) \\ k(\mathbf{x}, \mathbf{z}_2, \Sigma_2) \\ \vdots \\ k(\mathbf{x}, \mathbf{z}_M, \Sigma_M) \end{bmatrix}. \tag{9}$$

The benefits of introducing Localized Automatic Relevance Determination (LARD) into the Hilbert Maps framework can be seen in Fig. 1e, where it has been switched off in relation to Fig. 1d. Without LARD, only the distance from each cluster is taken into consideration, and therefore classification is restricted to circular areas around the centroids. With LARD, the shape of each cluster dictates classification area, which allows for a better representation of different structures in the environment and interpolation in unobserved areas of the input space.

III. INCREMENTAL LEARNING

The training methodology present in the Hilbert Maps framework, as shown in [1], lends itself naturally for online learning, where new data is continuously obtained and used to build a discriminative model of the environment. However, the modifications proposed in Sec. II, namely the clustering of data to produce the mean $\mathbf{z}_m$ and covariance Σ_M values for each feature coordinate, introduce some obstacles for the straightforward application of Hilbert Maps for online learning. Every time new data $\mathcal{D}' = \{\mathbf{x}_i, y\}_{i=1}^{P}$ is obtained, it falls under one of the following two categories:

1) **Unobserved areas**: As new areas of the input space are observed, extra clusters should be produced to populate these areas and incorporate this new information.
2) **Revisited areas**: As areas of the input space are observed again, this new information should be added to the clusters already populating these areas.

Category 1 can be performed by clustering the new data, to produce a new parameter dataset $\mathcal{P}'$, and append it to the current parameter dataset $\mathcal{P} = \{\mathcal{P}, \mathcal{P}'\}$ as new coordinates of $\Phi(\mathbf{x})$. The initial weight parameters for these coordinates are set to $\mathbf{w} = \mathbf{0}$ and then trained by iteratively running the SGD update step shown in Eq. 4.

Category 2 involves determining if an area of the input space with new information has already been visited (i.e. has clusters attached to it). This is done by first clustering the new data, to produce a new parameter dataset $\mathcal{P}'$, and then compare each new cluster to the ones present in $\mathcal{P}$, searching for proximity. An initial filter eliminates matches that are further away than ζ in the Euclidean space. The potential matches are compared in length-scale space, using their respective Σ matrices. If their coordinates fall both within one standard deviation of each other, they are considered the same cluster and merged together to produce new mean $\mathbf{z}$ and covariance Σ values for that particular feature coordinate. Pseudo-code for this incremental learning technique can be found in Alg. 2.

Algorithm 2 Incremental Learning for LARD-HM

```
Input:  new data points $\mathcal{D}^n = \{X, \mathbf{y}\}$
        current parameter dataset $\mathcal{P}^c = \{Z, \mathbf{\Sigma}, \mathbf{w}\}$
        number of clusters $m$ and iterations $t$ , batch size $b$
        distance function $d(\mathbf{x}_i, \mathbf{x}_j, \Sigma)$ , threshold $\zeta$
Output: updated parameter dataset $\mathcal{P}^c = \{Z, \mathbf{\Sigma}, \mathbf{w}\}$
 1: $Z^n \leftarrow kmeans(X^n, m, t, b, d(\mathbf{x}_i, \mathbf{x}_j, I)$
 2: for $\mathbf{z}^n \in Z^n$ do
 3:     $merged \leftarrow false$
 4:     for $\mathbf{z}^c \in Z^c$ do
 5:         if $d(\mathbf{z}^n, \mathbf{z}^c, I) < \zeta$ then
 6:             if $d(\mathbf{z}^n, \mathbf{z}^c, \Sigma^c) < 1$ and $d(\mathbf{z}^n, \mathbf{z}^c, \Sigma^n) < 1$ then
 7:                 $\mathbf{z}^c \leftarrow \mathbf{z}^{n+c}$   // Update mean vector
 8:                 $\Sigma^c \leftarrow \Sigma^{n+c}$   // Update covariance matrix
 9:                 $merged \leftarrow true$
10:                 break
11:             end if
12:         end if
13:     end for
14:     if not $merged$ then
15:         $\mathcal{P}^c \leftarrow \{\mathbf{z}^n, \Sigma^n, 0\}$   // Add new cluster
16:     end if
17: end for
18: $\mathbf{w}^c \leftarrow updateSGD(\mathcal{D}^n, \mathcal{P}^c)$   // Update weight parameters
```

IV. EXPERIMENTS

In this section we test the proposed framework in three different datasets: a simulated 3D environment, data from a RIEGL laser scanner and data from a popular dense visual SLAM algorithm, LSD-SLAM [21]. The proposed framework is compared against one of the current state-of-the-art 3D mapping techniques, OctoMap [7], and a localized Gaussian process (GP) [14] solution, that takes each cluster of points described in Sec. IIb and trains an independent GP model for that particular region of the input space. Inference is performed by querying the test point on its k nearest neighbors and taking the inverse weighted average of each mean value in relation to its respective variance value.

In all experiments, unless noted otherwise, the squared exponential kernel in Eq. 8 was used, the sparse feature vector in Eq. 9 was obtained using $k = 5$ nearest neighbors, the learning rate in Eq. 4 was set to $\eta = 0.1$, a threshold of 0.5 was used to differentiate between occupied and free areas, and a resolution of $0.05m$ was selected for space discretization. For plotting purposes, when using the HM and GP frameworks, the marching cubes algorithm [22] was used, with (when applied) each vertex colored by the information present in its nearest neighbor[3]. All computations were performed on a 2.50x8 GHz notebook with OpenMP parallelization when applicable.

A. Simulated 3D Environment

The first dataset addressed here was composed of $50k$ points and simulated 3 scan lasers taken from an indoor environment of about $40m^2$ (random noise $\sigma^2 = \mathcal{N}(0, 0.1)$ was added to each data point), as seen in Fig. 2a. The processing times for each major task present in the LARD-HM framework can be found in Table I. It is clear that most of the processing time is spent on clustering ($m = 500$) and feature calculation, while an almost negligible amount is spent on actually training weight parameters and querying test points. A similar comparison, now between different 3D scene reconstruction methods, can be found in Table II, where we can see that LARD-HM, while slower than Octomap, manages to increase computational speed by an order of magnitude over the original HM framework proposed in [1]. It also requires significantly less memory, since LARD-HM maintains one weight parameter for each cluster, while the original HM framework maintains weights distributed on a grid covering the entire input space.

[3]All coding and plotting for this paper was done using CVPP, a home-grown C++ library freely available in *https://bitbucket.org/vguizilini/cvpp*. A demo of LARD-HM is included in this library.

TABLE I: Average Processing times over 10 runs (msec) for different tasks in the LARD-HM framework, on the simulated dataset.

Task	Total time	Individual time
k-means clustering	471 ± 8	5.68×10^{-3} / point
Length-scale calculation	23 ± 4	4.60×10^{-2} / cluster
Training features calculation	116 ± 12	1.40×10^{-3} / feature
SGD training	4 ± 1	4.82×10^{-5} / feature
Grid features calculation	873 ± 121	1.37×10^{-3} / feature
Grid features query	17 ± 4	2.66×10^{-5} / feature

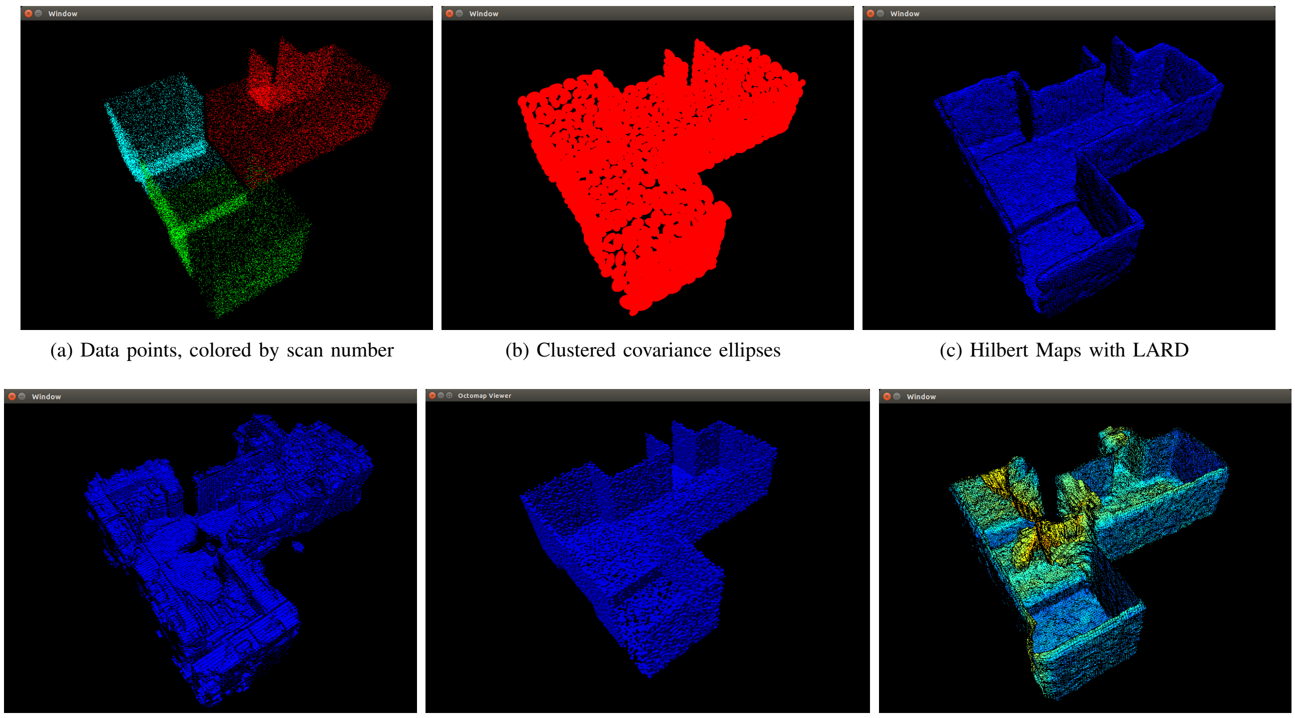
The 3D scene reconstruction results for LARD-HM, alongside Original HM (OHM), OctoMap and Localized Gaussian Processes (LGP) are shown in Fig. 2b, 2c and 2d respectively. The first noticeable aspect is that LARD-HM was able to seamlessly incorporate data from different sensors into a single scene (the processing order was green, blue and red, see Fig. 2a), which testifies to the effectiveness of the incremental learning technique described in Sec. III. It is also clear that LARD-HM is better suited for interpolation than OctoMap, and even the original HM framework, since it is capable to fill in gaps in the training data to smooth out inference results.

This effect becomes more prominent as we reduce the number of training points, as depicted in Fig. 3, where different ratios were set aside as ground-truth for testing. At 99% training points, LARD-HM reaches virtually 100% correct classifications, while OHM reaches 97% and OctoMap and LGP converge to roughly 89% and 82%, respectively. As we decrease the percentage of training points, LARD-HM and LGP remain stable until about 40% before starting to steeply

TABLE II: Average processing times over 10 runs (msec) for different methods of 3D scene reconstruction, on the simulated dataset.

Method	Training Time (82842 points)	Query Time (638683 points)
Original HM	3440 ± 285	25473 ± 5987
Local GP	7241 ± 873	72121 ± 10141
LARD-HM	216 ± 41	849 ± 55
OctoMap	130 ± 22	11 ± 2

Preprint submitted to 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received February 27, 2016.



(a) Data points, colored by scan number (b) Clustered covariance ellipses (c) Hilbert Maps with LARD

(d) Original Hilbert Maps (e) OctoMap (f) Localized GP (colored by variance)

Fig. 2: 3D scene reconstruction on a simulated environment using different techniques.

decline, while OHM and OctoMap consistently decrease in a roughly linear fashion. At 5% training points, LARD-HM still remains at about 78% correct classifications, while LGP, OHM and Octomap reaches about 66%, 62% and 55%, respectively.

The LGP framework has the interesting property of also providing variance estimates, which can be used to measure how certain the underlying model is of each particular surface estimate. These variance estimates serve as color texture for Fig. 2d, where we can see that variance increases as we move away from training points, into areas of the input space where there is still no available information. The HM framework can be extended to provide this variance information by substituting the LR classifier with a Bayesian regression technique, however this was not explored here.

B. RIEGL Dataset

The second dataset considered here was obtained using a RIEGL sensor, composed of about 1.4 million points spread over an area of $400m^2$. For this dataset $m = 1500$ clusters were automatically selected and all points were used for training. Fig. 4 shows the 3D scene reconstruction results using LARD-HM, with some zoomed in areas depicted in the surrounding. The effects of changing the ratio of training points (inducing sparsity) and number of clusters is depicted in Fig. 6. At 70% of training data the percentage of correct classifications stabilize at round 91%, and the same behavior happens at around 1000 clusters. We attribute this stabilization to the deterioration of each cluster information, since with a larger number of clusters fewer points are attributed to each one, which leads to worse mean and covariance

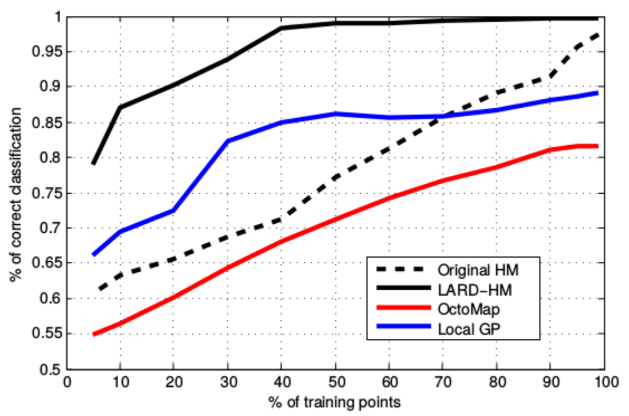


Fig. 3: Quantitative comparison on sparse simulated data, obtained by changing the ratio between training and testing points.

TABLE III: Percentage of correct classification for different methods, on real datasets (50% sparsity).

Method	RIEGL dataset	LSD-SLAM dataset
OctoMap	74.76%	68.23%
Original HM	81.48%	73.47%
LARD-HM	89.25%	83.89%

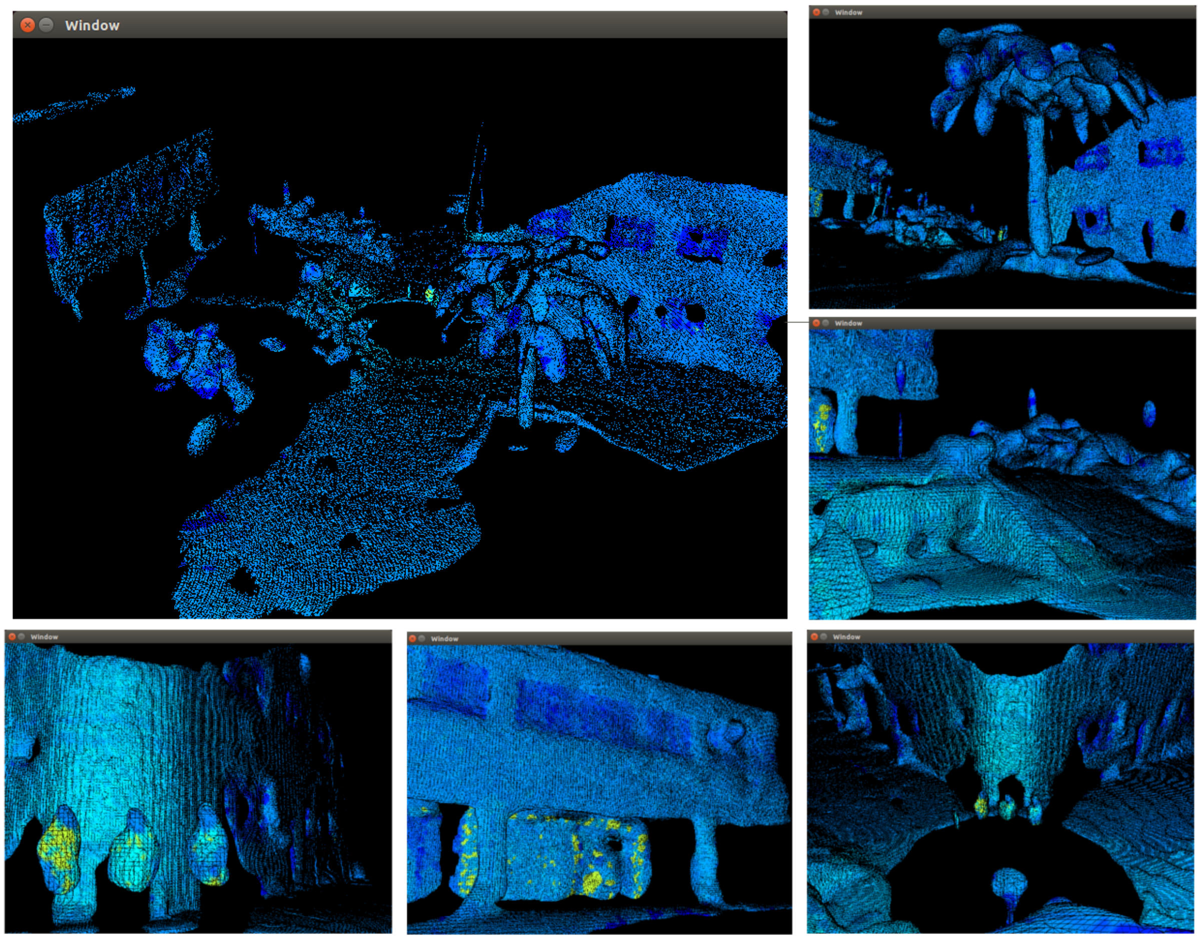


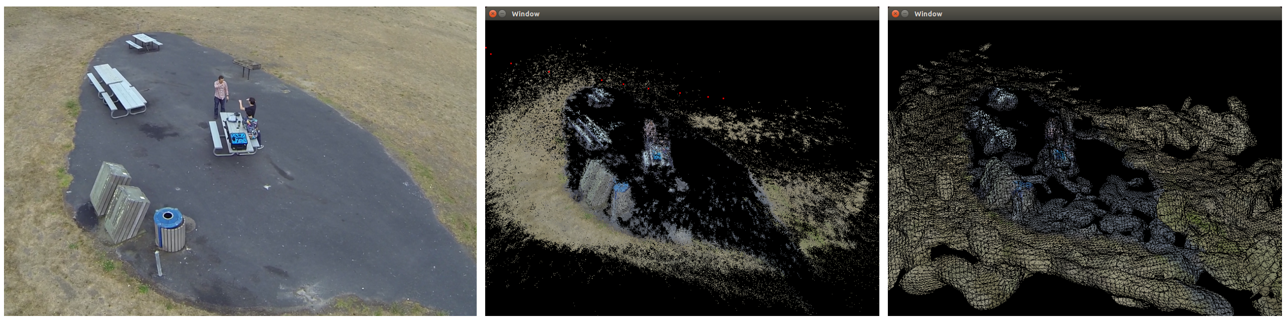
Fig. 4: 3D scene reconstruction results on the RIEGL dataset using LARD-HM.

estimates. In fact, to avoid numerical instability clusters with fewer than 5 points were merged with their nearest neighbors, effectively decreasing the number of considered clusters.

C. LSD-SLAM Dataset

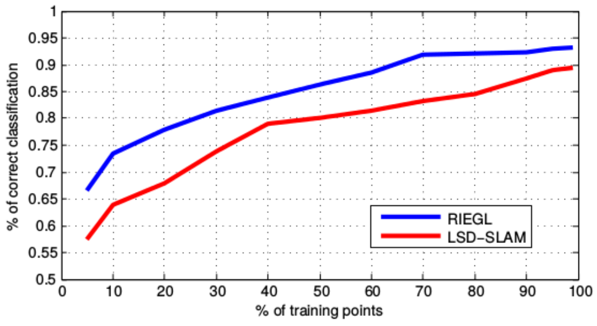
The last dataset considered here was obtained using visual information collected from a standard calibrated GoPro camera attached to an UAV (Solo, from 3DRobotics). This visual information was processed using LSD-SLAM [21] to produce a dense 3D pointcloud (Fig. 5b), composed of 15 keyframes obtained from different viewpoints of a (roughly) static environment, as seen in Fig. 5a. Each keyframe contributed with an average of $80k$ data points.

Note that a GoPro camera is not ideal for this type of application, and therefore the resulting 3D pointcloud has



(a) Original scene (b) Data points (c) LARD-HM results

Fig. 5: 3D scene reconstruction results on the LSD-SLAM dataset using LARD-HM.



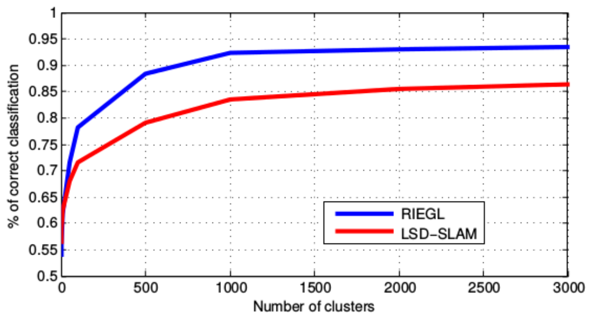


Fig. 6: Quantitative comparison for the RIEGL and LSD-SLAM datasets within the LARD-HM framework, showing the effects of sparse data and changing the number of clusters (80% of training data was used for cluster comparison).

substantial noise and alignment issues, due to drift accumulation and model inaccuracies. Even so, the LARD-HM framework was able to accurately reconstruct the structures present in the scene, as it can be seen in Fig. 5c. The information in each keyframe was incrementally added to the model, with 100 new clusters being calculated at each iteration, and either incorporated as new feature coordinates or merged with previous clusters.

V. Conclusion

This paper presented an extension of the Hilbert Maps framework to higher-dimensional spaces, addressing some of the challenges brought by the curse of dimensionality. A novel methodology for automatic feature coordinate selection was presented, and we introduced the concept of localized automatic relevance determination, that induces non-stationarity on the input space by maintaining different length-scale values for each feature coordinate. The result is a very efficient probabilistic classification technique, that rivals other state-of-the-art 3D scene reconstruction tools. Future work will focus on speeding up clustering and feature calculation, that are the bottleneck of the current implementation, the use of different kernel functions, and also explore the use of the proposed framework in other classification problems.

Acknowledgement

The authors would like to thank Lionel Ott, for kindly providing code for the original HM framework, and 3DRobotics, for the equipment used to obtain the LSD-SLAM dataset.

References

[1] F. T. Ramos and L. Ott, "Hilbert maps: Scalable continuous occupancy mapping with stochastic gradient descent." in *Robotic: Science and Systems (RSS)*, 2015.

[2] A. Elfes, "Occupancy grids: A probabilistic framework for robot perception and navigation," Ph.D. dissertation, Pittsburgh, USA, 1989.

[3] H. P. Moravec, "Robot spatial perception by stereoscopic vision and 3d evidence grids," Carnegie Mellon University: Robotics Institute, Tech. Rep., 1996.

[4] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, "6d slam: 3d mapping outdoor environments," *Journal of Field Robotics*, vol. 24, pp. 699–722, 2007.

[5] R. Hadsell, J. A. Bagnell, D. F. Huber, and M. Hebert, "Space-carving kernels for accurate rough terrain estimation." vol. 29, pp. 981–996, 2010.

[6] B. Douillard, J. Underwood, N. Melkumyan, S. Singh, S. Vasudevan, and A. Quadros, "Hybrid elevation maps: 3d surface models for segmentation," in *International Conference on Intelligent Robots and Systems (IROS)*. Proceedings of IEEE, 2010, pp. 1532–1538.

[7] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, vol. 34, pp. 189–206, 2013.

[8] S. T. O'Callaghan, F. T. Ramos, and H. F. Durrant-Whyte, "Contextual occupancy maps using gaussian processes." in *International Conference on Robotics Research (ICRA)*. Proceedings of IEEE, 2009, pp. 1054–1060.

[9] S. Dragiev, M. Toussaint, and M. Gienger, "Gaussian process implicit surfaces for shape estimation and grasping," in *International Conference on Robotics Research (ICRA)*. Proceedings of IEEE, 2011, pp. 2845–2850.

[10] B. Scholkopf and A. J. Smola, *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*, 2001.

[11] D. A. Freedman, *Statistical Models: Theory and Practice*. Cambridge University Press, 2005.

[12] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *International Conference on Computational Statistics (COMPSTAT)*, vol. 19. Springer-Verlag, 2010, pp. 177–187.

[13] ——, "Stochastic gradient descent tricks." in *Neural Networks: Tricks of the Trade (2nd Edition)*. Lecture Notes in Computer Science, 2012, vol. 7700, pp. 421–436.

[14] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. The MIT Press, 2005.

[15] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate gaussian process regression," *The Journal of Machine Learning Research (JMLR)*, vol. 6, pp. 1939–1959, 2005.

[16] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: Analysis and implementation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 24, pp. 881–892, 2002.

[17] D. Sculley, "Web-scale k-means clustering," in *International Conference on World Wide Web (WWW)*, vol. 19. ACM Digital Library, 2010, pp. 1177–1178.

[18] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song, "Scalable kernel methods via doubly stochastic gradients," in *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, Eds. Curran Associates, Inc., 2014, pp. 3041–3049.

[19] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *International Conference on Computer Vision Theory and Application (VISAPP)*, vol. 4. INSTICC Press, 2009, pp. 331–340.

[20] D. Wipf and S. S. Nagarajan, "A new view of automatic relevance determination," in *Advances in Neural Information Processing Systems (NIPS)*. Curran Associates, Inc., 2008, vol. 20, pp. 1625–1632.

[21] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *European Conference on Computer Vision (ECCV)*, vol. 13, 2014, pp. 834–849.

[22] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," in *Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, vol. 14. ACM Digital Library, 1987, pp. 163–169.

Preprint submitted to 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems. Received February 27, 2016.